GP 를 이용한 Assembly 코드 자동 생성 시스템 설계

김경임*, 원일용*
*서울호서전문학교 사이버해킹보안
quffhrhrh@gmail.com, clccclcc@shoseo.ac.kr

A System for Assembly Code Auto Generation using GP

Kyung-im Kim*, Ill-young Weon *
*Dept. of Cyber security, Hoseo Technical College

요 약

유전 프로그래밍 기법을 이용하여 완성된 프로그램 코드를 생성하는 연구는 최근까지 실용적으로 사용할 만한 결과를 만들지 못하고 있다. 우리는 완성된 프로그램 코드를 만들지 않고, 특정 기능을 갖는 함수 코드를 자동 생성하는 연구에 중점을 두었다. 실제 구현을 위한 시스템을 설계하고 타당성을 검토 하였다.

1. 서론

유전 프로그래밍은 코드를 자동으로 생성할 수 있는 분야의 연구에서 많이 사용되고 있다[1,2]. 유전 프로그래밍의 원리는 가능한 모든 해의 집합공간에서 자신이 원하는 최적의 해를 찾아가는 최적화 기법을 쓰고 있다. 이것은 복잡한 알고리즘이 있어야 하는 기존의 인공지능 방법들과 비교해 매우 단순한 접근 방법이다.

유전 프로그래밍을 이용하여 완성된 실용적인 코드를 생성하는 연구는 최근까지 좋은 결과를 내지 못하고 있다. 그 주된 이유는 코드를 생성하기 위해서, 언어에 대한 많은 조건을 광범위하게 적용해야 하는 어려움이 있기 때문으로 예측된다.

본 연구는 유전자 프로그래밍 기법을 이용하여 어셈블리 코드를 생성하는 방법에 대한 것이다. 우리는 완성된 프로그램 코드를 생성하지 않고, 특정 목적을 가지는 함수를 설계하고 이 제한된 영역에서 코드를 자동 생성시키는 시스템을 제안하고자 한다. 이렇게 접근 하는 이유는 전체 프로그램을 만드는 복잡도를 감소 시켜 좀 더 실용적인 예제로 접근하고자 하기 때문이다.

우리가 자동 생성하고자 하는 프로그램의 언어는 어셈블리 언어이다. 어셈블리어는 다른 언어에 비해 구조적으로 더 단순하여 의미적 제약조건을 만족하는 진화적 알고리즘을 설계하는 것이 상대적으로 쉽기 때문이다[3].

본 연구는 다음과 같이 구성되어 있다. 2 장에서는 관련 연구를, 3 장에서는 제안하는 시스템의 구조를 상세하게 설명하였다. 4 장 에서는 결론 및 향후 과제를 언급 하였다.

2. 관련 연구

2.1 유전 프로그래밍

유전 프로그래밍(Genetic Programming, GP)은 진화 알고리즘인 유전 알고리즘(Genetic Algorithm, GA)의 확장으로 두 프로그래밍은 많은 유사성을 지니고 있다. 두 프로그래밍의 가장 큰 차이점은 개체를 표현하는 방법이다. GP 는 트리 구조를 이용하여 컴퓨터 프로그램 형태로 표현하고, GA 는 고정된 길이의 비트 문자열 형태로 표현한다. 컴퓨터 프로그램의 표현은 함수 집합(Function set)과 터미널 집합(Terminal set)으로 구성된다. 여기서 함수 집합은 다양한 수학 연산자, 반복 연산자, 조건 연산자 등으로 구성되며, 터미널 집합은 숫자 혹은 변수, Boolean 상수로 구성된다[4,5].

GP 에서 가장 중요한 개념은 적합도 함수이다. 적합도 함수는 문제에 따라 달라지며, 프로그램이 문제를 잘 해결 할 수 있는지를 평가한다.

그리고 GP 는 자연 선택 방법으로 컴퓨터 프로그램을 진화시킨다. 자연 선택 방법에는 선택(Selection), 교차(Crossover), 돌연변이(Mutation)가 있다.

선택은 자연 선택 현상을 모델링 한 것으로 환경에 잘 적응한 해는 남고, 적응하지 못한 해들은 도태되는 과정이다. 교차는 자연계에서 생식에 의하여 새로운 개체가 생성되는 개념을 도입한 것이다. 해 집단에 존재하는 개체들이 가지고 있는 정보를 교차를 통해 서로 교환한다. 돌연변이는 개체들이 진화함으로써 잃어버린 유전 형질을 되돌리는 과정이다. 복제와 교차를 반복하며 개체는 적합도가 높은 유전 형질만 남게 된다. 잃어버린 유전 형질 중 fitness 를 도달 할

수 있는 인자가 있을 수 있기에 변이가 필요하다.

2.2 어셈블리 언어

어셈블리 언어는 기계어와 일대일 대응인 언어이다. 기계어는 CPU 종류에 따라 달라지며 어셈블리어도 통일된 규격은 없다. 어셈블리 언어는 CPU 가 지원하는 각종 연산 기능들을 이용하여 레지스터나 메모리에 데이터를 저장하고, 프로그램의 흐름을 변경하는 단순한 작업을 한다. 레지스터는 범용 레지스터, 세그먼트 레지스터, 포인트 레지스터, 인덱스 레지스터, 플래그 레지스터가 있다.

본 연구에서 기계어인 어셈블리어를 사용하는 이유는 다음과 같다. 고급 언어는 문법을 사용하여 정의되기 때문이다. 프로그램 전체적으로 자동 생성을 하면 구문적 제약 조건은 성립하지만, 의미적 제약조건을 성립하지 못할 가능성이 크다. 하지만 기계어인 어셈블리 언어를 이용해 프로그램을 자동 생성하면 의미적 제약 조건이 성립할 가능성이 높다[3].

3. 어셈블리 프로그램 자동 생성 시스템

3.1 전체 작업 구조

제안한 시스템의 전체적인 알고리즘은 <표 1> 과 같다. 시스템은 먼저 해당 함수의 후보 코드를 여러 개 생성한다. 그리고 이렇게 생성된 함수 후보 코드가 목적을 달성하는 기능을 보이면 종료하고, 그렇지 않으면 일정 비율로 각각의 개별 코드를 선택하여 crossover, mutation 등의 진화 연산을 수행한다. 이후 다시 종료 조건을 확인하며 위의 과정을 반복하게 된다.

<표 1> 전체 흐름 의사코드

```
create template file for generating function code
create fitness check script
generate first function code population

repeat
    for all individual code
        if individual code is satisfied code constraint
            make binary file using template code and individual code
            check fitness value for binary file using fitness check script
            if it is a terminal condition then stop procedure
            regenerate function code population using genetic operators
```

3.2 소스코드 생성과 제약 조건

GP 를 이용해 생성된 어셈블리 코드는 구문적 제약 조건과 의미적 제약 조건을 고려해야 한다. 제안된 시스템은 해당 어셈블리 코드가 만들어지면 구문적 제약 조건을 체크해서 위배되는 후보 코드는 모두 버려야 한다. 아래의 예는 어셈블리언어에서 우리가 사용하는 구문적 제약 조건의 예이다.

<표 2> movl 명령어 제약조건 예시

movl 레지스터, 레지스터
movl 메모리, 레지스터
movl 레지스터, 메모리
movl 메모리, 상수 값
movl 레지스터, 상수 값

의미적 제약 조건을 사용하기에는 어려움이 있다. 제안한 시스템에서 우리가 사용하는 의미적 제약 조건은 만들어진 코드를 실행해서 system crash 가 발생하면 위배, 발생하지 않으면 위배되지않는다.

어셈블리 코드를 GP 에서 생성하기 위해서는 해를 표현하기 위해 사용되는 인자들과 돌연변이 발생률, 교차율과 같은 요소들을 결정해야 한다. 우리는 아래와 같은 해 표현 조건을 사용한다.

<표 3> 해 표현의 조건

함수 집합	JAE, cmp, jmp, mov, call, ret, add, sub, and, or, xor
터미널 집합	ax, bx, cx, LB, RB, result_rax, result_rbx, result_rcx, end_game
해 트리의 최대 깊이	N
교차율	A %
돌연변이율	B %

이렇게 만들어진 코드의 예는 아래 <표 4>와 같다. 아래 모듈은 3 개의 숫자 중 가장 큰 숫자를 리턴하는 함수 이다.

<표 4> 생성된 해의 표현형 예

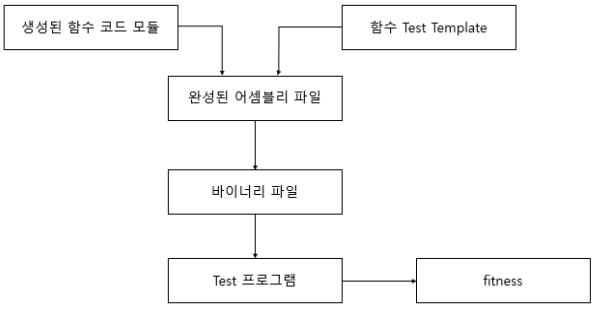
```
push ebp
mov ebp,esp
sub esp,28

mov eax,dword ptr ss:[ebp+8]
mov dword ptr ss:[ebp-C],eax
mov eax,dword ptr ss:[ebp-C]
cmp eax,dword ptr ss:[ebp+C]
jge project2.4016BD
mov eax,dword ptr ss:[ebp+C]
mov dword ptr ss:[ebp-C],eax
mov eax,dword ptr ss:[ebp-C]
cmp eax,dword ptr ss:[ebp+10]
jge project2.4016CB
mov eax,dword ptr ss:[ebp+10]
mov dword ptr ss:[ebp-C],eax
mov eax,dword ptr ss:[ebp-C]

leave
ret
```

3.3 fitness 계산

생성된 코드의 fitness 를 계산하기 위해서 시스템은 생성된 코드를 포함할 수 있는 템플릿이 필요하다. 템플릿은 생성된 코드의 기능을 사용하는 어셈블리 코드를 포함하고 있으며, 실행 파일 테스트 모듈과 연동할 입출력 양식을 가지고 있다.



(그림 1) fitness 측정 흐름도

이렇게 완성된 어셈블리 소스 파일은 어셈블러를 통해 실행파일로 만들어지게 된다. 실행파일 테스터 모듈은 완성된 바이너리 파일을 실행하면서 입출력을 통해 fitness 를 측정하게 된다.

예를 들어 생성하고자 하는 모듈이 3 개의 숫자를 입력 받고 가장 큰 숫자를 리턴하는 기능이라고 가정한다. 템플릿은 외부로부터 3 개의 숫자를 입력 받고 생성된 모듈을 호출하여 리턴 값을 받는다. 그 값이 테스터 모듈에 전달하는 코드가 된다. 테스터 프로그램은 6 가지의 경우의 수를 만들어 바이너리를 실행하게 된다. 리턴 값을 보고 fitness 를 계산하는 스크립트를 가지고 있다.

<표 5> fitness 계산한 결과의 예

Case1: Input:1, 2, 3	Return: 1
Case2: Input:1, 3, 2	Return: 3 (ok)
Case3: Input:2, 1, 3	Return: 3 (ok)
Case4: Input:2, 3, 1	Return: 2
Case5: Input:3, 1, 2	Return: 1
Case6: Input:3, 2, 1	Return: 3 (ok)

Fitness = Number of Correct output = 3 / 6 = 0.5

만들고자 하는 서브 모듈의 목적에 따라 템플릿 파일 및 테스트 프로그램은 설계가 달라진다. 따라서 fitness 를 계산하는 방식도 달라져야 한다.

4. 결론 및 향후 과제

유전 프로그래밍 기법을 이용하여 프로그램을 생성하는 일은 다른 인공지능 알고리즘보다 계산 없이 최적화 문제로 다룰 수 있다는 장점이 있다. 그러나 한 번에 완성된 프로그램 코드를 생성하는 일은 복잡하고 어려운 일이다. 본 논문은 완성된 코드의 일부분 함수를 유전 프로그래밍 기법으로 코드를 생성하는 시스템에 대해 구체적 설계를 제시하였다.

제시된 시스템의 실현가능성을 위해 이후 요소 하나 하나를 구현하고 검증하는 과정이 필요하다. 또 이미 만들어져 있는 함수를 유전형으로 변경한 후 유사 기능의 다른 코드로 구현하는 연구도 필요하다.

참고문헌

[1] 김영균, 서기성 “ Genetic Programming 을 이용한 코너 검출자의 자동생성” 한국지능시스템학회 vol.19 No.4, pp.580-585, 2009.

[2] Kisung Seo, soohwan Hyun “ Automatic Gait Generation for Quadruped Robot Using a GP Based Evloutionary Method in Joint Space” Journal of Institute of Control vol.14 No.6, pp.573-579, 2008.

[3] Eric Collom “Applying Genetic Programming to Bytecode and Assembly” Scholarly Horizons: University of Minnesota Morris Undergraduate Journal, vol.1 No.2, 2014

[4] 서원준, 박철수 “유전프로그래밍과 BEMS 데이터를 이용한 실내온도 기계학습 모델” 대한건축학회논문집 계획계 vol.32 No.06, pp105-112, 2016.

[5] 최치원, 민경식, 이병정 “자동결함정정에 스택오버플로우 유사코드를 활용한 유전 프로그래밍 적용 기법” 한국정보과학회 학술발표논문집 vol.46 No.01, pp.1734-1736, 2019.