

Principios del Lenguajes de Programación

Clase 3 - Análisis Sintáctico

Sandra Roger

Facultad de Informática
Universidad Nacional del Comahue

Primer Cuatrimestre

Contenidos

Repaso

Introducción

Descripción de la sintaxis

Definición formal de lenguajes

Bibliografía

¿Qué vimos?

PROGRAMA ANALÍTICO:

Unidad 1. *Introducción a los conceptos de los lenguajes de programación*

Concepto del del lenguaje de programación. Buenos Lenguajes y Atributos. Historia y Evolución de los Lenguajes de Programación. Clasificación de lenguajes. Criterios de diseño y de implementación de un lenguaje de programación. Concepto de Paradigma: imperativo, funcional, lógico, orientado a objetos. Máquinas virtuales: Jerarquías. Sintaxis y semántica. Conceptos de Intérpretes y Compiladores.

Técnicas formales de descripción sintáctica. Introducción a la Semántica Operacional, Axiomática y Denotacional. Atributos y Tiempos de Ligadura.

¿Qué vimos?

Ortogonalidad:

Capacidad de combinar varias características de un lenguaje y que tengan significado.

- ▶ Más ortogonales entonces mayor cantidad de combinaciones de caracteres, por lo tanto menos excepciones y menos casos especiales que recordar
- ▶ Desventaja: Compilación sin errores con combinaciones lógicamente incoherente o ejecución ineficiente

¿Qué vimos?

Ortogonalidad:

Capacidad de combinar varias características de un lenguaje y que tengan significado.

- ▶ Más ortogonales entonces mayor cantidad de combinaciones de caracteres, por lo tanto menos excepciones y menos casos especiales que recordar
- ▶ Desventaja: Compilación sin errores con combinaciones lógicamente incoherente o ejecución ineficiente

Introducción

MI VECINA ME VINO A RECLAMAR
QUE MI PERRO PERSIGUE A SU
HIJO EN BICICLETA.
JAJAJAJA QUE TONTA, MI PERRO
NI SABE ANDAR EN BICI.

Introducción

MI VECINA ME VINO A RECLAMAR
QUE MI PERRO PERSIGUE A SU
HIJO EN BICICLETA.
JAJAJAJA QUE TONTA, MI PERRO
NI SABE ANDAR EN BICI.



THE PROBLEM ABOUT BEING A PROGRAMMER

My mom said:

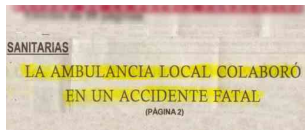
"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6"

I came back with 6 bottles of milk.

She said: "Why the hell did you buy 6 bottles of milk?"

I said: "BECAUSE THEY HAD EGGS!!!!"

Introducción



Introducción

Lourdes no quiere a su tía porque es muy envidiosa.

¿Quién es envidiosa, Lourdes o su tía?

El pez está listo para comer.

¿El pez está listo para ser comido o está listo para que le den de comer?

Estuve esperándote en el banco.

¿En qué banco? ¿Sentado en un banco o en una institución financiera?

Se debe limpiar aquí.

¿Es una orden para alguien que necesita limpiarse o ese lugar necesita una limpieza?

Introducción

Lourdes no quiere a su tía porque es muy envidiosa.

¿Quién es envidiosa, Lourdes o su tía?

El pez está listo para comer.

¿El pez está listo para ser comido o está listo para que le den de comer?

Estuve esperándote en el banco.

¿En qué banco? ¿Sentado en un banco o en una institución financiera?

Se debe limpiar aquí.

¿Es una orden para alguien que necesita limpiarse o ese lugar necesita una limpieza?

Introducción

Lourdes no quiere a su tía porque es muy envidiosa.

¿Quién es envidiosa, Lourdes o su tía?

El pez está listo para comer.

¿El pez está listo para ser comido o está listo para que le den de comer?

Estuve esperándote en el banco.

¿En qué banco? ¿Sentado en un banco o en una institución financiera?

Se debe limpiar aquí.

¿Es una orden para alguien que necesita limpiarse o ese lugar necesita una limpieza?

Introducción

Lourdes no quiere a su tía porque es muy envidiosa.

¿Quién es envidiosa, Lourdes o su tía?

El pez está listo para comer.

¿El pez está listo para ser comido o está listo para que le den de comer?

Estuve esperándote en el banco.

¿En qué banco? ¿Sentado en un banco o en una institución financiera?

Se debe limpiar aquí.

¿Es una orden para alguien que necesita limpiarse o ese lugar necesita una limpieza?

Introducción

Lourdes no quiere a su tía porque es muy envidiosa.

¿Quién es envidiosa, Lourdes o su tía?

El pez está listo para comer.

¿El pez está listo para ser comido o está listo para que le den de comer?

Estuve esperándote en el banco.

¿En qué banco? ¿Sentado en un banco o en una institución financiera?

Se debe limpiar aquí.

¿Es una orden para alguien que necesita limpiarse o ese lugar necesita una limpieza?

Introducción

- ▶ Informalidad - Formalidad: Necesidad de contar con una descripción más precisa (formal) de un lenguaje de programación.
- ▶ Razonar matemáticamente sobre un programa lo cual requiere una verificación formal del comportamiento de un LdP.
- ▶ Estandarización: Independencia de la implementación y de la máquina.

Introducción

La especificación precisa de un lenguaje de programación es esencial para describir el **comportamiento computacional** del mismo.

Sin una **notación clara** del efecto de los constructores del lenguaje es imposible determinar dicho **comportamiento**.

Introducción

- ▶ **Alfabeto**: conjunto de cadenas de caracteres elegidas de una conjunto finito y fijo de símbolos .
- ▶ **Frases** o **Constructores**: Las cadenas que pertenecen al lenguaje

Introducción

Descripción del lenguaje

Sintaxis

Descripción del formato de los programas del lenguaje

Descripción de cómo se forman las frases

Semántica

Estudia el comportamiento de los programas

Manejo del “significado” de las frases

Introducción

Pragmática

Estudia aspectos relacionados con las técnicas empleadas para la construcción de programas.

Manejo del uso práctico de las frases

Grado de éxito con el que un programa cumple sus objetivos tanto en su fidelidad con el modelo de computación subyacente como su utilidad para los programadores Ej. El traductor necesita proveer opciones al usuario, para debugear, para interactuar con el sistema operativo y tal vez para interactuar con un entorno de desarrollo de software. Estas opciones conforman el pragmatismo del traductor (en general, estas facilidades forman parte de la definición del lenguaje)

Introducción

Construcción Sintáctica:

Ejemplo: Asignación de una expresión

$\langle \textit{asig} \rangle \rightarrow \langle \textit{variable} \rangle \langle \textit{signo_asig} \rangle \langle \textit{expresin} \rangle ;$

$\langle \textit{variable} \rangle \rightarrow \langle \textit{identificador} \rangle$

$\langle \textit{signo_asig} \rangle \rightarrow := | \dots$

$\langle \textit{expresion} \rangle \rightarrow \dots$

Introducción

Construcción Semántica:

Ejemplo: Asignación de una expresión

$a := b + 1$

- ▶ **Variable (lado izquierdo):** lugar en la memoria.
- ▶ **Expresión (lado derecho):** computación de un valor basado en operandos (lugares en memoria, literales, constantes, etc.) y operaciones
- ▶ **Operación de Asignación:** guardar el valor calculado de la expresión en el lugar de la memoria que representa el nombre de la variable

Construcción Pragmática:

Ejemplo: Asignación de una expresión ¿Cuál es el “uso” de la asignación?:

- ▶ setear el valor de una expresión que necesito usar luego más de una vez.
- ▶ comunicar valores de un lugar a otro del programa
- ▶ modificar parte de la estructura de un objeto de datos.
- ▶ guardar valores acumulados en una iteración
- ▶ Etc.

Introducción

Errores según la fase de traducción en la que ocurren:

- ▶ **Los errores léxicos** ocurren durante el análisis léxico:
 - ▶ caracteres ilegales Ej. (en C) $x@ = 2$
 - ▶ errores ortográficos: Ej while en lugar de while
- ▶ **Los errores sintácticos** incluyen tokens faltantes y expresiones mal organizadas
- ▶ **Los errores semánticos** pueden ser.
 - ▶ **Estáticos**: (es decir, detectados antes de la ejecución). Tipos incompatibles o variables no declaradas;
 - ▶ **Dinámicos**: (detectados durante la ejecución). Un subíndice fuera de rango o la división entre cero.

Introducción

Errores según la fase de traducción en la que ocurren:

- ▶ **Los errores léxicos** ocurren durante el análisis léxico:
 - ▶ caracteres ilegales Ej. (en C) $x@ = 2$
 - ▶ errores ortográficos: Ej while en lugar de while
- ▶ **Los errores sintácticos** incluyen tokens faltantes y expresiones mal organizadas
- ▶ **Los errores semánticos** pueden ser.
 - ▶ **Estáticos**: (es decir, detectados antes de la ejecución). Tipos incompatibles o variables no declaradas;
 - ▶ **Dinámicos**: (detectados durante la ejecución). Un subíndice fuera de rango o la división entre cero.

Introducción

Errores según la fase de traducción en la que ocurren:

- ▶ **Los errores léxicos** ocurren durante el análisis léxico:
 - ▶ caracteres ilegales Ej. (en C) $x@ = 2$
 - ▶ errores ortográficos: Ej while en lugar de while
- ▶ **Los errores sintácticos** incluyen tokens faltantes y expresiones mal organizadas
- ▶ **Los errores semánticos** pueden ser.
 - ▶ **Estáticos:** (es decir, detectados antes de la ejecución).
Tipos incompatibles o variables no declaradas;
 - ▶ **Dinámicos:** (detectados durante la ejecución). Un subíndice fuera de rango o la división entre cero.

Introducción

Errores según la fase de traducción en la que ocurren:

- ▶ **Los errores léxicos** ocurren durante el análisis léxico:
 - ▶ caracteres ilegales Ej. (en C) $x@ = 2$
 - ▶ errores ortográficos: Ej while en lugar de while
- ▶ **Los errores sintácticos** incluyen tokens faltantes y expresiones mal organizadas
- ▶ **Los errores semánticos** pueden ser.
 - ▶ **Estáticos:** (es decir, detectados antes de la ejecución). Tipos incompatibles o variables no declaradas;
 - ▶ **Dinámicos:** (detectados durante la ejecución). Un subíndice fuera de rango o la división entre cero.

Introducción

Errores según la fase de traducción en la que ocurren:

- ▶ **Los errores léxicos** ocurren durante el análisis léxico:
 - ▶ caracteres ilegales Ej. (en C) $x@ = 2$
 - ▶ errores ortográficos: Ej while en lugar de while
- ▶ **Los errores sintácticos** incluyen tokens faltantes y expresiones mal organizadas
- ▶ **Los errores semánticos** pueden ser.
 - ▶ **Estáticos:** (es decir, detectados antes de la ejecución). Tipos incompatibles o variables no declaradas;
 - ▶ **Dinámicos:** (detectados durante la ejecución). Un subíndice fuera de rango o la división entre cero.

Introducción

Otros tipo de errores:

- **Los errores lógicos:** Estos son aquellos que comete el programador y que hacen que el programa se comporte de una manera errónea o no deseable. Ej ciclos infinitos si u y v valen 1

```
x = u;  
y = v;  
while ( y , != 0)  
t = y;  
y = x * y;  
x = t;
```

- No hay violación en la fase semántica
- NO son errores de la traducción

Definición formal de lenguajes

Reconocedores:

- ▶ Un dispositivo de reconocimiento lee cadenas de entrada de un lenguaje y decide si la entrada pertenece o no al lenguaje
- ▶ Ejemplo: analizador sintáctico de un compilador.

Generadores:

- ▶ Un dispositivo que genera las oraciones (sentencias) de un lenguaje
- ▶ Uno puede determinar si la sintaxis de una sentencia particular es correcta, comparándolas con la generada por un generador
- ▶ Ejemplo: gramática

Definición formal de lenguajes

Reconocedores:

- ▶ Un dispositivo de reconocimiento lee cadenas de entrada de un lenguaje y decide si la entrada pertenece o no al lenguaje
- ▶ Ejemplo: analizador sintáctico de un compilador.

Generadores:

- ▶ Un dispositivo que genera las oraciones (sentencias) de un lenguaje
- ▶ Uno puede determinar si la sintaxis de una sentencia particular es correcta, comparándolas con la generada por un generador
- ▶ Ejemplo: gramática

Descripción de la sintaxis

Terminología:

- ▶ Una **sentencia (oración)** es una cadena de caracteres pertenecientes a un alfabeto.
- ▶ Un **lenguaje** es un conjunto de sentencias
- ▶ Un **lexema** es la unidad sintáctica básica de un lenguaje (Ej, *, suma, begin, etc.)
- ▶ Un **token (componente léxico)** es una categoría de lexemas (Ej, identificador, literal, etc.)

Métodos formales para describir la sintaxis

Tres mecanismos describen el diseño e implementación de los lenguajes de programación:

1. Expresiones regulares
 2. Gramáticas formales
 3. Gramáticas de atributos
- ▶ Las expresiones regulares nos permitirán definir los **lexemas** o **tokens**(trabajo del analizador léxico)
 - ▶ Las gramáticas formales definirán la sintaxis y
 - ▶ Las gramáticas de atributos, definirán la **semántica estática**.

Métodos formales para describir la sintaxis

- ▶ Gramáticas
- ▶ Forma de Backus-Naur (Backus-Naur Form) y Gramáticas Libres de Contexto
- ▶ Forma de Backus-Naur Extendida (Extended BNF)

Métodos formales para describir la sintaxis

EXAMPLE 3.1

A Grammar for a Small Language

```
<program> → begin <stmt_list> end  
<stmt_list> → <stmt>  
              | <stmt> ; <stmt_list>  
<stmt> → <var> = <expression>  
<var> → A | B | C  
<expression> → <var> + <var>  
               | <var> - <var>  
               | <var>
```

Forma de Backus-Naur (BNF)

- ▶ Backus-Naur Form (1959)
- ▶ Desarrollada por John Backus para describir al Algol 58
- ▶ Equivalente a las gramáticas libres de contexto
- ▶ Meta-lenguaje utilizado para describir otro lenguaje

EXAMPLE 3.5

BNF and EBNF Versions of an Expression Grammar

BNF:

```
<expr> → <expr> + <term>
        | <expr> - <term>
        | <term>
<term> → <term> * <factor>
        | <term> / <factor>
        | <factor>
<factor> → <exp> ** <factor>
          | <exp>
<exp> → ( <expr> )
        | id
```

EBNF:

```
<expr> → <term> {(+ | -) <term>}
<term> → <factor> {( * | / ) <factor>}
<factor> → <exp> { ** <exp>}
<exp> → ( <expr> )
        | id
```

Derivación

- ▶ Cada **cadena de símbolos** en una derivación está en **forma sentencial**
- ▶ Una **sentencia** (frase u oración) es una **forma sentencial** que sólo tiene **símbolos terminales**
- ▶ Una **derivación izquierda** es aquella en la cual se expande el no terminal más a la izquierda de cada forma sentencial
- ▶ Una derivación puede ser a izquierda o a derecha

Derivación

- ▶ Cada **cadena de símbolos** en una derivación está en **forma sentencial**
- ▶ Una **sentencia** (frase u oración) es una **forma sentencial** que sólo tiene **símbolos terminales**
- ▶ Una **derivación izquierda** es aquella en la cual se expande el no terminal más a la izquierda de cada forma sentencial
- ▶ Una derivación puede ser a izquierda o a derecha

Derivación

- ▶ Cada **cadena de símbolos** en una derivación está en **forma sentencial**
- ▶ Una **sentencia** (frase u oración) es una **forma sentencial** que sólo tiene **símbolos terminales**
- ▶ Una **derivación izquierda** es aquella en la cual se expande el no terminal más a la izquierda de cada forma sentencial
- ▶ Una derivación puede ser a izquierda o a derecha

Derivación

- ▶ Cada **cadena de símbolos** en una derivación está en **forma sentencial**
- ▶ Una **sentencia** (frase u oración) es una **forma sentencial** que sólo tiene **símbolos terminales**
- ▶ Una **derivación izquierda** es aquella en la cual se expande el no terminal más a la izquierda de cada forma sentencial
- ▶ Una derivación puede ser a izquierda o a derecha

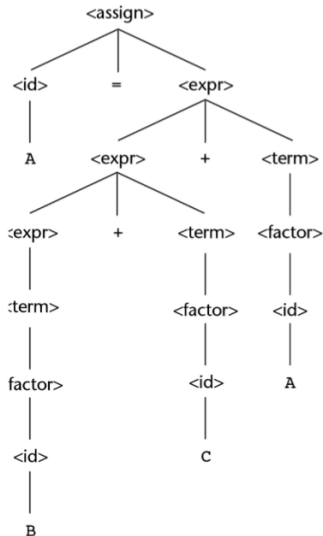
Ejemplo de derivación

```
<program> => begin <stmt_list> end  
=> begin <stmt> ; <stmt_list> end  
=> begin <var> = <expression> ; <stmt_list> end  
=> begin A = <expression> ; <stmt_list> end  
=> begin A = <var> + <var> ; <stmt_list> end  
=> begin A = B + <var> ; <stmt_list> end  
=> begin A = B + C ; <stmt_list> end  
=> begin A = B + C ; <stmt> end  
=> begin A = B + C ; <var> = <expression> end  
=> begin A = B + C ; B = <expression> end  
=> begin A = B + C ; B = <var> end  
=> begin A = B + C ; B = C end
```

Árbol sintáctico

Figure 3.4

A parse tree for $A = B + C + A$ illustrating the associativity of addition



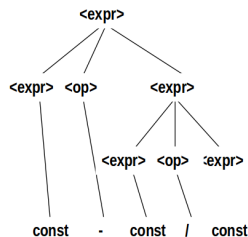
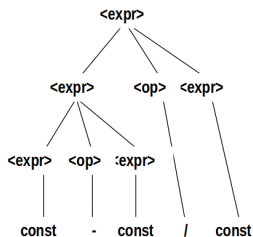
Árbol sintáctico

$$\begin{aligned}\langle \textit{expr} \rangle &\rightarrow \langle \textit{term} \rangle \langle \textit{op} \rangle \langle \textit{term} \rangle \mid \textit{const} \\ \langle \textit{op} \rangle &\rightarrow / \mid -\end{aligned}$$

Árbol sintáctico

$\langle \text{expr} \rangle \rightarrow \langle \text{term} \rangle \langle \text{op} \rangle \langle \text{term} \rangle \mid \text{const}$

$\langle \text{op} \rangle \rightarrow / \mid -$



BNF y E-BNF

► BNF

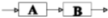

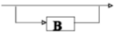
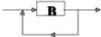
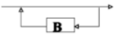
$$\begin{aligned} \langle \textit{expr} \rangle &\rightarrow \langle \textit{expr} \rangle + \langle \textit{term} \rangle \\ &\quad | \quad \langle \textit{expr} \rangle - \langle \textit{term} \rangle \\ &\quad | \quad \langle \textit{term} \rangle \\ \langle \textit{term} \rangle &\rightarrow \langle \textit{term} \rangle * \langle \textit{factor} \rangle \\ &\quad | \quad \langle \textit{term} \rangle / \langle \textit{factor} \rangle \\ &\quad | \quad \langle \textit{factor} \rangle \end{aligned}$$

► E-BNF

$$\begin{aligned} \langle \textit{expr} \rangle &\rightarrow \langle \textit{term} \rangle (+|-) \langle \textit{term} \rangle \\ \langle \textit{term} \rangle &\rightarrow \langle \textit{factor} \rangle (*|/) \langle \textit{factor} \rangle \end{aligned}$$

Diagramas de Vías

Diagramas sintácticos: muestran las cadenas válidas en forma gráficas

Operación	BNF	Diagrama de Comway
Yuxtaposición	AB	
Opción	$A B$	
	ϵB	
Repetición	1 o más veces $\{B\}$	
	0 o más veces $[B]$	

Ejemplo en BNF

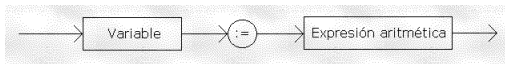
```
<comando asignación> ::= <variable> := <expresión aritmética>  
<expresión aritmética> ::= <término> | <expresión aritmética> + <término> | <expresión aritmética> - <término>  
<término> ::= <primario> | <término> * <primario> | <término> / <primario>  
<primario> ::= <variable> | <número> | (<expresión aritmética>)  
<variable> ::= <identificador> | <identificador> [<lista subíndice>]  
<lista subíndice> ::= <expresión aritmética> | <lista subíndice> , <expresión aritmética>
```

Ejemplo en BNF Extendido

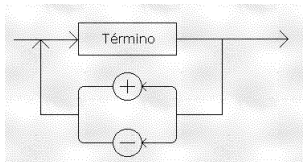
```
<comando asignación> ::= <variable> := <expresión aritmética>  
<expresión aritmética> ::= <término> { [+|-] <término> }*  
<término> ::= <primario> { [*|/] <primario> }  
<primario> ::= <variable> | <número> | (<expresión aritmética>)  
<variable> ::= <identificador> | <identificador> [<lista subíndice>]  
<lista subíndice> ::= <expresión aritmética> { , <expresión aritmética> }*
```

Ejemplo en Diagramas de Vías

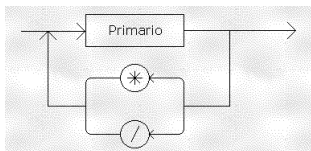
Comando asignación



Expresión aritmética.

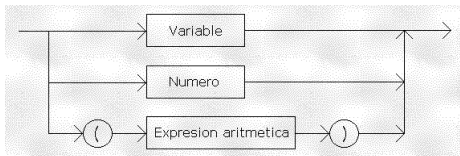


Término.



Ejemplo en Diagramas de Vías

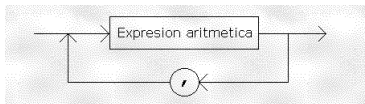
Primario.



Variable.



Lista subíndice.



Ambigüedad sintáctica

Algunas características para determinar si una gramática es ambigua:

- ▶ si la gramática genera una sentencia con más de una derivación a izquierda.
- ▶ si la gramática genera una sentencia con más de una derivación a derecha.

Gramáticas de Atributos

► Semántica estática:

- Una gramática de atributos es un mecanismo utilizado para describir más de lo que se puede con la gramática libre de contexto pero que pueden ser evaluados en compilación
- Es una extensión de las gramáticas libres de contexto, permite la descripción de ciertas reglas del lenguaje como la compatibilidad de tipos.
- La idea es agregarle a la gramática tradicional, información semántica a lo largo del árbol de parsing.

► Gramática de Atributos se define como

- una gramática libre de contexto a la cual se le han agregado atributos, funciones para la evaluación de los atributos y funciones predicado.

Gramáticas de Atributos

- ▶ **Atributo:**
 - ▶ asociados a los símbolos de la gramática, son similares a las variables (en el sentido de que pueden tener valores asociados)
- ▶ **Funciones para la evaluación de atributos:**
 - ▶ llamadas funciones semánticas, están asociadas a las reglas de la gramática. Son utilizadas para indicar cómo los atributos especificados son computados.
- ▶ **Funciones predicado:**
 - ▶ determinan la regla semántica estática del lenguaje, están asociadas a reglas de la gramática.

Gramáticas de atributos

Definición

Una gramática de atributos es una gramática libre de contexto $G=(S,N,T,P)$ con los siguientes agregados:

- ▶ Para cada símbolo de la gramática X , hay un conjunto de atributos.
- ▶ El conjunto se divide en dos conjuntos disjuntos **atributos sintetizados y heredados** respectivamente
- ▶ Hay atributos intrínsecos en las hojas del árbol de derivación
- ▶ Cada regla tiene un conjunto de **funciones** que definen ciertos atributos de los no-terminales en la regla.
- ▶ Cada regla tiene un conjunto (posiblemente vacío) de **predicados** para chequear la consistencia de los atributos.

EXAMPLE 3.6

An Attribute Grammar for Simple Assignment Statements

1. Syntax rule: $\langle \text{assign} \rangle \rightarrow \langle \text{var} \rangle = \langle \text{expr} \rangle$
Semantic rule: $\langle \text{expr} \rangle.\text{expected_type} \leftarrow \langle \text{var} \rangle.\text{actual_type}$
2. Syntax rule: $\langle \text{expr} \rangle \rightarrow \langle \text{var} \rangle[2] + \langle \text{var} \rangle[3]$
Semantic rule: $\langle \text{expr} \rangle.\text{actual_type} \leftarrow$
 if $(\langle \text{var} \rangle[2].\text{actual_type} = \text{int})$ and
 $(\langle \text{var} \rangle[3].\text{actual_type} = \text{int})$
 then int
 else real
 end if

Predicate: $\langle \text{expr} \rangle.\text{actual_type} == \langle \text{expr} \rangle.\text{expected_type}$
3. Syntax rule: $\langle \text{expr} \rangle \rightarrow \langle \text{var} \rangle$
Semantic rule: $\langle \text{expr} \rangle.\text{actual_type} \leftarrow \langle \text{var} \rangle.\text{actual_type}$
Predicate: $\langle \text{expr} \rangle.\text{actual_type} == \langle \text{expr} \rangle.\text{expected_type}$
4. Syntax rule: $\langle \text{var} \rangle \rightarrow A \mid B \mid C$
Semantic rule: $\langle \text{var} \rangle.\text{actual_type} \leftarrow \text{look-up}(\langle \text{var} \rangle.\text{string})$

The look-up function looks up a given variable name in the symbol table and returns the variable's type.

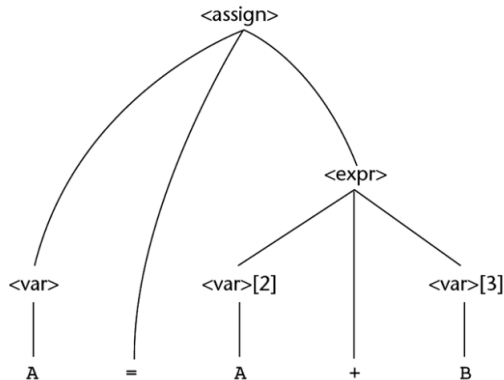
Gramáticas de Atributos

Ejemplo: $A = A + B$

1. Crear el árbol sintáctico

Figure 3.6

A parse tree for
 $A = A + B$



Gramáticas de Atributos

2. Agregar los atributos intrínsecos

3. Heredar atributos

EXAMPLE 3.6

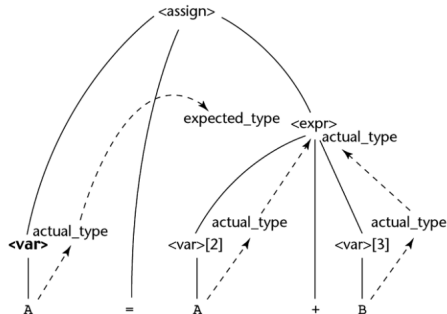
An Attribute Grammar for Simple Assignment Statements

1. Syntax rule: $\langle \text{assign} \rangle \rightarrow \langle \text{var} \rangle = \langle \text{expr} \rangle$
Semantic rule: $\langle \text{expr} \rangle.\text{expected_type} \leftarrow \langle \text{var} \rangle.\text{actual_type}$
2. Syntax rule: $\langle \text{expr} \rangle \rightarrow \langle \text{var} \rangle[2] + \langle \text{var} \rangle[3]$
Semantic rule: $\langle \text{expr} \rangle.\text{actual_type} \leftarrow$
 if $\langle \text{var} \rangle[2].\text{actual_type} = \text{int}$ and $\langle \text{var} \rangle[3].\text{actual_type} = \text{int}$
 then int
 else real
 end if
Predicate: $\langle \text{expr} \rangle.\text{actual_type} == \langle \text{expr} \rangle.\text{expected_type}$
3. Syntax rule: $\langle \text{expr} \rangle \rightarrow \langle \text{var} \rangle$
Semantic rule: $\langle \text{expr} \rangle.\text{actual_type} \leftarrow \langle \text{var} \rangle.\text{actual_type}$
Predicate: $\langle \text{expr} \rangle.\text{actual_type} == \langle \text{expr} \rangle.\text{expected_type}$
4. Syntax rule: $\langle \text{var} \rangle \rightarrow A \mid B \mid C$
Semantic rule: $\langle \text{var} \rangle.\text{actual_type} \leftarrow \text{look-up}(\langle \text{var} \rangle.\text{string})$

The look-up function looks up a given variable name in the symbol table and returns the variable's type.

Figure 3.7

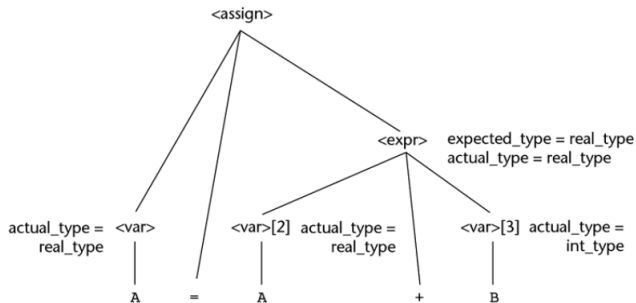
The flow of attributes in the tree



4. Sintetizar atributos

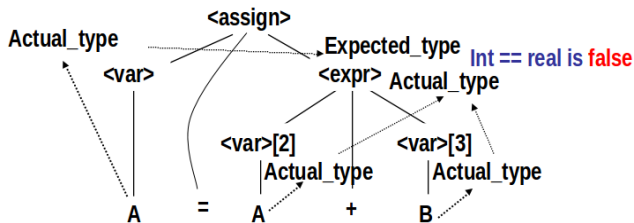
Figure 3.8

A fully attributed
parse tree



Gramáticas de Atributos

5. Determinar si es correcto



Desafíos

- Dada la siguiente gramática ambigua, dar otra que no lo sea.

$$\begin{aligned} \langle \text{ifSent} \rangle &\Rightarrow \text{if } \langle \text{expLog} \rangle \text{ then } \langle \text{sent} \rangle \\ &\quad | \quad \text{if } \langle \text{expLog} \rangle \text{ then } \langle \text{sent} \rangle \text{ else } \langle \text{sent} \rangle \\ \langle \text{sent} \rangle &\Rightarrow \langle \text{ifSent} \rangle \end{aligned}$$

- Dar una gramática ambigua y otra que no lo es para la asignación cuyas operaciones pueden ser sumas y divisiones solamente. El orden de precedencia es primero la división y luego la suma. Las expresiones que se pueden sumar o dividir son identificadores solamente.

Bibliografía

- ▶ Sebesta, Robert, Concepts of Programming Languages, 9th Edition. 2009. Capítulo 1 y 3.
- ▶ Pratt, Terrance W., Programming Languages: Design and Implementation, 3rd Edition. Capítulo 3 (y Cap. 4 de la 4ta edición).
- ▶ Material de apoyo en pedco.
- ▶ (opcional) Louden, Kenneth, Programming Languages: principles and practices. 3rd Edition 2011. Cengage Learning. Capítulo 1 y 6 .