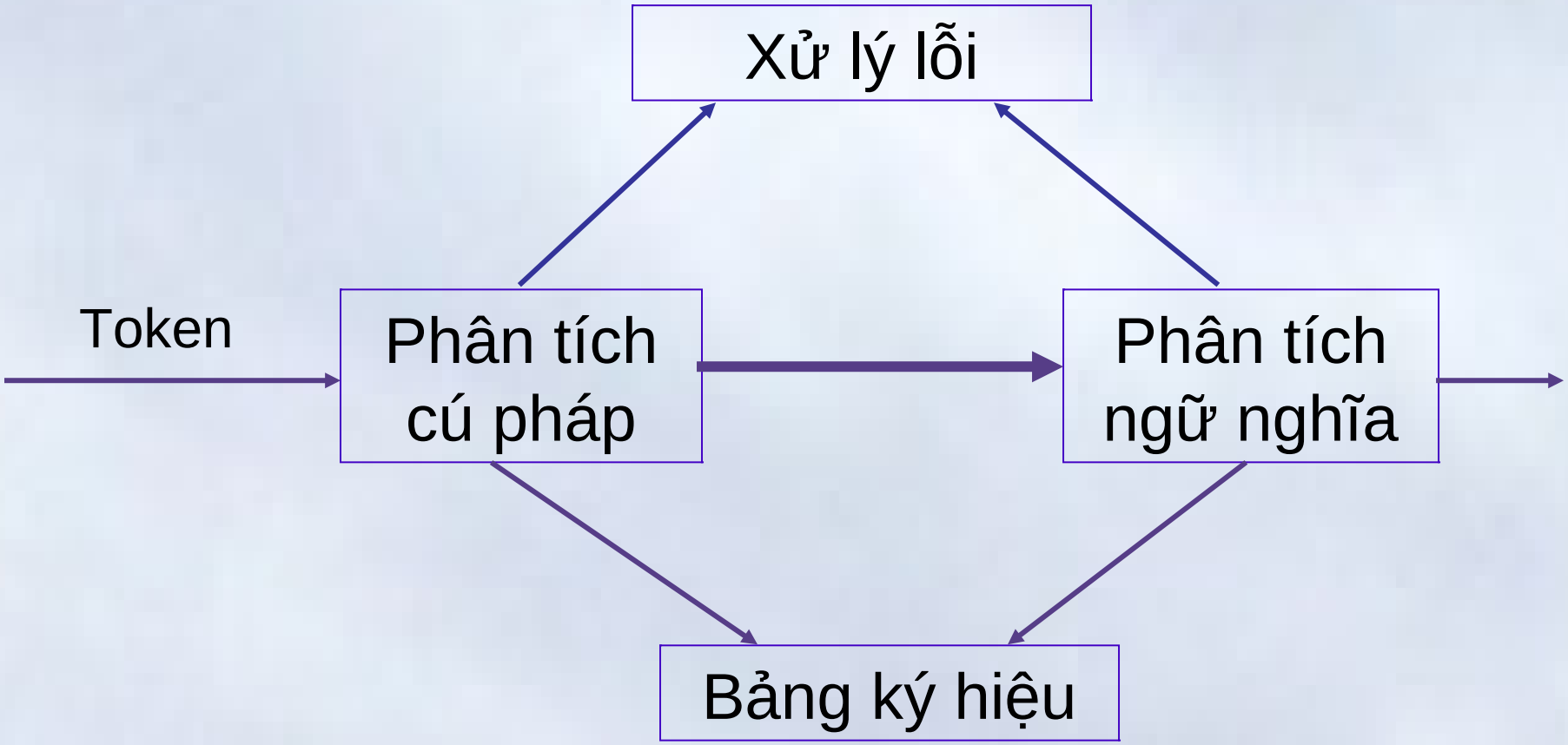
Thực hành
CHƯƠNG TRÌNH DỊCH
Bài 3: Phân tích cú pháp

Phạm Đăng Hải
haipd@soict.hust.edu.vn

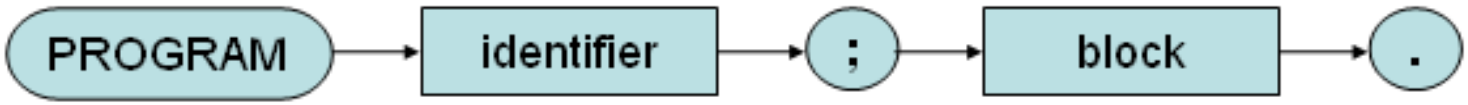
Nhiệm vụ của bộ ptcp



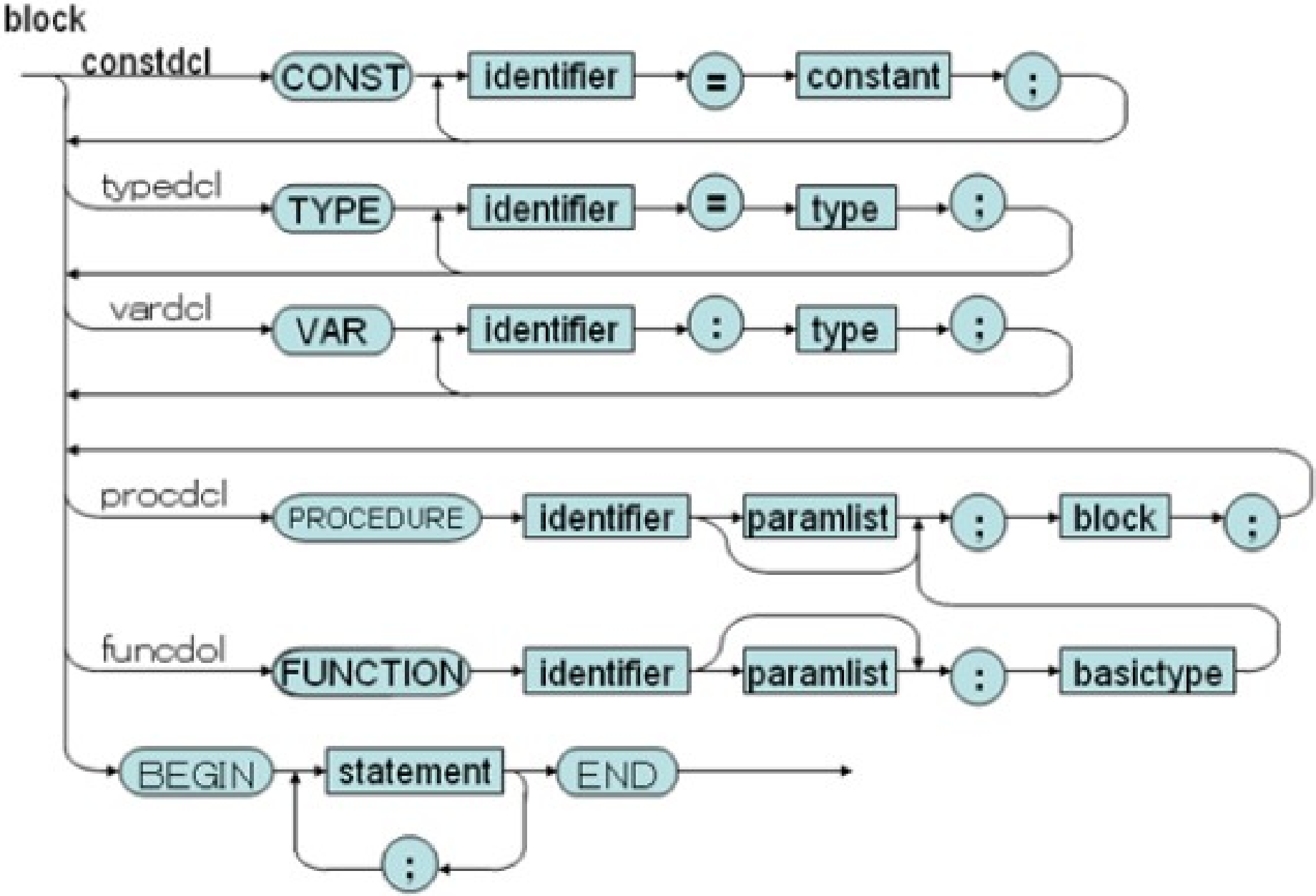
- Kiểm tra cấu trúc ngữ pháp của chương trình
- Kích hoạt bộ phân tích ngữ nghĩa và bộ sinh mã

Sơ đồ cú pháp của KPL

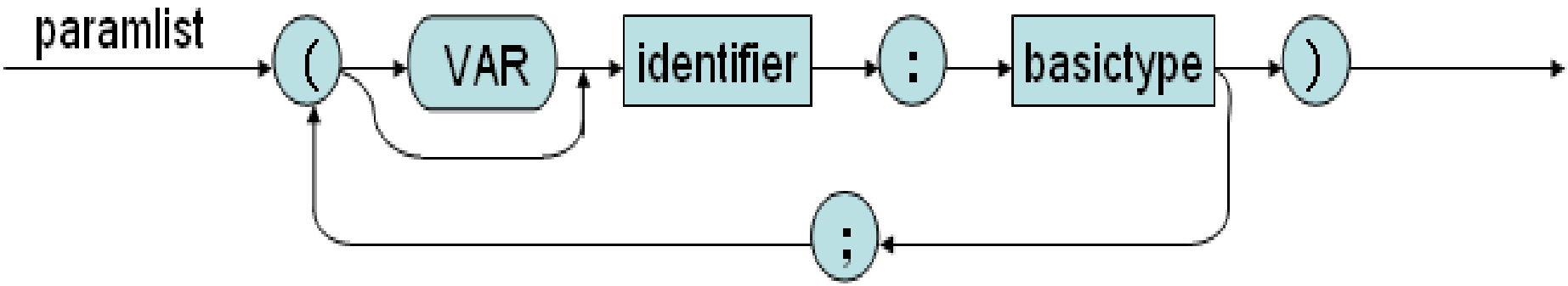
program

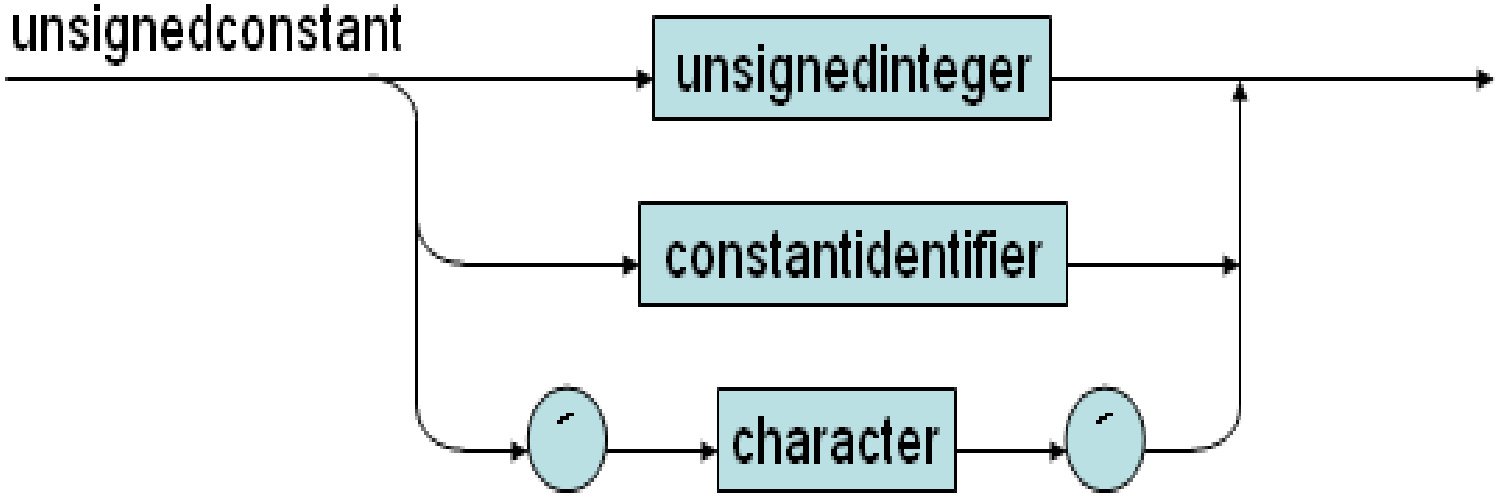


Sơ đồ cú pháp cho ngôn ngữ KPL

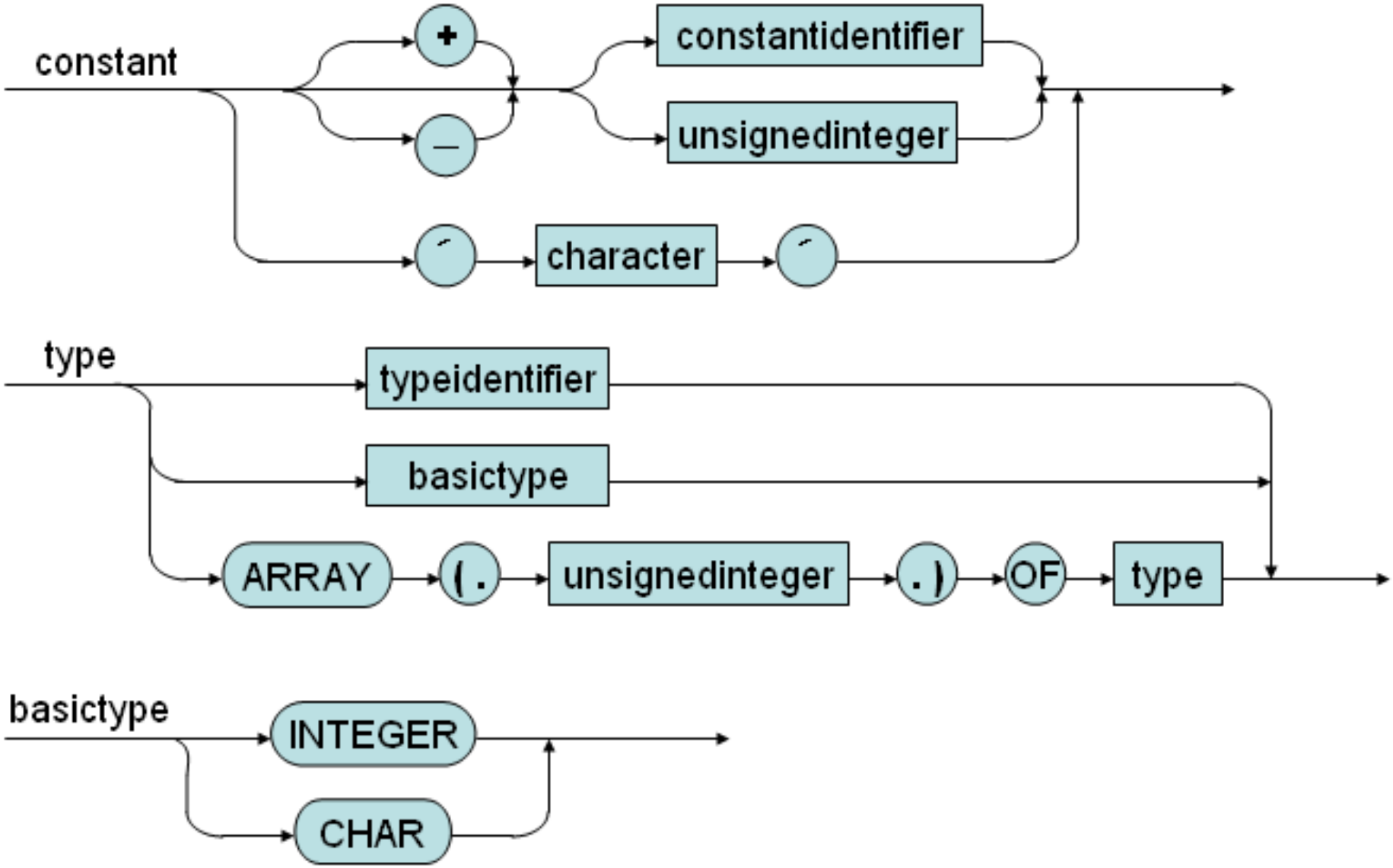


Sơ đồ cú pháp cho ngôn ngữ KPL

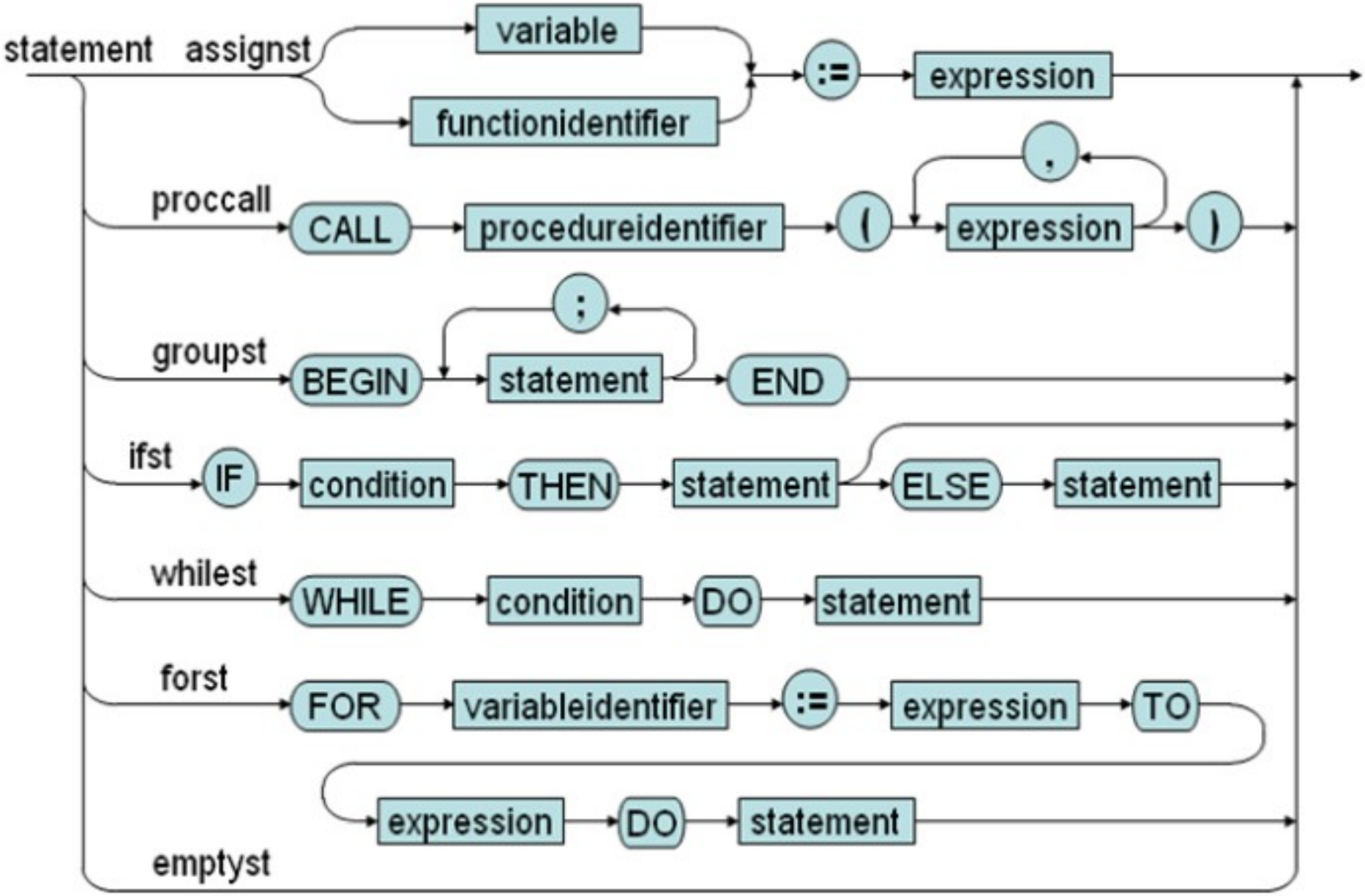




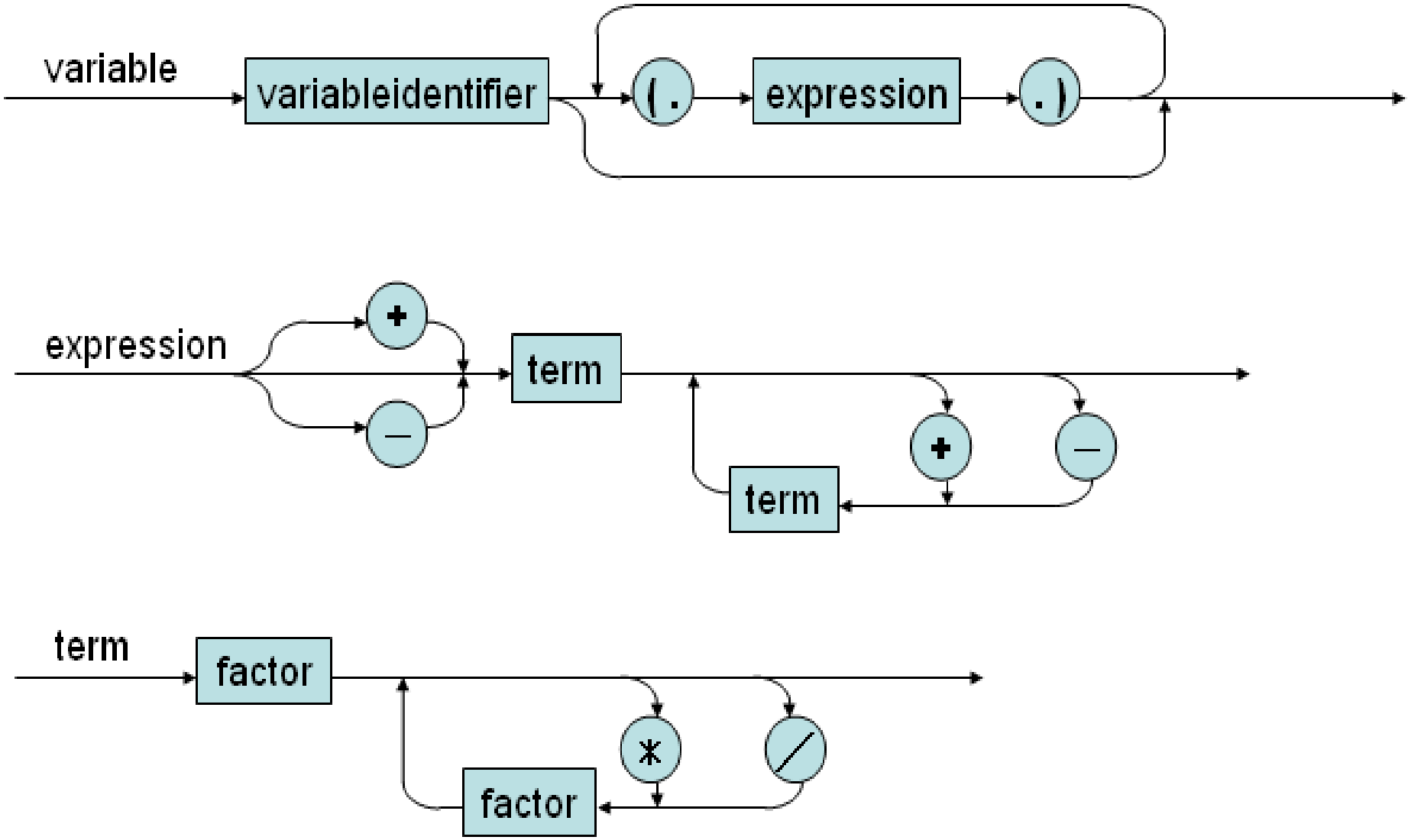
Sơ đồ cú pháp cho ngôn ngữ KPL



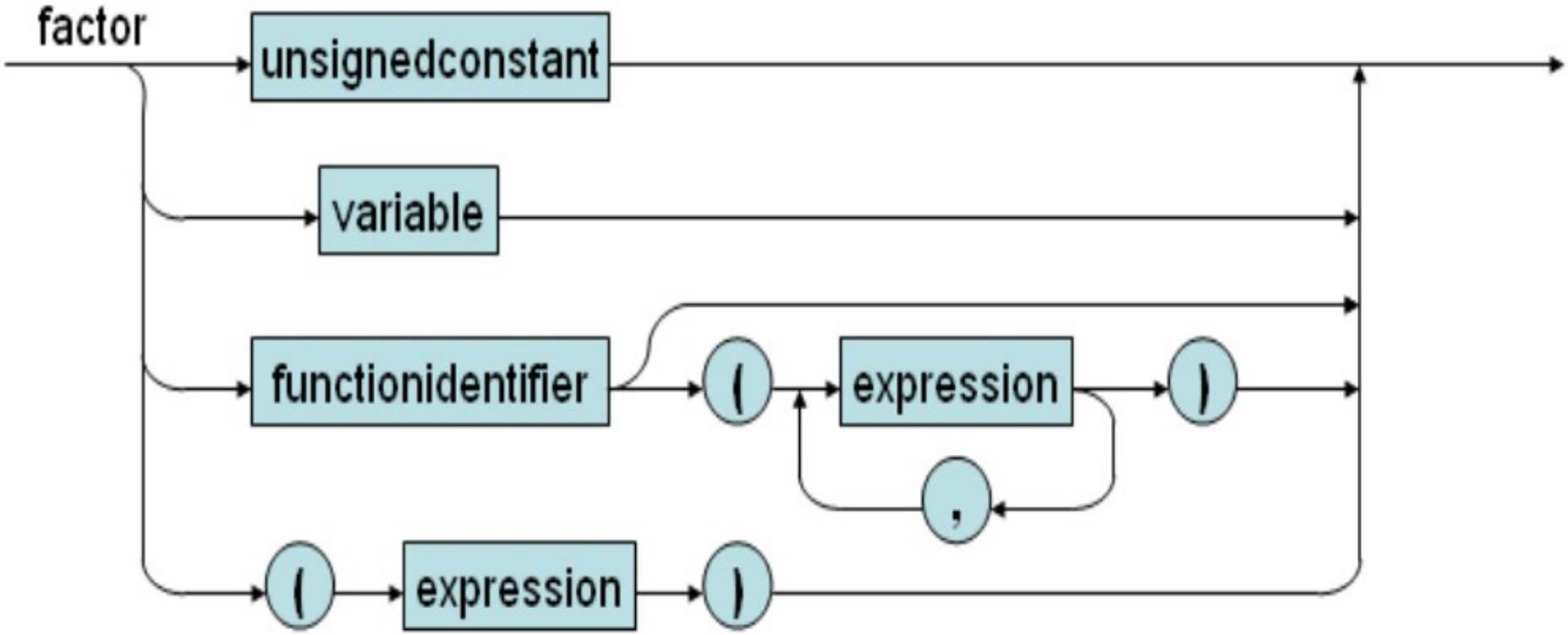
Sơ đồ cú pháp cho ngôn ngữ KPL

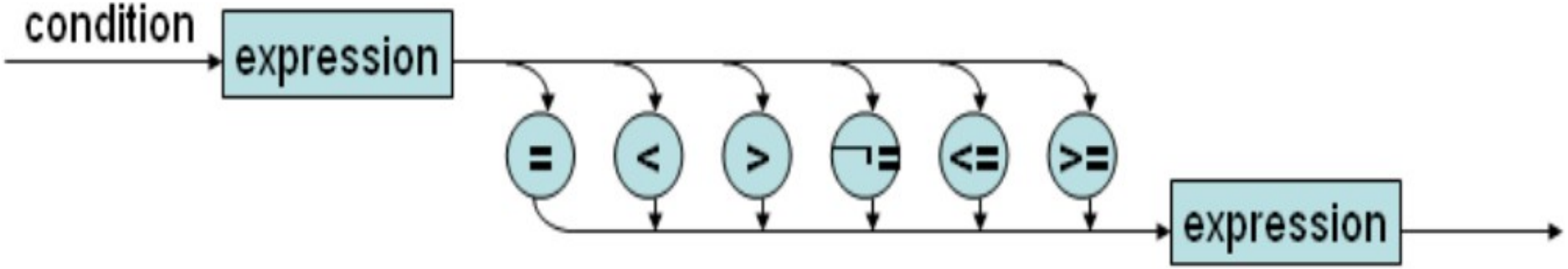


Sơ đồ cú pháp cho ngôn ngữ KPL



Sơ đồ cú pháp cho ngôn ngữ KPL





Sơ đồ cú pháp cho ngôn ngữ KPL

identifier

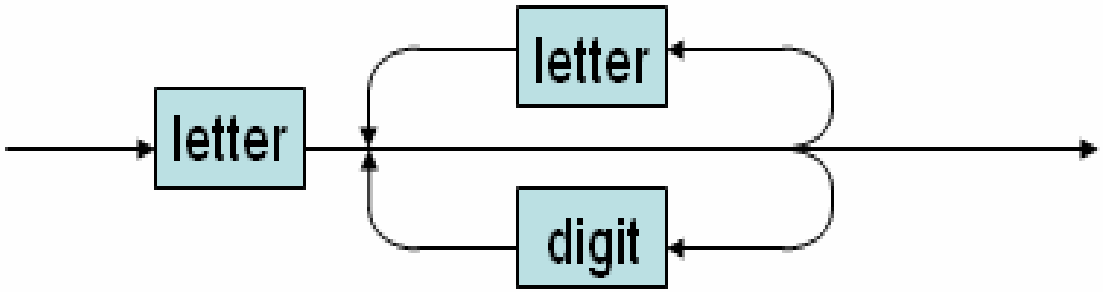
constantidentifier

typeidentifier

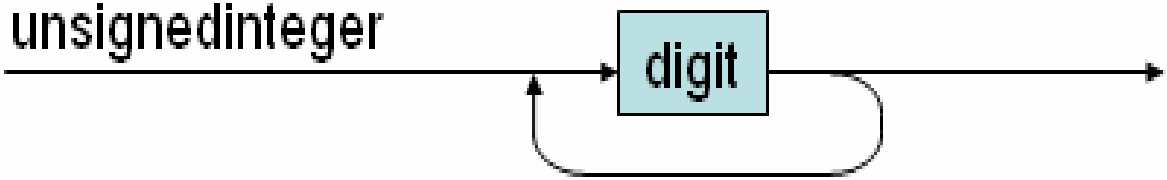
procedureidentifier

functionidentifier

variableidentifier



unsignedinteger



Chuyển đổi sang văn phạm BNF

- Thực hiện loại bỏ đệ quy trái
- Thực hiện nhân tử trái

Văn phạm BNF

1. Prog ::= **KW_PROGRAM Ident SB_SEMICOLON** Block **SB_PERIOD**
2. Block ::= **KW_CONST** ConstDecl ConstDecls Block2
3. Block ::= Block2
4. Block2 ::= **KW_TYPE** TypeDecl TypeDecls Block3
5. Block2 ::= Block3
6. Block3 ::= **KW_VAR** VarDecl VarDecls Block4
7. Block3 ::= Block4
8. Block4 ::= SubDecls Block5
9. Block5 ::= **KW_BEGIN** Statements **KW_END**

Văn phạm BNF

10. ConstDecls::= ConstDecl ConstDecls
11. ConstDecls::= ε
12. ConstDecl ::= **Ident SB_EQUA**L Constant SB_SEMICOLON
13. TypeDecls ::= TypeDecl TypeDecls
14. TypeDecls ::= ε
15. TypeDecl ::= **Ident SB_EQUAL** Type **SB_SEMICOLON**
16. VarDecls ::= VarDecl VarDecls
17. VarDecls ::= ε
18. VarDecl ::= **Ident SB_COLON** Type **SB_SEMICOLON**
19. SubDecls ::= FunDecl SubDecls
20. SubDecls ::= ProcDecl SubDecls
21. SubDecls ::= ε

Văn phạm BNF

22. FunDecl ::= **KW_FUNCTION Ident** Params **SB_COLON** BasicType **SB_SEMICOLON** Block **SB_SEMICOLON**
23. ProcDecl ::= **KW_PROCEDURE Ident** Params SB_SEMICOLON Block SB_SEMICOLON
24. Params ::= **SB_LPAR** Param Params2 **SB_RPAR**
25. Params ::= ε
26. Params2 ::= **SB_SEMICOLON** Param Params2
27. Params2 ::= ε
28. Param ::= **Ident SB_COLON** BasicType
29. Param ::= **KW_VAR Ident SB_COLON** BasicType

Văn phạm BNF

30. Type ::= **KW_INTEGER**
31. Type ::= **KW_CHAR**
32. Type ::= TypeIdent
33. Type ::= **KW_ARRAY SB_LSEL Number SB_RSEL KW_OF** Type
34. BasicType ::= **KW_INTEGER**
35. BasicType ::= **KW_CHAR**
36. UnsignedConstant ::= **Number**
37. UnsignedConstant ::= ConstIdent
38. UnsignedConstant ::= ConstChar
39. Constant ::= **SB_PLUS** Constant2
40. Constant ::= **SB_MINUS** Constant2
41. Constant ::= Constant2
42. Constant ::= ConstChar

Văn phạm BNF

43. Constant2::= ConstIdent
44. Constant2::= **Number**
45. Statements ::= Statement Statements2
46. Statements2 ::= **SB_SEMICOLON** Statement Statement2
47. Statements2 ::= ε
48. Statement ::= AssignSt
49. Statement ::= CallSt
50. Statement ::= GroupSt
51. Statement ::= IfSt
52. Statement ::= WhileSt
53. Statement ::= ForSt
54. Statement ::= ε

Văn phạm BNF

55. AssignSt ::= Variable **SB_ASSIGN** Expession
56. AssignSt ::= FunctionIdent **SB_ASSIGN** Expression
57. CallSt ::= **KW_CALL** ProcedureIdent Arguments
58. GroupSt ::= **KW_BEGIN** Statements **KW_END**
59. IfSt ::= **KW_IF** Condition **KW_THEN** Statement ElseSt
60. ElseSt ::= **KW_ELSE** statement
61. ElseSt ::= ε
62. WhileSt ::= **KW_WHILE** Condition **KW_DO** Statement
63. ForSt ::= **KW_FOR** VariableIdent **SB_ASSIGN** Expression **KW_TO** Expression **KW_DO** Statement

Văn phạm BNF

64. Arguments ::= **SB_LPAR** Expression Arguments2 **SB_RLAR**
65. Arguments ::= ε
66. Arguments2::= **SB_COMMA** Expression Arguments2
67. Arguments2::= ε
68. Condition ::= Expression Condition2
69. Condition2::= **SB_EQ** Expression
70. Condition2::= **SB_NEQ** Expression
71. Condition2::= **SB_LE** Expression
72. Condition2::= **SB_LT** Expression
73. Condition2::= **SB_GE** Expression
74. Condition2::= **SB_GT** Expression

Văn phạm BNF

75. Expression ::= SB_PLUS Expression2
76. Expression ::= SB_MINUS Expression2
77. Expression ::= Expression2
78. Expression2 ::= Term Expression3
79. Expression3 ::= SB_PLUS Term Expression3
80. Expression3 ::= SB_MINUS Term Expression3
81. Expression3 ::= ε

Văn phạm BNF

82. Term ::= Factor Term2
83. Term2 ::= SB_TIMES Factor Term2
84. Term2 ::= SB_SLASH Factor Term2
85. Term2 ::= ε
86. Factor ::= UnsignedConstant
87. Factor ::= Variable
88. Factor ::= FunctionApptication
89. Factor ::= SB_LPAR Expression SB_RPAR
90. Variable ::= VariableIdent Indexes
91. FunctionApplication ::= FunctionIdent Arguments
92. Indexes ::= SB_LSEL Expression SB_RSEL Indexes
93. Indexes ::= ε

Văn phạm KPL

- Tính FIRST và FOLLOW cho các ký hiệu không kết thúc?
- Về cơ bản KPL là một ngôn ngữ LL(1)
  - Có thể phân tích bởi phương pháp đệ quy trên xuống
- Thiết kế một bộ phân tích đệ quy trên dưới
  - Token ***lookAhead*** *//Token xem trước*
  - Duyệt ký hiêu kết thúc
  - Duyệt ký hiệu không kết thúc

Xây dựng Parser

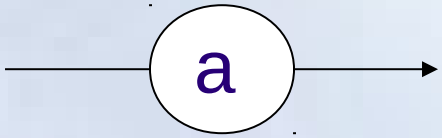
STT	Tên tệp	Nội dung
1	Makefile	Project
2	scanner.c	Tệp chính
3	reader.h, reader.c	Đọc mã nguồn
4	charcode.h, charcode.c	Phân loại ký tự
5	token.h, token.c	Phân loại và nhận dạng token, từ khóa
6	error.h, error.c	Thông báo lỗi
7	parser.c parser.h	Duyệt các cấu trúc chương trình nguồn
8	main.c	Chương trình chính

Xem trước một token

```
Token *currentToken;   // Token vừa đọc
Token *lookAhead;      // Token xem trước
void scan(void) {
  Token* tmp = currentToken;
  currentToken = lookAhead;
  lookAhead = getValidToken();//Thêm vào bộ pttv
  free(tmp);
}
```

```
Token* getValidToken(void) {
  Token *token = getToken();
  while (token->tokenType == TK_NONE) {
    free(token);
    token = getToken();
  }
  return token;
}
```

Duyệt ký hiệu kết thúc

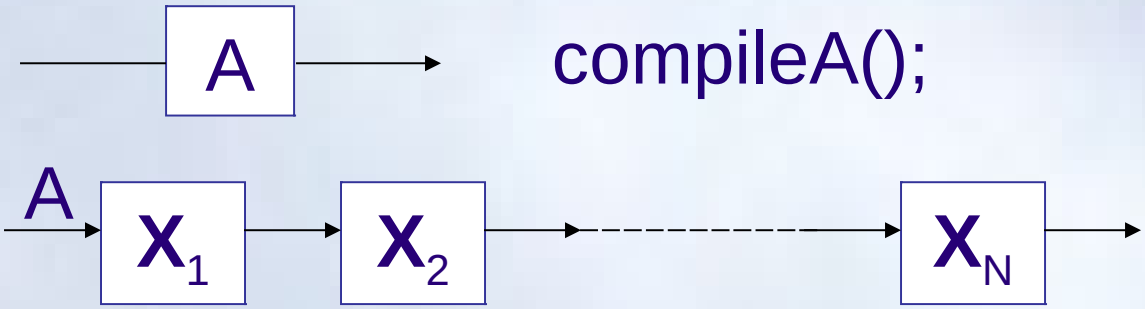


If Ch = a Then nextCh

Else Error (Đang đợi ký hiệu a)

```
void eat(TokenType tokenType) {
  if (lookAhead->tokenType == tokenType) {
      printToken(lookAhead);
      scan();
  } else missingToken(tokenType,
    lookAhead->lineNo, lookAhead->colNo);
}
```

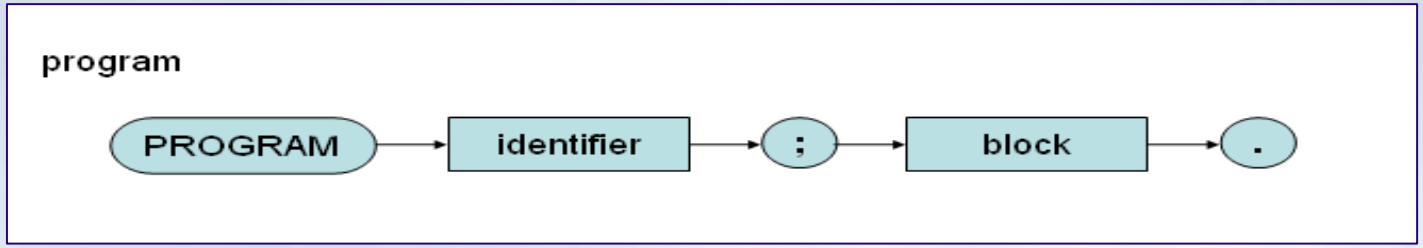
Duyệt ký hiệu không kết thúc và một sơ đồ




```
void compileA(){
    assert("parsing a A..");
    T(X1);
    T(X2); .....
    T(XN);
    assert("A parsed..");
}
```

```
//Thông báo quá trình
void assert(char *msg) {
  printf("%s\n", msg);
}
```

Duyệt sơ đồ→Ví dụ sơ đồ Program




```
void compileProgram(void) {
    assert("Parsing a Program ....");
    eat(KW_PROGRAM);
     eat(TK_IDENT);
     eat(SB_SEMICOLON);
    compileBlock();
    eat(SB_PERIOD);
  assert("Program parsed!");
}
```

Kích hoạt bộ ptcp

```
int compile(char * fileName) {
    if (openInputStream(fileName) == IO_ERROR)
            return IO_ERROR;
    currentToken = NULL;
    lookAhead = getValidToken();
    compileProgram();
    free(currentToken);
    free(lookAhead);
    closeInputStream();
  return IO_SUCCESS;
}
```

Ví dụ Statement

FIRST(Statement) = {TK_IDENT, KW_CALL, KW_BEGIN, KW_IF, KW_WHILE, KW_FOR, ε}

FOLLOW(Statement) = {SB_SEMICOLON, KW_END, KW_ELSE}

/* Predict parse table for Expression */

Input	Production	Action
TK_IDENT	→ Statement ::= AssignSt	→ compileAssignSt();
KW_CALL	→ Statement ::= CallSt	→ compileCallSt();
KW_BEGIN	→ Statement ::= GroupSt	→ compileGroupSt();
KW_IF	→ Statement ::= IfSt	→ compileIfSt();
KW_WHILE	→ Statement ::= WhileSt	→ compileWhileSt();
KW_FOR	→ Statement ::= ForSt	→ compileForSt();
SB_SEMICOLON	→ ε	→ do nothing (break;)
KW_END	→ ε	→ do nothing (break;)
KW_ELSE	→ ε	→ do nothing (break;)
Others	Error	

Ví dụ Statement

```
void compileStatement(void) {
  switch (lookAhead-> tokenType){
  case TK_IDENT:
    compileAssignSt();
    break;
  case KW_CALL:
    compileCallSt();
    break;
  case KW_BEGIN:
    compileGroupSt();
    break;
  case KW_IF:
    compileIfSt();
    break;
  case KW_WHILE:
    compileWhileSt();
    break;
  case KW_FOR:
     compileForSt();
     break;
     // check FOLLOW tokens
   case SB_SEMICOLON:
   case KW_END:
   case KW_ELSE:
     break;
     // Error occurs
   default:
     error(ERR_INVALIDSTATEMENT,
lookAhead->lineNo, lookAhead-
>colNo);
     break;
  }
}
```

Tuần 1

- Dịch chương trình với
  - Khai báo hằng
  - Khai báo kiểu
  - Khai báo biến
  - Thân hàm rỗng

Tuần 2

- Dịch chương trình với
  - Khai báo hằng
  - Khai báo kiểu
  - Khai báo biến
  - Các lệnh

Tuần 3

- Dịch chương trình với đầy đủ sơ đồ cú pháp