개별연구 보고서

수행 학기	2019학년도 1학기			
교과목 정보	교과목명	프로그램 자동 생성 연구	분반	-
교과담당교수	소속	컴퓨터공학과	교수명	손윤식
학생정보	이름	양승영	학번	2014112096
	학과	컴퓨터공학과	전화번호	010-5034-3548
	학년	4	이메일	yang354833@naver.com

요약 보고서

작품명 (프로젝트명)	프로그램 자동 생성 연구
1. 연구·개발 동기/목적/ 필요성 및 개발 목표	□ 프로그래밍 교육에 대한 관심과 수요 증가 □ 간단한 알고리즘 문제 풀이가 가능한 프로그램 생성 자동화
2. 최종 결과물 소개	□ 별 찍기 프로그램의 실행 결과 및 구조 분석 □ 별 찍기 프로그램 자동 생성 프로그램 구현
3. 프로젝트 추진 내용	□ 프로그램 자동 생성의 필요성 및 기법 조사 □ 별 찍기 프로그램의 실행 결과 패턴 분석 □ 별 찍기 프로그램의 구조적 분석
4. 기대효과	□ 간단한 기능의 프로그램을 자동 생성함으로써 개발 속도 향상 □ 프로그래밍 지식이 부족한 사용자들도 쉽게 원하는 프로그램 생성 가능

목 차

1. 서론

최근 국내에서는 프로그래밍에 대한 관심이 많아지고 있다. 이에 대한 대표적인 예시로 2019년부터 초등학교 5, 6학년을 대상으로 연간 17시간 소프트웨어 교육이 의무화 되었다. 따라서 앞으로 프로그래밍에 대한 수요는 더욱 증가될 것으로 예측된다. 전통적으로 프로그래밍 입문자들을 위한 간단한 연습 문제로는 별 찍기가 있다. 별 찍기 문제는 단순하지만 그 내부는 수학적인 알고리즘 요소로 구성되어 있기 때문에, 실력 향상을 원하는 대부분의 입문자들이 거쳐 가는 문제이다.

프로그램 자동 생성이란 명세 없이도 간단한 입력과 출력의 예시만으로 그에 맞는 프로그램을 자동으로 생성해주는 것을 말한다. 이러한 프로그램 자동 생성에 관한 연구는 오래전부터 진행되어 왔는데, 다양한 방법들에 대한 연구가 진행되고 있으며 최근에 와서는 머신러닝 기술을 이용한 방법이 활발히 연구되고 있다.

본 연구에서는 기존에 진행되었던 프로그램 자동 생성 연구들을 간략히 소개하고, 프로그래밍 지식이 부족하거나 관련 경험이 거의 없는 사용자들을 위한 간단한 별 찍기 프로그램을 자동으로 생성하기 위한 방법을 소개한다.

2. 관련/배경 연구

프로그램 자동 생성은 크게 두 가지 세부 분야로 나뉜다. 하나는 프로그램 합성(Synthesis)이고, 다른 하나는 프로그램 축소(Reduction)이다. 프로그램 합성은 제시된 입력과 출력을 바탕으로 해당 조건을 만족하는 새로운 프로그램을 생성하는 것을 말한다. 다음으로 프로그램 축소는 제시된 입력과 출력뿐만 아니라 원본 프로그램을 바탕으로 제시된 입력과 출력을 만족하는 최소한의 크기로 프로그램을 생성하는 것을 말한다. 이어지는 2.1절과 2.2절에서 각각의 세부 분야를 간략히 소개한다.

2.1 프로그램 합성

프로그램 합성 기술에는 연구자들 사이에 합의된 표준화 포맷인 Syntax-guided-Synthesis (SyGuS)가 존재한다. SyGuS 포맷은 모든 가능한 프로그램의 공간을 정의하는 구문적 제약조건(Syntactic specification)과 참/거짓을 판단할 수 있도록 논리식의 형태로 제공되는 의미적 제약조건(Semantic specification)으로 구성된다.

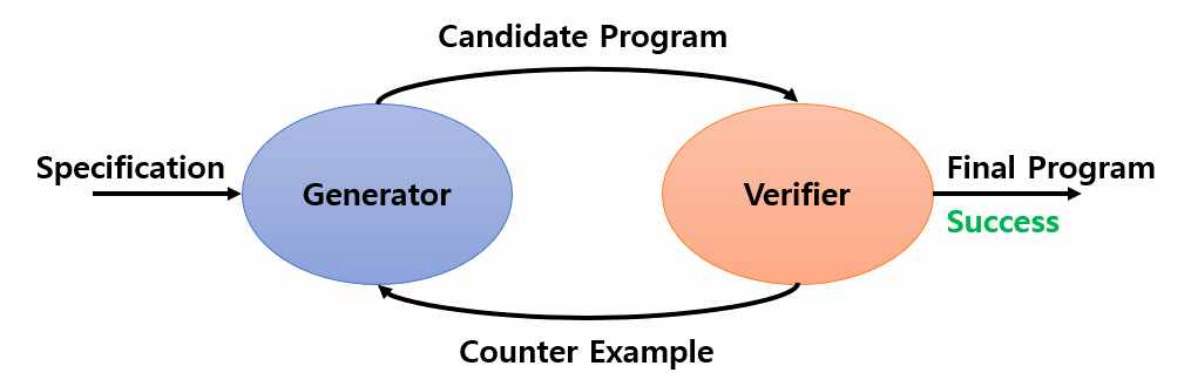
아래 <표 1>은 SyGuS 포맷의 예시이다. 변수 x와 y 중 더 큰 값을 찾는 함수인 max2를 생성하기 위한 명세들이 입력으로 주어진다. 먼저 나오는 명세는 프로그램을 생성할 수 있는 구문적 제약조건이고, 다음으로 나오는 명세는 의미적 제약조건이다. 해당하는 구문적 제약조건

을 이용해 프로그램을 생성해나가고, 프로그램에서 non-terminal symbol인 I와 B가 모두 제거되면 프로그램 생성이 종료된다. 그 후 생성된 프로그램이 의미적 제약조건을 만족한다면 올바른 프로그램을 생성한 것이다.

```
;; Name and signature of the function to be synthesized
(synth-fun max2 ((x Int) (y Int)) Int

    ;; Declare the non-terminals that would be used in the grammar
    ((I Int) (B Bool))

    ;; Define the grammar for allowed implementations of max2
    ((I Int (x y 0 1
             (+ I I) (- I I)
             (ite B I I)))
     (B Bool ((and B B) (or B B) (not B)
              (= I I) (<= I I) (>= I I))))
)

(declare-var x Int)
(declare-var y Int)

;; Define the semantic constraints on the function
(constraint (>= (max2 x y) x))
(constraint (>= (max2 x y) y))
(constraint (or (= x (max2 x y)) (= y (max2 x y))))

(check-synth)
```

<표 1> SyGuS 포맷 예시

이러한 프로그램 합성 과정은 가장 널리 쓰이는 합성 알고리즘인 반례기반 프로그램 합성(Counter Example Guided Inductive Synthesis; CEGIS) 알고리즘을 이용하여 진행된다. 이 알고리즘은 아래 <그림 1>에서 볼 수 있듯이, 후보 해(Solution candidate)를 찾는 탐색 부분(Generator)과 찾아낸 후보 해가 주어진 요구조건을 만족하는 올바른 프로그램인지 확인하는 검증 부분(Verifier)으로 구성된다. Generator는 Verifier로부터 제공되는 피드백을 활용하여 주어진 조건을 만족하는 Solution candidate를 찾아내고, Verifier는 찾아진 Solution candidate가 요구조건을 만족하는지 검증하여 만족하지 않는다면 반례를 생성해 Generator에게 피드백 한다. 이 과정은 Generator가 모든 요구조건을 만족하는 프로그램을 찾아내거나, 요구조건을 만족하는 프로그램이 주어진 탐색 공간에 없다는 것이 확인될 때까지 반복된다.



<그림 1> CEGIS 알고리즘

CEGIS 알고리즘의 성능은 Generator에 적용되는 알고리즘에 따라 성능이 결정되는데, 따라서 Generator에 적용할 알고리즘에 대한 연구가 진행되고 있다.

2.2 프로그램 축소

프로그램 축소는 현대 소프트웨어의 규모 및 복잡도의 기하급수적인 증가에 따른 성능 저하, 유지 관리 난이도 증가, 보안 위협을 해결하기 위해 등장하였다. 물론 성능 및 보안 상 이점을 목표로 개발자들이 특정 사용 목적에 한정시켜 수동으로 프로그램 크기를 줄이는 시도는 있었다. BusyBox와 같이 기존의 프로그램에 비해 크기가 월등히 작고, 알려진 보안 취약점도 없는 성공적인 프로그램 축소 사례도 있지만, 그 과정이 수동으로 진행되기에 적용할 수 있는 규모의 한계가 존재한다.

이 절에서는 자동 프로그램 축소를 위한 소프트웨어 중 하나인 Chisel을 소개한다. Chisel은 원본 C 프로그램과 임의의 프로그램을 받아 그 프로그램이 바람직한 성질을 만족시키는지에 따라 참/거짓을 반환하는 성질 테스트 함수를 입력으로 받는다. 성질 테스트 함수는 각 Test script들을 이용해 축소된 프로그램이 요구 기능을 정상적으로 수행하는지 검사한다. 이 소프트웨어는 프로그램의 동작을 보존하려 하는 기존의 프로그램 축소 도구들과는 달리, 불필요한 기능을 제거함으로써 프로그램의 행동을 변화시킨다. Chisel은 프로그램의 원하는 행동과 어떤 부분에서 정확성이 제공되어야 하는지를 지정하여 프로그램을 효율적으로 축소하고 있다.

아래의 <표 2>는 Chisel에 사용되는 성질 테스트 함수의 예시인데, 성질 테스트 함수는 크게 3가지 항목을 검사하며 검사 결과가 모두 참일 때, 축소된 프로그램이 최종적으로 참임을 반환한다. 검사하는 각각의 항목은 정상적인 컴파일의 여부, 원래 버전과 같은 입력에 동일한 출력의 여부, 원래 버전에서 지원하지는 않는 입력이 들어왔을 때 새로운 오류의 유발 여부이다. 두 번째와 세 번째 항목의 입력에 대한 검사는 사용자가 직접 Test script 작성함으로써 간단하게 추가할 수 있고, 프로그램을 실행시키면 최종적으로 요구된 기능들을 만족하는 경량화 된 프로그램이 자동으로 생성된다.

```
#!/bin/bash

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

# tests for the desired functionalities
function core {
  # 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 non_core {
  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
```

<표 2> Chisel의 성질 테스트 함수 예시

3. 본론

본 연구에서는 Baekjoon Online Judge(https://www.acmicpc.net/)에서 지원하는 별 찍기 문제를 대상으로 한다. 대상으로 하는 별 찍기 문제의 수는 총 12개이며, 하나의 프로그램 자동 생성 프로그램을 이용해 각 12개 문제에 대한 솔루션 프로그램의 생성을 목표로 한다.

3.1 별 찍기 패턴 분석

<표 3>은 각각의 별 찍기 프로그램의 입력과 출력 예시이다. 별 찍기 이름에 붙어있는 번호는 Baekjoon Online Judge의 문제 이름을 참고하여 작성하였다. 각 프로그램의 입력 예시는 첫 번째 줄의 3으로 동일하고, 이 입력으로 인해 출력되는 별 찍기 패턴은 모두 다르다.

3
*
**
***
	3
	  *
	  **
	  ***
		3
		***
		**
		*
			3
			***
			**
			*
				3
				  *
				  ***
				*****
					3
					*****
					  ***
					  *

별 찍기 1	별 찍기 2	별 찍기 3	별 찍기 4	별 찍기 5	별 찍기 6

3
  *
  ***
*****
  ***
  *
	3
	* *
	** **
	******
	** **
	* *
		3
		*****
		  ***
		  *
		  ***
		*****
			3
			  *
			  **
			  ***
			  **
			  *
				3
				*
				**
				***
				**
				*
					3
					  ***
					  ***
					  ***

별 찍기 7	별 찍기 8	별 찍기 9	별 찍기 12	별 찍기 13	별 찍기 14

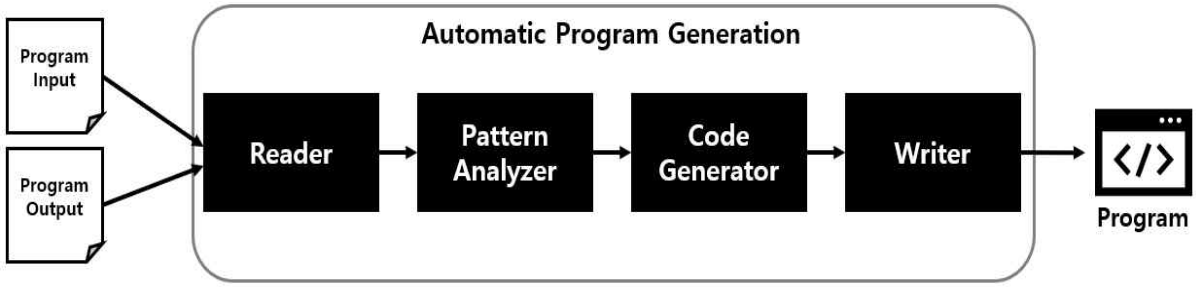
<표 3> 별 찍기 프로그램의 입력과 출력 예시

각각의 별 찍기 패턴은 크게 5가지로 분류할 수 있다. 먼저 출력되는 줄의 수를 i, 각 줄에 출력되는 별의 개수의 최댓값을 j라고 한다. 프로그램의 입력으로 들어오는 값을 n이라고 할 때, (i=n, j=n) 이면 패턴 1, (i=n, j=2*n-1) 이면 패턴 2, (i=2*n-1, j=n) 이면 패턴 3, (i=2*n-1, j=2*n-1) 이면 패턴 4, (i=2*n-1, j=2*n) 이면 패턴 5이다. 최종적으로 각 별 찍기 패턴은 다음과 같이 분류된다.

- 패턴 1: 별 찍기 1, 별 찍기 2, 별 찍기 3, 별 찍기 4, 별 찍기 14
- 패턴 2: 별 찍기 5, 별 찍기 6
- 패턴 3: 별 찍기 12, 별 찍기 13
- 패턴 4: 별 찍기 7, 별 찍기 9
- 패턴 5: 별 찍기 8

3.2 구현

별 찍기 패턴들의 분석 결과를 바탕으로, 별 찍기 프로그램 자동 생성 프로그램을 구현한다. 자동 생성 프로그램의 입력은 3.1절에서 예시로 보인 것과 같은 형식의 텍스트 파일로 주어지며, 최종적으로 생성되어 출력되는 결과물은 C++ 형식의 프로그램이다. <그림 2>는 구현한 프로그램의 전체적인 구성 및 흐름을 나타낸다.



<그림 2> 별 찍기 프로그램 자동 생성 프로그램의 구성 및 흐름

자동 생성 프로그램의 핵심인 모듈인 패턴 분석기(Pattern Analyzer)와 코드 생성기(Code Generator)는 모든 별 찍기 패턴에 대한 입력을 정확하고 효율적으로 분석하여 코드를 생성할 수 있어야 한다. 이를 위해 먼저 패턴 분석기는 출력되는 줄의 수 i와 각 줄에 출력되는 별의 개수 최댓값 j를 파악하여 1에서 5 사이의 패턴 중 하나로 분류한다. 그 후 별 찍기 패턴의 모양을 직사각형으로 봤을 때, 해당하는 각 4개의 꼭짓점에 '*'이 찍히는지 공백이 찍히는지를 확인하여 세부 별 찍기의 번호를 판단하고, 정해진 별 찍기 번호의 모양과 입력된 별 찍기 모양이 일치하는지 확인한다. 일치한다면 코드 생성기에 별 찍기의 번호를 전달하고 일치하지 않는다면 잘못된 입력 파일임을 출력한다.

코드 생성기는 전달받은 별 찍기의 번호에 해당하는 솔루션 프로그램을 생성한다. 각각의 별 찍기 모양은 다양하지만, 이를 출력하는 프로그램의 형식은 서로 비슷하다. 따라서 이 사실을 바탕으로 하나의 자동 생성 프로그램으로 다양한 별 찍기 문제의 솔루션 프로그램들을 생성할 수 있다.

기본적으로 별 찍기 프로그램의 소스코드에서 사용되는 Statement로는 for 문과 if-else 문이 있다. 패턴 1과 2에는 2중 for 문과 if-else 문의 조합 하나를 사용해 솔루션 프로그램을 만들 수 있고, 나머지 패턴 3, 4, 5는 2중 for 문과 if-else 문의 조합이 2개 사용된다. 이 조합으로 소스코드의 큰 틀을 만들고, 각 Statement에 들어가는 조건들을 변화시킴으로써 각각의 조건을 만족하는 서로 다른 별 찍기 프로그램이 생성된다. 이 조건에 사용되는 변수로는 'start'와 'finish'이고, 패턴 분석기의 패턴 분석을 통해 전달 받은 별 찍기의 번호를 바탕으로 코드 생성기에서 값을 선언해준다.

아래의 <표 4>는 자동 생성 프로그램을 사용해 만들어진 별 찍기 문제의 솔루션 프로그램 소스코드 예시이다.

```
#include<iostream>
using namespace std;

int main() {
  int n;
  cin >> n;
  int start = 0, finish = 0, s = 0, f = 1;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      if (j >= start && j <= finish)
        cout << '*';
    }
    cout << '\n';
    start += s;
    finish += f;
  }

  return 0;
}
```

별 찍기 1의 솔루션 프로그램 소스코드

```
#include<iostream>
using namespace std;

int main() {
  int n;
  cin >> n;
  int start1 = 0, finish1 = 0, f1 = 1;
  int start2 = 2*n-1, finish2 = 2*n-1, s2 = -1;
  for (int i = 0; i < 2 * n - 1; i++) {
    for (int j = 0; j < 2 * n; j++) {
      if ((j >= start1 && j <= finish1)
        || (j >= start2 && j <= finish2))
        cout << '*';
      else
        cout << ' ';
    }
    cout << '\n';
    if (i + 1 < input) {
      finish1 += f1;
      start2 += s2;
    }
    else {
      finish1 -= f1;
      start2 -= s2;
    }
  }

  return 0;
}
```

별 찍기 8의 솔루션 프로그램 소스코드

<표 4> 별 찍기 문제의 솔루션 프로그램 소스코드 예시

3.3 평가

자동 생성 프로그램의 평가는 생성된 각각의 별 찍기 문제의 솔루션 프로그램을 Baekjoon Online Judge의 실제 각 문제의 솔루션 코드로 제출해봄으로써 채점을 통해 이루어진다.

아래의 <그림 3>은 자동 생성된 각각의 별 찍기 문제의 솔루션 프로그램 소스코드를 Baekjoon Online Judge의 채점 프로그램을 통해 채점한 결과이다.

<그림 3> 자동 생성된 별 찍기 문제의 솔루션 프로그램 소스코드 채점 결과

총 12개의 별 찍기 문제 중 8개는 성공, 4개는 실패했는데, 실패한 문제들은 앞서 분류했던 패턴 2, 4에 속하는 문제들임을 알 수 있다. 실패의 원인을 분석해본 결과, Baekjoon Online Judge의 채점 프로그램에서 원하는 정답 코드는 별 찍기의 각 줄에서 마지막 문자들이 공백으로 끝나는 경우에는 이 공백들을 출력하지 않고 바로 개행 문자가 오길 원하는데, 자동 생성된 솔루션 프로그램 소스코드들은 공백까지 모두 출력해버리기 때문에 차이가 나타났다.

하지만 Baekjoon Online Judge의 채점 프로그램을 통과하지 못했다고 하더라도, 전체적으로 생성된 솔루션 프로그램들이 기대한대로 잘 동작함을 확인할 수 있었다.

4. 결론 및 향후 연구

프로그램 자동 생성의 두 세부 분야인 프로그램 합성과 프로그램 축소에 대한 연구들을 간단히 살펴본 후, 별 찍기 프로그램의 입력과 출력을 분석해 올바른 별 찍기 문제의 솔루션 프로그램 자동 생성 기법을 소개하였다. 실제로 이 프로그램을 사용해 생성된 솔루션 프로그램을 Baekjoon Online Judge의 채점 프로그램으로 평가하고, 그 결과를 분석해보았다. 구현한 프로그램을 사용하면 어느 정도 사용자가 원하는 프로그램을 생성할 수 있었으며, 사용자가 프로그래밍의 경험이 얼마 없는 초보자이더라도 손쉽게 사용할 수 있기 때문에 프로그래밍 공부에도 도움이 될 수 있을 것으로 기대된다.

참고 문헌

[1] https://sygus.org/
[2] 이우석, “확률 모델을 이용한 프로그램 자동 생성 가속화”, 한국정보과학회지 제37권 제3호(통권 제358호), pp.40-47, 2019.
[3] https://chisel.cis.upenn.edu/
[4] https://www.acmicpc.net/