Teoría de Autómatas y Compiladores [ICI-445]
Capítulo 1: Lenguajes y Gramáticas Formales

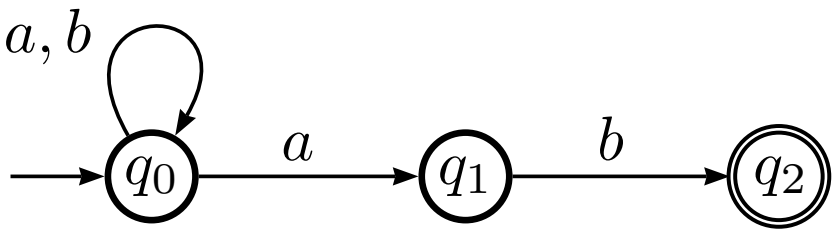
Dr. Ricardo Soto
[ricardo.soto@ucv.cl]
[http://www.inf.ucv.cl/~rsoto]

Escuela de Ingeniería Informática
Pontificia Universidad Católica de Valparaíso

Marzo, 2010

1. Introducción

Un **autómata** es una máquina teórica que lee instrucciones en forma de símbolos y cambia de estado según éstas.



Áreas de aplicación de la teoría de autómatas:

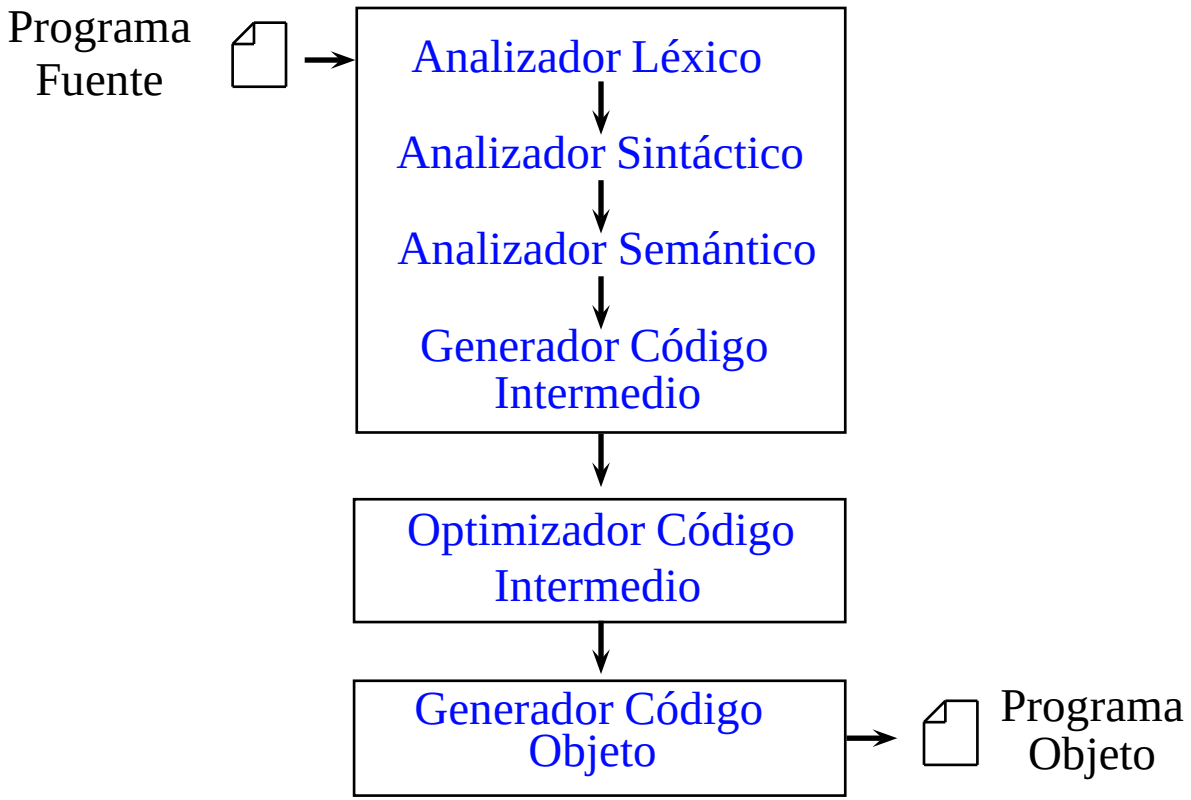
- Comunicaciones.
- Teoría de Control.
- Circuitos secuenciales.
- Recocimiento de Patrones.

⋮

- **Compiladores**.

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:

- $|\omega|$ denota la longitud de la cadena ω.
- 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_0 denota al conjunto cuyo único elemento es la palabra vacía, es decir, $V_0 = \{\lambda\}$.
- 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 $a_1 a_2 \ldots a_n$ 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)**
  $L = \{a(bc)^n | n >= 1\}$

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 $V_T \cap V_N = \varnothing$, 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 $L = \{a(bc)^n | n >= 1\}$:
- Solución:
  $S \rightarrow aB$
  $B \rightarrow bcB|bc$
  donde $V_N = \{S, B\}$ y $V_T = \{a, b, c\}$.

Ejercicios

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

4.1 Jerarquía de Chomsky

Noam Chomsky clasificó las gramáticas en cuatro familias:

- **Tipo 3 (Gramáticas regulares).**
  - Lineales por la derecha. Producciones del tipo:
    $A \to \lambda$
    $A \to \alpha$
    $A \to \alpha B$
    donde $A, B \in V_N$ y $\alpha \in V_T$.
  - Lineales por la izquierda. Producciones del tipo:
    $A \to \lambda$
    $A \to \alpha$
    $A \to B\alpha$
    donde $A, B \in V_N$ y $\alpha \in V_T$.
  - Los lenguajes generados por estas gramáticas se llaman lenguajes regulares (lenguajes de la clase $\mathcal{L}_3$).

4.1 Jerarquía de Chomsky

- **Tipo 2 (Gramáticas libres de contexto).**
  - Producciones del tipo:
    $A \rightarrow \alpha$
    donde $A \in V_N$ y $\alpha \in (V_N \cup V_T)^*$.
  - Los lenguajes generados por estas gramáticas se llaman lenguajes libres de contexto (lenguajes de la clase $\mathcal{L}_2$).
  - La mayoría de los lenguajes de programación pertenecen a esta categoría.
- **Tipo 1 (Gramáticas sensibles al contexto).**
  - Producciones del tipo:
    $\alpha A \beta \rightarrow \alpha \gamma \beta$
    $S \rightarrow \lambda$
    donde $A, S \in V_N$,
    $\alpha, \beta \in V^*$ y $\gamma \in V^+$.
  - Los lenguajes generados por estas gramáticas se llaman lenguajes sensibles al contexto (lenguajes de la clase $\mathcal{L}_1$).
  - La sensibilidad al contexto se puede interpretar en la 1era regla de producción: A puede ser reemplazada por γ sólo cuando A aparezca en el contexto de α y β.

4.1 Jerarquía de Chomsky

- **Tipo 0 (Gramáticas sin restricciones o Gramáticas con estructura de frase).**
  - Producciones del tipo:
    $\alpha \rightarrow \beta$
    donde $\alpha \in (V^* \cdot V_N \cdot V^*)$ y $\beta \in V^*$
  - Los lenguajes generados por estas gramáticas se llaman lenguajes libres de contexto (lenguajes de la clase $\mathcal{L}_0$).

Clases de lenguajes

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

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 )* ["." "*"] ";"
```

Ejercicios

- 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

Ejercicios

Defina los siguientes lenguajes mediante expresiones regulares:

- $L_1 = \{(ab)^n | n > 1\}$
- $L_2 = \{a^n b^m | n \geq 4 \ y \ m \geq 3\}$
- 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