컴파일러개론 4주차 실습

RECURSIVE DESCENT PARSER

2024. 10. 04.

TA: 박정필

✉: 202350941@o.cnu.ac.kr

공지, 질문 방법

- 강의자료, 동영상
  - 사이버캠퍼스 업로드 예정
- 공지
  - 카카오톡 오픈채팅, 사이버캠퍼스
    - https://open.kakao.com/o/gkfF7JMg
      - 오픈프로필 사용 가능
      - 참여코드: 2024cp01
- 질문
  - 수업시간, 카카오톡 오픈채팅에 질문
  - 과제 관련 질문은 제출기한 전날까지만 가능 (당일 질문은 답변 X)
  - 이메일
    - 이메일 제목은 "[컴파일러개론][분반] ...", 제목 반드시 준수
    - 교수님 : eschough@cnu.ac.kr
    - TA : 202350941@o.cnu.ac.kr

목차

- 4주차 실습
  - Recursive Descent Parser
  - Recursive Descent Parser Pseudo Code & Example

- 4주차 과제
  - Recursive Descent Parser Implementation in Java

Recursive Descent Parser

- Recursive Descent Parser (재귀적 하강 구문 분석기)
  - Top Down 구문 분석기
  - recursive procedure의 집합으로 구성
  - 각 non-terminal과 terminal에 대한 procedure를 작성한 뒤 통합하는 방식으로 구현
  - LL 문법만 파싱 가능

Recursive Descent Parser Pseudo Code

1. Recursive Descent Parser

- 모든 terminal 심볼 a 에 대한 파서 코드

```
procedure pa;
begin
    if nextSymbol = ta   then get_nextSymbol
                         else error
end; /*pa*/
```

- 모든 non-terminal 심볼 A 에 대한 파서 코드

```
procedure pA;
begin
    case nextSymbol of
      LOOKAHEAD(A→X1X2...Xm):  for i:=1 to m do pXi;
      LOOKAHEAD(A→Y1Y2...Yn):  for i:=1 to n do pYi;
      LOOKAHEAD(A→Z1Z2...Zr):  for i:=1 to r do pZi;
      LOOKAHEAD(A→ε ):  ;
      otherwise : error
    end
end; /*pA*/
```

Compiler (컴파일러개론)
4. Syntax Analysis-Top Down
[5] 구문분석기를 만들기 (직접 구현)

충남대학교
조은선

LL(1) 파서 구현 방법

1. Recursive descent parser
   - recursion 이용
   - 각 non-terminal 마다 한개의 procedure를 둠
   - 장점 : 직관적, 쉽다.
   - 단점: 생성규칙이 바뀌면 구문 분석기를 고쳐야함
2. Predictive parser
   - 이론적으로 PDA (push down automata) 에 기반
   - 생성 규칙이 바뀌더라도 구문 분석기는 고치지 않음
     - 단지 구문 분석기의 행동을 나타내는 파싱 테이블만 수정

* 조은선 교수님 컴파일러개론 이론 자료

Recursive Descent Parser Example (1/4)

- CFG (Context Free Grammar)

$$\mathrm{S} \rightarrow aAb$$
$$A \rightarrow aS \mid b$$

Recursive Descent Parser Example (2/4)

$$S \rightarrow aAb$$
$$A \rightarrow aS \mid b$$

- Terminal
  - symbol a
  - symbol b
  - …

```
1 procedure pa;
2   begin
3   if nextSymbol = qa    then get_nextSymbol
4                         else error
5 end;
```

Procedure of terminal symbol a

```
1 procedure pb;
2   begin
3   if nextSymbol = qb    then get_nextSymbol
4                         else error
5 end;
```

Procedure of terminal symbol b

Recursive Descent Parser Example (3/4)

$$S \rightarrow aAb$$
$$A \rightarrow aS \mid b$$

- Non-terminal
  - symbol S
  - symbol A
  - …

```
1 procedure pS;
2   begin
3     if nextSymbol = qa  then
4                             begin pa; pA; pb end;
5                           else error
6 end;
```

Procedure of non-terminal symbol S

```
1 procedure pA;
2   begin
3     case nextSymbol of
4       qa: begin pa; pS end;
5       qb: pb;
6       otherwise: error
7   end
8 end;
```

Procedure of non-terminal symbol A

Recursive Descent Parser Example (4/4)

$$S \rightarrow aAb$$
$$A \rightarrow aS \mid b$$

- Main

```
1  begin
2    get_nextSymbol;
3    pS;
4    if nextSymbol = q$     then accept
5                           else error
6  end;
```

Procedure of main

Recursive Descent Parser Example 실행 결과 (1/5)

- 입력 문자열 : abb

```
CFG rules:
        1. S->aAb
        2. A->aS
        3. A->b
input: abb
char: a
char: b
char: b
char:

OK
```

Recursive Descent Parser Example 실행 결과 (2/5)

- 입력 문자열 : aaabbb

```
CFG rules:
        1. S->aAb
        2. A->aS
        3. A->b
input: aaabbb
char: a
char: a
char: a
char: b
char: b
char: b
char:

OK
```

Recursive Descent Parser Example 실행 결과 (3/5)

- 입력 문자열 : aaaaabbbb

```
CFG rules:
        1. S->aAb
        2. A->aS
        3. A->b
input: aaaaabbbb
char: a
char: a
char: a
char: a
char: a
char: b
char: b
char: b
char: b
char:

OK
```

Recursive Descent Parser Example 실행 결과 (4/5)

- 입력 문자열 : aaa

```
CFG rules:
        1. S->aAb
        2. A->aS
        3. A->b
input: aaa
char: a
char: a
char: a
char:

FAIL
```

Recursive Descent Parser Example 실행 결과 (5/5)

- 입력 문자열 : aaaaabbbbb

```
CFG rules:
        1. S->aAb
        2. A->aS
        3. A->b
input: aaaaabbbbb
char: a
char: a
char: a
char: a
char: a
char: b
char: b
char: b
char: b
char: b
FAIL
```

4주차 과제: Recursive Descent Parser (1/3)

- Java로 Recursive Descent Parser 만들기

- CFG:

$$S \rightarrow aA \mid bB$$
$$A \rightarrow aBb \mid bBb \mid cBb$$
$$B \rightarrow d \mid e \mid f$$

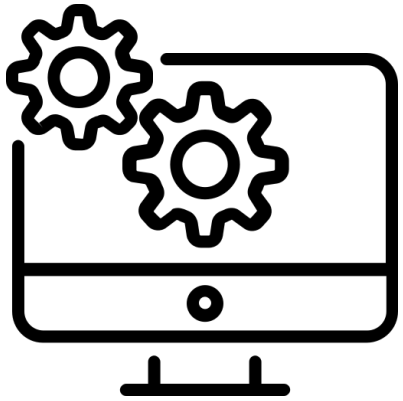
- 실행 예시:

```
youjeong@Plas-Youjeong:/mnt/c/Users/admin/Desktop/workspace/Compiler/week3/Java$ javac Main.java
youjeong@Plas-Youjeong:/mnt/c/Users/admin/Desktop/workspace/Compiler/week3/Java$ java Main
CFG:
        S -> aA ¦ bB
        A -> aBb ¦ bBb ¦ cBb
        B -> d ¦ e ¦ f

input: abdb
char: a
char: b
char: d
char: b
char:

OK
```

4주차 과제: Recursive Descent Parser (2/3)



1. Run Java Program

```
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> javac .\Main.java
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> java Main
```

```
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> javac .\Main.java
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> java Main
abdb
```

2. Console 입력 받기

```
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> javac .\Main.java
PS C:\Users\admin\Desktop\workspace\java-workspace\RDParser> java Main
abdb
OK
```

3. 결과 Console 출력

4주차 과제: Recursive Descent Parser (3/3)

- 과제 세부사항
  - 이전 실습 과제들 동일한 방식으로,
    - 사이버캠퍼스 4주차 과제 란에 제출
    - 여러 개 파일로 구현 가능, 하지만 package 사용 금지
    - main 함수 있는 파일 이름은 `Main.java`
    - `.java` 파일들 **`학번.zip`**으로 압축해서 제출
  - 코드에 주석 잘 달기 (필수 X)
  - 맨 앞에 CFG 출력 x (이해를 돕기 위한 출력일 뿐임)
  - input: 출력 x (이해를 돕기 위한 출력일 뿐임)
  - char: a ... 출력 x (이해를 돕기 위한 출력일 뿐임)
  - [OK] or [FAIL] 만 출력 (슬라이드 16 참고)

- 마감
  - **2024**년 **10**월 **11**일 금요일 **23**시 **59**분 (기한 엄수)
  - 추가 제출 기한 없음