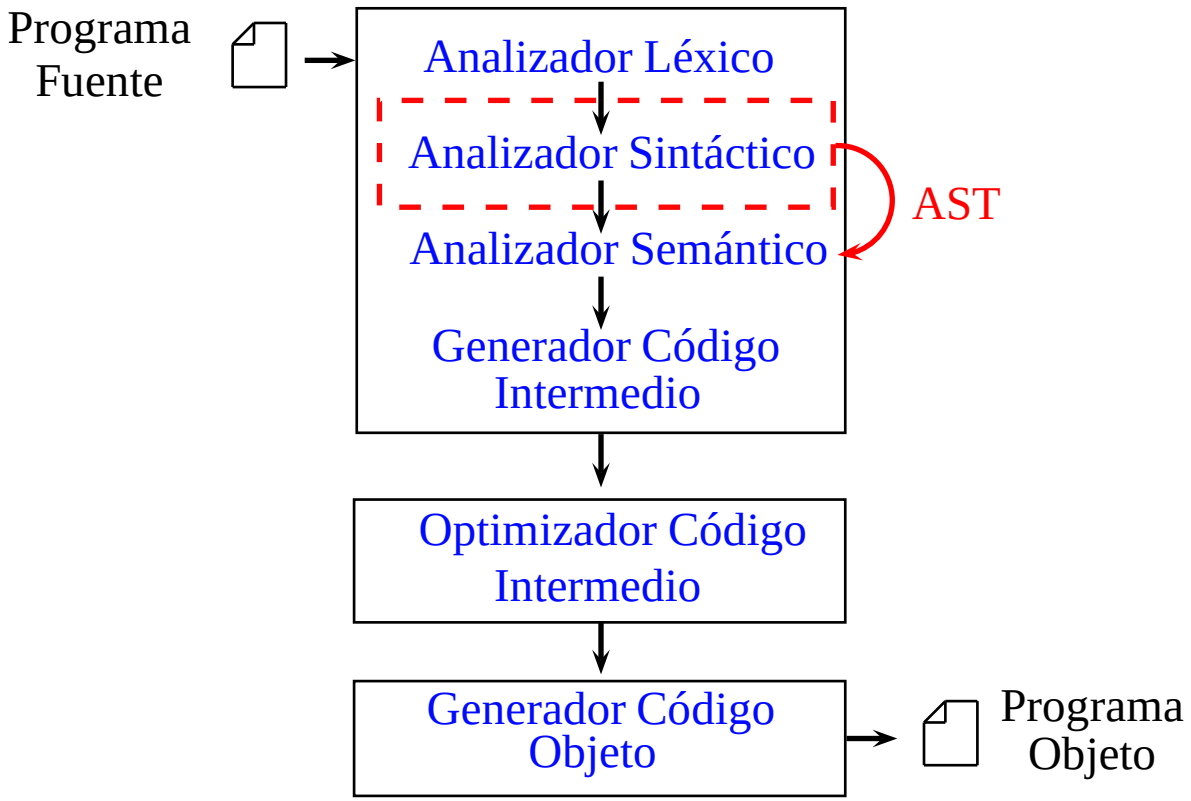
Desarrollo de Lenguajes Orientados a Objeto y Compiladores Avanzados [MII-779] Capítulo 3: Análisis Sintáctico

Dr. Ricardo Soto
[ricardo.soto@ucv.cl]
[http://www.inf.ucv.cl/~rsoto]

Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

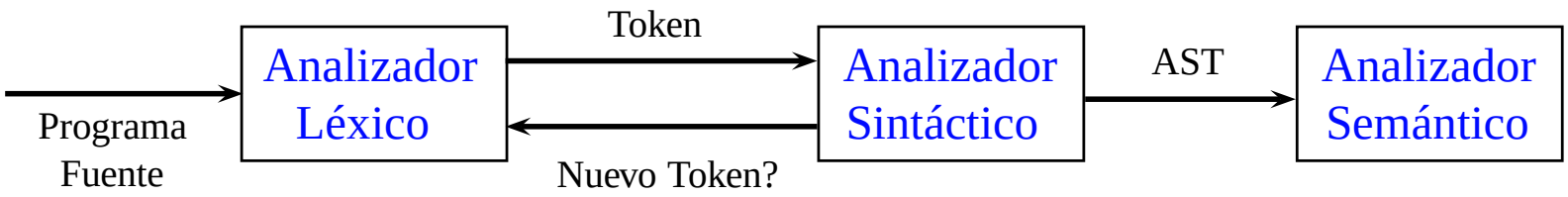
1. Introducción

El **analizador sintáctico** es la fase que sigue al análisis léxico. En esta fase se construye un AST (árbol de sintaxis abstracta) para capturar la jerarquía de la entrada.



2. Funciones del Analizador Sintáctico

- **Construir** un AST a partir de los **tokens** recibidos por el análizador léxico.



- Detección de **errores sintácticos**

Nota

El **analizador sintáctico** también se conoce como parser.

3. Herramientas para implementar analizadores sintácticos

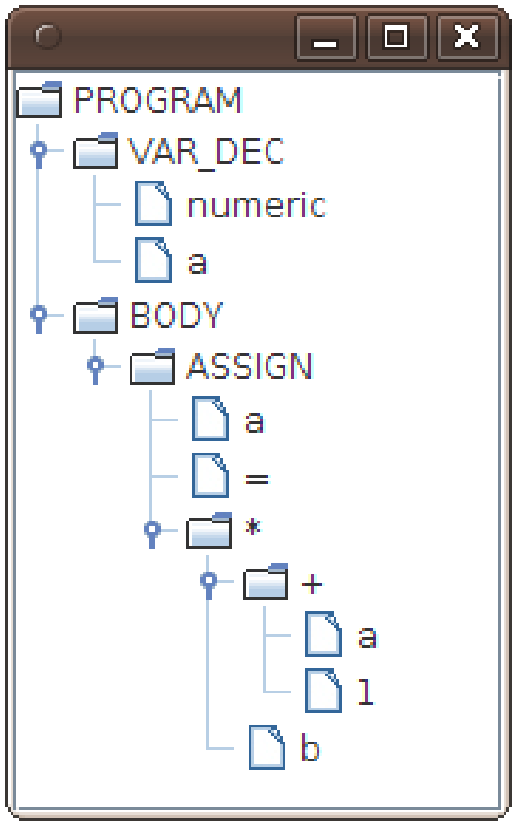
- Generadores de analizadores sintácticos:
  - Yacc (http://dinosaur.compilertools.net/)
  - Bison (http://www.gnu.org/software/bison/)
  - PLY (Python Lex-Yacc) (http://www.dabeaz.com/ply/)
  - ANTLR (http://www.antlr.org/)
  - ...

4. Implementación de analizadores sintácticos en ANTLR

Definición de tokens y reglas

```
tokens {
  PROGRAM
  VAR_DEC
  ASSIGN
  ...
}
...

program : VAR_RW! var_dec BEGIN_RW! body END_RW!
          {## = #( #[PROGRAM, "PROGRAM"] ,##);};

var_dec : (type IDENT SEMICOLON!)*
          {## = #( #[VAR_DEC, "VAR_DEC"] ,##);};

type:     NUMERIC_TYPE|STRING_TYPE;

assign : IDENT ASSIG expr SEMICOLON!
        {## = #( #[ASSIGN, "ASSIGN"] ,##);};
```

AST



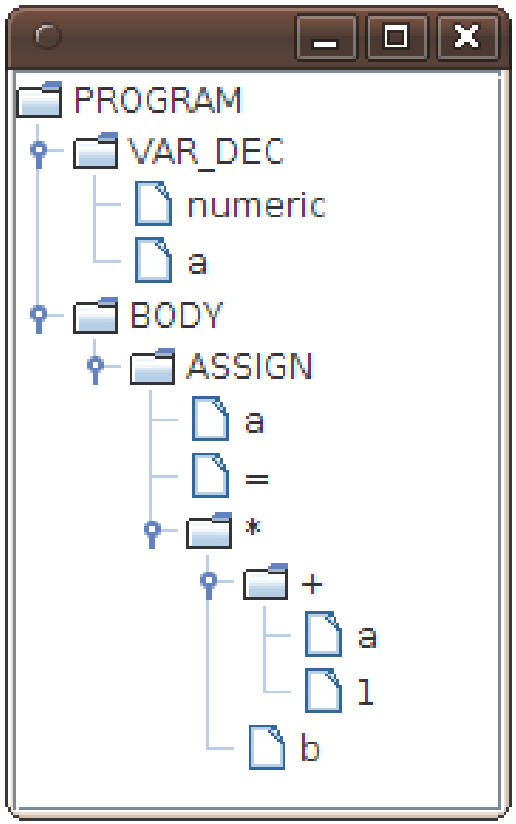
4. Implementación de analizadores sintácticos en ANTLR

Definición de tokens y reglas

```
tokens {
  PROGRAM
  VAR_DEC
  ASSIGN
  ...
}
...

program : VAR_RW! var_dec BEGIN_RW! body END_RW!
          {## = #( #[PROGRAM, "PROGRAM"] ,##);};

var_dec : (type IDENT SEMICOLON!)*
          {## = #( #[VAR_DEC, "VAR_DEC"] ,##);};

type:     NUMERIC_TYPE|STRING_TYPE;

assign : IDENT ASSIG expr SEMICOLON!
        {## = #( #[ASSIGN, "ASSIGN"] ,##);};
```

```
var
  numeric a;
begin
  a = (a + 1) * b;
end
```

AST



5. Ejercicios

Implemente el analizador sintáctico del lenguaje MiLe++ (Micro Lenguaje)

```
//grammatica mile++

program ::= main_class_dec (class_dec)*
main_class_dec ::= "main" class_dec;

class_dec ::= "class" IDENT ("extends" IDENT)?
            "(" (param)? ")" "{" class_body "}"

param ::= type IDENT ("," type IDENT)*
basic_type ::= ("numeric"|"string")

var_dec ::= (basic_var_dec| object_dec)*
basic_var_dec ::= basic_type IDENT ";"
object_dec ::= "object" IDENT IDENT ";"

type ::= basic_type|IDENT
```

```
method_dec ::= ("method" type IDENT "(" (param)? ")"
             "{" var_dec body return_st "}")*;
return_st ::= "return" expr ";"

class_body ::= var_dec method_dec
body ::= (assign |st | print | read)*

st ::= (for_st|if_st);
assign ::= IDENT "=" expr ";"

for_st ::= "for" for_header "{" body "}"
for_header ::= "(" assign expr ";" number ")"
if_st ::= "if" "(" exp ")" "{" body "}" else_st?
else_st ::= "else" "{" body "}"
read ::= "read" "(" ident ")" ";"
print ::= "print" "(" string "," ident ")" ";"
```