Introducción al análisis léxico

Clase 16

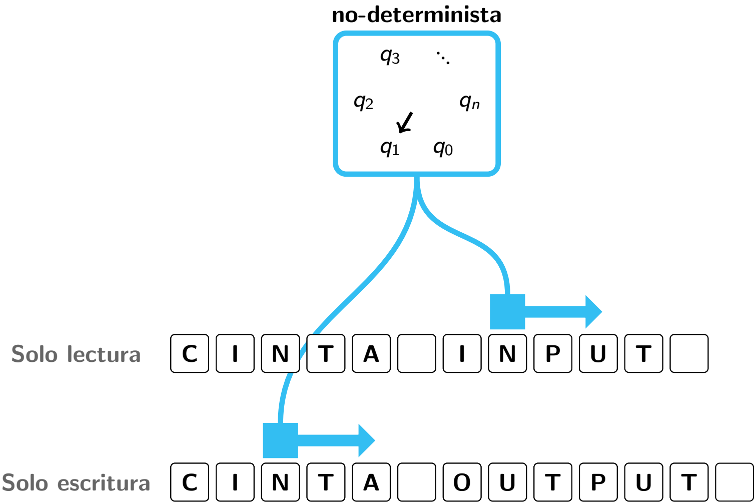
IIC2223 / IIC2224

Prof. Cristian Riveros

Outline

Outline

Transductores (recordatorio)



Definición de transductor (recordatorio)

Definición

Un transductor (*en inglés*, transducer) es una tupla:

$$\mathcal{T} \quad = \quad (Q, \Sigma, \Omega, \Delta, I, F)$$

- Q es un conjunto finito de estados.
- Σ es el alfabeto de input.
- Ω es el alfabeto de **output**.
- $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times Q$ es la **relación de transición**.
- $I \subseteq Q$ es un conjunto de estados iniciales.
- $F \subseteq Q$ es el conjunto de estados finales.

Función definida por un transductor (recordatorio)

Sea $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ un transductor y $u, v \in \Sigma^*$.

Definiciones

- $\mathcal{T}$ **entrega** v con **input** u si existe una configuración **inicial** (q_0, u, ϵ) y una configuración **final** (q_f, ϵ, v) tal que:

$$(q_0, u, \epsilon) \vdash_{\mathcal{T}}^{*} (q_f, \epsilon, v)$$

- Se define la función $[\![\mathcal{T}]\!] : \Sigma^* \to 2^{\Omega^*}$:

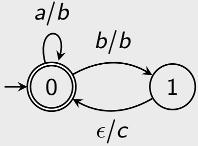
$$[\![\mathcal{T}]\!](u) \;=\; \{v \in \Omega^* \mid \mathcal{T} \text{ entrega } v \text{ con input } u\}$$

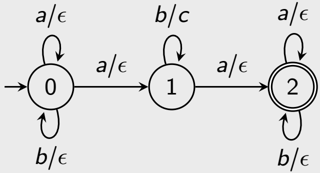
- Se dice que $f : \Sigma^* \to 2^{\Omega^*}$ es una **función racional** si existe un transductor $\mathcal{T}$ tal que $f = [\![\mathcal{T}]\!]$.

Un transductor define una función de palabras a conjunto de palabras.

Función definida por un transductor (recordatorio)

¿qué función define cada transductor?

1. 

2. 

Funciones versus relaciones

Dos interpretaciones para un transductor

1. $\mathcal{T}$ define la **función** $[\![\mathcal{T}]\!] : \Sigma^* \to 2^{\Omega^*}$:

$$[\![\mathcal{T}]\!](u) \;=\; \{v \in \Omega^* \mid \mathcal{T} \text{ entrega } v \text{ con input } u\}$$

2. $\mathcal{T}$ define la **relación** $[\![\mathcal{T}]\!] \subseteq \Sigma^* \times \Omega^*$:

$$(u, v) \in [\![\mathcal{T}]\!] \quad \text{si, y solo si,} \quad \mathcal{T} \text{ entrega } v \text{ con input } u$$

Desde ahora, hablaremos de función o relación **indistintamente** y hablaremos de las **relaciones racionales** (definidas por un transductor).

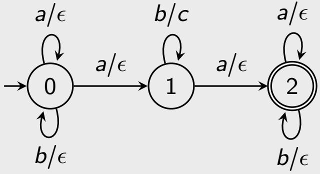
Lenguaje de input y lenguaje de output

Definiciones

Para una relación $R \subseteq \Sigma^* \times \Omega^*$ se define:

- $\pi_1(R) \; = \; \{ \; u \in \Sigma^* \mid \exists v \in \Omega^*. \; (u, v) \in R \; \}$.
- $\pi_2(R) \; = \; \{ \; v \in \Omega^* \mid \exists u \in \Sigma^*. \; (u, v) \in R \; \}$.

¿cuál es el lenguaje definido por $\pi_1([\![\mathcal{T}]\!])$ y $\pi_2([\![\mathcal{T}]\!])$?



Lenguaje de input y lenguaje de output

Definiciones

Para una relación $R \subseteq \Sigma^* \times \Omega^*$ se define:

- $\pi_1(R) \;=\; \{\; u \in \Sigma^* \mid \exists v \in \Omega^*.\ (u, v) \in R \;\}.$
- $\pi_2(R) \;=\; \{\; v \in \Omega^* \mid \exists u \in \Sigma^*.\ (u, v) \in R \;\}.$

Teorema

Si $\mathcal{T}$ es un transductor,
entonces $\pi_1(\llbracket \mathcal{T} \rrbracket)$ y $\pi_2(\llbracket \mathcal{T} \rrbracket)$ son lenguajes regulares sobre Σ y Ω, resp.

Demostración: $\pi_1(\llbracket \mathcal{T} \rrbracket)$

Para $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$, defina $\mathcal{A}_1 = (Q, \Sigma, \Delta_1, I, F)$ tal que:

$$(p, a, q) \in \Delta_1 \quad \text{si, y solo si,} \quad \exists b \in \Omega \cup \{\epsilon\}.\ (p, a, b, q) \in \Delta$$

y demuestre que $\mathcal{L}(\mathcal{A}_1) = \pi_1(\llbracket \mathcal{T} \rrbracket)$. ■

Operaciones de relaciones

Teorema

Sea $\mathcal{T}_1$ y $\mathcal{T}_2$ dos transductores con Σ y Ω alfabetos de input y output.

Las siguientes son **relaciones racionales**.

1. $[\![\mathcal{T}_1]\!] \cup [\![\mathcal{T}_2]\!] = \{(u, v) \in \Sigma^* \times \Omega^* \mid (u, v) \in [\![\mathcal{T}_1]\!] \vee (u, v) \in [\![\mathcal{T}_2]\!]\}$.
2. $[\![\mathcal{T}_1]\!] \cdot [\![\mathcal{T}_2]\!] = \{(u_1 u_2, v_1 v_2) \in \Sigma^* \times \Omega^* \mid (u_1, v_1) \in [\![\mathcal{T}_1]\!] \wedge (u_2, v_2) \in [\![\mathcal{T}_2]\!]\}$.
3. $[\![\mathcal{T}_1]\!]^* = \bigcup_{k=0}^{\infty} [\![\mathcal{T}_1]\!]^k$.

Demostración.

Operaciones de relaciones

Teorema

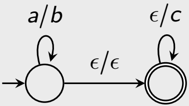
Existen transductores $\mathcal{T}_1$ y $\mathcal{T}_2$ sobre Σ y Ω tal que:

$$[\![\mathcal{T}_1]\!] \cap [\![\mathcal{T}_2]\!] \;=\; \{(u,v) \in \Sigma^* \times \Omega^* \mid (u,v) \in [\![\mathcal{T}_1]\!] \wedge (u,v) \in [\![\mathcal{T}_2]\!]\}$$

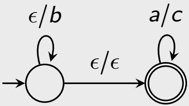
NO es una relación racional.

Demostración

Considere los siguientes transductores:



$[\![\mathcal{T}_1]\!] = \{(a^n, b^n c^m) \mid n \geq 0, m \geq 0\}$



$[\![\mathcal{T}_2]\!] = \{(a^n, b^m c^n) \mid n \geq 0, m \geq 0\}$

pero $[\![\mathcal{T}_1]\!] \cap [\![\mathcal{T}_2]\!] = \{(a^n, b^n c^n) \mid n \geq 0\}$,
y por lo tanto $[\![\mathcal{T}_1]\!] \cap [\![\mathcal{T}_2]\!]$ no es racional (¿por qué?)

Outline

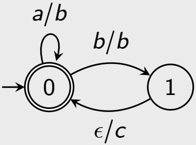
Transductores para funciones de palabras a palabras

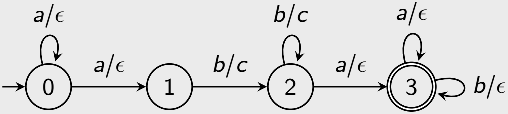
Definición

Decimos que un transductor $\mathcal{T}$ define una función (parcial) si:

$$\text{para todo } u \in \Sigma^* \text{ se tiene que } |\,[\![\mathcal{T}]\!](u)\,| \leq 1.$$

Ejemplos

- 

- 

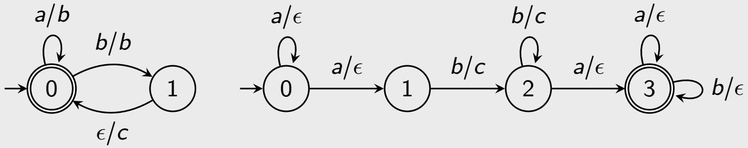
Transductores para funciones de palabras a palabras

Definición

Decimos que $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ es **determinista** si cumple que:

1. $\mathcal{T}$ define una función parcial $[\![\mathcal{T}]\!] : \Sigma^* \to \Omega^*$.
2. para todo $(p, a_1, b_1, q_1) \in \Delta$ y $(p, a_2, b_2, q_2) \in \Delta$,
   si $a_1 = a_2$, entonces $b_1 = b_2$ y $q_1 = q_2$.
3. si $(p, \epsilon, b, q) \in \Delta$, entonces
   para todo $(p, a', b', q') \in \Delta$, se tiene que $(a', b', q') = (\epsilon, b, q)$.

¿cuál transductor es determinista?



Transductores para funciones de palabras a palabras

Definición

Decimos que $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ es **determinista** si cumple que:

1. $\mathcal{T}$ define una función parcial $[\![\mathcal{T}]\!] : \Sigma^* \to \Omega^*$.
2. para todo $(p, a_1, b_1, q_1) \in \Delta$ y $(p, a_2, b_2, q_2) \in \Delta$,

   si $a_1 = a_2$, entonces $b_1 = b_2$ y $q_1 = q_2$.
3. si $(p, \epsilon, b, q) \in \Delta$, entonces

   para todo $(p, a', b', q') \in \Delta$, se tiene que $(a', b', q') = (\epsilon, b, q)$.

Propiedad

Si $\mathcal{T}$ es un transductor determinista, entonces $\mathcal{T}$ define una función parcial.

¿es verdad que si $\mathcal{T}$ define una función parcial, entonces $\mathcal{T}$ es deterministas?

Transductores para funciones de palabras a palabras

Definición

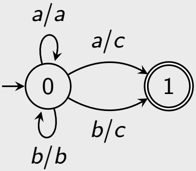
Decimos que $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ es **determinista** si cumple que:

1. $\mathcal{T}$ define una función parcial $[\![\mathcal{T}]\!] : \Sigma^* \to \Omega^*$.
2. para todo $(p, a_1, b_1, q_1) \in \Delta$ y $(p, a_2, b_2, q_2) \in \Delta$,

   si $a_1 = a_2$, entonces $b_1 = b_2$ y $q_1 = q_2$.
3. si $(p, \epsilon, b, q) \in \Delta$, entonces

   para todo $(p, a', b', q') \in \Delta$, se tiene que $(a', b', q') = (\epsilon, b, q)$.

Contraejemplo



¿cuál es la ventaja de los transductores deterministas?

Outline

Sintaxis y semántica de un lenguaje de programación

Definición

1. La **sintaxis** de una lenguaje es un conjunto de reglas que describen los programas válidos que tienen significado.

¿cuáles son programas válidos en Python?

- ```
  myint = 7
  print myint
  ```
- ```
  mystring = 'hello"
  print(mystring)
  ```

Sintaxis y semántica de un lenguaje de programación

Definición

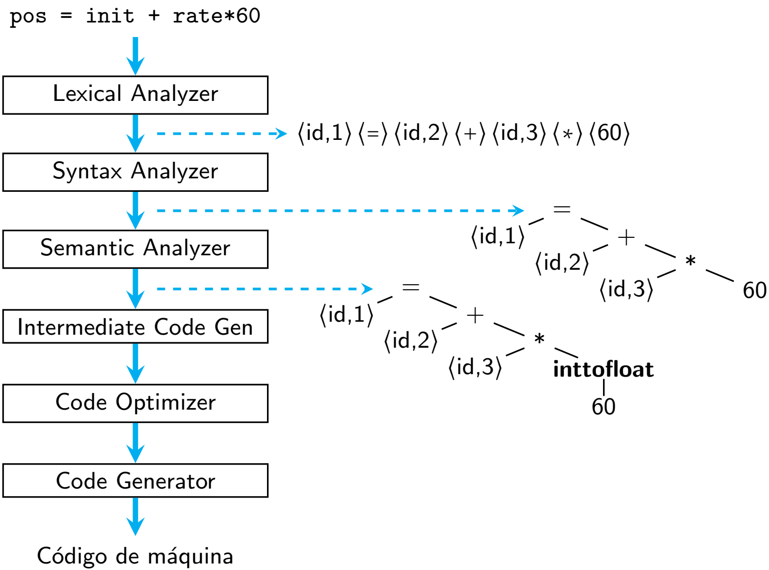
1. La **sintaxis** de una lenguaje es un conjunto de reglas que describen los programas válidos que tienen significado.

2. La **semántica** de un lenguaje define el significado de un programa correcto según la sintaxis.

¿cuál es la semántica de este programa en Python?

```
mylist = []
mylist.append(1)
mylist.append(2)
for x in mylist:
    print(x)
```

La estructura de un compilador



Verificación de sintaxis

En este proceso se busca:

- verificar la sintaxis de un programa.
- entregar la estructura de un programa (árbol de parsing).

Consta de tres etapas:

1. Análisis léxico (**Lexer**).
2. Análisis sintáctico (**Parser**).
3. Análisis semántico.

Por ahora, solo nos interesará el **Lexer**.

(el funcionamiento del **Parser** lo veremos cuando veamos gramáticas)

Análisis léxico (Lexer)

- El análisis léxico consta en dividir el programa en una sec. de **tokens**.
- Un **token** (o lexema) es un substring (válido) dentro de un programa.
- Un **token** esta compuesto por:
  - tipo.
  - valor (el valor mismo del substring).

Análisis léxico (Lexer)

Tipos usuales de **tokens** en lenguajes de programación:

- **number** (constante): 2, 345, 495, ...
- **string** (constante): 'hello', 'iloveTDA', ...
- **keywords**: `if`, `for`, ...
- **identificadores**: pos, init, rate ...
- **delimitadores**: '{', '}', '(', ')', ',', ...
- **operadores**: '=', '+', '<', '<=', ...

Análisis léxico (Lexer)

Ejemplo

```
pos = init + rate * 60
```

Tipo	Valor
id	`pos`
EQ	`=`
id	`init`
PLUS	`+`
id	`rate`
MULT	`*`
number	60

Todos los espacios, saltos de línea y caracteres de formato son **omitidos**

Cierre de clase

En esta clase vimos:

1. Propiedades de transductores.
2. Modelo de transductor determinista.
3. Introducción al análisis léxico.

Próxima clase: Aplicación de transductores al análisis léxico