

Sintaxis y Semántica del Lenguaje

2° Ing. En Sistemas de
Información – UTN FRVM

Integrantes

■ Docentes

- ❑ Ing. Mario Rinaldi
- ❑ Dr. Jorge A. Palombarini

■ Ayudantes Alumnos

- ❑ Ignacio Sarmiento
 - ❑ Mateo Digliodo
-

Sintaxis: Regularización de la materia

■ Tipos de actividades:



- 6 Trabajos Prácticos Grupales de Entrega Obligatoria

- 2 Parciales + Tests

- Actividades de Cátedra: lectura y visualización de material

Regularización y Aprob. Directa



Aprobación Directa

- Entregar en **tiempo** y **forma** 6 TPS.
 - **Tiempo:** Versión Impresa entregada antes del final de la clase en que vence el TP. Versión Digital hasta las 23:59 del mismo día.
 - **Forma:** Una versión digital y una versión impresa en carpeta con carátula especificando: Nro. De Grupo, Integrantes, Nro. de Práctico.
 - **Puntaje Final de la Materia** = $0.33(\text{Puntaje por entrega en tiempo y forma de la totalidad de los TPS}) + 0.33(\text{Puntaje Parcial 1} + \text{Tests}) + 0.33(\text{Puntaje Parcial 2} + \text{Tests})$
-
- Aprobar los dos parciales o un parcial más el recuperatorio correspondiente (60 pts. o más), y obtener un **puntaje final de la materia** de **80 puntos o más**.

- Asistir al 75% de las clases.

Regularización y Aprobación No Directa

Aprobar los dos parciales o un parcial más el recuperatorio correspondiente (60 pts. o más), y obtener un **puntaje final de la materia** de entre **60 y 79** puntos.

- Asistir al 75% de las clases.

- Rendir un Examen Final Teórico Práctico.

Consideraciones

- Los TPs son grupales (4 personas por grupo).

Planificación

Semana	Clase 1	Clase 2	Actividad
1	15/8/2018	16/8/2018	Presentación Materia. Lenguajes Regulares y ERs. Lógica. Trabajar sobre TP 1.
2	22/8/2018	23/8/2018	Trabajar sobre TP 1. Estados y AFD.
3	29/8/2018	30/8/2018	Librería automata-lib. Trabajar TP2.
4	5/9/2018	6/9/2018	Trabajar TP 2. Entrega TP 1.
5	12/9/2018	13/9/2018	AFD. Trabajar TP2.
6	19/9/2018	20/9/2018	Trabajar TP2.
7	26/9/2018	27/9/2018	AFN. Operaciones entre autómatas. Convertir AFN - AFD. Trabajar TP3. Entrega TP2.
8	3/10/2018	4/10/2018	Trabajar TP3. Programación Lógica.
9	10/10/2018	11/10/2018	Equivalencia con AF. Bloques. GNFA. Trabajar TP4. Entrega TP3.
10	17/10/2018	18/10/2018	Parcial 1.
11	24/10/2018	25/10/2018	Trabajar TP4.
12	31/10/2018	1/11/2018	Gramáticas y Lenguajes Libres de Contexto. FNC. Ambigüedad. Trabajar TP5. Entrega TP4.
13	7/11/2018	8/11/2018	Trabajar TP5.
14	14/11/2018	15/11/2018	Consultas. Entrega TP5 y TP6
15	21/11/2018	22/11/2018	Parcial 2.
16	28/11/2018	29/11/2018	Recuperatorio 1 ó 2.
Tests: Se avisará con una semana de anticipación			

Consideraciones Previas

- Se dan por comprendidos conceptos generales de Teoría de Conjuntos y Lógica Proposicional. Existen dos guías optativas de repaso.
- También existe una guía optativa de repaso general.

Repositorio y Mail de la cátedra

- <https://github.com/sintaxisfrvm/sintaxis>
 - sintaxisfrvm@gmail.com
 - ❑ Consultas.
 - ❑ Trabajos Prácticos.
-

Bibliografía

- Autómatas y Lenguajes. Un enfoque de diseño. R. Brena
 - Introduction to the theory of computation – M. Sipser.
 - Introduction to automata theory, languages, and computation – J. Hopcroft
-

Actividad I

- Armar grupos de trabajo de 4 personas.
 - En una lista escribir los nombres de los integrantes y sus respectivos e-mails.
-

Lógica Proposicional

■ $p, q, r, \dots \rightarrow$ Proposiciones

■ \sim

■ \wedge

■ \vee

■ \rightarrow

■ \leftrightarrow

p	q	$\sim p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Si no vienes ya, nos vamos a desayunar.

Tiene coche y, sin embargo, no sabe conducir.

Juan canta sólo si está contento.

Teoría de Conjuntos

- Un **conjunto** es una agrupación de **elementos** que comparten alguna propiedad en común que permite definirlos.
- No es lo mismo un conjunto que un elemento de un conjunto. Diferenciar \in , y relacionarlo con diferenciar un **único símbolo inicial** de un **conjunto de símbolos finales**.

Teoría de Conjuntos

- Los conjuntos se pueden definir por comprensión y por extensión.
- Por comprensión: se utiliza la forma $\{x/x \dots\}$ que corresponde a la utilización de conjuntos formadores.
- Por ejemplo, un lenguaje es un **conjunto** de palabras.
- Existe el **conjunto vacío**. Cuál es? Cómo se simboliza?
- Existen operaciones entre conjuntos: unión, intersección, diferencia.

Teoría de Lenguajes

- Nociones más elementales:
 - **símbolo**, que es simplemente una representación distinguible de cualquier información. Los símbolos pueden ser cualesquiera, como w , 9 , $\#$, a, b, c , etc. Un **símbolo** es una entidad indivisible.
 - Un **alfabeto** es un conjunto no vacío de símbolos. Así, el alfabeto del idioma español,
 $\Sigma = \{a, b, c, \dots, z\}$.

Cadenas o Palabras

- Con los símbolos de un alfabeto es posible formar secuencias o cadenas de caracteres, tales como *mxzxptlk*, *balks*, *r*, etc. 4 Las cadenas de caracteres son llamadas también *palabras*.
- Un caso particular de cadena es la palabra vacía, ε o λ , la cual no tiene ninguna letra.
- La longitud de una palabra: cantidad de letras que contiene, contando las repeticiones;
 - Se denota $|w|$ para una palabra w . Ej. $|\text{perro}| = 5$. Cuando escribimos varias palabras o caracteres uno a continuación de otro, se supone que forman una sola palabra (se concatenan). La notación usada para denotar la concatenación de dos cadenas α y β es $\alpha\beta$.
- Por ejemplo, si $w = \text{abra}$ y $v = \text{cada}$, entonces $wvbra$ es la palabra *abracadabra*.

Concatenación

- La concatenación de palabras es *asociativa*, esto es, $(xy)z = x(yz)$, pero no **conmutativa**.
- La longitud de una concatenación cumple la propiedad: $|uv| = |u| + |v|$.
- Una palabra v es **subcadena** de otra w cuando existen cadenas x, y - posiblemente vacías - tales que $xvy = w$. Por ejemplo, “bora” es subcadena de “víbora”, y ε es *subcadena* de toda palabra.
- El conjunto de todas las palabras que se pueden formar con un alfabeto Σ es denotado convencionalmente por Σ^* . Por ejemplo, si $\Sigma = \{a, b\}$, $\Sigma^* = \{\varepsilon, a, aa, aaa, aaaa, \dots, b, bb, \dots, ab, aba, abb, \dots\}$. El conjunto Σ^* es infinito, pero enumerable.

Lenguajes, operaciones con lenguajes

- Un ***lenguaje*** es simplemente un conjunto de palabras.
- Así, ***{abracadabra}*** es un lenguaje (de una sola palabra), ***{ali, baba, y, sus, cuarenta, ladrones}*** es otro, Σ^* es otro, etc.
- Todo lenguaje es subconjunto de un Σ^*
- Puesto que los lenguajes son conjuntos, podemos efectuar con ellos todas las operaciones de los conjuntos (unión, intersección, diferencia).
- Definiremos además la operación de concatenación de lenguajes, escrita como $L1 \bullet L2$, como una extensión de la concatenación de palabras: **$L1 \bullet L2 = \{w | w = xy, x \in L1, y \in L2\}$** .

Ejemplo

- Por ejemplo, dados los lenguajes $L_1 = \{ca, ma\}$ y $L_2 = \{nta, sa\}$, la concatenación $L_1 L_2$ seria ***{canta, casa, manta, masa}***.
- Como se ve en este ejemplo, para calcular la concatenación de dos lenguajes hay que concatenar cada palabra del primero de ellos con cada una del segundo.

Estrella de Kleene

- Una operación más complicada es la llamada “*estrella de Kleene*” o “*cerradura de Kleene*”.
- **Definición.-** Si L es un lenguaje, L^* , llamado “*cerradura de Kleene*” de L , es el más pequeño conjunto que contiene:
 - La palabra vacía, ε
 - El conjunto L
 - Todas las palabras formadas por la concatenación de miembros de LPor ejemplo, si $L = \{abra, cadabra\}$, $L^* = \{\varepsilon, abra, abraabra, abracadabra, cadabraabra, \dots\}$

Ejemplos

- **Potencias de un alfabeto:** Conjunto de todos los strings de una cierta longitud que podemos formar a partir del alfabeto. Σ^k

$$\Sigma^0 = \{\epsilon\}$$

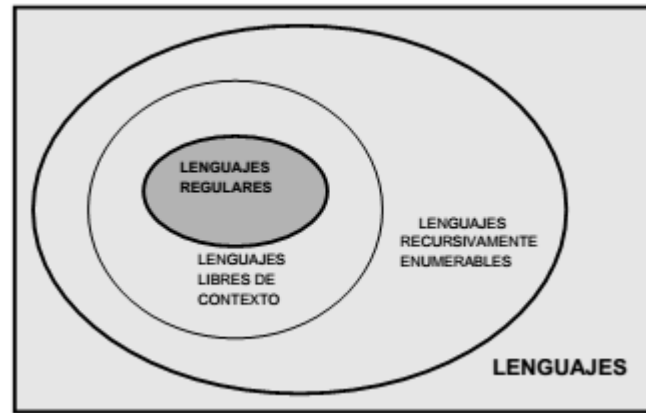
$$\text{If } \Sigma = \{0, 1\}, \text{ then } \Sigma^1 = \{0, 1\}, \Sigma^2 = \quad \Sigma^3 =$$

- El conjunto de todos los strings sobre un alfabeto se denota Σ^* .

$$\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}.$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup$$

Jerarquía de Chomsky



- Los “**Lenguajes Regulares**”, que es la clase más pequeña, e incluye a los lenguajes más simples. Un ejemplo de lenguaje regular es el conjunto de todos los números binarios.
- Los “**Lenguajes Libres de Contexto**”, que incluyen a los Lenguajes Regulares. Por ejemplo, la mayoría de los lenguajes de programación son Lenguajes Libres de Contexto.
- Los “**Lenguajes Recursivamente Enumerables**”, que incluyen a los Libres de Contexto (y por lo tanto a los Lenguajes Regulares)

Lenguajes Regulares

- Los *lenguajes regulares* se llaman así porque sus palabras contienen “regularidades” o repeticiones de los mismos componentes, como por ejemplo en el lenguaje ***L1*** siguiente:
- ***L1 = {ab, abab, ababab, abababab, . . .}***
- En este ejemplo se aprecia que las palabras de *L1* son simplemente repeticiones de “*ab*” cualquier numero de veces.
- Otro ejemplo más complicado sería el lenguaje *L 2*:
L2 = {abc, cc, abab, abccc, ababc, . . .}
- La regularidad en ***L2*** consiste en que sus palabras comienzan con repeticiones de “*ab*”

Lenguajes Regulares

- Adicionalmente a las repeticiones de esquemas simples, vamos a considerar que los lenguajes finitos son también regulares por definición.
- Por ejemplo, el lenguaje $L3 = \{\textit{anita}, \textit{lava}, \textit{la}, \textit{tina}\}$ es regular.
- Finalmente, al combinar lenguajes regulares uniéndolos o concatenándolos, también se obtiene un lenguaje regular.
- Por ejemplo, $L1 \cup L3 = \{\textit{anita}, \textit{lava}, \textit{la}, \textit{tina}, \textit{ab}, \textit{abab}, \textit{ababab}, \textit{abababab}, \dots\}$ es regular.
- También es regular una concatenación como $L3L3 = \{\textit{anitaanita}, \textit{anitalava}, \textit{anitala}, \textit{anitatina}, \textit{lavaanita}, \textit{lavalava}, \textit{lavala}, \textit{lavatina}, \dots\}$

Ejemplos

1. El lenguaje de todos los strings que inician con una n cantidad de 0's, y siguen con una n cantidad de 1's, y $n \geq 0$: $\{\epsilon, 01, 0011, 000111, \dots\}$
2. El lenguaje formado por todos los strings de 0's y 1's tal que ambos aparecen la misma cantidad de veces en el string: $\{\epsilon, 01, 10, 0011, 0101, 1001, \dots\}$
3. Σ^* y \emptyset , son lenguajes sobre cualquier alfabeto Σ
4. $\{\epsilon\}$, es un lenguaje sobre cualquier alfabeto
5. $\{\epsilon\} \neq \emptyset$

Conjuntos formadores

- Es común definir un lenguaje utilizando un conjunto formador: $\{w \mid \text{condiciones que debe cumplir } w \text{ para pertenecer al lenguaje}\}$
- Ejemplos:
 1. $\{w \mid w \text{ consists of an equal number of 0's and 1's}\}$.
 2. $\{w \mid w \text{ is a binary integer that is prime}\}$.
 3. $\{w \mid w \text{ is a syntactically correct C program}\}$.
- Es posible reemplazar w por una expresión con parámetros
 1. $\{0^n 1^n \mid n \geq 1\}$ Se lee: "El conjunto de 0 a la n 1 a la n tal que n es mayor o igual a 1". El lenguaje esta formado por los strings $\{01, 0011, 000111, \dots\}$. Al igual que con los alfabetos en la expresión parametrizada se puede elevar un símbolo a una potencia para representar n copias de dicho símbolo.
 2. $\{0^i 1^j \mid 0 \leq i \leq j\}$ El lenguaje esta formado por los strings que comienzan con una n cantidad de 0's (incluyendo 0 cantidad de 0's) y siguen inmediatamente después de dichos ceros, con una cantidad igual o mayor de 1's. No hay más cadenas después de dichos 1's. ***(Obsérvese que es más sencillo expresar el lenguaje como una expresión parametrizada para evitar la ambigüedad del castellano)***

Un lenguaje o un problema?

- Lenguajes y problemas son realmente lo mismo. Cuál sea el término utilizado depende del punto de vista.
- Conjuntos de strings pertenecientes a un lenguaje $\{0^n 1^n \mid n \geq 1\}$
- Interpretamos la semántica de las construcciones

```
For i = 0 To lvTareas.Items.Count - 1  
    If lvTareas.Items(i).Text <> Task Then
```

Expresiones Regulares

- En aritmética podemos utilizar operadores como + y x, para construir expresiones como $(5+3) \times 4$ por ejemplo.
- Similarmente, podemos utilizar operaciones regulares para construir expresiones que describan lenguajes, las cuáles son llamadas “Expresiones regulares”.

$(0 \cup 1)0^*$.

Expresiones Regulares

- El valor de la expresión aritmética es el número 32.
- El valor de la expresión regular es un lenguaje. En este caso el lenguaje está formado por todas las cadenas que comiencen por 0 o 1, seguidas de cadenas que consistan en cualquier número de ceros.
- El símbolo de concatenación “.”, está implícito.

Expresiones regulares

- Gran importancia en la ciencia de la computación, por ejemplo en la búsqueda de patrones en grandes cantidades de texto.
- Comando GREP, lenguaje Perl, Python.

Ejemplo

$$(0 \cup 1)^*$$

- Si Σ es cualquier alfabeto, entonces:
- Σ^1 describe el lenguaje que consta de todos los strings de longitud 1 sobre ese alfabeto.
- Σ^* describe todos los strings de cualquier longitud sobre el alfabeto.
- Σ^*1 describe todos los strings que terminan en 1.
- $(0\Sigma^*) \cup (\Sigma^*1)$

Equivalencias

- $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$.
- $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$.

Precedencia

- Primero realizamos la operación estrella.
 - Luego la concatenación.
 - Por último, realizamos la unión
-

Definición formal

Say that R is a *regular expression* if R is

1. a for some a in the alphabet Σ ,
2. ϵ ,
3. \emptyset ,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*) , where R_1 is a regular expression.


Ejemplos

- $0^*10^* = \{w / w \text{ contiene un único uno}\}.$
 - $\Sigma^*1\Sigma^* = \{w / w \text{ tiene al menos un } 1\}.$
 - $\Sigma^*001\Sigma^* = \{w / w \text{ contiene la cadena } 001 \text{ como una subcadena}\}.$
 - $(01^+)^* = \{w / \text{todo } 0 \text{ en } w \text{ es seguido por al menos un uno}\}.$
 - $(\Sigma\Sigma)^* = \{w / w \text{ es una cadena de longitud par}\}.$
 - $(\Sigma\Sigma\Sigma)^* = \{w / \text{la longitud de } w \text{ es un múltiplo de tres}\}.$
 - $01 \cup 10 = \{01, 10\}.$
 - $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w / w \text{ comienza y termina con el mismo símbolo}\}.$
 - $(0 \cup \varepsilon)1^* = 01^* \cup 1^*.$ La expresión $0 \cup \varepsilon$ describe el lenguaje $\{0, \varepsilon\}$, por lo que la operación de concatenación añade ya sea un 0 o ε antes de cada cadena 1^* .
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}.$
- $1^*\emptyset = \emptyset$ Concatenando el conjunto vacío con cualquier otro, obtenemos el conjunto vacío.
 - $\emptyset^* = \{\varepsilon\}.$ La operación estrella permite concatenar cualquier número de cadenas del lenguaje para obtener una cadena en el resultado. Si el lenguaje es vacío, la operación estrella puede concatenar 0 cadenas, dando sólo la cadena vacía.

Identidades

$R \cup \emptyset = R$ (La unión del lenguaje vacío con cualquier lenguaje es este último lenguaje)

$R\varepsilon = R$ (La concatenación un lenguaje con epsilon es el primer lenguaje)

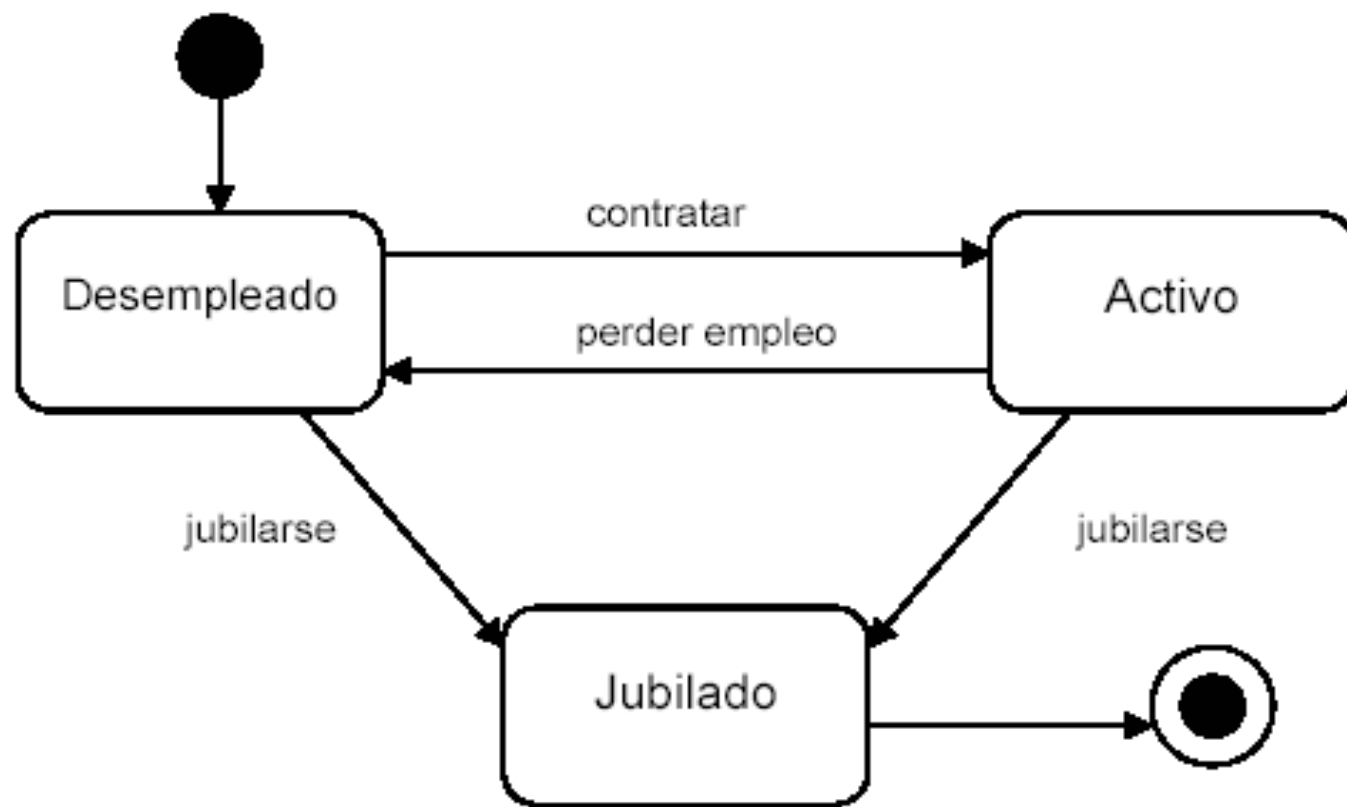

$$R \cup \varepsilon \neq R$$

$$R\emptyset \neq R$$

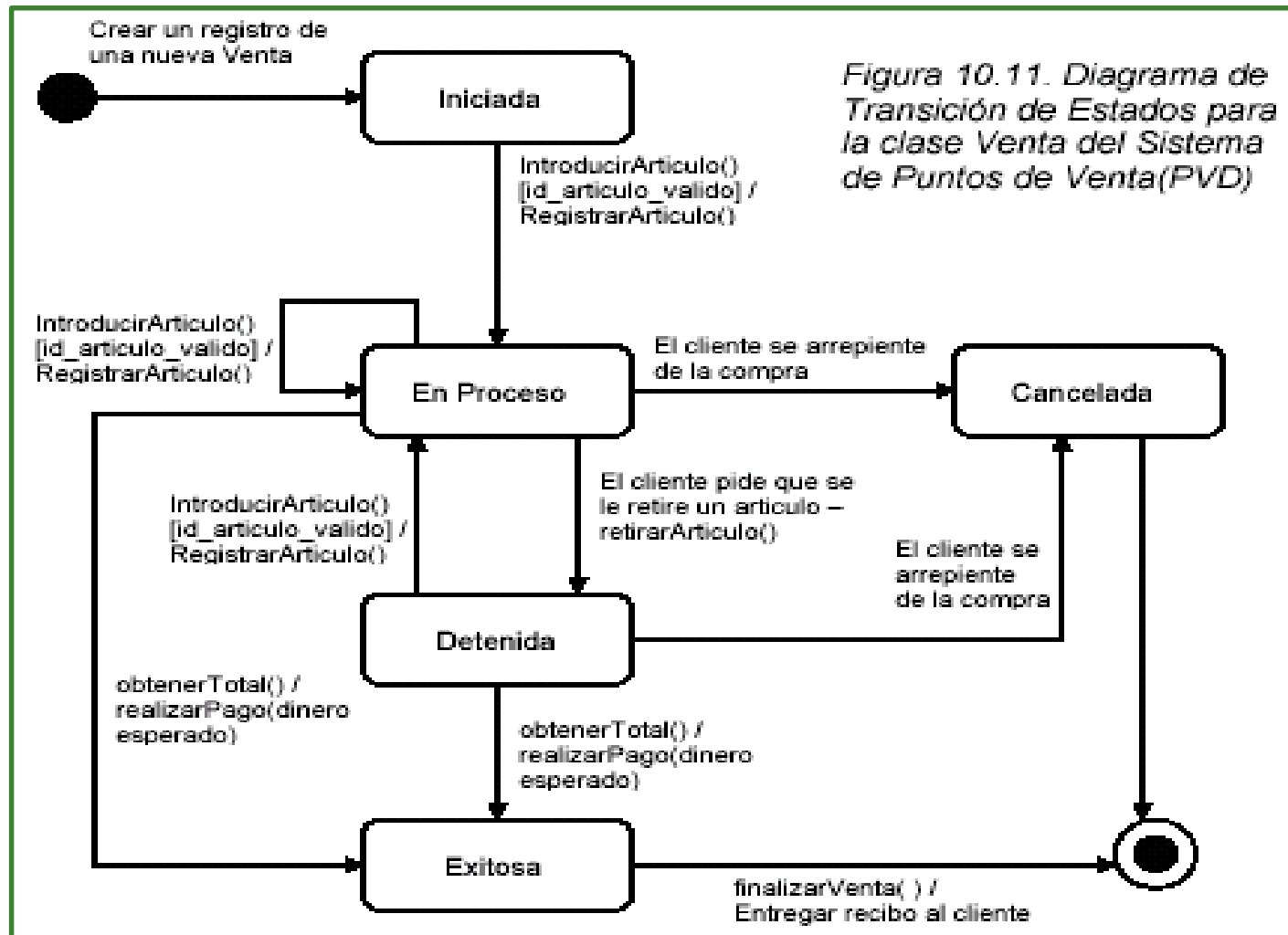
Dígrafos, Estados y Transiciones

- Los dígrafos son una representación gráfica de funciones.
 - Se utilizan para representar situaciones problemáticas reales.
 - Por ejemplo en Diseño de Sistemas se utilizan para realizar los DTE para ver los cambios de estado de un pedido que puede haber sido creado, cancelado, devuelto, en preparación, enviado, aceptado).
-

DTE



DTE II



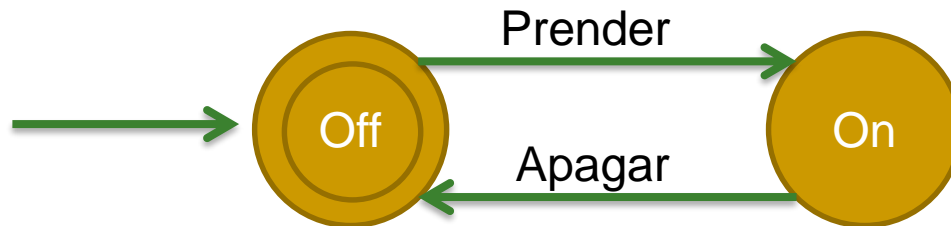
Dígrafos: Representación

- Los Dígrafos se componen de “estados” que se representan con un círculo, y de “transiciones” que se representan por una flecha que une el estado_actual y el estado_siguiente y que se etiquetan con la acción/actividad/cambio que origina esa transición.



Dígrafos: Representación (II)

- De manera abstracta se puede representar a los estados llamándolos con q_0, q_1, q_2, \dots y a las transiciones con a_1, a_2, \dots . Pero tienen que representar inequívocamente un conjunto de realidades diferentes y las condiciones que hacen variar entre una y otra.



Dígrafos Representación (III)

- Conviene que en cada situación a representar con dígrafos se identifique primero una única situación inicial, porque esta será el punto de partida para poder resolver el problema y por eso es importante que sea identificado.

Diagrama de Transición de Estados

- Puede haber varios estados finales o estados de aceptación, que se representarán con un círculo de doble línea.
- Cada “estado” tiene que reunir el conjunto de todas las variables que se necesiten para describir una situación en particular. Tener en cuenta que pueden ser varias circunstancias a considerar. Por ejemplo, para saber si gané un juego de “carreras de autos” tengo que saber que:
 - ❑ llegué al final de la pista y
 - ❑ el reloj del tiempo para la llegada todavía está corriendo y
 - ❑ ningún otro jugador llegó antes que yo

Diagrama de Transición de Estados

- Existen estados “parecidos” donde por ejemplo, se cumple que llegué al final de la pista y todavía el reloj está habilitado pero hay otro jugador que llegó antes que yo, u otra situación en la que llegué primero al final de la pista pero el reloj de tiempo ya llegó a 0 (estado en el cual ningún jugador ganó la carrera)
- No sería correcto considerar “estados” que sean GANE y NO-GANE, o estados LLEGUE_AL_FINAL_DE_LA_PISTA y NO_LLEGUE_AL_FINAL_DE_LA_PISTA porque por sí solos no describen una situación totalmente y se necesita de más información para conocer por completo el estado de la carrera de autos.

DTE

- Plantear el dígrafo que representa en forma de Estados y Acciones el comportamiento de
- La caja de cambios manual de un auto.
- El comportamiento de un semáforo.
- El Ta-Te-Ti
- Pensar en otras situaciones o dispositivos de la vida real cuyo comportamiento pueda ser representado a través de transiciones de estados.