

Clase #24 de 25

Ambigüedades Sintácticas, Restricciones Semánticas e Integración Lex/Yacc

Nov 12, Lunes

Agenda para esta clase

- Jerraquía de Chomsky
- Sintaxis, Restricciones Semánticas, y Semántica
- Árbol de Sintaxis Concreta & Árbol de Sintaxis Abstracta
- Ambigüedades Sintácticas
- Sintaxis de C – Unidad de Traducción
- Sintaxis de C – Sentencias
- Integración Lex/Yacc
- Resolución del Último Final

Jearquía de Chomsky

Jerarquía de Chomsky

Lenguaje	Gramática	Autómat a	Otros	Aplicación
Regular	GR	AF	ER → RegEx → Lex	Análisis Léxico
Independiente del Contexto	GIC	APD	BNF → EBNF → Yacc	Análisis Sintáctico
Sensible al Contexto	GSC	MT (Autómat a linealmente acotado)	-	-
Irrestrictivo	GI	MT	-	-

Sintaxis, Restricciones, y Semántica

Especificación de Sintaxis y Semántica en Estándares de Lenguajes de Programación

Especificación de Sintaxis y Semántica en Estándares de Lenguajes de Programación

1. Sintaxis

- **Reglas Sintácticas**
- Especificación del constructo sintáctico en notación basada en GIC, por ejemplo, K&R y BNF

2. Restricciones para que tenga semántica

- **Reglas semánticas**
- Restricciones semánticas **sensibles al contexto** para que el constructo sintáctico tengan semántica, expresadas en **lenguaje natural**, con referencia a los elementos sintácticos del constructo

3. Semántica

- Definición del **comportamiento observable**, expresado en **lenguaje natural**

Ejemplo de Sintaxis y Semántica: Sentencia de Selección

6.8.4 Selection statements

Syntax

selection-statement:

if (*expression*) *statement*

if (*expression*) *statement* **else** *statement*

switch (*expression*) *statement*

Semantics

A selection statement selects among a set of statements depending on the value of a controlling expression.

Ejemplo de Sintaxis y Semántica: Sentencia If

6.8.4.1 The **if** statement

Constraints

The controlling expression of an **if** statement shall have scalar type.

Semantics

In both forms, the first substatement is executed if the expression compares unequal to 0. In the **else** form, the second substatement is executed if the expression compares equal to 0. If the first substatement is reached via a label, the second substatement is not executed.

Árbol de Sintaxis Concreta & Árbol de Sintaxis Abstracta

Sintaxis Concreta y Abstracta

- Sintaxis concreta
 - Árbol de Sintaxis Concreto
 - Específica para un LP
 - Concretas
 - Con tokens
 - Forma
 - Producto de la derivación según reglas sintáticas
- Sintaxis abstracta
 - Árbol de Sintaxis Abstracto
 - General para varios LP
 - Conceptual
 - Sin tokens
 - Semántica
 - Necesaria para el análisis semántico

Ejercicio: Obtener los Árboles de Sintaxis Concreta y Abstracta

```
if ( a )  
f();
```

- ASC:
- ASA:

```
if ( b );  
g();  
else  
h();
```

- ASC:
- ASA:

Ambigüedad en la Definición Sintáctica

Ejercicio: Dibuje cada Árbol de Derivación Sintáctica

```
if ( a )
  if( b )
    g();
else
  h();
```

```
if ( a ) {
  if( b ){
    g();
  }
} else {
  h();
}
```

```
if ( a ) {
  if( b )
    g();
} else
  h();
```

```
if ( a )
  if( b )
    g();
else
  h();
```

```
if ( a ) {
  if( b ){
    g();
  } else {
    h();
  }
}
```

```
if ( a ) {
  if( b )
    g();
} else
  h();
```

¿Conclusión?

- Dos derivaciones posibles
- Dos semánticas posibles
- Ambigüedad:
 - Una sintaxis es ambigua cuando permite más de un árbol de derivación
- ¿Qué pasa en otros lenguajes?
- ¿Soluciones?
 - Estilo
 - Llaves siempre
 - Llaves cuando se las requiere
 - Corrección de la S&S de la sentencia if

Ejemplo: Resolución de la Única Ambigüedad en C

6.8.4.1 The **if** statement

Constraints

The controlling expression of an **if** statement shall have scalar type.

Semantics

In both forms, the first substatement is executed if the expression compares unequal to 0. In the **else** form, the second substatement is executed if the expression compares equal to 0. If the first substatement is reached via a label, the second substatement is not executed.

An **else** is associated with the lexically nearest preceding **if** that is allowed by the syntax.



Intervalo

20 minutos

Sintaxis de C – Unidad de Traducción

Axioma de un Programa C

Reglas de Sintaxis para Cualquier Programa C

- ¿Qué es un “programa”?
- ¿Qué incluir?
- ¿Dónde comenzar?
- ¿Cuál es la forma general?

Unidad de traducción y definición de función

unidad-de-traducción

declaración-externa

unidad-de-traducción declaración-externa

declaración-externa

definición-de-función

declaración

definición-de-función

especificadores-de-declaración? declarador sentencia-compuesta

Sintaxis de C – Sentencias

Sentencia

sentencia:

sentencia-expresión

sentencia-compuesta

sentencia-de-selección

sentencia-de-iteración

sentencia-etiquetada

sentencia-de-salto

sentencia-expresión:

expresión? ;

Sentencia compuesta (¿C11?)

sentencia-compuesta:

{ *lista-de-declaraciones?* *lista-de-sentencias?* }

lista-de-declaraciones:

declaración

lista-de-declaraciones *declaración*

lista-de-sentencias:

sentencia

lista-de-sentencias *sentencia*

Sentencia de selección

sentencia-de-selección:

if (*expresión*) *sentencia*

if (*expresión*) *sentencia* **else** *sentencia*

switch (*expresión*) *sentencia*

Sentencia de iteración (¿C11?)

sentencia-de-iteración:

while (*expresión*) *sentencia*

do *sentencia* **while** (*expresión*) ;

for (*expresión?* ; *expresión?* ; *expresión?*) *sentencia*

Sentencia etiquetada y de salto

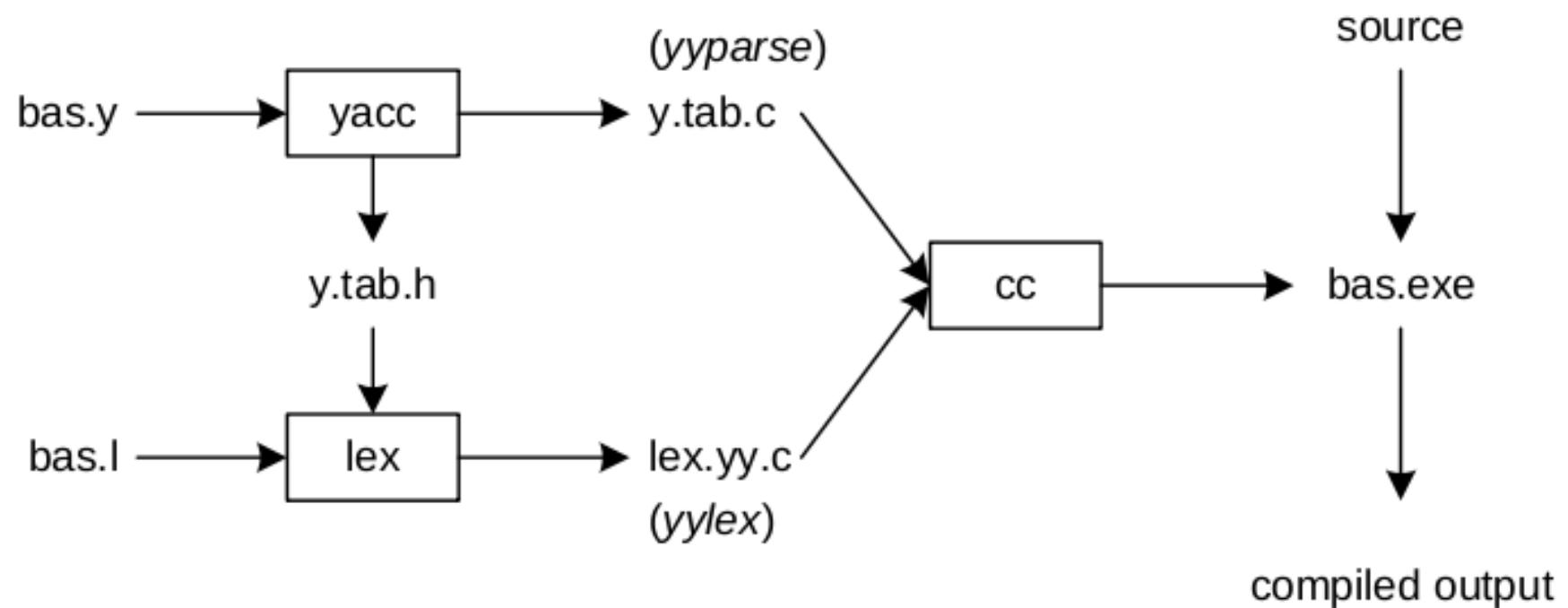
sentencia-etiquetada:

case *expresión-constante* : *sentencia*
default : *sentencia*
identificador : *sentencia*

sentencia-de-salto:

continue ;
break ;
return *expresión?* ;
goto *identificador* ;

Integración entre Lex & Yacc



Lex & Yacc: %token & y.tab.h

- **%token** indica los terminales o tokens de la gramática
- El header **y.tab.h** lo genera yacc cuando se indica la opción **-d** desde la línea de comando, y contiene las declaraciones de tokens
- El header lo debe incluir el programa resultante de lex

Generación de Programa desde especificaciones Lex y Yacc

- Scanner.l
- Lex
 - Scanner.c
- CC
 - *Scanner.obj*
- Parser.y
- Yacc
 - Parser.tab.c
 - Parser.tab.h
- CC
 - *Parser.tab.obj*
- Prog.c
- CC
 - Prog.obj
- +
 - *Parser.tab.obj*
 - *Scanner.obj*
- Link
 - Prog.exe
- Funciones
 - *yyacc()*
 - *yylex()*

Lenguaje Ejemplo

$$L = \left\{ ({}^n \cdot \text{número} \cdot)^n / n \geq 0, \text{número} \in \text{Número} \right\}$$

$$\text{Número} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^*$$

- Ejercicios:
 - Defina formalmente la GIC
 - ¿Hay AF?
 - Defina formalmente el APD
- Escribir un programa que informe el valor y cuántos paréntesis lo rodean:
 - Makefile
 - main.c
 - Yacc.y
 - Scanner.l

Extracto de Especificación Lex

```
%%
[ \t\n]  ;
[0-9]+ {yyval = atoi(yytext);return NUMBER;}
"("    {yyval = *yytext;      return LPAR;}
")"    {yyval = *yytext;      return RPAR;}
.      {printf("Lex error.\n");exit(EXIT_FAILURE);}
%%
```

Extracto de Especificación Yacc

```
%%
S
: NUMBER          {n=$1;}
| LPAR S RPAR   {++ns;}
;
%%
```

Programa Completo – Makefile

```
CFLAGS = -std=c11 -Weverything
YACC   = bison
RM     = rm -f

.PHONY: clean all

all: prog

prog: main.o Parser.tab.o Scanner.o
$(CC) -o $@ $^
$(RM) main.o Parser.tab.h Parser.tab.o Scanner.o Scanner.c

%.tab.c %.tab.h: %.y
$(YACC) -d $<

Scanner.c : Parser.tab.h

clean:
$(RM) prog main.o Parser.tab.h Parser.tab.o Scanner.o Scanner.c
```

Programa Completo – main.c

```
#include <stdio.h>

int main(void){
    int yyparse(void);
    switch(yyparse()){
        case 0:{
            extern unsigned ns;
            extern int n;
            printf("Se encontró %d rodeado de %d paréntesis.\n",n,ns);
            return 0;
        }
        case 1:
            printf("Error sintático.\n");
            return 1;
        default:
            printf("Otro error.\n");
            return 2;
    }
}
```

Programa Completo – Parser.y

```
%{  
#include <stdio.h>  
unsigned ns;  
int n;  
int yyerror(const char *);  
%}  
  
%token NUMBER LPAR RPAR  
  
%%  
S  
: NUMBER {n=$1;}  
| LPAR S RPAR {++ns;}  
;  
%%  
int yyerror(const char *s){}
```

Programa Completo – Scanner.l

```
%option noyywrap
%{
#include <stdlib.h>
#include "Parser.tab.h"
%}

%%
[\t\n]  ;
[0-9]+ {yyval = atoi(yytext);return NUMBER;}
 "("   {yyval = *yytext;      return LPAR;}
 ")"   {yyval = *yytext;      return RPAR;}
.     {printf("Lex error.\n");exit(EXIT_FAILURE);}
%%
```

Resolución del Último Final

Práctica y Autodiagnóstico

<https://josemariasola.wordpress.com/ssl/exams/>



¿Consultas?



Fin de la clase