



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2018-0122870
(43) 공개일자 2018년11월14일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01)

(52) CPC특허분류
G06F 11/3604 (2013.01)

(21) 출원번호 10-2017-0057043

(22) 출원일자 2017년05월04일

심사청구일자 2017년05월04일

(71) 출원인

고려대학교 산학협력단

서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)

(72) 발명자

오학주

서울특별시 성북구 아리랑로 89, 102동 103호(돈암동, 일신건영휴먼빌아파트)

소순범

서울특별시 성북구 개운사길 75-30, 005호(안암동5가)

(74) 대리인

김등용, 김홍석

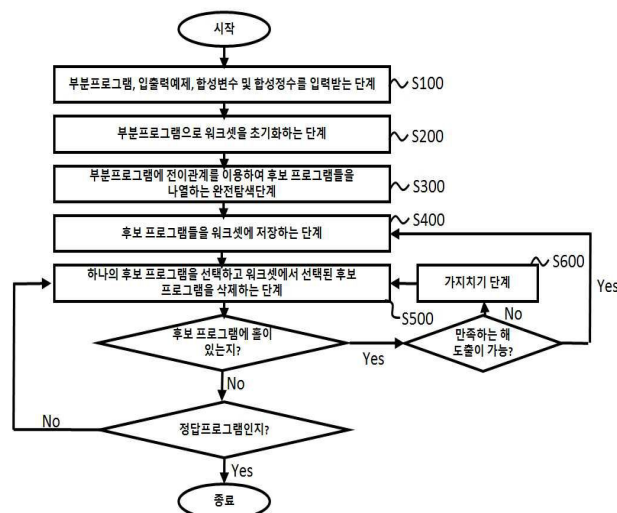
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 프로그램 합성 방법

(57) 요약

본 발명의 실시예에 따른 프로그램 합성방법은 프로그램 합성기에 의해서 수행되는 명령형 프로그램을 자동으로 합성하는 방법으로서, 미완성 프로그램인 부분프로그램, 입출력 예제, 합성변수 및 합성정수를 입력받는 단계, 상기 부분프로그램으로 워크셋(workset)을 초기화하는 단계, 상기 부분프로그램에 전이관계를 이용하여 후보 프로그램들을 나열하는 완전탐색단계, 상기 후보 프로그램들을 상기 워크셋에 저장하는 단계, 상기 후보 프로그램들 중 하나의 후보 프로그램을 선택하고, 상기 워크셋에서 선택된 후보 프로그램을 삭제하는 단계 및 선택된 상기 후보 프로그램에 홀이 있는 경우, 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계를 포함한다.

대표도 - 도3



이 발명을 지원한 국가연구개발사업

과제고유번호 R0190-16-2011

부처명 미래창조과학부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정보보호핵심원천기술개발사업

연구과제명 IoT 소프트웨어 보안 취약점 자동 분석 기술 개발

기 여 율 1/1

주관기관 고려대학교 산학협력단

연구기간 2016.06.01 ~ 2017.05.31

명세서

청구범위

청구항 1

프로그램 합성기에 의해서 수행되는 명령형 프로그램을 합성하는 방법으로서,

미완성 프로그램인 부분프로그램, 입출력 예제, 합성변수 및 합성정수를 입력받는 단계;

상기 부분프로그램으로 워크셋(workset)을 초기화하는 단계;

상기 부분프로그램에 전이관계(transition relation)를 이용하여 후보 프로그램들을 나열하는 완전탐색단계;

상기 후보 프로그램들을 상기 워크셋에 저장하는 단계;

상기 후보 프로그램들 중 하나의 후보 프로그램을 선택하고, 상기 워크셋에서 선택된 후보 프로그램을 삭제하는 단계; 및

선택된 상기 후보 프로그램에 홀이 있는 경우, 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 2

제1항에 있어서,

상기 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계 이후,

판정결과 입출력 예제를 만족하는 해가 도출되는 것이 가능한 경우, 전이관계(transition relation)를 통해 재차 후보 프로그램을 추출하여 상기 워크셋에 추가하는 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 3

제1항에 있어서,

상기 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계 이후,

판정결과 입출력 예제를 만족하는 해가 도출되는 것이 불가능한 경우, 홀이 있는 상기 후보 프로그램을 삭제하는 가지치기(pruning) 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 4

제3항에 있어서,

상기 가지치기 단계는 상기 부분 프로그램의 홀이 모두 채워져 완성되었을 때 가능한 모든 실제 실행 의미(concrete semantics)를 대표하는 요약 의미(abstract semantics)를 정의하고, 상기 요약 의미가 출력예제를 만족하지 못하는 프로그램을 삭제하는 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 5

제1항에 있어서,

상기 부분프로그램에 전이관계(transition relation)를 이용하여 후보 프로그램들을 나열하는 완전탐색단계는

상기 부분프로그램과 입력된 상기 합성정수 및 합성변수를 이용하여 후보 프로그램들을 나열하는 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 6

제1항에 있어서,

상기 후보 프로그램들을 상기 워크셋에 저장하는 단계는

상기 후보 프로그램들 중 중복되는 프로그램을 제거하도록 코드 최적화하여 저장하는 단계를 포함하는 것을 특징으로 하는 프로그램 합성 방법.

청구항 7

제1항 내지 제6항 중 어느 한 항의 방법을 수행하기 위한 프로그램이 저장된 컴퓨터로 판독 가능한 저장매체.

발명의 설명

기술 분야

[0001] 본 발명은 프로그램을 합성하는 방법에 관한 것으로서, 보다 상세하게는 명령형 프로그램을 자동으로 합성하는 방법에 관한 것이다.

배경 기술

[0002] 최근 컴퓨터 프로그래밍 교육, 즉 코딩교육에 대한 관심이 지고 있는 상황에서 우리나라도 2018년부터 소프트웨어 교육을 의무화하는 방안이 검토중에 있다. 그러나 현재는 프로그래밍 교육 전문 교사들의 부재로 인해 내실 있는 교육이 이루어지기는 어려운 실정이다. 또한, 교사 1명이 학급의 모든 학생들에게 맞춤 교육을 제공하는 과목을 특성상 쉽지 않은 실정이고, 교사들도 때때로 실수를 할 수 있다. 온라인 프로그램을 교육하는 웹사이트가 현재에도 존재하기는 하나 자동 채점 시스템에 기반을 두고 있다. 자동 채점 시스템은 학생이 주어진 문제에 대해 작성한 프로그램을 제출하면 맞았는지 틀렸는지 여부를 알려주는데 국한되어 있다.

[0003] 자동 채점 시스템을 통해서 학생들은 자신이 궁금해 하는 부분에 대한 피드백을 받지 못하고, 결과를 확인하는 수준에 그치고 있다.

[0004] 이러한 상황으로 인해 사용자가 원하는 프로그램을 자동으로 생성하는 기술인 프로그램 합성기술은 날로 필요성이 커지고 있는 기술이다.

[0005] 프로그램을 합성하는 기술은 국내에서는 시도된 바가 없으며, 2010년 초반에 들어서야 세계적으로 관심을 받아, 연구가 진행되고 있다.

[0006] 일반적인 프로그래밍 언어로 구성된 프로그램을 합성하는 것은 매우 어려운 문제이다. 최근 해외 주요학회에서 출판된 논문들에서는 주로 목적 특화 언어(Domain Specific Language)를 정의함으로써 특정 목적만을 위한 프로그램을 합성하는데 집중하였다. 일례로 Sumit Gulwani et al. Automating string processing in spreadsheets using input-output examples, In proceedings of the 38th annual ACM SIGPLAN SIGACT Symposium on Principles of Programming Languages, 2011)에서는 주어진 문자열의 각 대문자를 추출하여 이음으로써 새로운 문자열을 생성하는 프로그램을 자동으로 합성하는 연구결과를 선보였다.

[0007] 그러나 이러한 프로그램의 합성기술에서 프로그램 합성을 위한 연산속도에 대한 고려는 전혀 없어 프로그램을 합성하는데 너무 많은 시간이 소요되는 문제가 있다.

선행기술문헌

특허문헌

[0008] (특허문헌 0001) 미국등록특허 US 7,577,935 B2(2009.08.18)

(특허문헌 0002) 대한민국 공개특허 KR 2007-0061326(2007.06.13)

발명의 내용

해결하려는 과제

[0009] 본 발명의 목적은 목적 특화 언어로 정의되는 프로그램을 합성하는 대신 보다 일반적인 문법을 지닌 언어로 된 프로그램을 합성하는 프로그램 합성방법을 제공하는데 있다.

과제의 해결 수단

- [0010] 본 발명의 실시예에 따른 프로그램 합성방법은 프로그램 합성기에 의해서 수행되는 명령형 프로그램을 자동으로 합성하는 방법으로서, 미완성 프로그램인 부분 프로그램, 입출력 예제, 합성변수 및 합성정수를 입력받는 단계, 상기 부분프로그램으로 워크셋(workset) 을 초기화하는 단계, 상기 부분프로그램에 전이관계(transition relation)에 의해 후보 프로그램들을 나열하는 완전 탐색(enumerative search) 단계, 상기 후보 프로그램들을 상기 워크셋에 저장하는 단계, 상기 후보 프로그램들 중 하나의 후보 프로그램을 선택하고, 상기 워크셋에서 선택된 후보 프로그램을 삭제하는 단계 및 선택된 상기 후보 프로그램에 홀이 있는 경우, 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계를 포함한다.
- [0011] 여기서, 상기 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계 이후, 판정결과 입출력 예제를 만족하는 해가 도출되는 것이 가능한 경우, 전이관계(transition relation)를 통해 재차 후보 프로그램을 추출하여 상기 워크셋에 추가하는 단계를 포함한다.
- [0012] 여기서, 상기 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계 이후, 판정결과 입출력 예제를 만족하는 해가 도출되는 것이 불가능한 경우, 홀이 있는 상기 후보 프로그램을 삭제하는 가지치기(pruning) 단계를 포함한다.
- [0013] 여기서, 상기 가지치기 단계는 상기 부분 프로그램의 홀이 모두 채워져 완성되었을 때 가능한 모든 실제 실행 의미(concrete semantics)를 대표하는 요약 의미(abstract semantics)를 정의하고, 상기 요약 의미가 출력예제를 만족하지 못하는 프로그램을 삭제하는 단계를 포함한다.
- [0014] 여기서, 상기 부분프로그램에 전이관계(transition relation)를 이용하여 후보 프로그램들을 나열하는 완전탐색 단계는 상기 부분프로그램과 입력된 상기 합성정수 및 합성변수를 이용하여 상기 부분 프로그램에 후보 프로그램들을 나열하는 단계를 포함한다.
- [0015] 여기서, 상기 후보 프로그램들을 상기 워크셋에 저장하는 단계는 상기 후보 프로그램들 중 중복되는 프로그램을 제거하도록 코드 최적화하여 저장하는 단계를 포함한다.

발명의 효과

- [0016] 본 발명의 실시예에 따른 프로그램 합성 방법에 의하면 일반적인 문법을 지닌 언어로 된 프로그램을 합성할 수 있다.
- [0017] 또한, 본 발명의 실시예에 따른 프로그램 합성 방법에 의하면 프로그램을 합성하는 속도가 개선될 수 있다.

도면의 간단한 설명

- [0018] 도 1은 합성프로그램 예시이다.
- 도 2는 문맥자유문법에 의해 정의된 명령형 언어이다.
- 도 3은 본 발명의 실시예에 따른 프로그램 합성방법의 순서도이다.
- 도 4는 전이관계를 표현한 도식이다.
- 도 5는 본 발명의 실시예에 따른 완전탐색과정을 통해서 도출된 후보 프로그램의 예시이다.
- 도 6은 코드 최적화가 완료된 후보 프로그램들이다.
- 도 7은 입출력 예제를 만족하는 해가 도출될 가능성이 없는 경우의 유형을 설명하기 위한 프로그램이다.
- 도 8은 프로그램 실행 요약 의미를 표현한 도면이다.
- 도 9는 본 발명의 실시예에 따른 프로그램 합성 방법을 이용하여 실제 실험한 결과표이다.

발명을 실시하기 위한 구체적인 내용

- [0019] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시 예들에 대해서 특정한 구조적 또는 기능적 설명은 단지 본 발명의 개념에 따른 실시 예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시 예들은 다양한 형태로 실시될 수 있으며 본 명세서에 설명된 실시 예들에 한정되지 않는다.

- [0020] 본 발명의 개념에 따른 실시 예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시 예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시 예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.
- [0021] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.
- [0022] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.
- [0023] 본 명세서에서 사용한 용어는 단지 특정한 실시 예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.
- [0024] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.
- [0025] 본 발명인 프로그램 합성 방법에 대해서 상세히 설명하기에 앞서 본 발명인 프로그램 합성 방법이 실제로 실행된 예를 통해서 본 발명에 대한 대략적인 내용을 살펴본다.
- [0026] 도 1은 합성프로그램 예시이다.
- [0027] 먼저, 입력된 숫자를 서로 뒤집어서 출력하는 프로그램을 작성하기 위해, 프로그램의 일부내용이 생략된 부분프로그램을 사용자가 아래와 같이 작성했다고 가정한다.
- [0028] Reverse (n) {r:=0; while (?) {?}; return r;}
- [0029] 위 부분프로그램에서 reverse라는 함수는 n을 입력받아 r에 저장된 값을 내보내야 하는데 while-loop의 조건과 몸체를 어떻게 작성할지 몰라 ""로 표시한 상황이다.
- [0030] 본 발명의 실시예에 따른 프로그램 합성방법에 의하여 프로그램을 완성하려면 부분프로그램 외에 추가로 입출력 예제, 합성변수, 합성정수를 이용한다.
- [0031] 입출력예제로 {1 → 1, 12 → 21, 123 → 321}이 주어지고, 합성변수는 {n,r,x}, 합성정수는 {0,1,10}으로 주어졌다고 가정하면, 본 발명의 실시예에 따른 프로그램 합성 방법에 의해서 도 1과 같은 프로그램이 완성된다.
- [0032] 본 발명의 실시예에 따른 프로그램 합성 방법에 의한 합성 대상인 후보 프로그램의 명령형 언어는 문맥 자유문법(Context Free Grammar)에 의해 정의된다.
- [0033] 도 2는 문맥자유문법에 의해 정의된 명령형 언어이다.
- [0034] 도 2에 도시된 바와 같이 ℓ -value(ℓ)는 변수 (x) 또는 배열 원소 참조 표현식 (x[y])이다. 산술 표현식(a)은 상수인 정수(n), l-value(l), 혹은 이항 산술 연산식($\circ$)이다.
- [0035] 논리 표현식(b)은 논리상수(true, false), 이항관계 연산식($<$), 논리부정($\neg$), 논리곱($\wedge$), 논리합($\vee$)을 포함한다.

- [0036] 명령문(c)은 할당문($\ell := a$), 스킵(skip), 시퀀스($c_1; c_2$), 조건문(if b c_1 c_2), 반복문(while b c)을 포함한다.
- [0037] 특히 일반적인 명령형 프로그램 언어와 달리 홀(hole) 표현($\diamond, \triangle, \square$)을 이용하여 모르는 부분을 표시할 수 있다. 각 산술 표현식, 논리 표현식, 명령문에 대한 홀($\diamond, \triangle, \square$) 표현들은 문맥 자유 문법에 의해 엄밀하게 구분되지만 실제 사용자 인터페이스에서는 간단하게 물음표(?)를 이용하여 표시한다.
- [0038] 도 3은 본 발명의 실시예에 따른 프로그램 합성 방법의 순서도이다.
- [0039] 도 3에 도시된 바와 같이 본 발명의 실시예에 따른 프로그램 합성 방법은 프로그램 합성기에 의해서 수행되는 명령형 프로그램을 자동으로 합성하는 방법으로서,
- [0040] 미완성 프로그램인 부분프로그램, 입출력 예제, 합성변수 및 합성정수를 입력받는 단계(S100), 상기 부분프로그램으로 워크셋(workset)을 초기화하는 단계(S200), 상기 부분프로그램에 전이관계(transition relation)를 이용하여 후보 프로그램들을 나열하는 완전탐색단계(S300), 상기 후보 프로그램들을 상기 워크셋에 저장하는 단계(S400), 상기 후보 프로그램들 중 하나의 후보 프로그램을 선택하고, 상기 워크셋에서 선택된 후보 프로그램을 삭제하는 단계(S500) 및 선택된 상기 후보 프로그램에 홀이 있는 경우, 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정하는 단계를 포함한다.
- [0041] 앞서도 언급했던 부분 프로그램이란 홀 표현을 가지고 있는 프로그램을 의미한다. 또한, 입출력 예제는 최종적으로 완성된 프로그램이 만족시켜야 할 명제이다. 다시 말하면 부분 프로그램을 입력으로 받아서, 각 입력 예제에 대해 짝지어진 출력 예제 결과를 내는 프로그램으로 완성시키는 역할을 한다. 본 발명의 실시예에 따른 합성 방법에서는 각 입출력예제가 정수 혹은 정수의 배열들로 구성되며, 목적에 따라 문자열이 추가될 수 있다.
- [0042] 합성변수는 부분 프로그램을 입출력 예제를 만족하는 프로그램으로 완성시키기 위해 사용되는 변수들로서 본 발명의 실시예에 다른 합성방법에서 변수의 타입은 사용자로부터 구분되어 입력 받게 된다. 즉, 정수형 변수인지 배열 변수인지를 구별하여 입력 받는다.
- [0043] 합성정수는 부분 프로그램을 입출력 예제를 만족하는 프로그램으로 완성시키기 위해 사용될 정수들이다.
- [0044] 이와 같이 본 발명의 실시예에 따른 프로그램 합성 방법은 부분 프로그램, 입출력 예제, 합성변수 및 합성정수를 입력받는다(S100).
- [0045] 다음으로 본 발명의 실시예에 따른 프로그램 합성 방법은 부분 프로그램으로 워크셋을 초기화하는 단계(S200)를 포함한다.
- [0046] 워크셋이란 부분 프로그램을 완성된 프로그램으로 완성하기 위해서 후보 프로그램들 즉, 후보 프로그램 후보군들의 모임이다. 부분 프로그램이 입력되면 부분 프로그램에 따라서 워크셋이 초기화된다. 초기화를 통해서 워크셋에 부분 프로그램이 저장된다.
- [0047] 본 발명의 실시예에 따른 프로그램 합성 방법은 상기 부분 프로그램에 전이관계(transition relation)를 이용하여 후보 프로그램들을 나열하는 완전탐색단계를 포함한다(S300). 보다 상세하게 완전탐색단계는 상기 부분프로그램과 입력된 상기 합성정수 및 합성변수를 이용하여 상기 부분 프로그램에 후보 프로그램들을 나열하는 단계를 포함한다.
- [0048] 도 4는 전이관계를 표현한 도식이다.
- [0049] 완전탐색단계(S300)는 부분 프로그램과 후보 프로그램을 나열하는 단계로서, 도 4에 도시된 전이관계를 이용하여 후보 프로그램을 탐색한다.
- [0050] 도 4에서 하나의 홀은 같은 종류의 표현식(a, b, c)으로 치환될 수 있다. 부분 프로그램 p가 주어졌을 때, 프로그램 p와 합성이 가능한 프로그램은 도 4의 전이관계에 의해서 전이된 프로그램들의 집합으로 구해진다. 기 입력된 합성변수의 조합들로 ℓ -value(ℓ)은 사용자로부터 입력받은 합성변수의 조합들로 구해지고, n은 사용자로부터 입력받은 합성정수들에서의 임의의 원소에 해당한다.
- [0051] 도 5는 본 발명의 실시예에 따른 완전탐색과정을 통해서 도출된 후보 프로그램의 예시이다.
- [0052] 도 5에서는 합성정수의 세트를 (Gamma)라하고, 정수타입의 합성변수의 세트를 Σ (Sigma)라 한다. 또한, 최초의 프로그램을 p_0 라 한다.

- [0053] $p_0 = \text{pgm}(x)\{\square; r := 1; r := \diamond; \text{return } r\}$
- [0054] 이때, 앞서 설명한 바와 같이 최초의 워크셋은 $\{p_0\}$ 가 된다.
- [0055] 도 5는 주어진 합성정수와 합성변수에 의해서 전이관계에 의한 완전탐색단계를 수행한 결과이다.
- [0056] 도 5에 도시된 것은 부분 프로그램 p_0 의 $\square$ (명령문)와 $\diamond$ (산술표현식)와 같은 홀을 채울 수 있는 모든 후보 프로그램을 부분 프로그램 p_0 에 도입한 결과이다.
- [0057] 완전 탐색을 수행하면서 종료하지 않은 프로그램을 합성하는 경우도 발생하는데 이러한 모든 프로그램들에 대해 종료여부를 판단할 수 있는 일반적인 알고리즘은 존재하지 않는다(정지 문제(halting problem)). 따라서 반복문(while-loop)을 합성할 때는 3가지 제약조건을 준다. 첫째, 조건문으로서 x (입력변수) $< y$ (출력변수) 혹은 $x > n$ (n 은 정수)의 형태만 허용하고, 둘째, 반복문 몸체(loop body)의 마지막에서는 카운팅 변수가 증가(조건문이 $x < y$ 일 경우) 혹은 감소(조건문이 $x > n$ 인 경우) 해야 하며, 셋째, 나머지 몸체에서는 조건문에 들어간 변수들이 재 정의 되어서는 안된다는 제약조건을 둘 수 있다.
- [0058] 본 발명의 실시예에 따른 프로그램 합성 방법은 다음 단계로, 상기 후보 프로그램들을 워크셋에 저장하는 단계(S400)를 거친다. 즉, 전이관계에 의한 완전탐색단계를 수행한 도 5와 같은 결과를 워크셋에 저장하는 단계(S400)이다.
- [0059] 또한, 워크셋에 저장하는 단계(S400)는 상기 후보 프로그램들 중 중복되는 프로그램을 제거하도록 코드 최적화하여 저장하는 단계를 포함한다.
- [0060] 도 6은 코드 최적화가 완료된 후보 프로그램들이다.
- [0061] 예를 들어 $\dots; r := 0; r := x^*0; \square; \dots$ 과 같은 프로그램은 코드 최적화를 통해서 $\dots; r := 0; \dots$ 과 같이 의미가 보존되면서 더 간단한 프로그램으로 변형된다.
- [0062] 본 발명의 실시예에 따른 코드 최적화 기술로는 constant propagation, copy propagation, dead code elimination, expression simplification 등 4가지의 코드 최적화 기술일 수 있으나, 반드시 이에 제한되는 것은 아니다.
- [0063] 다음으로, 상기 후보 프로그램들 중 하나의 후보 프로그램을 선택하고, 워크셋에서 선택된 후보 프로그램을 삭제하는 단계(S500)는 완전탐색을 통해서 후보 프로그램들을 생성하고 워크셋에 저장되어 있는 상태에서 하나의 프로그램을 선택하여 워크셋에서 꺼내고, 워크셋내에서 이를 삭제하는 단계이다.
- [0064] 하나의 프로그램을 선택하는 것은 가장 크기가 작은 프로그램부터 선택하여 꺼낼 수 있으며, 이는 최종적으로 합성되는 프로그램이 주어진 예제에 대해 과적합(overfitting)하게 되는 문제를 해결하기 위함이다.
- [0065] 다음으로 선택된 후보 프로그램에 홀이 있는 경우, 입출력 예제를 만족하는 해가 도출되는 것이 불가능한지 여부를 판정한다.
- [0066] 만일 홀이 없는 경우는 입력 받았던 입출력 예제와 일치하는지를 판단하여 정답 프로그램인지를 확인하게 되고, 정답 프로그램이 아닌 경우는 워크셋에 저장되어 있는 다른 후보 프로그램을 선택하여 홀이 있는 경우와 없는 경우를 판정하는 과정을 반복적으로 거친다.
- [0067] 만일 홀이 있는 경우는 판정결과 입출력 예제를 만족하는 해가 도출되는 것이 가능한 경우와 입출력 예제를 만족하는 해가 결코 도출될 수 없는 경우로 나뉜다.
- [0068] 홀이 있으면서 입출력 예제를 만족하는 해의 도출이 가능하다고 판정하는 경우에는 앞서 설명한 전이관계를 통해 재차 후보 프로그램을 추출하여 워크셋에 추가하는 단계를 거치게 되고, 입출력 예제를 만족하는 해가 결코 도출될 수 없는 경우라면 홀이 있는 후보 프로그램을 삭제하는 가지치기(pruning)단계(S600)를 거친다.
- [0069] 입출력 예제를 만족하는 해의 도출이 가능하다고 판정하여 전이관계를 통해 재차 후보 프로그램을 추출하고, 워크셋에 추가할 경우에는 이전에 설명했던 중복되는 프로그램을 제거하는 코드 최적화를 수행하여 워크셋에 추가할 수 있다.
- [0070] 이하 본 발명의 실시예에 따른 가지치기 단계에 대해서 상세히 살펴본다.

- [0071] 도 7은 입출력 예제를 만족하는 해가 도출될 가능성이 없는 경우의 유형을 설명하기 위한 프로그램이다.
- [0072] 도 7의 유형 1은 입출력 예제 중 (1,1)이 있다고 가정하고 유형 1의 프로그램이 1을 입력받으면 1을 출력해야 하는데 산술 표현식 $홀(\Diamond)$ 이 어떠한 표현식으로 구체화 된다고 해도 입력 1이 주어졌을 때 절대로 1을 출력값으로 내보낼 수 없다. 반복문의 조건식($n > 0$)과 반복문 몸체의 명령문 $r := n+1$;에 의해 리턴되는 r 의 값이 2 이상임이 확실하기 때문이다.
- [0073] 도 7의 유형 2는 마찬가지로 입출력 예제 중 (1,1)이 있다고 가정하고, 유형 2의 프로그램이 명령문 $홀(\Box)$ 이 어떠한 명령문으로 구체화 된다고 해도 입력 1이 주어졌을 때 절대로 1을 출력값으로 내보낼 수 없다. $x * 10 = 1$ 을 만족할 수 있는 x 값은 존재하지 않기 때문이다.
- [0074] 도 7의 유형 1은 구간분석(interval analysis)에 의해서 가지치기(pruning)를 하며, 유형 2는 심볼릭 분석(symbolic analysis)에 의해서 가지치기를 한다.
- [0075] 부분 프로그램의 홀이 모두 채워져 완성되었을 때 가능한 모든 실제 실행의미(concrete semantics)를 대표하는 요약 의미(abstract semantics)를 정의하고, 요약의미가 출력예제를 만족하지 못하는 프로그램을 삭제하는 단계가 가지치기(pruning)단계인 것이다.
- [0076] 즉, 남아 있는 홀이 어떠한 방식으로 교체되더라도 절대로 입출력 예제를 만족할 수 없는 프로그램을 정적 분석을 통해 확인하고 쳐내는 것이다.
- [0077] 정적 분석에서 요약 도메인 (abstract value domain) 은 아래와 같이 정의된다.
- [0078] $\hat{m} \in \hat{\mathbf{M}} = \mathbf{X} \rightarrow \hat{\mathbf{V}}, \hat{v} \in \hat{\mathbf{V}} = \mathbf{I} \times \mathbf{S}$
- [0079] $\hat{m}$ 은 프로그램 변수 ($\mathbf{X}$)에서 요약된 값 $\hat{\mathbf{V}}$ 으로의 함수이다. 요약된 값 (abstract value, $\hat{v}$) 는 구간 도메인 (lattice of intervals, $\mathbf{I}$) 과 심볼릭 도메인 (symbolic lattice, $\mathbf{S}$) 로 구성된다. 구체적으로, 구간 도메인 $\mathbf{I}$ 는 아래와 같이 정의된다:
- [0080]
$$\mathbf{I} = (\{\perp\} \cup \{[l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty, +\infty\} \wedge l \leq u\})$$
- [0081] 심볼릭 도메인 $\mathbf{S}$ 는 아래와 같이 정의된다:
- [0082] $\mathbf{S} = \mathbf{SB} \cup \{\perp, \top\}$ where $\mathbf{SB} \rightarrow n \mid \beta_x (x \in \mathbf{X}_i) \mid \mathbf{SB} \oplus \mathbf{SB}$
- [0083] 심볼릭 표현식 $\mathbf{SB}$ 는 정수이거나 (n), 심볼이거나 (β_x), 심볼릭 표현식들 간의 이항 산술 연산이다 ($\oplus$).
- [0084] $\mathbf{X}_i$ 는 정수 타입의 변수 집합을 의미하며, β_x 각 정수 타입의 변수 별로 심볼을 도입함을 의미한다.
- [0085] 실제의 값에서 요약된 값으로 변경하는 요약 함수 α 는 아래와 같이 정의된다:
- [0086]
$$\alpha(n) = ([n, n], n)$$
- [0087]
$$\alpha(n_1 \dots n_k) = ([\min\{n_1, \dots, n_k\}, \max\{n_1, \dots, n_k\}], \top)$$
- [0088] 위의 경우는 정수값에 대한 요약을 의미하며, 아래의 경우는 정수 배열에 대한 요약을 뜻한다. 요약 의미 함수의 도메인은 아래와 같다:
- [0090] $\hat{\mathbf{B}}$ 는 요약 논리 lattice (abstract boolean lattice, $\{\widehat{true}, \widehat{false}, \perp, \top\}$)를 뜻한다. 각 산술 표현식

(α), 논리 표현식 (b), 명령문 (C) 에 대한 요약의미 함수로서 아래 도 8에 도시되어 있다.

[0091] 도 8은 프로그램 실행 요약 의미를 표현한 도면이다.

[0092] 도 8에 기술되어 있는 요약 의미의 핵심은, 각 홀에 대해 ($\Diamond$, Δ , $\square$) 안전하게 의미를 정의함으로써, 부분 프로그램이 완성되었을 때의 모든 가능성을 포섭하도록 실제 실행 의미를 과대 추정 (over-approximation) 하는 것이다.

[0093] 각 배열 원소들은 하나의 원소로 요약된다. 따라서, 배열 원소 참조 표현식 ($x[u]$) 에 대한 요약 의미는 배열의 인덱스 u 를 관여하지 않는다. while-loop 반복문에 대해서는 고정점 연산 (fixed point computation, $\widetilde{fix}$)을 수행한다. 반복문 분석 시 일정 횟수 이상 이후에도 고정점에 도달하지 못할 경우, 프로그램 분석의 종료를 보장하고 속도를 빠르게 하기 위해, 구간 도메인과 심볼릭 도메인에 대해 각각 넓히기 연산 (widening)을 수행한다. 구간 도메인에 대해서는 "Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In POPL, 1977" 에 소개된 표준 넓히기 연산을 사용하며, 심볼릭 도메인에 대해서는 무조건적으로 $\top$ 의 값으로 변이시킨다.

[0094] 이제 정의한 요약 의미함수를 가지고, 아래의 절차에 따라 가지치기 ($prune(p)$) 하는 법을 설명한다. 설명의 용이성을 위해 프로그램의 단일 파라미터 x 를 가정하며, return 변수는 l 라 한다. 또, 임의의 입력 예제를 v_i , 짝지어지는 출력 예제를 v_o 라 하자. 부분 프로그램의 전체 명령문은 s 로 주어졌다고 가정하자.

[0095] 1. 각 입력 예제에 대해, 요약 의미 함수로 실행하고 분석 결과 itv_s, se_s 를 얻는다:

[0096]
$$(itv_s, se_s) = (\widehat{C}[s])([x \mapsto \alpha(v_i)])(y)$$

[0097] 2. 출력 예제에 대해 구간 요약의미를 얻는다:

[0098]
$$([l_o, u_o], -) = \alpha(v_o)$$

[0099] 3. 아래의 제약 조건 C_{v_i, v_o}^p 을 생성해낸다:

[0100]
$$C_{v_i, v_o}^p = (l_s \leq l_o \wedge u_o \leq u_s) \wedge (se \in SB \Rightarrow l_o \leq se \leq u_o)$$

[0101] 최종적으로, 가지치기($prune(p)$)는 아래와 같이 정의된다:

[0102]
$$prune(p) \Leftrightarrow C_{v_i, v_o}^p \text{를 만족하지 못하는 입출력 예제의 쌍 } (v_i, v_o) \text{가 존재한다.}$$

[0103] 분석 결과는 출력 예제를 반드시 과대 추정 (over-approximation) 한 것이어야 함을 의미한다. 가지치기 ($prune(p)$)의 충족 여부 (satisfiability) 는 SMT solver를 이용해 쉽게 확인할 수 있다.

[0104] 도 9는 본 발명의 실시예에 따른 프로그램 합성 방법을 이용하여 실제 실험한 결과표이다.

[0105] 도 9는 30개의 벤치마크 프로그램들에 대한 실험결과로서 각 벤치마크 프로그램은 입문자 프로그래밍 교육 수준에 적합한 프로그램으로 구성하였다. 각각 주어진 부분 프로그램들은 한 개의 논리 홀 표현(Δ)과 한 개 혹은 두개의 명령문 홀($\square$)을 지닌 채 실험을 진행하였다.

[0106] $\perp$ 은 제한시간 (1시간) 내에 합성하지 못했음을 의미한다.

- [0107] "IVars" 는 주어진 정수 타입 변수의 개수, "AVars" 는 주어진 배열 타입 변수의 개수, "Ints" 는 주어진 정수의 개수, "EXs" 는 입-출력 예제의 개수, "Time" 은 합성하는데 소요되는 시간을 초로 표기한 것이다.
- [0108] 모든 실험은 MacBook Pro (Intel Core i7/16GB RAM)에서 수행되었다.
- [0109] "Base" 는 코드 최적화 기술 없이 완전 탐색을 수행했을 때의 결과, "Base+Opt" 는 코드 최적화 기술을 더했을 때의 결과, "Ours" 는 본 발명의 실시예에 따른 프로그램 합성 방법인 정적 분석에 기반한 가지치기까지 더해졌을 때의 최종 결과를 의미한다. 정적 분석에 기반한 가지치기까지 ("Ours") 더해졌을 경우, "Base" 열의 평균 결과치보다 약 93.7 배의 성능 향상을 이룰 수 있었다.
- [0110] 본 발명은 도면에 도시된 실시 예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시 예가 가능하다는 점을 이해할 것이다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

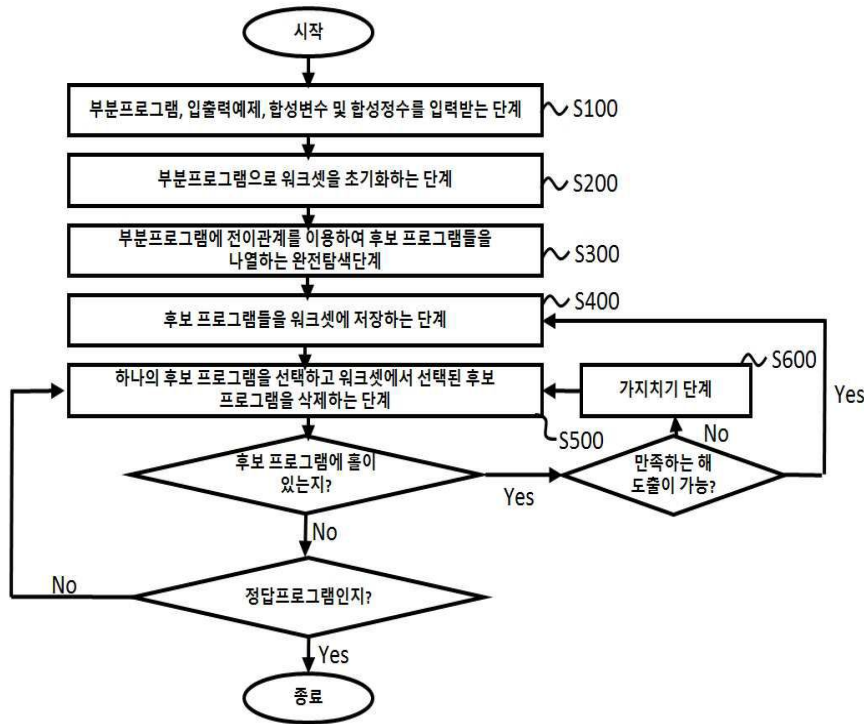
도면1

```
reverse (n) {
  r := 0;
  while (n > 0) {
    x := n % 10;
    r := r * 10;
    r := r + x;
    n := n / 10;
  };
  return r;
}
```

도면2

$$\begin{aligned} & \oplus \rightarrow + \mid - \mid * \mid / \mid \% , \quad \prec \rightarrow = \mid > \mid < \\ & l \rightarrow x \mid x[y] , \quad a \rightarrow n \mid l \mid l_1 \oplus l_2 \mid l \oplus n \mid \Diamond \\ & b \rightarrow true \mid false \mid l_1 \prec l_2 \mid l \prec n \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \mid \neg b \mid \Delta \\ & c \rightarrow l := a \mid skip \mid c_1 ; c_2 \mid if \ b \ c_1 \ c_2 \mid while \ b \ c \mid \square \end{aligned}$$

도면3



도면4

$a \rightarrow_a a'$	$c_1 \rightarrow c'_1$	$c_2 \rightarrow c'_2$	$b \rightarrow_b b'$	$c_1 \rightarrow c'_1$	$c_2 \rightarrow c'_2$	$b \rightarrow_b b'$
$l := a \rightarrow l := a' \quad c_1; c_2 \rightarrow c'_1; c'_2 \quad c_1; c_2 \rightarrow c_1; c'_2 \quad \text{if } b \ c_1 \ c_2 \rightarrow \text{if } b' \ c_1 \ c_2 \quad \text{if } b \ c_1 \ c_2 \rightarrow \text{if } b \ c'_1 \ c_2 \quad \text{if } b \ c_1 \ c_2 \rightarrow \text{if } b \ c_1 \ c'_2 \quad \neg b \rightarrow_b \neg b'$						
$b \rightarrow_b b'$	$c \rightarrow c'$	$b_1 \rightarrow_b b'_1$	$b_2 \rightarrow_b b'_2$	$b_1 \rightarrow_b b'_1$	$b_2 \rightarrow_b b'_2$	
$\text{while } b \ c \rightarrow \text{while } b' \ c \quad \text{while } b \ c \rightarrow \text{while } b \ c' \quad b_1 \wedge b_2 \rightarrow_b b'_1 \wedge b'_2 \quad b_1 \wedge b_2 \rightarrow_b b_1 \wedge b'_2 \quad b_1 \vee b_2 \rightarrow_b b'_1 \vee b'_2 \quad b_1 \vee b_2 \rightarrow_b b_1 \vee b'_2$						
$\Box \rightarrow l := \Diamond$	$\Box \rightarrow \text{skip}$	$\Box \rightarrow \Box; \Box$	$\Box \rightarrow \text{if } \Delta \Box \Box$	$\Box \rightarrow \text{while } \Delta \Box$	$\Delta \rightarrow_b \text{true}$	$\Delta \rightarrow_b \text{false}$
$\Delta \rightarrow_b l < n$	$\Delta \rightarrow_b \Delta \wedge \Delta$	$\Delta \rightarrow_b \Delta \vee \Delta$	$\Delta \rightarrow_b \neg \Delta$	$\Diamond \rightarrow_a n$	$\Diamond \rightarrow_a l$	$\Diamond \rightarrow_a l_1 \oplus l_2$
$\Diamond \rightarrow_a l \oplus n$						

도면5

$$\begin{aligned} & \{(x := \Diamond; r := 1; r := \Diamond), \\ & (skip; r := 1; r := \Diamond), \\ & (\Box; \Box; r := 1; r := \Diamond), \\ & (if \ \Delta \ \Box \ \Box; r := 1; r := \Diamond), \\ & (while \ \Delta \ \Box; r := 1; r := \Diamond), \\ & (\Box; r := 1; r := 1;), \\ & (\Box; r := 1; r := 2;), \\ & (\Box; r := 1; r := x), \\ & (\Box; r := 1; r := x + x), \\ & (\Box; r := 1; r := x - x), \\ & (\Box; r := 1; r := x * x), \\ & (\Box; r := 1; r := x / x), \\ & (\Box; r := 1; r := x \% x), \\ & (\Box; r := 1; r := x + 1), \\ & (\Box; r := 1; r := x + 2), \\ & (\Box; r := 1; r := x - 1), \\ & (\Box; r := 1; r := x - 2), \\ & (\Box; r := 1; r := x * 1), \\ & (\Box; r := 1; r := x * 2), \\ & (\Box; r := 1; r := x / 1), \\ & (\Box; r := 1; r := x / 2), \\ & (\Box; r := 1; r := x \% 1) \\ & (\Box; r := 1; r := x \% 2)\}. \end{aligned}$$

도면6

$\{ (x := \Diamond; r := 1; r := \Diamond),$
 $(skip; r := 1; r := \Diamond),$
 $(\Box; \Box; r := 1; r := \Diamond),$
 $(if \ \Delta \ \Box \ \Box; r := 1; r := \Diamond),$
 $(while \ \Delta \ \Box; r := 1; r := \Diamond),$
 $(\Box; r := 1;),$
 $(\Box; r := 2;),$
 $(\Box; r := x),$
 $(\Box; r := x + x),$
 $(\Box; r := 0),$
 $(\Box; r := x * x),$
 $(\Box; r := x + 1),$
 $(\Box; r := x + 2),$
 $(\Box; r := x - 1),$
 $(\Box; r := x - 2),$
 $(\Box; r := x * 2),$
 $(\Box; r := x/2),$
 $(\Box; r := x \% 2) \}.$

도면7

<pre> example1(n){ r := 0; while (n > 0){ r := n + 1; n := $\Diamond$; }; return r; } </pre>	<pre> example2(n) { r := 0; while (n > 0){ $\Box$; r := x * 10; n := n / 10; }; return r; } </pre>
유형 1	유형2

도면8

$\hat{\mathcal{A}}[n](\hat{m})$	$=$	$([n, n], n)$
$\hat{\mathcal{A}}[l](\hat{m})$	$=$	$\hat{m}(x) \ (l = x \text{ or } x[y])$
$\hat{\mathcal{A}}[l_1 \oplus l_2](\hat{m})$	$=$	$\hat{\mathcal{A}}[l_1](\hat{m}) \oplus \hat{\mathcal{A}}[l_2](\hat{m})$
$\hat{\mathcal{A}}[l \oplus n](\hat{m})$	$=$	$\hat{\mathcal{A}}[l](\hat{m}) \oplus n$
$\hat{\mathcal{A}}[\Diamond](\hat{m})$	$=$	$([-\infty, +\infty], \top)$
$\hat{\mathcal{B}}[true](\hat{m})$	$=$	$\widehat{true} \ (\hat{\mathcal{B}}[false](\hat{m}) = \widehat{false})$
$\hat{\mathcal{B}}[l_1 < l_2](\hat{m})$	$=$	$\hat{\mathcal{A}}[l_1](\hat{m}) \prec \hat{\mathcal{A}}[l_2](\hat{m})$
$\hat{\mathcal{B}}[l < n](\hat{m})$	$=$	$\hat{\mathcal{A}}[l](\hat{m}) \prec n$
$\hat{\mathcal{B}}[b_1 \wedge b_2](\hat{m})$	$=$	$\hat{\mathcal{B}}[b_1](\hat{m}) \wedge \hat{\mathcal{B}}[b_2](\hat{m})$
$\hat{\mathcal{B}}[b_1 \vee b_2](\hat{m})$	$=$	$\hat{\mathcal{B}}[b_1](\hat{m}) \vee \hat{\mathcal{B}}[b_2](\hat{m})$
$\hat{\mathcal{B}}[\neg b](\hat{m})$	$=$	$\neg \hat{\mathcal{B}}[b](\hat{m})$
$\hat{\mathcal{B}}[\Delta](\hat{m})$	$=$	$\top$
$\hat{\mathcal{C}}[x := \Diamond](\hat{m})$	$=$	$\hat{m}[x \mapsto ([-\infty, +\infty], \beta_x)]$
$\hat{\mathcal{C}}[x[y] := \Diamond](\hat{m})$	$=$	$\hat{m}[x \mapsto ([-\infty, +\infty], \top)]$
$\hat{\mathcal{C}}[x := a](\hat{m})$	$=$	$\hat{m}[x \mapsto \hat{\mathcal{A}}[a](\hat{m})]$
$\hat{\mathcal{C}}[x[y] := a](\hat{m})$	$=$	$\hat{m}[x \mapsto \hat{\mathcal{A}}[a](\hat{m}) \sqcup \hat{m}(x)]$
$\hat{\mathcal{C}}[skip](\hat{m})$	$=$	$\hat{m}$
$\hat{\mathcal{C}}[c_1; c_2](\hat{m})$	$=$	$(\hat{\mathcal{C}}[c_2] \circ \hat{\mathcal{C}}[c_1])(\hat{m})$
$\hat{\mathcal{C}}[if \ b \ c_1 \ c_2](\hat{m})$	$=$	$\widehat{cond}(\hat{\mathcal{B}}[b], \hat{\mathcal{C}}[c_1], \hat{\mathcal{C}}[c_2])(\hat{m})$
$\hat{\mathcal{C}}[while \ b \ c](\hat{m})$	$=$	$(\widehat{fix} \ \hat{F})(\hat{m})$
where $\hat{F}(g)$		$\widehat{cond}(\hat{\mathcal{B}}[b], g \circ \hat{\mathcal{C}}[c], \lambda x.x)$
$\hat{\mathcal{C}}[\Box](\hat{m})(x)$	$=$	$\begin{cases} ([-\infty, +\infty], \beta_x) & x \in \mathbb{X}_i \\ ([-\infty, +\infty], \top) & x \in \mathbb{X}_a \end{cases}$

도면9

Domain	No	Description	Vars		Ints	Exs	Time (sec)		
			IVars	AVars			Base	Base+Opt	Ours
Integer	1	Given n , return $n!$.	2	0	2	4	0.0	0.0	0.0
	2	Given n , return $n!!$ (i.e., double factorial).	3	0	3	4	0.0	0.0	0.0
	3	Given n , return $\sum_{i=1}^n i$.	3	0	2	4	0.1	0.0	0.0
	4	Given n , return $\sum_{i=1}^n i^2$.	4	0	2	3	122.4	18.1	0.3
	5	Given n , return $\prod_{i=1}^n i^2$.	4	0	2	3	102.9	13.6	0.2
	6	Given a and n , return a^n .	4	0	2	4	0.7	0.1	0.1
	7	Given n and m , return $\sum_{i=n}^m i$.	3	0	2	3	0.2	0.0	0.0
	8	Given n and m , return $\prod_{i=n}^m i$.	3	0	2	3	0.2	0.0	0.1
	9	Count the number of digit for an integer.	3	0	3	3	0.0	0.0	0.0
	10	Sum the digits of an integer.	3	0	3	4	5.2	2.2	1.3
	11	Calculate product of digits of an integer.	3	0	3	3	0.7	2.3	0.3
	12	Count the number of binary digit of an integer.	2	0	3	3	0.0	0.0	0.0
	13	Find the n th Fibonacci number.	3	0	3	4	98.7	13.9	2.6
	14	Given n , return $\sum_{i=1}^n (\sum_{m=1}^i m)$.	3	0	2	4	$\perp$	324.9	37.6
	15	Given n , return $\prod_{i=1}^n (\prod_{m=1}^i m)$.	3	0	2	4	$\perp$	316.6	86.9
	16	Reverse a given integer.	3	0	3	3	$\perp$	367.3	2.5
Array	17	Find the sum of all elements of an array.	3	1	2	2	8.1	3.6	0.9
	18	Find the product of all elements of an array.	3	1	2	2	7.6	3.9	0.9
	19	Sum two arrays of same length into one array.	3	2	2	2	44.6	29.9	0.2
	20	Multiply two arrays of same length into one array.	3	2	2	2	47.4	26.4	0.3
	21	Cube each element of an array.	3	1	1	2	1283.3	716.1	13.0
	22	Manipulate each element into 4th power.	3	1	1	2	1265.8	715.5	13.0
	23	Find a maximum element.	3	1	2	2	0.9	0.7	0.4
	24	Find a minimum element.	3	1	2	2	0.8	0.3	0.1
	25	Add 1 to each element.	2	1	1	3	0.3	0.0	0.0
	26	Find the sum of square of each element.	3	1	2	2	2700.0	186.2	11.5
	27	Find the multiplication of square of each element.	3	1	1	2	1709.8	1040.3	12.6
	28	Sum the products of matching elements of two arrays.	3	2	1	3	20.5	38.7	1.5
	29	Sum the absolute values of each element.	2	1	1	2	45.0	50.5	12.1
	30	Count the number of each element.	3	1	3	2	238.9	1094.1	0.2
Average							> 616.8	165.5	6.6