

Desarrollo de lenguajes y Compiladores [MII-771]

Capítulo 1: Lenguajes y Gramáticas Formales

Dr. Ricardo Soto

[ricardo.soto@ucv.cl]

[<http://www.inf.ucv.cl/~rsoto>]

Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

Marzo, 2010



1. Introducción

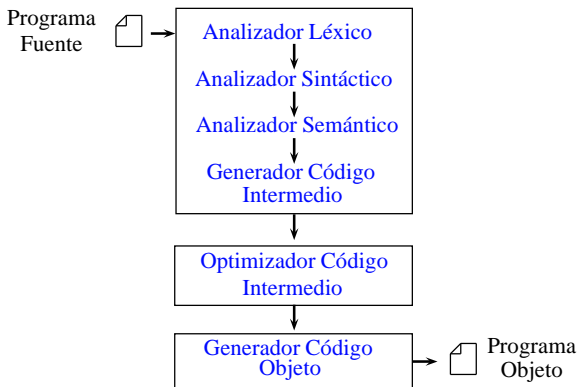
- El desarrollo de un compilador genera comúnmente la impresión de ser una tarea ardua... **muchas líneas de código** y de conocimientos avanzados en **teoría de autómatas**.
- En la actualidad, existen diversas herramientas que permiten al usuario **abstraerse de complejos** algoritmos...

Motivación??...

- Necesidad de la industria de definir lenguajes propios...
 - **Estandarizar** el desarrollo de sus productos.
 - **Modelar** problemas y distribuir el conocimiento de manera apropiada.
 - **Facilitar** las tareas de los programadores.

1. Introducción

Un **compilador** es un programa que traduce un programa escrito en un lenguaje *a* (lenguaje fuente) a un lenguaje *b* (lenguaje objeto).



2. Alfabetos y palabras

Un **alfabeto** es un conjunto finito y no vacío de elementos llamados símbolos o letras.

Una **palabra** o cadena sobre un alfabeto V es una cadena finita de símbolos del alfabeto.

Notaciones:

- λ denota a una cadena de longitud 0, también conocida como palabra vacía.
- V_n denota al conjunto de todas las palabras de longitud n sobre V
- V^* denota al conjunto de todas las cadenas de cualquier longitud sobre V .
- V^+ denota al conjunto de todas las cadenas de cualquier longitud sobre V , excepto la vacía.
- Un elemento de V_n es una cadena del tipo $a_1 a_2 \dots a_n$ donde cada $a_i \in V$.

3. Lenguajes Formales

Llamamos **lenguaje** sobre el alfabeto V a cualquier subconjunto de V^* .

Especificación de lenguajes:

- **Extensión (lenguajes finitos)**

$L = \{a, aa, aaa\}$ es un lenguaje sobre el alfabeto $V = \{a\}$

$L = \{aba, cab, aaabc\}$ es un lenguaje sobre el alfabeto $V = \{a, b, c\}$

- **Comprensión (lenguajes infinitos)**

$L = \{a(bc)^n \mid n \geq 1\}$

4. Gramáticas Formales

El uso de **gramáticas** es otra forma de describir un lenguaje en forma general y rigurosa.

Definiciones:

- Una gramática es una cuadrupla $G = (V_N, V_T, S, P)$ donde:
 - V_T es el alfabeto de símbolos terminales.
 - V_N es el alfabeto de símbolos no terminales, de forma que $V_T \cap V_N = \emptyset$, y denotamos con V al alfabeto total de la gramática, esto es, $V = V_N \cup V_T$
 - S es el símbolo inicial y se cumple que $S \in V_N$
 - P es un conjunto finito de reglas de producción.

4. Gramáticas Formales

Definiciones:

- Una regla de producción es un par ordenado (α, β) de forma que:
 - $\alpha = \gamma_1 A \gamma_2$, donde:
 - $\gamma_1, \gamma_2 \in (V_N \cup V_T)^*$
 - $A \in V_N$
 - $\beta \in (V_N \cup V_T)^*$
- Una regla de producción (α, β) se suele escribir como $\alpha \rightarrow \beta$

Ejemplo

- Definir una gramática para el lenguaje $L = \{a(bc)^n | n \geq 1\}$:
- Solución:
 $S \rightarrow aB$
 $B \rightarrow bcB | bc$
donde $V_N = \{S, B\}$ y $V_T = \{a, b, c\}$.

- Definir una gramática para los siguientes lenguajes:
 - $L_1 = \{a^n b^m \mid n \geq 4 \text{ y } m \geq 3\}$
 - $L_2 = \{a^n b^n \mid n > 0\}$
 - $L_3 = \{a^n b^{2n} \mid n > 0\}$
 - $L_4 = \{a^n b^n c^m d^m \mid n > 0 \text{ y } m > 0\}$

4.2 Notación BNF (Backus-Naus-Form)

- BNF es una metasintaxis utilizada para definir gramáticas
- BFN y sus extensiones son ampliamente utilizadas para definir gramáticas de lenguajes de programación.
- En BNF, símbolos no terminales se definen entre ángulos ($\langle \rangle$) y producciones se definen utilizando el símbolo ::=.

Ejemplos

```
1. <S> ::= a<B>  
   <B> ::= bc<B>|bc
```

```
2. <class-dec> ::= class <identifier> { <class-body> }
```

4.3 Notación EBNF (Extended BNF)

- EBNF introduce el uso de paréntesis cuadrados para símbolos opcionales y llaves para repeticiones.
- El uso de ángulos para símbolos no terminales no es obligatorio.
- Introduce el uso de comillas en terminales para evitar ambigüedad con símbolos reservados de EBNF.

Símbolo opcional

```
class-dec ::= "class" identifier ["extends" identifier]
           "{" class-body "}"
```

Repetición

```
number ::= {digit}
```

4.4 Otras convenciones

- $*$ denota desde cero a n repeticiones
- $+$ denota desde una a n repeticiones
- $()$ para agrupaciones

Repetición

```
number ::= digit+
```

Opción y repetición

```
import-dec ::= identifier ( "." identifier ) * [ "." "*" ] ";"
```

- Utilize EBNF para construir las siguientes gramáticas:
 - Número entero.
 - Número real.
 - Letra.
 - Palabra.
 - Dirección Postal.
Ej: Juan Maldonado Perez,
6 norte 1234,
Viña del Mar,
Chile
 - Una expresión.
 - `if-else` en Java.
 - `for` en Java.
 - Clase Java.

4.5 Expresiones Regulares

- Las expresiones regulares también permiten especificar lenguajes regulares.
- Las expresiones regulares son de gran utilidad en editores de texto y aplicaciones para buscar y manipular textos.
- En la actualidad existe gran soporte para el uso de expresiones regulares (Perl, PHP, bibliotecas Java, bibliotecas .NET, shell Unix, etc).
- Similar a EBNF:
 - * denota desde cero a n repeticiones
 - + denota desde una a n repeticiones
 - { n } denota n repeticiones
 - { m, n } denota de m a n repeticiones
 - ? denota elemento opcional
 - () para agrupaciones

Defina los siguientes lenguajes mediante expresiones regulares:

- $L_1 = \{(ab)^n | n > 1\}$
- $L_2 = \{a^n b^m | n \geq 4 \text{ y } m \geq 3\}$
- Todas las palabras que empiezen con a y terminen con o
- Todas las palabras que empiezen con a, tengan una s y terminen con o
- Todas las palabras que tengan entre 5 y 8 letras