Journal of The Korea Society of Computer and Information
Vol. 21 No. 11, pp. 159-166, November 2016

www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2016.21.11.159

Development of a Code Generation Support System in Integrated Development Environment of an Educational Compiler

Jung-Hoon Kwon*, Jong-Min Bae**

Abstract

Compiler course is one of the important courses in computer science. It requires more efficient learning environment because of its large coverage scale and complexity. One of its solutions is to provide the integrated development environment for educational compilers which is enable to give practice-oriented class and enhance student's interest. This paper presents the code generation support system developed in an integrated development environment of educational compiler. Our system helps students to understand the process of code generation and visualizes the relation among the source language, AST, and the target language. It makes students develop their own compilers more easily.

▸ Keywords : Educational compilers, Code generation, Code generation support system, LEGO Mindstorms NXT

I. Introduction

컴파일러 교과목은 오랜 기간 동안 컴퓨터과학의 주요한 교과목으로, 많은 알고리즘과 컴퓨터 시스템을 이해하는데 도움을 주는 컴퓨터과학의 주요 교과목이다[1]. 특히 컴파일러 교과목의 실습을 통해 컴파일러를 개발하는 것은 학생에게 프로그래밍 언어를 이해하고 충분한 사고력을 키우는데 많은 도움이 된다. 뿐만 아니라 컴퓨터과학 분야에서 다루는 다양한 지식이 접목된 컴파일러 개발을 통하여 학생들은 전반적인 컴퓨터과학 분야에 대한 이해를 높일 수 있다.

그러나 컴파일러 교과목은 학생들에게는 컴퓨터과학 교과목 중 어려운 과목중 하나로 인식되고 있다[1]. 그리고 컴파일러 프로젝트를 진행하는 학생뿐만 아니라 컴파일러 교과목을 교육하는 교수자 역시 학생들에게 컴파일러의 주요한 교육적 요소를 이해시키는 것은 많은 시간과 노력을 요구한다. 특히 컴파일러의 교과목은 컴파일러 프로젝트를 통하여 학생들이 직접 컴파일러를 개발하면서 자신만의 언어를 설계하고 알고리즘을 고안해내는 것은 훌륭한 교육적 효과를 줄 수 있지만 짧은 기간 동안 높은 복잡도의 컴파일러 프로젝트를 진행하는 것은 학생들에게 현실적으로 부담이 될 수밖에 없다.

이러한 높은 복잡도로 인한 학생의 부담을 덜어주고, 컴파일러 교과목의 학습 성취도를 향상시키기 위한 많은 연구가 있는데, 그중에서 교육용 컴파일러 통합 개발환경에 대한 연구가 있다. 예를 들어, Bantam 컴파일러는 Java언어를 기반으로 한 Bantam Java 언어를 정의하여 언어에 집중하기 보다는, 학생들에게 컴파일러의 구성과 구현에 집중할 수 있도록 개발환경을 제공한다[2]. 그리고 Chirp on Crickets는 C 언어 기반의

• First Author: Jung-Hoon Kwon, Corresponding Author: Jong-Min Bae
*Jung-Hoon Kwon (boogierock27@gmail.com), Aero Master Corporation
**Jong-Min Bae (jmbae@gnu.ac.kr), Dept. of Computer Science, Gyeongsang National University
• Received: 2016. 08. 13, Revised: 2016. 09. 09, Accepted: 2016. 10. 27.

Chirp 언어를 통하여 Cricket 컨트롤러를 구동하는 교육용 컴파일러 개발환경을 제공하고, 학생들에게 컴파일러에 대한 이해를 돕기 위한 도구와 Cricket 컨트롤러가 없어도 실행 가능한 시뮬레이터를 제공하여 개발환경에 대한 접근성을 높였다[3]. Edu-IDEC은 이클립스 프레임워크 플러그인 기반으로 한 교육용 컴파일러 개발환경으로서, LEGO MindStorms NXT 로봇을 구동하는 프로그램에 대한 컴파일러를 제작할 때, 학생들의 부담을 줄이면서 학생들의 흥미를 끌 수 있도록 개발환경을 제공한다[6]. 이러한 도구들을 활용한 수업은 효과적인 것으로 평가되었다.

본 연구에서는 학생들에게 코드생성 과정에 대한 이해를 도와서, 컴파일러의 코드생성기를 쉽게 개발할 수 있도록, 코드생성지원시스템을 개발한 결과를 제시한다. 개발된 코드생성지원시스템은 원시언어, 구문트리, 목적언어 사이의 관계에 대한 시각화 기능을 제공한다. 이 시스템은 이클립스 플러그인 프레임워크 기반의 교육용 컴파일러 통합 개발환경인 Edu-IDEC[6]에 플러그인 되어서 수행된다. 이는 컴파일러 프로젝트 실습환경에 그대로 활용되어서 학생들의 컴파일러 교과목의 학습 성취도 향상에 기여할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 코드생성지원시스템의 관련 연구에 대하여 알아보고, 3장에서는 본 코드생성지원시스템의 구현환경을 설명한다. 그리고 4장에서는 코드생성지원시스템의 설계와 구현에 대하여 논한다. 마지막으로 5장에서 결론을 제시한다.

II. Related Works

본 장에서는 기존의 코드생성 시각화 도구에 대한 연구를 제시하고, 본 논문에서 제시하는 코드생성 지원 시스템과의 차별성을 설명한다.

Frances[4]는 학생들에게는 익숙하지 않은 목적언어에 대하여 친숙할 수 있도록 원시언어의 구조적 문법에 대한 목적언어 코드를 블록화하고 블록화된 코드에 대하여 코드 색칠 기능을 제공한다. 코드 색칠 기능은 블록화된 목적언어 코드를 다른 목적언어 코드 표현색과는 다른 색으로 표현하여 시각화 하는 방법이다. 그리고 원시언어의 제어문에 대한 목적언어 코드의 흐름을 시각화 하여 학생들의 목적언에 이해를 돕는다. 이를 통해서 학생들에게 코드생성 과정에 대한 이해와 목적언어에 대한 친숙할 수 있는 환경을 제공한다. 그리고 Frances에서는 원시언어로 C, C++, Fortran을 제공하고 있으며, 목적언어로 어셈블리어를 제공한다.

MieruCompiler[5]는 XCC 교육용 컴파일러에서 컴파일 과정으로 생성된 정보를 이용하여 코드생성 과정에 대한 시각화를 수행한다. 그리고 MieruCompiler에서는 웹기반의 시각화 도구를 이용하여 원시언어, 목적언어, 구문트리, 심볼테이블, 파싱스택을 시각화 하여 학생들에게 전반적인 컴파일 과정에 대한 이해를 돕는다. 그리고 원시언어로 C언어의 subset인 XC 언어 정의하여 제공하고 목적언어로 어셈블리어를 제공한다.

본 연구에서는 원시언어와 구문트리, 목적언어와의 상관관계를 시각화 하고, 목적언어에 대한 이해를 높일 수 있는 환경을 제공한다. 본 연구에서 제공하는 코드생성 시각화 방법으로는 원시언어 코드를 선택시, 선택된 원시언어 코드에 대응되는 구문트리의 노드와 목적언어 코드를 강조하고, 특정 구문트리 노드 선택시, 선택된 구문트리의 노드에 대응되는 원시언어 코드와 목적언어 코드를 강조하는 기능을 제공한다. 그리고 원시언어의 구조적 문법에 대한 목적언어 코드에 코드 색칠 기능을 제공하여 학생들의 목적언어에 대한 이해를 높이고, 실시간 대화 기능을 통하여 원시언어 코드 변경에 따른 구문트리와 목적언어 코드의 변화를 즉각적으로 확인할 수 있는 기능을 제공한다.

III. A Developing Environment of the Code Generation Support System

본 장에서는 본 연구의 기반이 되는 교육용 컴파일러 통합 개발환경인 Edu-IDEC[6]와 원시언어, 목적언어, 그리고 본 연구의 구현환경에 대하여 설명한다.

1. System Architecture of Edu-IDEC

Edu-IDEC는 이클립스 플러그인 프레임워크 기반의 컴파일러 교육을 위한 통합 개발환경이다. Fig. 1은 Edu-IDEC의 구조를 나타내는데, 크게 컴파일러 제작도구, 레퍼런스 컴파일러, 코드생성 시각화 도구, 목적언어 테스트 도구로 구성된다[6]. Edu-IDEC의 컴파일러 제작도구는 학생들의 컴파일러 프로젝트 진행을 위한 모듈로서, JFLEX[8] 편집기, CUP[9] 편집기, 구문트리 시각화 도구로 구성된다. JFLEX 편집기는 JFLEX 명세와 그외 어휘분석기 생성에 필요한 정보를 설정하여 어휘분석기를 생성하는 도구이다. CUP 편집기는 CUP에 필요한 명세와 그 외 구문분석기를 생성하기 위해 필요한 정보를 설정하고 구문분석기를 생성할 수 있는 도구이다. 그리고 구문트리 시각화도구는 구문트리를 시각적으로 표현하여 학생들에게 컴파일과정을 이해할 수 있도록 돕는 도구이다.

레퍼런스 컴파일러는 학생들의 컴파일러 프로젝트의 가이드라인 역할을 하며 어휘분석기, 구문분석기, 코드생성기로 구성된다. 레퍼런스 컴파일러의 원시언어는 C 언어와 유사한 간단한 C언어를 사용하고, 목적언어로는 LEGO 마인드스톰 NXT의 기계어인 NBC의 부분집합인 Mini NBC 언어를 사용한다.

www.kci.go.kr

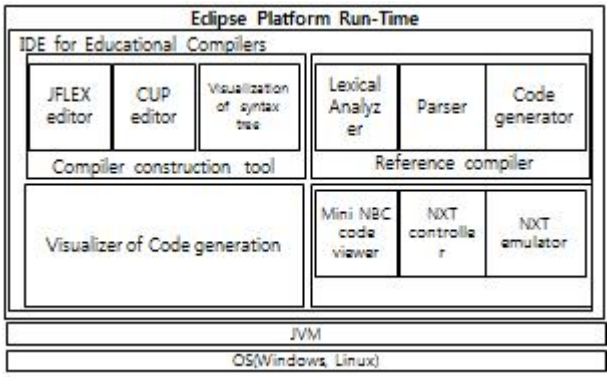

Fig. 1. System Architecture of Edu-IDEC

목적언어 테스트 도구는 학생들이 구현한 컴파일러의 목적언어를 실행하고 테스트하기 위한 모듈로서, Mini NBC 코드 뷰어, NXT 제어기, NXT 에뮬레이터로 구성된다. Mini NBC 코드 뷰어는 컴파일과정을 통해 생성된 목적언어 코드를 보여주기 위한 도구로, 목적언어 코드의 문법강조 기능을 통하여 목적언어에 익숙하지 않은 학생들에게 목적언어 코드의 구조를 시각적으로 쉽게 이해할 수 있도록 돕는다. NXT 제어기는 생성된 목적언어 코드를 NXT 로봇을 작동시킬 수 있는 바이트코드로 변환하고, NXT 로봇으로 바이트코드를 업로드하여 작동시키는 도구이다. NXT 제어기에는 이 과정에서 발생하는 에러의 종류나 위치를 이클립스 결과 창을 통해 알려주는 기능이 있다. NXT 에뮬레이터는 NXT 로봇이 없는 환경에서 가상의 NXT 로봇을 실행하여 NXT 로봇의 센서 값을 제어하고, LCD 출력창과 실행속도를 제어할 수 있는 실행환경을 제공한다. 이러한 환경을 제공하여, 실제 NXT 로봇이 없더라도 컴파일 과정을 통해 생성된 목적언어 코드를 테스트하고 실행해 볼 수 있도록 돕는다.

2. The Source language

Edu-IDEC에서 사용하는 원시언어의 문법은 Fig. 2와 같다. 이는 C 언어와 유사한 문법적 구조를 가지고 있다. 문장의 종류로는 배정문, if문, 함수호출문, while문이 들어갈 수 있으며, 함수호출문은 목적언어의 내장함수를 호출할 수 있다. //는 주석을 의미하며 주석으로부터 해당 줄의 끝까지는 원시언어 코드로 인식하지 않는다. 변수명은 알파벳, 숫자, 밑줄로 이루어지며, 변수명의 첫 번째 글자는 알파벳이어야 한다. 그리고 if, else, while, int는 예약어로, 변수명으로 사용할 수 없다. 산술연산에는 사칙연산이 가능하며, 논리 연산에는 “equal”, “not equal”, “less than equal”, “greater than equal”을 제공한다.

2. The Object language

본 시스템의 목적언어는 LEGO MindStorm NXT[10] 로봇을 조종할 수 있는 저급언어인 NBC(Next Byte Codes) 언어를 축소하여 정의하였다. 문법 구조는 어셈블리어와 유사한 형태이지만, 목적언어를 익히는 시간을 최소화할 수 있도록 NBC의 부분집합을 정의하였으며 이를 Mini NBC로 칭한다. 여기에는 LEGO 마인드스톰 NXT 로봇을 조작할 수 있는 내장함수 및 상수를 포함한다.

```
program     ::=  nxt_control
nxt_control ::=  var_decl VOID NAME PARAM_L PARAM_R comp_stmt
var_decl    ::=  dcl_list
                 | ε
dcl_list    ::=  dcl_list decl
                 | decl
decl        ::=  INT var_list SEMICOLON

var_list    ::=  var_list COMMA NAME
                 | NAME
stmt        ::=  asgn_stmt
                 | if_stmt
                 | call_stmt
                 | while_stmt
                 | comp_stmt
asgn_stmt   ::=  NAME EQ exp SEMICOLON
if_stmt     ::=  IF PARAM_L cond PARAM_R stmt
                 | IF PARAM_L cond PARAM_R stmt ELSE stmt
while_stmt  ::=  WHILE PARAM_L cond PARAM_R stmt
call_stmt   ::=  NAME PARAM_L param_list PARAM_R SEMICOLON
param_list  ::=  exp_list
                 | ε

exp_list    ::=  exp_list COMMA exp
                 | exp
comp_stmt   ::=  BRACE_L stmt_list BRACE_R
stmt_list   ::=  stmt_list stmt
                 | stmt
cond        ::=  exp LTEQ exp
                 | exp GTEQ exp
                 | exp NEQ exp
                 | exp EQ exp

exp         ::=  exp PLUS term
                 | exp MINUS term
                 | term
term        ::=  term MUL factor
                 | term DIV factor
                 | factor
factor      ::=  NAME
                 | NUMBER
                 | HEXA
                 | PARAM_L exp PARAM_R
```

Fig. 2. Grammar of Source Language

Table 1은 Mini NBC의 일부 명령어이다. 변수 선언 블록과 실행문 블록으로 나뉜다. 변수 선언 블록은 dseg segment로 시작하고 dseg ends로 끝이 나며, 변수 선언 블록 내부에서 선언할 수 있는 자료형은 unsigned 8 비트 값을 가지는 byte형을 제공한다. 변수 선언 블록이 끝나면, 실행문 블록 코드가 나오고 실행문 블록은 thread main으로 시작하여 endt로 실행문 블록을 마친다. 실행문 블록 내부에는 데이터 이동과 산술연산, 논리연산, 무조건 분기, 조건 분기 명령 등이 들어간다.

www.kci.go.kr

Table 1. A Part of Object Language

object language instructions	description
dseg segment	Start of variable declaration block
dseg ends	End of variable declaration block
thread	Start of execution statement block
endt	End of execution statement block
mov	Move 2nd argument value into 1st argument.
set	Set 1st argument with 2nd argument value.
add	1st arg = 2nd arg + 3rd arg
sub	1st arg = 2nd arg - 3rd arg
mul	1st arg = 2nd arg * 3rd arg
div	1st arg = 2nd arg / 3rd arg
mod	1st arg = 2nd arg % 3rd arg
cmp	Comparison of 3rd arg and 4th arg with 1st arg. 2nd arg has a result.
jmp	Unconditional jump to 1st arg label
brtst	Compare 3rd arg with 0 , and jump to 2nd arg if true.

4. Implementation environment of the code generation support system

Fig. 3은 개발된 코드생성지원시스템의 구현환경을 나타낸다. 전체적으로는 이클립스 통합 개발환경에 바탕을 두고 있다. Edu-IDEC는 이클립스 플러그인 프레임워크 기반의 교육용 컴파일러 개발환경인데, 제시하는 코드생성지원시스템은 이클립스 플랫폼 런타임과 Edu-IDEC를 기반으로 한다.

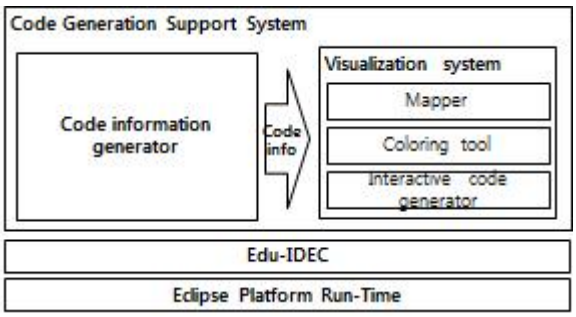

Fig. 3. Implementation Environment of Code Generation Support System

IV. Design and Implementation of the Code Generation Support System

본 장에서는 코드생성지원시스템의 설계와 구현에 대하여 논한다.

1. The Structure of the code generation support system

Fig. 4는 코드생성지원시스템의 전체적인 구성이다. 이는 크게 코드정보생성기와 코드생성시각화도구로 구성된다. 코드정보생성기는 파서분석기와 코드생성분석기로 구성되며, 코드생성시각화 도구는 사상기(Mapper)와 컬러링(Coloring) 도구, 그리고 대화형 코드생성기로 구성된다.

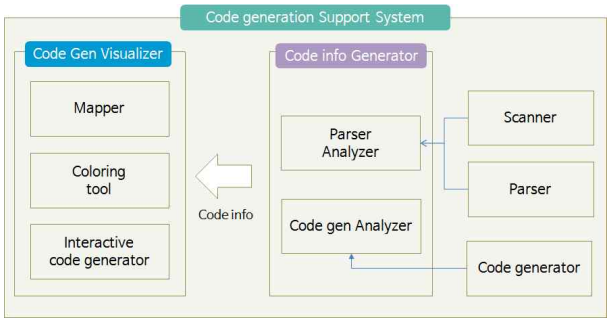

Fig. 4. Architecture of Code Generation Support System

코드정보생성기에서는 구문트리의 특정 노드에 대응되는 원시 프로그램 소스상의 위치정보와 구문트리의 특정 노드에 대응되는 목적 프로그램 소스상의 위치정보를 생성할 필요가 있다. 이러한 코드정보는 구문트리의 노드에 저장되는데, 이 정보는 코드생성시각화도구에서 필요하다. 파서분석기에서는 특정 노드에 대응되는 원시언어 코드의 위치정보를 생성할 필요가 있는데, 이는 어휘분석기와 파서에서 원시 프로그램에 대한 정보를 얻어 생성한다. 그리고 코드생성 분석기에서는 목적언어 코드의 위치정보를 생성할 필요가 있는데, 이는 코드생성기를 통해서 목적언어 코드에 대한 정보를 얻어 생성한다. 파서분석기에서는 구문트리의 특정 노드에 대응되는 원시언어 위치정보를 생성하기 위해서 어휘분석기에서 원시 프로그램 소스를 얻고, 파서로부터 파싱과정에 대한 정보와 구문트리를 얻어 원시언어 코드 위치 정보를 결정한다.

코드 생성 시각화 도구는 코드정보 생성기로부터 받은 코드정보를 통하여 코드생성 과정에 대한 이해를 돕고, 목적언어에 대한 이해와 원시언어-구문트리-목적언어에 대한 이해를 돕는 기능을 한다. 우선, 코드생성 시각화 도구의 Mapper는 사용자가 원시 프로그램 소스상의 원시언어 코드를 선택하면, 선택된 원시언어 코드에 대응되는 구문트리의 노드와 목적언어 코드를 강조한다. 그리고 사용자가 구문트리의 노드를 선택하게 되면, 선택된 구문트리의 노드에 대응되는 원시언어 코드와 목적언어를 강조하는 기능을 수행한다. 그리고 Coloring 도구는 목적언어 코드의 구조를 시각화하기 위하여, 원시언어의 구조적 문법에 대한 목적언어 코드를 다른 목적언어 코드와는 다른 색으로 코드를 표현하는 기능을 수행한다. 그리고 대화형 코드생성기는 원시언어 코드의 변경에 따른 구문트리와 목적언어 코드의 변화를 즉각적으로 확인할 수 있는 기능을 제공한다.

2. Node structure of the syntax tree to support the code generation

본 시스템은 원시코드와 구문트리, 그리고 목적코드에 대한 상호 관계를 보여 줌으로써, 이 관계에 대한 학생들의 이해를 높이고자 한다. 이를 구현하는 방법으로 구문트리에 코드생성

www.kci.go.kr

에 대한 정보를 저장하고자 한다. 이를 위해서 코드생성 지원 시스템의 구문트리의 노드에는 특정 노드에 대응되는 원시언어 코드와 목적언어 코드 위치정보를 표현할 수 있어야 한다. 그리고 코드 색칠 기능을 제공하기 위해서 해당 노드에 대응되는 목적언어 코드의 색에 대한 정보를 표현할 수 있는 있도록 해야 한다. 그리고 파싱스택에는 문법기호(터미널 혹은 논터미널)와 상태번호가 한 쌍으로 저장되는데, 이를 Symbol이라는 클래스로 추상화한다. 이 클래스에는 문법기호와 이 문법기호가 표현되는 원시언어 코드의 위치정보가 저장되어 있다.

```
public abstract Class Node implements Iterator<Node>{
        private int pos = 0;
        public Color color = Color.black;
        public Node[] child;

        public Symbol sym = null;
        public int startPos = 0;
        public int endPos = 0;

        public int target_StartPos = 0;
        public int target_EndPos = 0;
```

Fig. 5. A Part of Node Structure of Syntax Tree

Fig. 5는 구문트리의 노드구조의 일부이다. 여기에서 startPos는 해당 노드에 대응되는 원시언어 코드의 원시 프로그램 소스상의 시작 위치를 의미하고, endPos는 해당 노드에 대응되는 원시언어 코드의 원시언어 프로그램 소스상의 끝 위치를 의미한다. 또한 target_StartPos은 해당 노드에 대응되는 목적언어 코드의 목적 프로그램 소스상의 시작 위치를 의미하고, target_EndPos는 해당 노드에 대응되는 목적언어 코드의 목적언어 프로그램 소스상의 끝 위치를 의미한다. 그리고 color는 해당 노드에 대응되는 목적언어 색깔을 의미한다. 이는 코드생성 시각화 도구의 coloring도구에서 제공하는 기능인 코드 색칠 기능하는데 사용된다. 그리고 sym은 해당 노드에 대응되는 파싱스택의 내용에 대한 추상화인 Symbol 객체를 참조하기 위한 필드 객체이다.

3. Corresponding relation between parsing stack and nodes of syntax tree

파서분석기는 특정 노드에 대응하는 원시언어 코드 위치 정보를 얻어야 하는데, 어휘분석기가 그 정보를 제공하는 것이 편리하다. 따라서 어휘분석기에는 파서분석기에게 해당 어휘에 대한 원시언어 코드 위치 정보를 제공하는 기능을 포함시킨다.

Fig. 6은 구문트리의 노드와 파싱스택에 저장되는 내용인 Symbol과의 관계를 나타낸다. 클래스 Symbol은 파싱스택에 저장되는 내용을 추상화한 것이다. 클래스 Symbol의 기본구조는 (문법기호, 상태번호)의 쌍을 나타내는데, Fig. 6에서 sym은 문법기호의 종류를 의미하고 parse_state는 상태 번호를 의미한다.

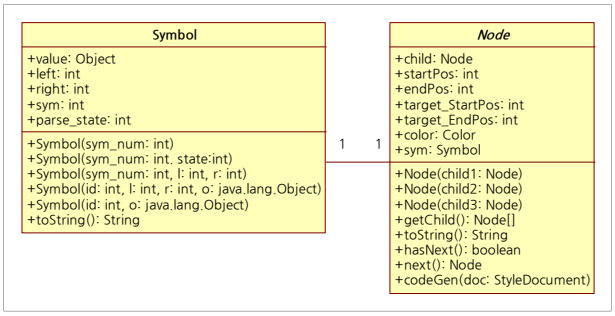

Fig. 6. Corresponding Relation between Parsing Stack and Nodes of Syntax Tree

left와 right는 해당 Symbol 객체가 표현하고 있는 (문법기호, 상태번호)의 쌍에 대응되는 원시 프로그램 소스상의 위치를 나타낸다. value는 Symbol 객체에 대응되는 구문트리의 노드에 대한 참조이다. 원시언어와 구문트리 사이의 관계를 분석하기 위해서는 구문트리 노드에 대응되는 Symbol 객체를 참조하고 있어야 하는데 클래스 Node의 sym객체가 그 역할을 담당한다.

4. Parser analyzer

파서분석기에서는 특정 노드에 대응되는 원시언어 코드의 위치 정보를 해당 노드에 저장하는 역할을 한다. 이를 위해서는 현재 단계의 파싱스택과 다음 단계의 파싱스택을 비교를 통해 감축되어 생성되는 노드의 자식노드를 얻는다. Table 2는 파싱과정에서 해당 문법규칙에서 현재 단계 파싱스택과 다음 단계 파싱스택의 내용인데, 이를 비교·분석하여 특정 노드의 자식노드를 결정할 수 있다. Table 2의 문법 규칙은 TermNode / FactorNode가 ArithExpNode로 감축되는 예이다. 이 문법 규칙은 원시 프로그램 코드 “20 / 10”에 대응된다.

TermNode/FactorNode가 ArithExpNode로 감축될 때 생성하는 노드는 Fig. 7의 ArithExpNode이다. 이 노드의 startPos필드와 endPos필드에 20과 10에 대응되는 원시코드 위치정보를 저장해야한다.

Table 2. Example Contents of Parsing Stack

rule	ArithExpNode ::= TermNode / FactorNode
(A) parsing stack	VarDeclNode void NameNode ( ) { NameNode = ExpNode + TermNode / FactorNode
(B) parsing stack	VarDeclNode void NameNode ( ) { NameNode = ExpNode + ArithExpNode:DIV

source program	int VAR;
	void main()
	{
	VAR = 20 + 20 / 10;
	SetSensorLight(IN_4);
	}

www.kci.go.kr

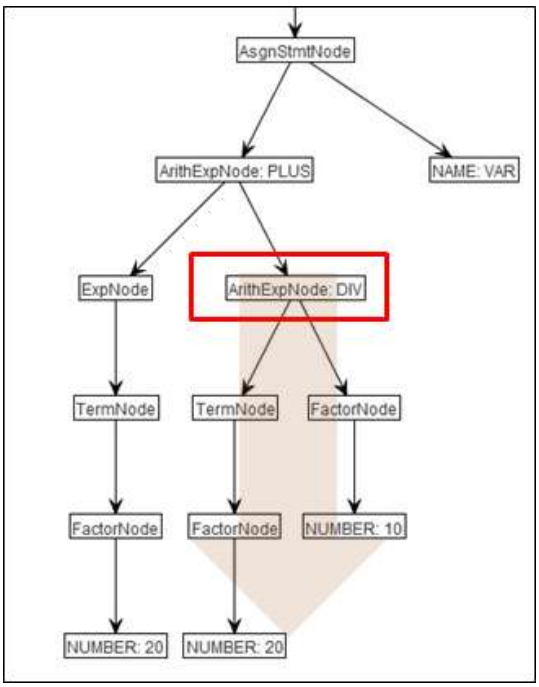

Fig. 7. Identifying the location information in Tree Node of Source Code

Fig. 7에서 ArithExpNode에 대응되는 원시언어 코드 위치 정보를 결정하기 위하여, ArithExpNode의 후손 노드 중에서 단말 노드를 찾으면 터미널 20을 표현하는 좌단 단말 노드인 NumberNode이다. 이 노드에 표현되어 있는 원시언어 코드 시작 위치를 얻는다. 또한, 터미널 10을 표현하고 있는 우단 단말 노드인 NumberNode를 찾고, 이 노드에서 원시언어 코드 위치 정보를 얻을 수 있다.

5. Analyzer of code generation

코드생성 분석기에서는 구문트리의 노드에 대응되는 목적언어 코드 위치정보를 결정하고 이를 해당 노드에 저장하는 기능을 한다. 그리고 코드 색칠 기능을 위하여 각 노드에 대응되는 목적언어 코드를 생성할 때 텍스트 컬러 정보도 함께 생성한다. 특정 노드에 대응되는 목적언어 코드 위치 정보를 결정하기 위하여 단위 모듈당 목적언어 코드 생성 코드를 실행 이전에, 지금까지 생성된 목적언어 코드의 길이를 구하고, 단위 모듈에 대한 코드생성 코드를 실행 이후의 목적언어 코드의 길이를 구하면 된다. 이를 해당 노드의 목적언어 코드 위치 정보에 저장한다.

6. Implementation of a visualization tool for code generation process

코드생성 시각화 도구는 사상기, 컬러링 도구, 대화형 코드생성기로 구성된다. 이는 코드생성 분석기로부터 제공 받은 원시언어 코드 위치정보와 목적언어 코드 위치 정보가 저장된 구문트리를 입력으로 받는다.

6.1 사상기

Fig. 8은 선택된 원시언어 코드에 대응되는 구문트리의 노드와 목적언어 코드가 강조된 코드생성시각화도구의 사상기를 구현한 결과이다. 사상기는 원시언어 코드와 구문트리, 목적언어 코드의 연관관계를 표현하는 기능을 수행한다. 사용자가 왼쪽 화면에서 원시언어 코드를 선택하면, 선택된 원시언어 코드에 대응되는 구문트리의 노드와 목적언어 코드가 강조되어 표현된다. 그리고 사용자가 구문트리의 노드를 선택하면, 선택된 구문트리의 노드에 대응되는 원시언어 코드와 목적언어 코드가 강조되어 표현된다.

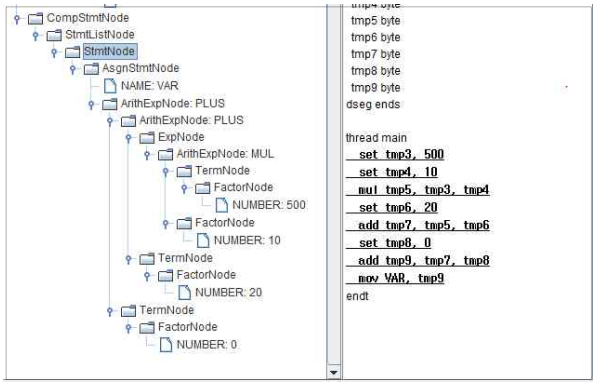

Fig. 8. Implementation Result of Mapper

사용자가 원시 프로그램 소스에서 특정 원시 코드를 선택하면, 선택된 코드의 시작위치 값과 끝 위치 값을 얻는다. 그리고 이 선택된 코드의 시작 위치 값을 sel_sp라 두고, 끝 위치 값을 sel_ep라 두면, 구문트리의 루트노드에서부터 노드의 start_pos 필드와 end_pos 필드를 조사하여, sel_sp >= start_pos && sel_ep <= end_pos 조건을 만족하는 최하위 노드를 결정한다. 그리고 구문트리 뷰어에서 이 노드를 표현하는 부분을 강조하고, 그 노드에서 목적언어 코드 위치 정보를 얻어서 목적언어 코드를 강조한다.

만약 사용자가 구문트리의 노드를 선택하면, 그 노드에 이미 원시언어 코드 위치정보와 목적언어 코드 위치정보가 저장되어 있기 때문에 이 정보를 가져와서 원시언어 코드와 목적언어 코드를 쉽게 강조할 수 있다.

6.2 컬러링 도구

컬러링 도구에서는 코드 색칠 기능을 제공한다. 코드 색칠 기능은 원시언어의 구조적 문법에 대한 목적언어 코드의 표현 색을 다른 목적언어 코드와는 다른 색으로 표현하여 시각화 하는 기능이다. 이러한 기능을 위해 구문트리의 노드에 목적언어 코드색 정보를 정의하는 정보를 추가하였다.

www.kci.go.kr

```
public class IfStmtNode extends AbstractStmtNode {

public IfStmtNode(Node child1, Node child2) {
        super(child1, child2);
        color = Color.red;
}

public IfStmtNode(Node child1, Node child2, Node
child3) {
        super(child1, child2, child3);
        color = Color.red;
}

…

}
```

Fig. 9. Color Decision for if Statement

Fig. 9는 원시언어의 IF문에 대하여 목적언어 코드 표현색을 정의하는 코드이다. 여기서는 IF문에 대한 목적언어 코드색은 빨강색으로 정의하고 WHILE문에 대해서는 파랑색으로 정의한다. 디폴트로는 검정색이다.

Fig. 10은 코드 색칠기능을 통하여, 원시언어의 IF문에 대하여 목적언어 코드를 빨강색으로 표현한 것이다. Fig. 10에서 목적언어 코드 뷰어를 보면 “brtst EQ, IF_ELSE_15, tmp2”와 “jmp IF_END_15”, “IF_ELSE_15:”, “IF_END15:”가 빨강색으로 표현되어(코드 끝에 ***로 표시함.) 원시 프로그램 소스의 IF문에 대응되는 목적언어 코드를 분명하게 확인할 수 있다.

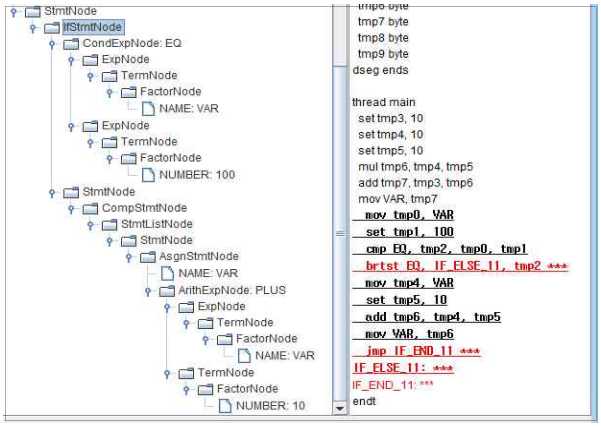

Fig. 10. Coloring Result for if Statement

6.3 대화형 코드생성기

대화형 코드생성기에서는 원시언어 코드의 변경에 따른 구문트리와 목적언어 코드의 변화를 즉각적으로 확인할 수 있는 기능을 제공한다. 이를 위하여 원시언어 코드에 대한 수정과 저장 기능을 제공한다.

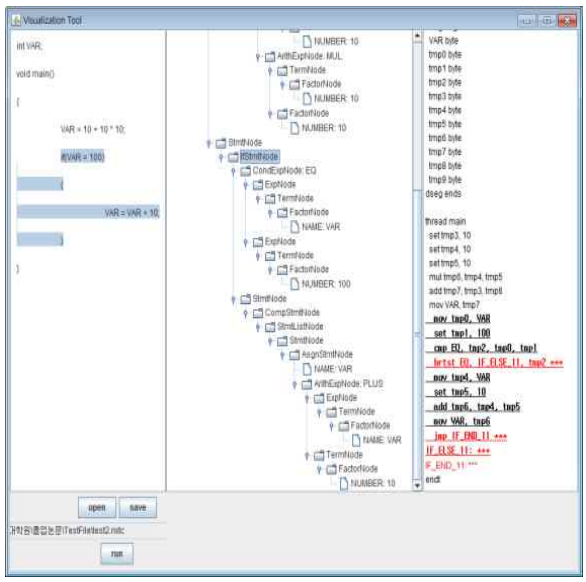

Fig. 11. Implementation Result for Visualization Tool

Fig. 11은 Fig. 10에서 원시언어 코드 수정 이후 대화형으로 시각화가 수행되어 표현된 코드생성 시각 도구의 화면이다. 이 기능은 사용자가 원시언어 코드를 수정하고 저장버튼을 클릭하여 저장하고 실행 버튼을 클릭하면 변경된 원시언어 코드에 대응되는 구문트리와 목적언어 코드를 제시한다. 이를 통해서 코드생성 시각화 도구 내에서 즉각적으로 구문트리와 목적언어 코드의 변화를 확인할 수 있다.

IV. Conclusions

컴파일러 교과목은 컴퓨터과학 분야의 주요한 교과목중 하나이다. 특히 컴파일러 프로젝트를 통하여 학생들에게 다양한 지식과 사고력을 키울 수 있는 환경을 제공한다. 그러나 충분한 실습환경 제공이 어렵고 학생들의 컴파일러 프로젝트에 대한 부담으로 이론 위주의 수업방식이 주로 이루어지고 있다. 이에 따라 교육용 컴파일러 통합 개발환경을 이용한 실습 위주의 수업모델에 대한 연구가 필요하다. 본 연구에서는 교육용 컴파일러 제작 실습을 효과적으로 수행할 수 있도록 도와주는 목적으로, 교육용 컴파일러 통합 개발환경에서 코드생성지원시스템을 개발한 결과를 제시하였다.

개발된 시스템은 C언어 기반의 원시언어, NBC 기반의 목적언어, 그리고 구문트리와의 관계를 시각화함으로써, 학생들의 목적언어에 대한 이해를 높이고, 코드생성 과정에 대한 이해를 도와서 코드생성기를 쉽게 만들 수 있는 환경을 제공한다. 그리고 이클립스 플러그인 프레임워크 기반의 교육용 컴파일러 통합 개발환경 Edu-IDEC에서 개발되어 확장성이 뛰어나고 배포가 용이하다.

www.kci.go.kr

REFERENCES

[1] Saumya Debray, “Making compiler design relevant for students who will (most likely) never design a compiler,” Proceedings of the. 33rd SIGCSE Technical Symposium on Computer Science Education, pp. 341-345, 2002.

[2] Marc L. Corliss and E Christopher Lewis, “Bantam: a customizable, java-based, classroom compiler,” Proceedings of the 39th SIGCSE technical symposium on Computer science education, pp. 38-42, 2008.

[3] Li xu and Fred G. Martin, “Chirp on crickets: teaching compilers using an embedded robot controller,” Proceedings of the 37th SIGCSE technical symposium on Computer science education, pp. 82-86, 2006.

[4] Tyler Sondag, Kian L. Pokorny and Hridesh Rajan. “Frances: a tool for understanding code generation,” Proceedings of the 41st ACM technical symposium on computer science education, pp. 12-16, 2010.

[5] Katsuhiko Gondow, Naoki Fukuyasu and Yoshitaka Arahori. “MieruCompiler: integrated visualization tool with horizontal slicing for educational compilers,” Proceedings of the 41st ACM technical symposium on Computer science education, pp. 7-11, 2010.

[6] Woo-Kyung Sung, Hyun-Syug Kang, and Jong-Min Bae, “Development of an Eclipse-based IDE for Educational Compilers,” Journal of Korean Association of Computer Education, Vol 14, No 6, pp., 9-18, September 2011.

[7] Eclipse Foundation, Inc. “About the Eclipse Foundation”, http://www.eclipse.org/org/

[8] Gerwin Klein, “JFlex - The Fast Scanner Generator for Java”, http://www.jflex.de/

[9] Scott E. Hudson, “CUP parser Generator for Java”, http://www2.cs.tum.edu/projects/cup/

[10] Lego. Inc., “8547 LEGO® MINDSTORMS® NXT 2.0”, http://mindstorms.lego,com

Authors

Jung-Hoon Kwon received the BS and MS degree in Computer Science Department from Gyeongsang National University, Korea, in 2012 and 2014 respectively.

He is currently working at Aero Master Corporation, Sacheon, Korea. He is interested in programming languages, compilers, and open sources.

Jong-Min Bae received the B.S. degree in mathematics education, and the M.S. and Ph.D. degrees in Computer Science and Statistics from Seoul National University, in 1980, 1983 and 1995, respectively

Dr. Bae joined the faculty of the Department of Computer Science at Gyeongsang National University, Jinju, Korea, in 1984. He is currently a Professor in the Department of Computer Science, Gyeongsang National University. He is interested in programming languages, computer education, and compilers.

www.kci.go.kr