프로그래밍 교육을 위한 순서도 생성기 구현
(Implementation of A Flow Chart Generator for Teaching Programming)

최성권, 류시혁, 신승철
Sung Kwon Choi, Si-Heok Ryu, Seung Cheol Shin
동양대학교 컴퓨터공학부

요 약 프로그래밍 교육을 할 때 작성된 프로그램에 대하여 순서도를 자동 생성하여 보여줄 수 있다면 매우 유용할 것이다. 본 논문은 간단한 명령형 언어 While의 프로그램을 입력받아 순서도를 작성해 주는 방법을 제안한다. While프로그램을 순서도 작성에 적합한 언어인 Flow Chart언어 프로그램으로 전환하는 번역기를 생성한다. 이렇게 생성된 FCL을 가지고 그래픽 라이브러리를 이용하여 순서도를 자동으로 생성하는 방법을 제시한다. 본 연구의 결과를 이용하면 프로그램의 초보자들도 쉽게 프로그램의 흐름을 이해할 수 있다.

Abstract When we are teaching the programming education, it is very useful if we show that a flow chart generate automatically owing to a written out of the program. In this paper, we introduce the method of a flow chart generating from the simple imperative language for the While program. After the While program is translated by the Flow Chart Language program, it is generated automatically by making use of the java graphic library. Taking advantage of our results of the study, beginners can understand the flow of the program easily.

1. 서론

순서도란 컴퓨터로 처리하고자 하는 문제를 분석하여 그 처리 순서를 단계화하여 상호간의 관계를 약속된 기호와 흐름선을 사용하여 알기 쉽게 나타낸 그림이다. 순서도를 작성하는 것은 사람의 손으로 혹은 툴을 가지고 수동적으로 순서의 흐름을 표현한다. 자신이 원하는 프로그램의 순서도를 보여주기 위해서는 직접 사용자가 그려서 표현을 하게 된다. 또한 이해하기 힘든 프로그램의 흐름을 제대로 알지 못할 경우 순서도를 그리는 것은 힘이 들었다. 프로그램의 소스로부터 순서도 작성을 자동적으로 하는 것은 프로그램의 흐름을 정확히 이해할 수 있으며 언어 숙달에 효과가 있다.

본 연구에서는 프로그램 순서도를 자동으로 그리기 위하여 자바를 기반으로 만들어진 Java Compiler Compiler(이하 JavaCC)[1, 9-13]를 이용하여 **While** 언어[14]에 대한 순서도 작성기를 구현한다. 일반 프로그래밍 언어를 대상으로 하는 것은 본 논문의 목적인 순서도 생성기법을 개발하기에 부차적인 부분이 많이 포함되므로 핵심언어 중 하나인 While언어에 대하여 순서도 생성 기법을 제안한다. **While** 언어는 if문, while문, assign문, skip문, comp문 등의 최소 집합으로 이루어진 작은 언어이다. 이러한 **While** 프로그램을 순서도로 표현하기 적합한 언어인 FCL(Flow Chart Language)[8]로 변환하여 순서도를 생성한다. 그리고 순서도를 표현하는데 있어서는 FCL 프로그램의 데이터를 인접리스트의 표현으로 변환한다. 이렇게 변환시킨 그래프 데이터를 가지고 순서도를 그래프의 표현식으로 구현을 한다.

논문의 구성은 2장은 순서도 생성기구조, 3장은 While-to-FCL 변환과정, 4장은 순서도 작도, 5장은 예제를 통한 실행결과 분석, 마지막으로 6장에서는 결론 및 발전방향 순으로 진행된다. 본 연구의 결과를 이용하면 **While** 프로그램의 순서도를 자동으로 생성하여 프로그램의 초보자들도 쉽게 프로그램의 흐름을 이해할 수 있다.

2. 순서도 생성기 구조

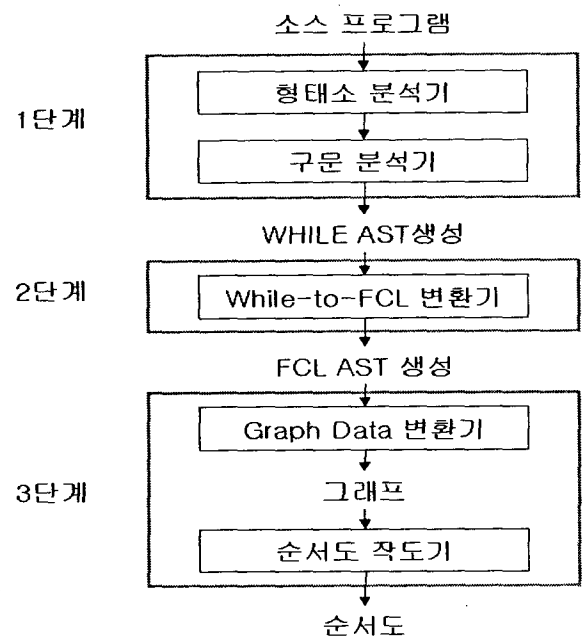
2.1 순서도 생성기 구조

컴파일러란 어떤 언어(source language)로 쓰여진 프로그램을 입력으로 받아들여서 대등한 목적 언어(target language)의 프로그램으로 바꿔주는 프로그램이다. (그림 1)은 순서도 생성기의 단계이다. 순서도 생성기의 단계는 컴파일러의 전반부의 순서(1단계)를 따라 처리되며 추상 구문 트리(abstract syntax tree)를 만든다. 후반부는 순서도를 생성하기 위하여 트리 변환, 그래프 데이터 변환, 순서도 작도의 단계를 거치게 된다. 전반부는 JavaCC로 구현을 하였고 후반부는 Java로 구현을 하였다.

2.2 구문 분석

구문분석은 (그림 2)의 While문법을 따르며 이 문법을 형

태소 분석기와 구문분석기를 통하여 파싱을 한다. 파싱후 AST를 만들게 된다.



(그림 1) 순서도 작성 시스템 구조

Syntax Domains

$n \in$ *numerals,NUM* $a \in$ Arithmetic expressions, Aexp

$x \in$ Variables,VAR $b \in$ boolean expressions,VAR

$S \in$ Statement, Stm

Gammar

a ::= **n** | **x** | **a1** + **a2** | **a1** * **a2** | **a1** − **a2**

b ::= **true** | **false** | **a1** = **a2** | **a1** <= **a2** | ¬**b** | **b1** ∧ **b2**

S ::= **x** := **a** | **skip** | *S1* ; *S2* | **if b then** *S1* **else** *S2* | **while b do** *S*

(그림 2) While 문법

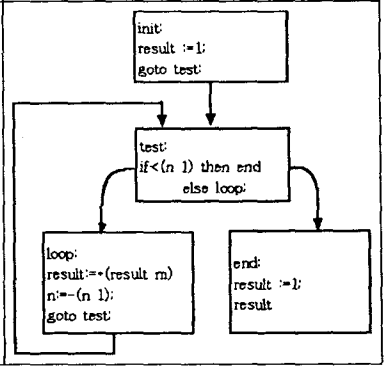
(그림 3)의 FCL는 순서도를 표현한 언어로서 명확한 의미 전달로 이해하기 쉽고 코드를 분석하고 연구하는데 충분한 언어이다. FCL의 예는 (표 1)과 같다.

3. While-to-FCL 변환과정

3.1 변환규칙의 개요

다음의 (그림 4)는 While AST를 FCL AST 변환 규칙이다. (그림 2)의 While 문법의 WhileAssign , WhileSkip, Whilecomp, WhileIf, WhileWhile문을 변환하는 규칙이며 Trf 펑션에 While AST를 아규먼트로 받아 트리 순행을 하면서 변환을 한다. 트리를 변환하는 이유는 FCL순서도의 의미가 명확하고 FCL AST를 그래프 데이터로 변환이 용이하다. 또한 순서도의 생성뿐만이 아닌 데이터 흐름 분석 및 애니메이션등 여러 가지 응용분야를 연구할 수 있다.

FCL 프로그램 소스	프로그램 순서도

(m n)
(init)
Init : result := 1;
goto test;
test : if < (n 1)
then end
else loop;
loop : result:=*(result m);
n := -(n 1);
goto test;
end : return result;
	

(표 1) FCL프로그램 소스 및 FCL 순서도

Syntax Domains

P ∈ *Program[FCL]* *X* ∈ *Variables[FCL]*

b ∈ *Blocks[FCL]* *e* ∈ *Expressions[FCL]*

l ∈ *Block-Labels[FCL]* *c* ∈ *Constants[FCL]*

a ∈ *Assignments[FCL]* *j* ∈ *Jumps[FCL]*

al ∈ *Assignments-Lists[FCL]* *O* ∈ *Operations[FCL]*

Grammar

p ::= (x*) (l) b +

b ::= **l** : **al j**

a ::= **x** := **e**;

al ::= **a al** | .

e ::= **c** | **x** | **o(e*)**

j ::= **goto l** | **return e** ; | **if e then l1 else l2**

(그림 4) FCL 문법

Assign문

Tr (Assign(var,e)) L_{in} L_{out}

-> [Block(L_{in}, [Assign(var,e)], L_{out})]

Skip문

Tr Skip L_{in} L_{out} -> [Block(L_{in}, [Skip], L_{out})]

Composition문

Tr (S_1;S_2) L_{in} L_{out} -> (Tr S_1 L_{in} L_{mid})@(Tr S_2 L_{mid} L_{out})

where L_{mid} is newlabel

If문

Tr (If(e,S_1,S_2)) L_{in} L_{out} -> Block(L_{in},[],If(e, L_{yes}, L_{no}) ::

(Tr S_1 L_{yes} L_{out})@(Tr S_2 L_{no} L_{out})

where L_{yes}, L_{no} are newlabel

While문

Tr (while(e, S_1)) L_{in} L_{out} ->

Block(L_{in}, [], If(e, L_{yes}, L_{out})) :: (Tr S_1 L_{yes} L_{in})

where L_{yes} is newlabel

Trans form

blocks -> (Tr Stmt $Start_{label}$ $Halt_{label}$)

TrF (WHILE(vars, Stmt)) ->

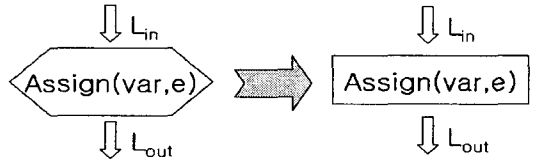
FCL(vars, $Start_{label}$, blocks@[Block($Halt_{label}$,[],Return)])

where $Start_{label}$ is newlabel and $Halt_{label}$ is lastlabel

(그림 4) WHILE-to-FCL 변환 규칙

3.2 Assign문, skip문 변환규칙

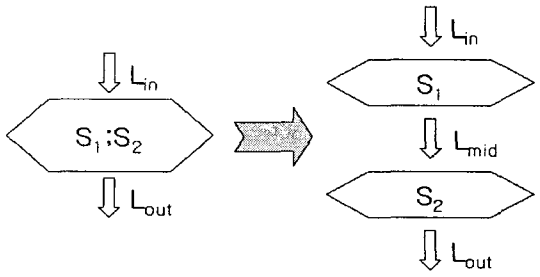
Assign문과 Skip문(그림 5)은 동일한 블록을 가진다. 여기서 L_{in}과 L_{out}은 label을 나타내며 이것은 1부터 순차적으로 증가한다.



(그림 5) Assign문 변환 규칙

3.3 Composition문 변환규칙

Composition문(그림 6)은 문장과 문장과의 연결을 말한다. 세미콜론과 같이 문장을 구분할 수도 있지만 While 문법에서는 WhileComp라고 하였다. (그림 6)과 같이 문장과 문장 사이에 L_{mid} label을 삽입하여 변환을 할 수 있다.

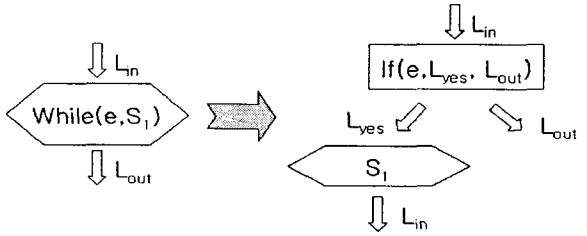


(그림 6) Composition문 변환 규칙

3.4 If문, While문 변환규칙

If문과 While문의 구조는 사실상 같다. (그림 7)과 (그림 8)에서 보듯이 While문의 경우는 Loop일 경우는 이전의 L_{in}을 다시 돌려 받으면 된다. While AST(그림 9) 변환된 FCL는 각각의 블록의 구조를 생성하고 이것들의 합성은 TrF(trans form)에서 블록들을 리스트로 하며 마지막으로 Halt 블록을 합성하여 FCL AST(그림 10)로 변환을 한다.

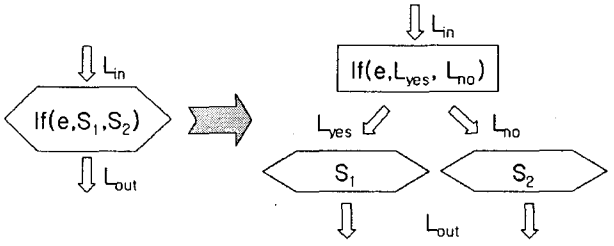
(그림 9)의 WHILE AST는 (그림 10)와 같이 변환된 FCL AST를 가지게 된다.



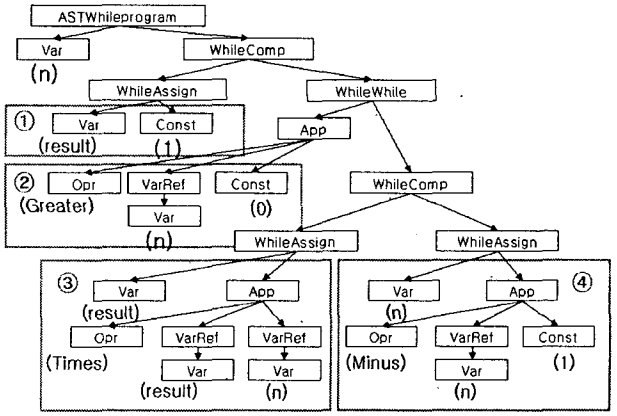
(그림 7) While문 변환 규칙

4. 순서도 작도

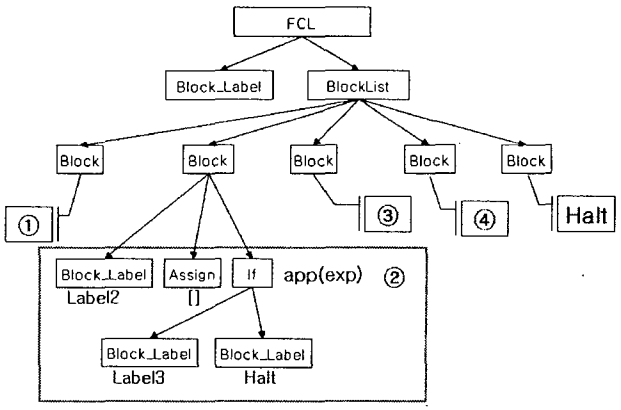
순서도를 그리기 위하여 우선 먼저 FCL AST 데이터를 읽어서 표현을 해야 한다. 그러나 FCL AST 데이터를 읽어서 그림으로 표현할 수도 있지만 프로그램의 복잡해지는



(그림 8) If문 변환 규칙



(그림 9) WHILE AST구조



(그림 10) FCL AST구조

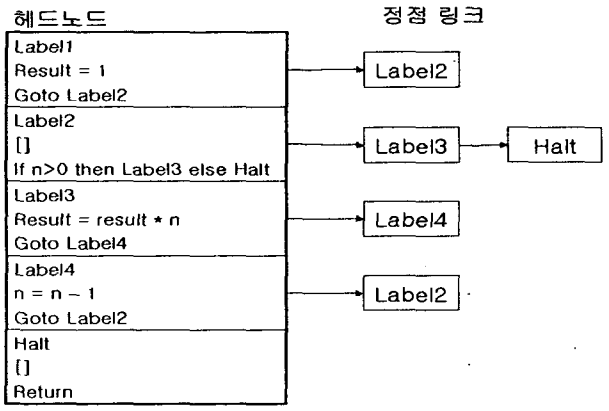
단점이 있다. 그림을 그릴 수 있는 최소 단위의 데이터인 그래프로 표현을 하고자 한다. 그래프의 표현은 다음과 같은 방법으로 표현을 할 것이다.

4.1 그래프 표현법

우선 자료구조에서 제시하는 그래프 표현법은 3가지 방법이 있다. 인접 행렬, 인접 리스트, 인접 다중 리스트의 3가지 방법이 있다. 이중에 본 논문의 그래프 표현 방법은 (그림 11)과 같이 인접리스트의 방법을 사용하였다.

(그림 11)에 보인 바와 같이 인접 리스트로 구현된다. 각 항(cell)들은 노드를 나타내는 것이다. 이 알고리즘은 그래프의 차수(degree)를 찾는데 적용되고 각 노드에 대한 입력차수(indegree)와 출력차수(outdegree)를 구할 수 있다. 다음의 (표 2)와 같이 차수를 찾고 이것으로 노드의 모양을

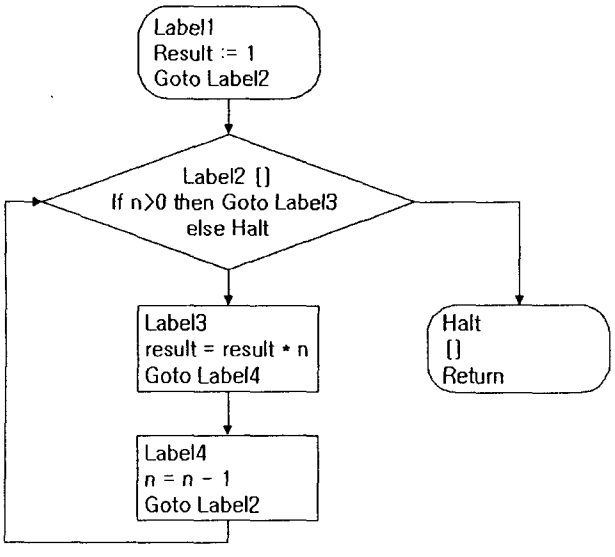
결정 할 수 있다. (그림 12)의 순서도와 같이 Start와 Exit는 둥근 사각형, Assign문은 사각형, Test는 마름모꼴로 표현 할 수 있다. 다중 If문 혹은 다중 While문의 경우 차수의 수가 1개 이상인 경우가 발생되는 점을 유의하여 노드를 표현해야 한다. Assin문의 경우도 차수의 수가 1개 이상이다. (표 2)의 순서도의 표현은 (그림 12)와 같다.



(그림 11) 그래프 인접리스트 표현

Label	indegree	outdegree	차수별 노드정의
Label1	0	1	Start
Label2	2	2	Test
Label3	1	1	Assign
Label4	1	1	Assign
Label5	1	0	Exit

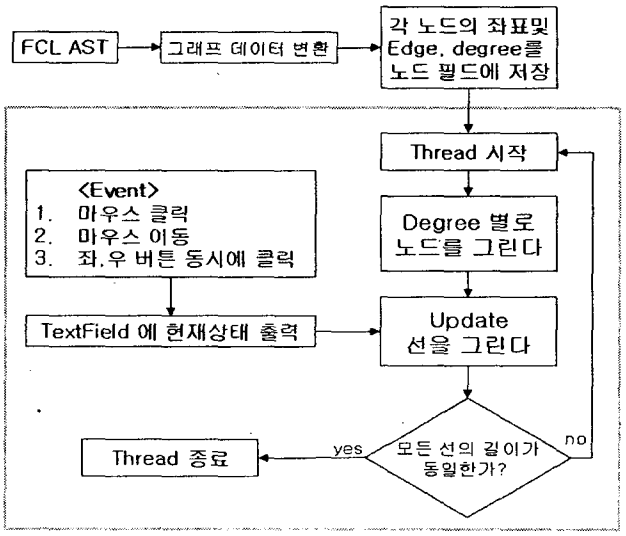
(표 2) 차수별 노드정의



(그림 12) 순서도

4.2 순서도 작도 과정

자바를 이용하여 순서도 작도 과정은 (그림 13)과 같다. FCL AST를 입력받아 트리 순행을 하면서 각 블록의 레이블을 조사하여 goto label과 $label_{in}$이 같으면 블록과 블록을 링크시킨다. 이렇게 (그림 11)처럼 인접리스트가 만들어지면 각 노드마다 좌표 및 Edge를 계산하여 각 객체에 저장을 한 다음 순서도를 그리게 된다. 각 노드를 연결하는 선을 그릴 때 노드의 선이 동일하지 않으면 노드의 수가 많아질수록 복잡해지므로 쓰레드를 이용하여 노드의 선을 체크한다.



(그림 13) 순서도 작도 과정

5. 결과 및 분석

5.1 예제 프로그램 소스

```
y := 1
while ¬(x = 1) do (y := x* y; x := x - 1)
```

(그림 14) factorial 예제 프로그램

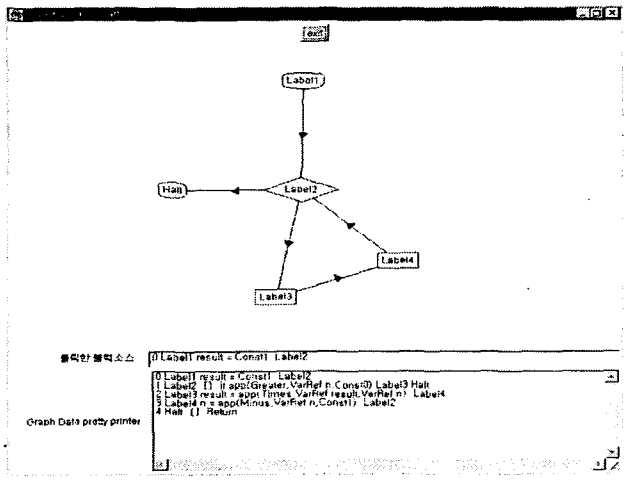
5.2 실행결과

(그림 14)의 소스 실행화면은 (그림 15)와 같이 실행된다. 우선 각 노드를 마우스로 선택을 하면 클릭한 블록의 프로그램은 TextField에 출력된다. 또한 While 프로그램을 분석한 그래프 데이터 중간코드는 TextArea창에 나타난다. 마우스 이벤트의 사용으로 노드를 간편하게 이동할 수도 있고 고정시킬 수도 있다. 또한 각 노드가 방향성을 가지고 있으므로 방향을 표시하기 위한 화살표를 삽입하여 순서의 흐름을 명확히 하였다.

5.3 문제점 분석

While 프로그램 소스를 입력받아 화면상으로 표현되어지는 순서도에는 여러 문제점이 있다. 그 중에서 가장 큰 어려운 문제는 순서도의 생성 시 While문이나 IF문의 경우 이전 노드로 되돌아가야 하는데 되돌아가는 선을 표현하기 어려운 점이다. 또한 화면상의 제약이 따른다. 예를 들어 프로그램의 소스가 커지면 커질수록 순서도를 복잡해지고 보기 힘들어지는 문제점을 안고 있다. (그림 12)처럼 순서도를 보이는 것이 작은 프로그램에는 할 수 있으나 위에서

언급한 문제를 해결하는데는 어려움이 있다.



(그림 15) factorial 예제 출력결과

6. 결론

본 논문은 자바를 기반으로 하고 있는 JavaCC와 JJTree를 이용하여 WHILE 언어에 대한 순서도를 자동 생성하는 컴파일러를 구현하였다. JavaCC는 자바와 동일한 규칙을 따르고 있으며 하향식 파싱을 지원한다. 하향식 파싱에는 여러 가지 제약이 따르게 되나 JJTree가 상향식 파싱으로 JavaCC의 미비점을 보완하는 역할을 하고 또한 AST를 구현하는 것을 JJTree 옵션으로 설정하여 쉽게 구현할 수 있다. While 프로그램을 입력받아 파싱을 하여 만들어진 AST를 가지고 의미분석을 하게 된다.

의미를 분석하여 순서도를 생성하기 위하여 여러 가지 방법을 사용하였다. 우선 FCL를 사용하였다. FCL는 간결하고 이해하기 쉽고 또한 순서도를 그리기 적합한 언어로서 표현이 자유롭고 명확한 언어이다. 이 언어로서 WHILE 언어를 순서도의 표현으로 변환시킬 수 있다는 것을 보여주며 데이터의 조작을 간편하게 한다.

그러나 화면상으로 표현되어지는 순서도 실행결과는 여러 가지 제약이 따르는 문제점을 보여주었다. 그 중에서 가장 큰 어려운 문제는 순서도 생성 시에 While문이나 IF문의 경우 이전 노드로 되돌아가야 하는데 되돌아가는 선을 표현하기 어려운 점이다. 또한 화면상의 제약이 따른다. 프로그램의 소스가 커지면 커질수록 프로그램의 이해도가 떨어진다.

이런 문제점을 보안하기 위해 그래프의 표현식을 사용하였다. 상대좌표를 이용하여 그래프의 표현식을 사용하였다. 상대좌표의 방법과 노드를 고정시켜 마우스로 이동 할 수 있는 여러 방법으로 순서의 흐름을 표현하였다. 최단 경로를 구하는 문제에서의 그래프 표현식을 착안하여 그래프의 형식으로 노드를 그리고 각 노드는 차수(degree)별로 특징점을 추출하여 모양을 표현했다. 또한 TextArea와 TextField를 사용하여 각 노드를 클릭하였을때 그 노드의 식이 Textfield에 나타나게 하여 화면상의 표현력을 보완하였다.

본 논문은 순서도를 자동 생성하여 프로그래밍 언어를 처음 배우는 학생들에게 프로그램의 소스를 순서도로 보여주게 되어 이해 및 언어 숙달에 도움을 줄 것이다. 또한 프로그램 언어 연구, GUI 디버깅 연구 및 데이터 흐름 분석 연구에 응용 할 수 있다. 우선과제는 문제점 분석에서의 여러 문제를 해결하여 그래프의 표현식이 아닌 순서도로 표현하고 프로그램 흐름을 명확히 해야한다. 또한 애니메이션(Animation) 처리를 통하여 데이터 흐름을 보이고 또한 While 언어의 확장으로 기타 다른 언어에서 순서도를 작성할 수 있도록 방법을 연구 할 것이다.

참 고 문 헌

[1] 윤태중, "JavaCC를 이용한 파스칼 컴파일러의 구현", 연세대학교 대학원 컴퓨터과학과 석사학위논문, 1998

[2] Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers Principles, Techniques and Tools, 사이버출판사, 한국어판, 1993

[3] Jeffrey D. Ullman, 오토마타와 계산이론, 홍릉과학출판사, 1993

[4] Robert W. Sebesta, Concepts of Programming Languages second edition, Addison Wesley, 1996

[5] Andrew W. Appel, Modern Compiler Implementation in Java, CAMBRIDGE UNIVERSITY PRESS, 1998

[6] Adam Drozdek, Data Structure and Algorithms in Java, BROOKS/COLE, 2001

[7] David A Watt & Deryck F Brown, Programming Language Processors in Java, Compilers and Interpreters, Prentice Hall, 2000

[8] John Hatcliff, "An Introduction to Online and Offline Partial Evaluation Using a Simple Flowchart Language", Department of Computing and Information Science Kansas State University, 1998

[9] http://www.javaworld.com/javaworld/jw-12-2000/jw-1229-cooltools_p.html, Build your own languages with JavaCC

[10] http://www.javaworld.com/javaworld/jw-12-1996/jw-12-jack_p.html, Looking for lex and yacc for Java? You don't know Jack

[11] http://www.alma-services.dial.pipex.com/informatics/source_code/java/JavaCC/Pretty/Pretty.html, About Pretty

[12] http://www.webgain.com/products/java_cc/documentation.html, JavaCC2_1 Documentation

[13] http://www.javasoft.com/products/index.html, Java Products & API

[14] Hanne Riis Nielson and Flemming Nielson, Semantics with Applications: A Formal Introduction. Wiley Professional Computing, pp.7, Wiley, 1992