

SDT 구현

2016112158 김희수



[날짜]

[회사 이름]

[회사 주소]

**1. 코드분석**

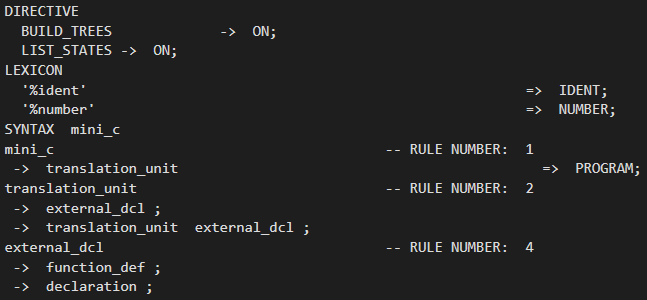
주어진 Parser.c에서 scanner를 사용하지만 scanner가 포함되어있지 않았다. 이전에 구현했던 LR파서에서 scanner부분을 가져와 붙였다.

parser()함수는 LR파서에서 구현했던 파서와 차이점만을 살펴보겠다. 이전과 달리 parser()함수의 데이터타입이 void에서 Node\*로 바뀌었다. Node구조체는 AST의 노드를 표현한다. Node\* parser()함수에선 semantic(~)다음에 ptr = buildTree(ruleName[ruleNumber], rightLength(ruleNumber)가 추가되었다.

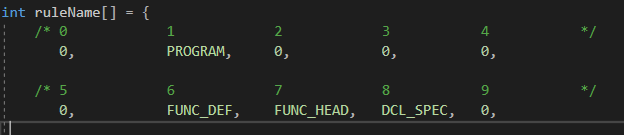
buildTree(int nodeNumber, int rhsLength)함수를 살펴보자. 이 함수는 AST를 생성하는 함수로 3개의 스텝으로 구성되어있다.

1. value stack에서 현재 reduce되는 생성 규칙의 rhs에 노드가 매달려 있는 인덱스를 찾는다. >> while(i<=sp && valueStack[i]==NULL) i++ 이 인덱스는 형제 노드로 연결할 노드의 첫번째 인덱스가 된다. >> start = i.
2. 형제노드를 연결하는 과정이다. 먼저 while(j<=sp&&valueStack[j]==NULL) j++ 로 형제노드로 연결할 노드의 다음 인덱스를 찾는다. 만약 다음 인덱스를 찾았으면, while(ptr->brother) ptr = ptr->brother; ptr->brother = valueStack[j]으로 형제노드를 연결한다. 그 후, 다음 인덱스를 찾기 위해 앞으로 이동한다 >> i = j. 이 과정의 끝으로 연결된 형제 노드들의 첫번째 노드의 포인터를 first에 저장한다
3. subtree를 만들고 son을 연결한다. 이 경우 의미있는 생성규칙이면 nonterminal 노드를 만든 후에 연결된 형제 노드를 son으로 연결하고 새로 만든 노드의 포인터를 복귀한다>> if(nodeNumber) 즉, if(ruleName[ruleNumber]) 절을 의미한다.

※ ruleName 배열은 “의미있는” 생성규칙과 그렇지 않은 생성규칙을 구분하는 배열이다. ruleName[0]이 의미가 없으면 0을 저장하고, ruleName[1]이 PROGRAM이라는 의미가 있으면 PROGRAM(열겨형)을 저장한다. 어떤 인덱스가 의미가 없는지, 의미가 있다면 무슨의미인지는 MiniC.lst을 보면서 결정한다.



위의 사진에서 RULE NUMBER 1은 의미(PROGRAM)가 있지만2,3,4,5는 의미가 없다. 이를 반영하면 다음과 같이 된다. RULE NUMBER 6,7,8은 의미가 있다.



즉, \*.lst파일에 나와있는 AST의 nodeName에 맞게 주어진 parser.c의 nodeNumber열거형과 nodeName배열을 수정해야하고 그에 맞게 ruleName을 확장해야한다. 올바르게 확장했다면 ruleName배열은 0~112까지의 인덱스를 갖는다.

다시 buildTree로 돌아가서 의미가 있으면 >> if(nodeNumber) nonterminal 노드를 생성하고 >> ptr->noderep = nonterm, 연결된 형제노드(fist)를 son으로 연결하고 >> ptr->son=first 이 노드를 리턴한다. 의미가 없으면 연결된 형제노드(first)만을 리턴한다.

parser()에서 사용되는 buildNode()함수는 토큰이 terminal인 경우 이 토큰으로 terminal 노드를 생성하는 함수이다.

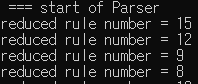
마지막으로 생성된 AST를 출력하는 printTree()는 먼저 parameter로 들어온 포인터의 노드를 출력하고 >> printNode(p, indent) 이 포인터가 terminal이면 이 포인터의 자식에 대해 printTree() 수행한다. 즉, printTree()함수는 root부터 시작해서 그 서브트리에 대해 재귀적으로 출력을 수행한다. printNode()함수는 parameter로 들어온 포인터가 terminal이냐 nonterminal이냐에 따라 그에 맞는 출력을 수행하는 함수이다.

**2. AST 생성 예시**

1) bubble.mc 파일

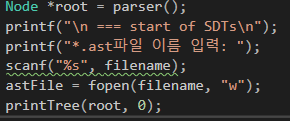
bubble.mc을 입력해서 scanner출력과 parser출력이 완료되면



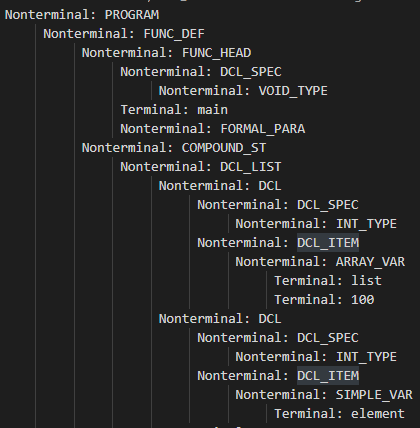


\*.ast파일의 이름을 입력받고 SDT를 실행시킨다.





Node \*root = parser()는 parser를 통해 만들어지는 AST의 첫번재 노드 root이다. 이 root를 printTree()에 파라미터로 넣어주면 AST출력이 시작된다.



그 결과 위와 같은 bubble.ast파일이 생성된다.