

## **Capítol 3 Anàlisi Sintàctica i us del generador de compiladors**

3.2 Expressar la gramàtica en BNF dels següents llenguatges:

- (a) Expressions amb operands números i variables. Les operacions possibles son: +, -, \*, /, (), -(unari).
- (b) Afegir a les expressions anteriors la possibilitat de fer crides a funcions amb paràmetres:
- (c) Afegir a les expressions anteriors la possibilitat de arrays i el factorial (n!): Si una funció NO pot retornar un array i si SI pot retornar un array:

## Capítol 3 Anàlisi sintàctica

3.2 Expressar la gramàtica en BNF dels següents llenguatges:

(a) Expressions amb operands números i variables. Les operacions possibles son:  
+, -, \*, /, (), -(unari).

$$\langle \text{exp} \rangle ::= \langle \text{ter} \rangle \{ ('+' | '-') \langle \text{ter} \rangle \}$$
$$\langle \text{ter} \rangle ::= \langle \text{fac} \rangle \{ ('*' | '/') \langle \text{fac} \rangle \}$$
$$\langle \text{fac} \rangle ::= \text{Num} | \text{Ident} | '-' \langle \text{fac} \rangle | '(' \langle \text{exp} \rangle ')'$$

b) Afegir a les expressions anteriors la possibilitat de fer crides a funcions amb paràmetres:

$$\langle \text{exp} \rangle ::= \langle \text{ter} \rangle \{ ('+' | '-') \langle \text{ter} \rangle \}$$
$$\langle \text{ter} \rangle ::= \langle \text{fac} \rangle \{ ('*' | '/') \langle \text{fac} \rangle \}$$
$$\langle \text{fac} \rangle ::= '-' \langle \text{fac} \rangle |$$
$$\text{Num} |$$
$$\text{Ident} [ '(' [ \langle \text{exp} \rangle \{ ',' \langle \text{exp} \rangle \} ] ')'] |$$
$$'(\langle \text{exp} \rangle)'$$

## Problema 3.2(cont.)

3.2(c) Afegir a les expressions anteriors la possibilitat de arrays i el factorial (n!):

$\langle \text{exp} \rangle ::= \langle \text{ter} \rangle \{ (+|-) \langle \text{ter} \rangle \}$

$\langle \text{ter} \rangle ::= \langle \text{fac1} \rangle \{ (*|/) \langle \text{fac1} \rangle \}$

$\langle \text{fac1} \rangle ::= '-' \langle \text{fac1} \rangle | \langle \text{fac2} \rangle$

Si una funció NO pot retornar un array:

$\langle \text{fac2} \rangle ::= (\text{Num} | \text{Ident} [ ( ' ( [ \langle \text{exp} \rangle \{ ', \langle \text{exp} \rangle \} ] ' ) | ( [ ' \langle \text{exp} \rangle \{ ', \langle \text{exp} \rangle \} ' ] ) ] | ' ( \langle \text{exp} \rangle ' ) ) ) \{ '!' \}$

Si una funció SI pot retornar un array:

$\langle \text{fac2} \rangle ::= (\text{Num} | \text{Ident} [ ( ' ( [ \langle \text{exp} \rangle \{ ', \langle \text{exp} \rangle \} ] ' ) | ( [ ' \langle \text{exp} \rangle \{ ', \langle \text{exp} \rangle \} ' ] ) ] | ' ( \langle \text{exp} \rangle ' ) ) ) \{ '!' \}$

## Instal·lar CoSeL i el modul Com.csm

- A la web de l'assignatura baixar els fitxers al mateix directori
  - Instal·lador de CrossVisions (CrossVisions Public setup.exe)
  - El Generador de compiladors (Com.csm)
- Guardar el fitxer Com.csm en el directori de treball
- Executar CrossVisions Public setup.exe
- Executar CoSeL.CIP del menú d'inici/CrossVisions
- Del menú de la aplicació seleccionar Fichero/Guardar Proyecto Como... i guardar el fitxer .cip en el directori de treball
- Sortir de l'aplicació
- Executar el fitxer .cip guardat

## Exemple d'un fitxer CoSeL (init.csl) que utilitza Com.csm

use com

// GRAMATICA =====

Var g=

BNF\_GRAMMAR <expressio>

TERMINALS + - \* / identificador numero ( ) !

BNF

Rule <expressio> ::= <terme> { (+|-) <terme> }

Rule <terme> ::= <factor> { (\*|/) <factor> }

Rule <factor> ::=

- <factor> | "(" <expressio> ")" |

Numero | Identificador

END;

## Exemple d'un fitxer CoSeL (init.csl) que utilitza Com.csm

// Càlculs de la gramàtica =====

g.BNFAProduccions();

g.CalcularAnullables();

g.CalcularPrimers();

g.CalcularSeguents();

// Visualització de resultats =====

g.VeureProduccions();

g.VeureAnullables();

g.VeurePrimers();

g.VeureSeguents();

## Produccions

<Expressio> --> <terme> <Expressio\_1>

<Expressio\_1> --> <Expressio\_2> <terme> <Expressio\_1>

<Expressio\_1> --> \$

<Expressio\_2> --> -

<Expressio\_2> --> +

<factor> --> <factor\_1>

<factor\_1> --> Identificador

<factor\_1> --> Numero

<factor\_1> --> ( <Expressio> )

<factor\_1> --> - <factor>

<terme> --> <factor> <terme\_1>

<terme\_1> --> <terme\_2> <factor> <terme\_1>

<terme\_1> --> \$

<terme\_2> --> /

<terme\_2> --> \*

## Anul·lables i Primers

Símbolos anulables: <terme\_1> <Expressio\_1>

Conjunts de primers dels símbols no terminals:

Primers(<Expressio>) = Identificador Numero ( -

Primers(<Expressio\_1>) = - + \$

Primers(<terme>) = - ( Numero Identificador

Primers(<Expressio\_2>) = + -

Primers(<terme\_1>) = / \* \$

Primers(<factor>) = Identificador Numero ( -

Primers(<terme\_2>) = \* /

Primers(<factor\_1>) = - ( Numero Identificador

## Seguents

Conjunts de seguents dels símbols no terminals:

Seguents(<Expressio>)= ) <SCHAR:EOF>

Seguents(<Expressio\_1>)= ) <SCHAR:EOF>

Seguents(<terme>)= ) + - <SCHAR:EOF>

Seguents(<Expressio\_2>)= Identificador Numero ( -

Seguents(<terme\_1>)= ) <SCHAR:EOF> - +

Seguents(<factor>)= ) \* / + - <SCHAR:EOF>

Seguents(<terme\_2>)= - ( Numero Identificador

Seguents(<factor\_1>)= ) <SCHAR:EOF> - + / \*

## Generació del parser

ga.VeureCollisionsLL1();

No hi ha Col·lisions LL(1)

ga.GenerarParser(`parserGA);

ParserGA("expressio.txt");

Anàlisi correcta del fitxer expressio.txt

## Arbre sintàctic

VeureArbreSintactic(); // Contingut del fitxer: 10+20\*b

<PARSER\_Expressio>

<PARSER\_terme>

<PARSER\_factor>

Numero#(10)

+

<PARSER\_terme>

<PARSER\_factor>

Numero#(20)

\*

<PARSER\_factor>

Identificador#(b)