문법과 언어

형식 언어

- **형식 언어(formal language)**
  - 잘 정의된(well-defined) 수학적 언어
  - 문장의 집합으로 정의
- **구성 요소**
  - 알파벳: 심볼들의 유한 집합
  - 스트링: 심볼을 연결한 것
- **연산자**
  - 연결(concatenation)
  - 합집합(union)

알파벳과 연산

- **알파벳 {a, b, c}**
  - 심볼: a, b, c
  - 스트링: ab, abc, bc, aa, bb, cb, a
    비교: ad, aξa (X)
- **연산자**
  - 연결: ab.bc = abbc
  - 합집합: {ab} ∪ {ac} = {ab, ac}
- **표기법**
  - a^2 = aa, a^3 = aaa
  - a^1 = a, a^0 = λ (또는 ε)

언어 = 문장의 집합

- **문장: 특정한 형태의 스트링**
  - 어떤 형태가 특정한 형태인가?
- **집합**
  - 집합의 표기
    - 원소 나열법
    - 조건 제시법
  - 유한 집합, 무한 집합
  - 무한 집합의 유한 표현?

언어의 예

- 알파벳이 **{a,b}**일 때
  - {a, aa, aaa}
  - {a^n | n은 짝수}
  - {$a^n b^n$ | $n \geq 1$}
  - {axb | $x \in$ {a,b}*}
- 알파벳이 **{0,1}**일 때
  - {x | $x \in$ {0,1}*, x는 ASCII 코드}
  - {x | $x \in$ {0,1}*, #(x)=65}

(aa)*

a(a+b)*b

#은 **2**진수를 **10**진수로 바꾸는 함수

*

❑ 심볼, 스트링

- ❖ 0번 이상 반복
- ❖ a*: a가 0번 이상 반복

  $= \{a^n \mid n \geq 0\} = \{\lambda, a, aa, aaa, \ldots\}$

❑ 집합

- ❖ 집합의 원소로 만들 수 있는 모든 스트링
- ❖ {a}* = a*
- ❖ {a, b}* = {λ, a, b, aa, ab, ba, bb, aaa, aab, …}

문법

- 무한 집합의 유한 표현
- 구성요소
  - 비단말 기호: N
  - 단말 기호: T
  - 생성 규칙: $\alpha \rightarrow \beta$
    - α는 β로 대치할 수 있다.
  - 시작 기호
- 문장
  - 문법으로부터 생성되면 문장이라 한다.

<로 >로 싸서 표기하기도 함

'와 '로 싸서 표기하기도 함

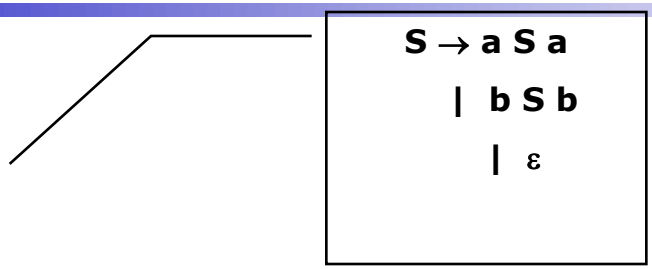
→대신에 ::=를 사용하기도 함

문맥 무관 문법

- 생성규칙의 제한
  - α를 비단말 기호로 제한
  - β는 단말기호와 비단말기호의 스트링
  - 즉, $A \rightarrow \beta$ 의 모양
- 왜 문맥무관인가?
  - $\gamma A \delta$는 항상 $\gamma \beta \delta$로 바뀔 수 있다.
  - γ나 δ에 상관 없이 ∴ 문맥무관

문법의 예

- 회문 언어
  - $S \rightarrow a\ S\ a$
  - $S \rightarrow b\ S\ b$
  - $S \rightarrow \varepsilon$



- 리스트 표기
  - $L \rightarrow a \mid (\ K\ )$
  - $K \rightarrow K\ ,\ L \mid L$

문장 만들기

- 회문 언어
  - $S \Rightarrow \varepsilon$
  - $S \Rightarrow aSa \Rightarrow aa$
  - $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$
- 리스트
  - $L \Rightarrow a$
  - $L \Rightarrow (K) \Rightarrow (L) \Rightarrow (a)$
  - $L \Rightarrow (K) \Rightarrow (K,L) \Rightarrow (K,L,L) \Rightarrow (L,L,L) \Rightarrow \Rightarrow \Rightarrow (a,a,a)$

문장 생성 연습

- **괄호**

  $P \rightarrow (\ P\ )\ P \mid \varepsilon$

  - 길이가 6인 스트링을 모두 구하시오.

- **수식**

  $E \rightarrow E + E \mid E * E \mid N$

  $N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5$

  - 길이가 5인 스트링을 모두 구하시오.

문법 만들기 연습

- 다음의 언어를 생성하는 문법은?
  - a^*
  - $\{a, b\}^*$
  - $\{a, aa, aaa\}$
  - $\{a^n \mid n$은 짝수$\}$
  - $\{a^n b^n \mid n \geq 1\}$
  - $\{axb \mid x \in \{a,b\}^*\}$

프로그래밍 언어

프로그래밍 언어

- 언어의 구성 요소
  - 구문(syntax): 요소들이 결합하는 방법
  - 의미(semantics): 요소들의 뜻
  - 실용(pragmatics): 언어의 부수적인 옵션
- 언어의 존재
  - 구문 + 의미
  - 컴파일러

실행문을 위한 문법

- 프로그래밍 언어에서의 실행문

```
<Stmt> ::= <AsgnStmt>     | <IfStmt>
         | <WhileStmt>    | <ForStmt>
         | <CallStmt>     | <RtrnStmt>
         | <CompStmt>
<AsgnStmt> ::= <Var> = <Expr>
<IfStmt>   ::= if ( <Cond> ) then <Stmt>
             | if ( <Cond> ) then <Stmt>
                             else <Stmt>
<WhileStmt> ::= while ( <Cond> ) <Stmt>
<ForStmt>  ::= for ( <Var> = <Expr> to <Expr> )
               <Stmt>
<CallStmt> ::= call <ProcName> ( <ParamList> )
<RtrnStmt> ::= return ( <Expr> )
<CompStmt> ::= begin <StmtList> end
<StmtList> ::= <StmtList> ; <Stmt> | <Stmt>
```

식을 위한 문법

- 프로그래밍 언어에서의 조건식과 수식

```
<Cond> ::= <Cond> & <Rel> | <Cond> '|' <Rel>
         | ! <Rel>         | <Rel>
<Rel>  ::= <Expr> < <Expr> | <Expr> <= <Expr>
         | <Expr> > <Expr> | <Expr> >= <Expr>
         | <Expr> = <Expr> | <Expr> != <Expr>

<Expr> ::= <Expr> + <Term> | <Expr> - <Term>
         | <Term>
<Term> ::= <Term> * <Fact> | <Term> / <Fact>
         | <Fact>
<Fact> ::= <Var> | <Number> | <FuncCall>
         | - <Fact> | ( <Expr> )
<FuncCall>  ::= <FuncName> ( <ParamList> )
<ParamList> ::= <ExprList> |
<ExprList>  ::= <ExprList> , <Expr> | <Expr>
```

ToyPL 언어의 문법

```
<Program> ::= program <Name> ;
              <SubPgmList>
              main <VarDecl> <CompStmt> .
<VarDecl> ::= var <DclList> ; |
<DclList> ::= <DclList> ; <Decl> | <Decl>
<Decl>    ::= <VarList> : <Type>
<VarList> ::= <VarList> , <Var> | <Var>
<Type>    ::= int | long
<Var>     ::= <Name>
<SubPgmList> ::= <SubPgmList> <SubPgm> |
<SubPgm>     ::= <ProcDecl> | <FuncDecl>
<ProcDecl>   ::= proc <ProcName> ( <FormParam> )
                 <VarDecl> <CompStmt>
<FuncDecl>   ::= func <FuncName> ( <FormParam> )
                 returns ( <Type> )
                 <VarDecl> <CompStmt>
<FormParam>  ::= <DclList> |
```

구성요소의 의미

- 형식매개변수(**<FormParam>**)는 다시 선언될 필요 없음
- 함수선언에서 **returns**는 반환 값의 형임
- **return** 문은 함수선언에서만 필요
- 프로그램의 실행은 **main**부터 시작
- **int**는 **4-byte, long**는 **8-byte**의 정수

※ **<IfStmt>**는 모호

ToyPL 언어의 어휘

- 키워드
  - program, main, proc, func, returns, var, int, long, if, then, else, while, for, to, call, return, begin, end
- 연산자 및 구분자
  - 연산자: = & | ! < <= > >= != + - * /
  - 구분자: ; : ( ) , .
- 이름과 숫자
  - 이름: 영문자로 시작, 영문자와 숫자가 이어짐
  - 숫자: 정수의 10진수 표기
- 주석
  - /*와 */ 사이의 모든 문자열

내장 함수

- 입출력은 **ToyPL**에서 정의되지 아니함
- 입출력은 내장함수에서 처리함
  - read(): 키보드로부터 숫자를 입력하여 반환
  - print(x): x를 화면에 출력, x는 변수 또는 상수

ToyPL 프로그래밍

- 가장 간단한 프로그램을 작성하시오.
  - 문법에 맞게 작성 (의미는 고려 안함)
- 필요없이 복잡한 프로그램 조각을 작성하시오.
  - 예: x+0+0+0+0+0+0+0+0+0+y
- 계승(factorial) 함수를 작성하시오
  - 반복(iterative)을 이용하여 작성
  - 순환(recursion)을 이용하여 작성

가장 간단한 프로그램

```
program p
main
begin
    a = 1
end .
```

불필요한 중복

- **begin begin begin a = 1 end end end**
- **(((((a+1)))))**
- **-------1**

계승함수 - “반복” 버전

계승함수 - “순환” 버전

문법 오류 찾기

```
program Sample
    proc Fact (n : long)
      var m : integer;
      begin
        m = n - 1;
        call Fact(m);
      end;
     main
    var a, b : integer;
    begin
      a = 0;
      call Fact(a);
    end .
```

로봇 제어 언어

- **로봇 제어 프로그램**
  - 프로그램으로 로봇을 제어하고자 함
  - 로봇 제어 언어로 작성된 프로그램
- **로봇의 동작**
  - 프로그램에 지시된 대로 움직임

로봇 제어 언어의 문법

```
RCP ::= robot at POS in the board { BRD }
        the control sequence is CTSQ .
BRD ::= size = NUM x NUM ;
        blocks = BLK ;
        destination = POS
BLK ::= POS , BLK
      | POS
POS ::= ( NUM , NUM )
CTSQ ::= CNTL CTSQ
       | CNTL
CNTL ::= go-ahead NUM
       | go-back NUM
       | turn-left
       | turn-right
```

언어의 의미

❑ 언어의 의미

- ❖ size는 판의 크기로 (가로크기)x(세로크기)임
- ❖ blocks는 장애물의 좌표임
- ❖ 판의 좌표는 왼쪽 아래가 (1,1) 임
- ❖ NUM은 1보다 크거나 같은 정수
- ❖ 초기에 로봇은 위쪽(북쪽)을 바라보고 있음

로봇 제어 프로그래밍

❑ 로봇 제어 프로그래밍

- ❖ 다음 그림과 같은 배치에서 목적지까지 도달하기 위하여 로봇이 움직여야 하는 방법을 프로그래밍하시오.