프로그래밍 언어
YACC 실습

YACC란?

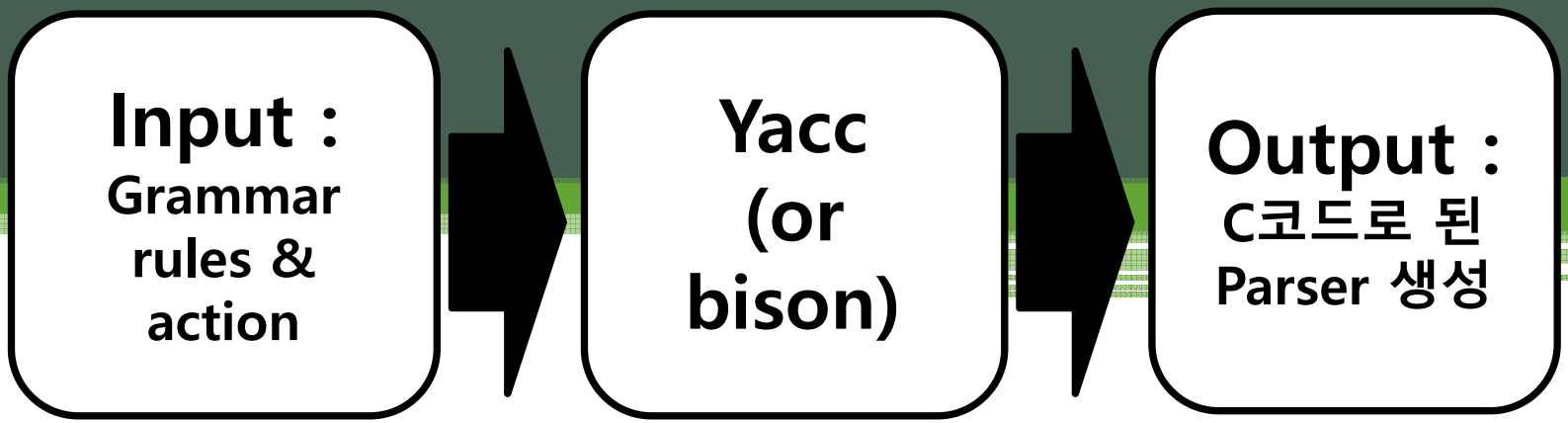
LEX와 같이 AT&T Bell 연구소에서 Mark Lesk가

UNIX 시스템의 유틸리티로 개발한 소프트웨어.

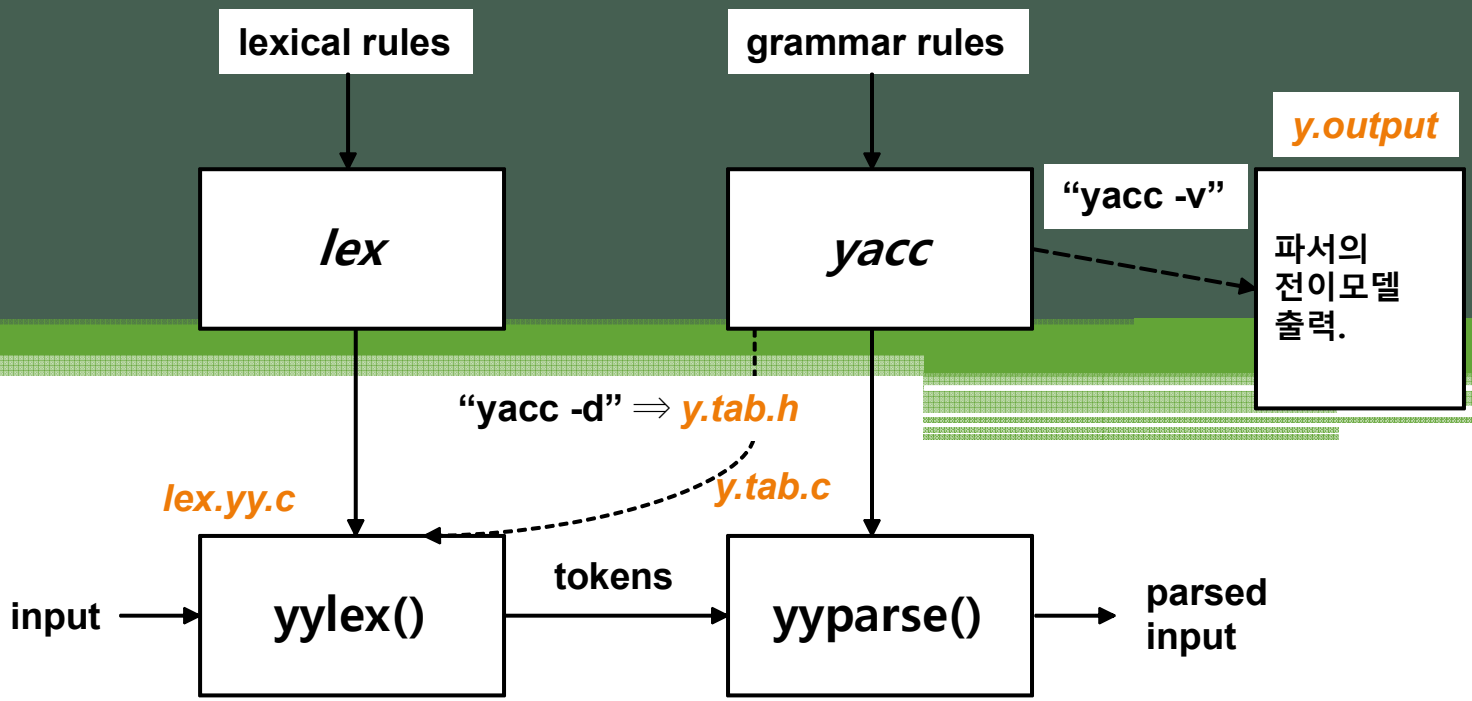
LEX가 정규표현식(Regular Expression)을 확인한다면,

YACC은 문법(Grammar)를 확인한다.

YACC란?


Input :
Grammar rules & action
Yacc (or bison)
Output :
C코드로 된 Parser 생성

YACC란?


lexical rules
grammar rules
y.output
lex
yacc
"yacc -v"
파서의
전이모델
출력.
"yacc -d" ⇒ y.tab.h
lex.yy.c
y.tab.c
input
yylex()
tokens
yyparse()
parsed
input

YACC 구조

- Yacc의 구조는 Lex와 같은 3개의 절로 구성

... 정의절 ...
%%
... 규칙절 ...
%%
...서브루틴...

YACC 구조_실습 1

int 변수를 카운팅하는 프로그램

YACC 구조_예제 1_Lex 코드 (정의절)

```
%{
#include <stdio.h>
#include "y.tab.h"
%}
```

"y.tab.h" : 프로그래머가 만든 Yacc의 토큰 번호 & 기타 변수의 정보를 가지고 있다.

YACC 구조_예제 1_Lex 코드 (정의절)

D는 숫자
L은 변수 첫 글자

YACC 구조_예제 1_Lex 코드 (규칙절)

```
"int"  {return INT;}
{L}({L}|{D})*    {return IDENTIFIER;}
{D}+      {return NUMBER;}
"="       {return '=';}
";"       {return ';';}
","       {return ',';}
[\t\v\n\f .]      {;}
%%
```

int, 변수가 가능한 문자, 숫자 ‘;’ 를 각각 반환하고 yacc.y파일에서 받아준다

YACC 구조_예제 1_Lex 코드 (규칙절)

```
int yywrap(void){
        return 1;
}
```

yywrap: EOF를만나면 호출
non zero - 끝
zero - 다음 파일을 더 받음

YACC 구조_예제 1_YACC 코드 (정의절)

```
%{
#include <stdio.h>
int yylex();
int intcount=0;
%}
```

intcount=0으로 초기화 해준다

YACC 구조_예제 1_YACC 코드 (정의절)

```
%token NUMBER INT IDENTIFIER
%start start_state
```

lex에서 받은 토큰들을 정의 및 start 설정

YACC 구조_예제 1_YACC 코드 (규칙절)

```
start_state : INT IDENTIFIER ';' {intcount++;}
            ;
```

int a;

YACC 구조_예제 2_YACC 코드 (규칙절)

```
start_state : INT IDENTIFIER '=' NUMBER ';'
{intcount++;}
                    ;
```

int a=숫자;

YACC 구조_예제 3_YACC 코드 (규칙절)

```
start_state : INT declaration_list ';'
            ;
declaration_list : declaration_list ',' declaration
            |  declaration
            ;
declaration : IDENTIFIER '=' NUMBER {intcount++;}
            | IDENTIFIER {intcount++;}
            ;
```

int a,b=0,c,d=1,g;

YACC 구조_예제 1_YACC 코드 (서브루틴절)

```
int main(void)
{
        printf("first int count is %d₩n₩n",
intcount);
        yyparse();
        printf("now int count is %d₩n",
intcount);
        return 0;
}
```

int a,b=0,c,d=1,g;

YACC 구조_예제 1

yyparse()에서 파싱에러가 발생하면 yyerror()호출
(주된 오류 syntax_error)

프로그래머의 역량에 따라 추가적인 오류처리를 할 수 있다.

```
void yyerror(const char *str)
{
    fprintf(stderr,"error : %s\n",str);
}
```

YACC 작동 원리

stack
&
State_machine

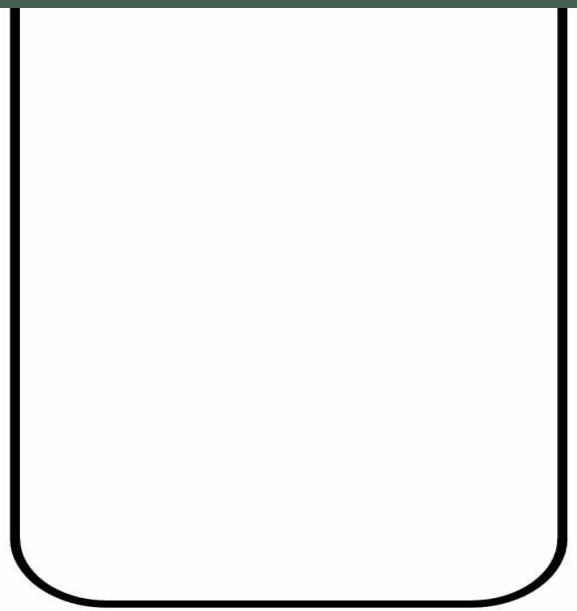
YACC 작동 원리_stack

1 + 10

NUMBER

ADD

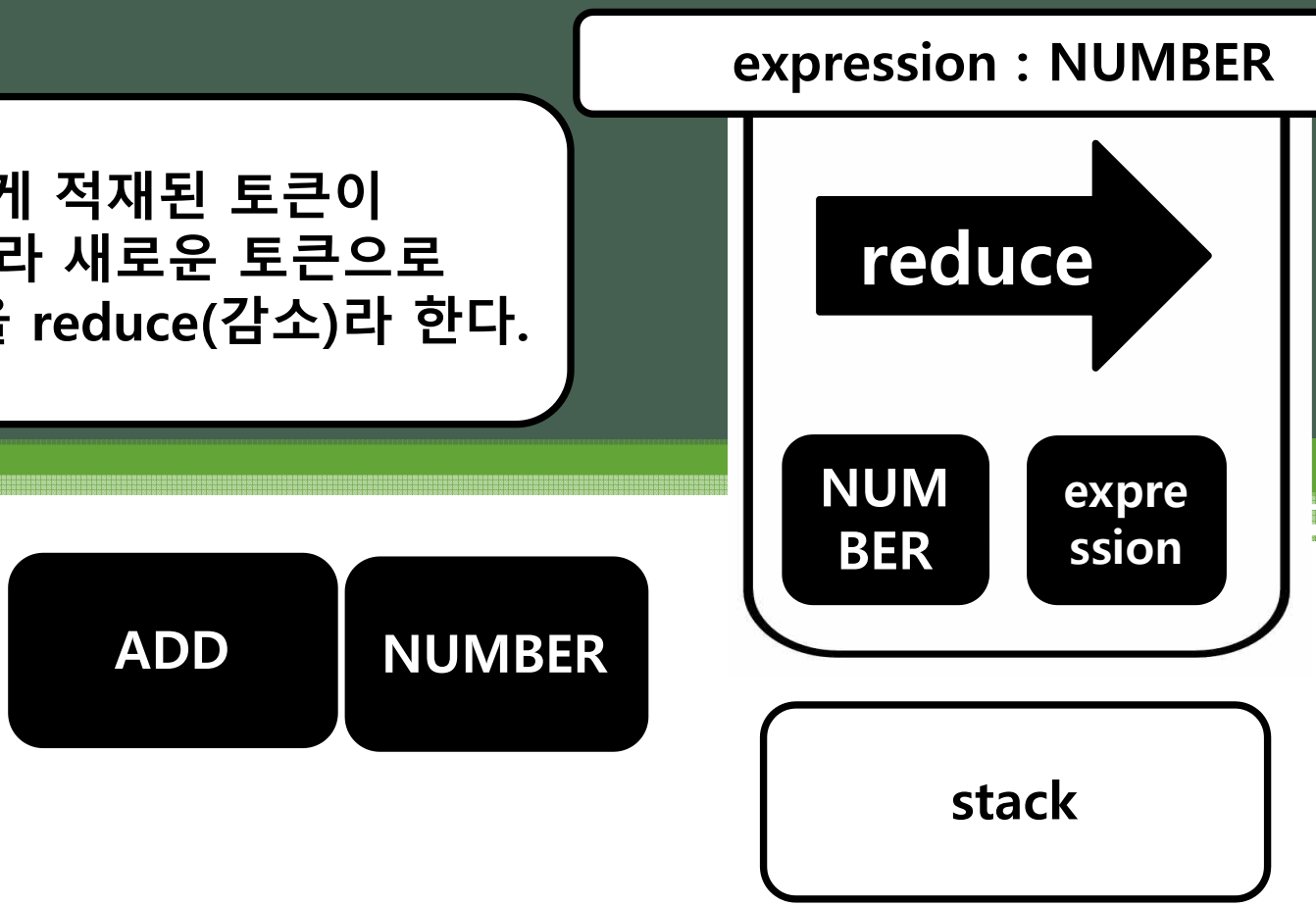
NUMBER



stack

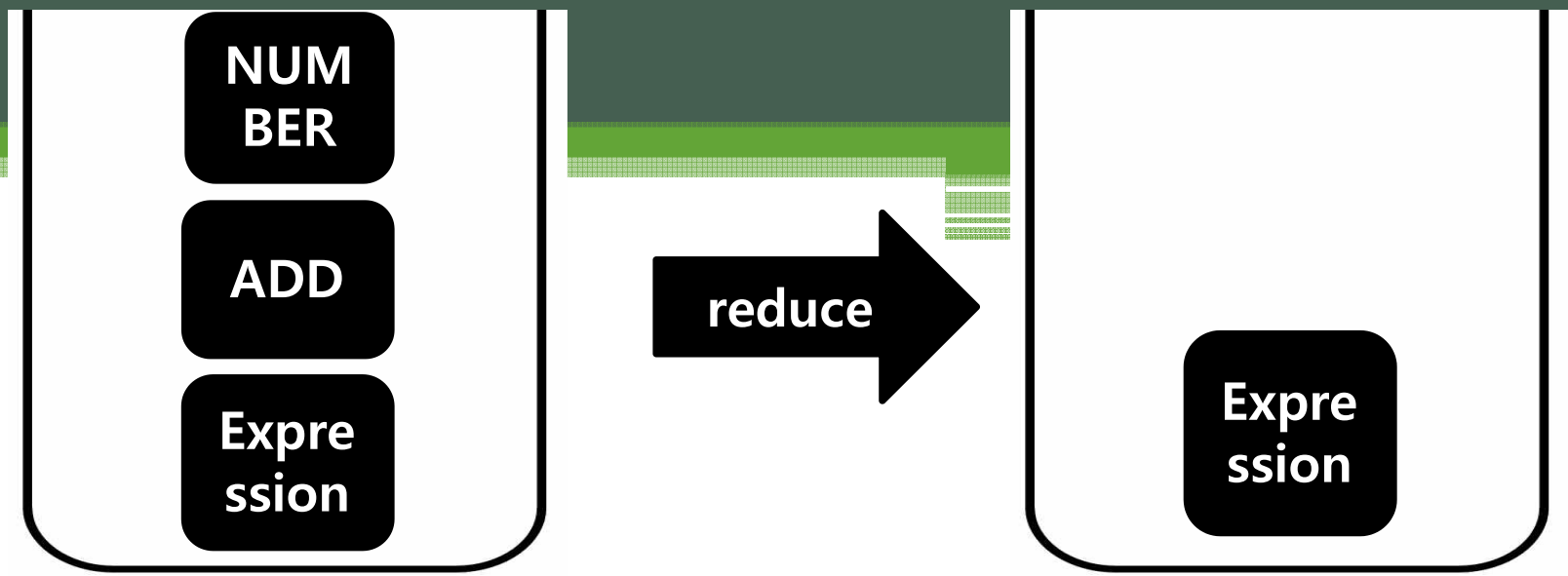
YACC 작동 원리_stack

이렇게 적재된 토큰이
룰에 따라 새로운 토큰으로
바뀌는 것을 reduce(감소)라 한다.



YACC 작동 원리_stack

expression :
expression ADD NUMBER

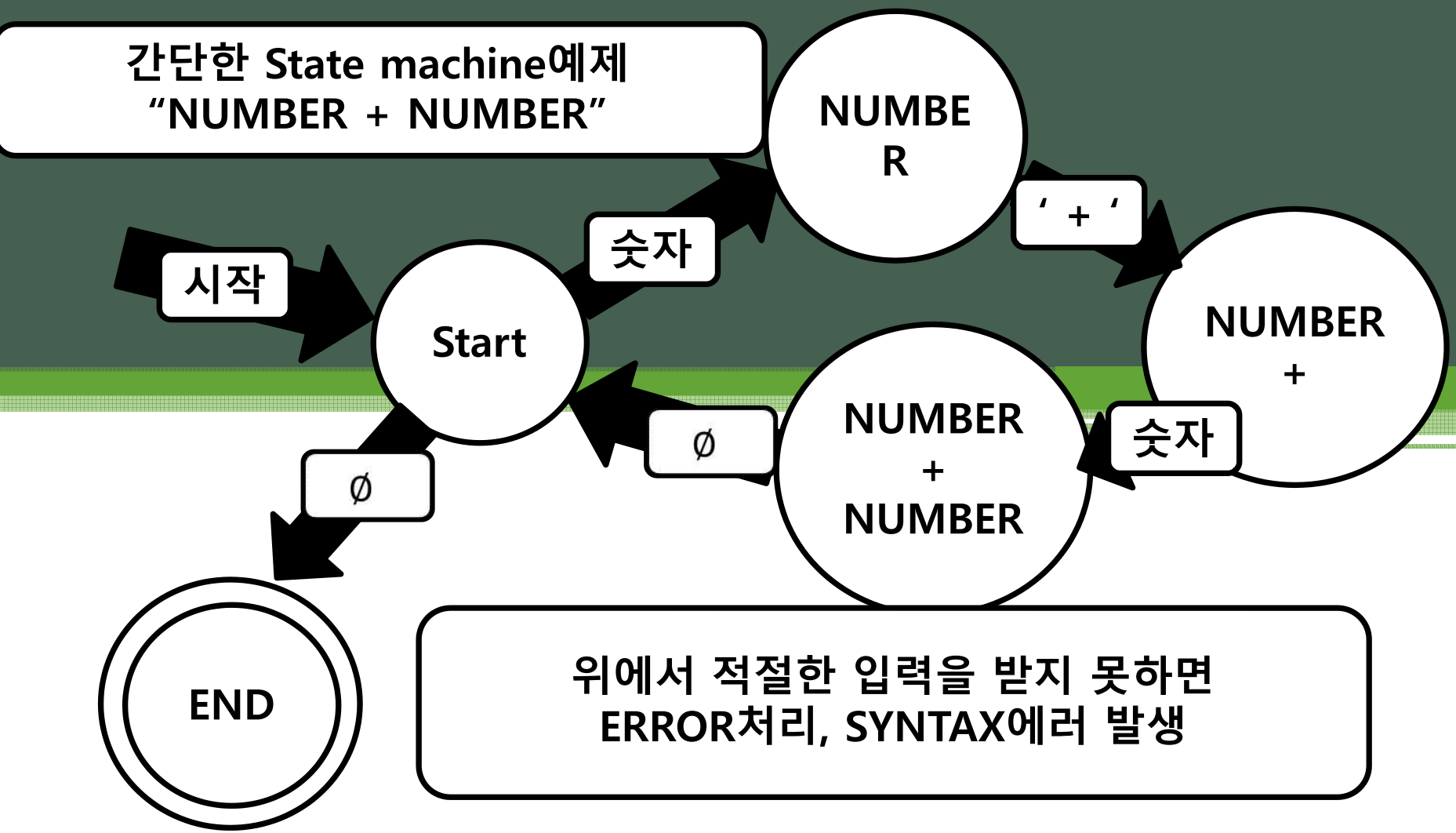


YACC 작동 원리_**State machine**

State Machine(상태 기계)?
여러 개의 상태와 이런 상태들간의
전이로 구성된 계산 모형.

stack의 내용과 입력을 받아 상태 전이

YACC 작동 원리_State machine



YACC 작동 원리

YACC은 HEAP, State machine을 이용하여 Parse한다.

YACC 작동 원리_관련오류

```
yacc: 4 shift/reduce conflicts
```

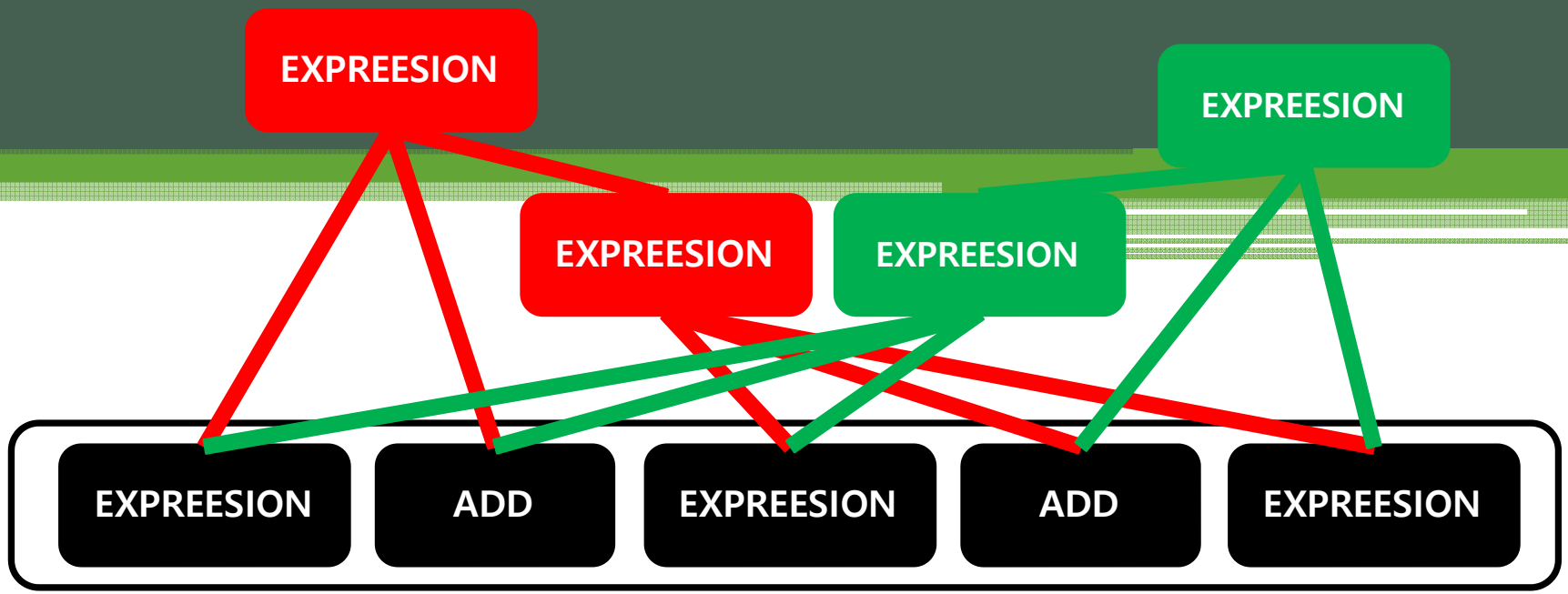
이를 “이동/감소” 충돌 이라고도 한다.
(비슷한 충돌로 “감소/감소”충돌이 있다.)

상태 전이도에서 새로운 token을 받느냐(shift)
stack에있는 내용을 symbol로 만드느냐(reduce)에
따라 다른 상태로 갈 수 있는 경우.(ambiguous)

YACC 작동 원리_관련오류

- reduce/reduce conflict

같은 입력에 대해 두 개의 TREE 가 존재할 수 있다 (모호성)



YACC 디버깅

- Yacc의 state 확인: 상태전이도 보여줌

$ yacc –d –v yacc.y

‘-v’를 추가하여 y.output을 생성한다.

컴파일 방법

```
$ lex lex.l
$ yacc –d yacc.y
$ cc y.tab.c lex.yy.c –o yacc
$ ./yacc < test.c
```

참고사이트

- documet: http://dinosaur.compilertools.net
- 한글설명 블로그 https://m.blog.naver.com/imisehi/150010452504
- 요약사이트 : https://www.epaperpress.com/lexandyacc/thy.html