Teoría de Autómatas y Lenguajes Formales

9 de marzo de 2017

Contenido

Objetivo General

Proporcionar al estudiante los fundamentos de la teoría de autómatas así como los de lenguajes formales. También se incluye una introducción a las máquinas de Turing.

Temario

1. Introducción
2. Autómatas Finitos
3. Expresiones Regulares y Lenguajes
4. Propiedades de los Lenguajes Regulares
5. Gramáticas Libres de Contexto y Lenguajes
6. Autómata de Pila
7. Propiedades de los Lenguajes Libres de Contexto
8. Introducción a las Máquinas de Turing

Referencias

1. John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman (2001). *Automata Theory, Language and Computation*. Addison-Wesley Publishing.
2. Michael Sipser (1997). *Introduction to the Theory of Computation*. PWS publishing company.

Teoría de Autómatas

- Estudio de dispositivos o máquinas de cómputo abstractas.
- Turing en los 30's estudió una máquina abstracta con las capacidades de las computadoras actuales (en lo que podían calcular).
- La meta de Turing era describir la frontera entre lo que una máquina podía hacer y lo que no, su conclusión aplica no solo a las máquinas de Turing, sino a las máquinas actuales.

Teoría de Autómatas

- En los 40's y 50's se estudiaron los Autómatas Finitos para modelar la función cerebral.
- Finales de los 50's, N. Chomsky inicia el estudio formal de las gramáticas.
- En 1969 S. Cook extiende el trabajo de Turing para estudiar lo que se podía y no calcular (*compute*), y lo que se podía resolver eficientemente o no (NP-duros).

Todos estos desarrollos tienen repercusiones muy importantes en lo que es la computación actualmente.

Introducción a Pruebas Formales

Para qué pruebas?

- Hacer hipótesis inductivas sobre partes de nuestros programas (iteración o recursión)
  - Hipótesis consistente con la iteración o recursión
- Entender como trabaja un programa correcto es esencialmente lo mismo que la prueba de teoremas por inducción
  - Pruebas deductivas (secuencia de pasos justificados)
  - Pruebas inductivas (pruebas recursivas de un estatuto parametrizado que se usa a sí mismo con valores más bajos del parámetro)

Pruebas Deductivas

- Secuencia de enunciados cuya verdad nos lleva de un enunciado inicial (hipótesis) a una conclusión. Son del tipo “If H Then C”.
- Ejemplo, probar que: *Si* $x \geq 4$ *Entonces* $2^x \geq x^2$. La idea general es probar que se cumple para 4 y probar que mientras x crece, el lado izquierdo duplica su valor con cada incremento, mientras que el lado derecho crece a una razón de $(\frac{x+1}{x})^2$ (crece más lento)
- En forma parecida podemos probar que *si* x es la suma de los cuadrados de 4 enteros positivos, *Entonces* $2^x \geq x^2$.

Pruebas por reducción a definiciones

- Si no se tiene muy claro cómo empezar una prueba, se pueden cambian los elementos de la hipótesis por sus definiciones
- *Ejemplo*: Sea S un subconjunto finito de un conjunto infinito U, si T es el complemento de S entonces T es infinito

Original	Transformado
S es finito	$\exists n$ entero tal que $\|\|S\|\| = n$
U es infiito	$\neg\exists p$ entero tal que $\|\|U\|\| = p$
T es complemento de S	$S \cup T = U$ y $S \cap T = \emptyset$

Prueba por Contradicción

- Sabemos: (i) $S \cup T = U$, (ii) $||S|| = n$ para un entero finito, y (iii) no existe un entero finito p tal que $||U|| = p$
- *Suponemos* que T es finito (lo contrario que queremos probar), i.e., $||T|| = m$ para un entero finito.
- *Entonces*: $||U|| = ||S|| + ||T|| = n + m$, lo cual contradice que no existe un entero finito p tal que $||U|| = p$.

Pruebas con cuantificadores

- A veces existen pruebas con cuantificadores, e.g., $\exists, \forall, \neg\exists, \neg\forall$.
- Esto es común en lógica, e.g., sabemos que todos los hombres son mortales y que Socrates en un hombre, entonces Socrates es mortal
- $\forall x \mathit{Hombre}(x) \rightarrow \mathit{Mortal}(x)$ y $\mathit{Hombre}(\mathit{Socrates})$ por *modus ponens*: $\mathit{Mortal}(\mathit{Socrates})$

Pruebas por Inducción

Frecuentemente el enunciado que tenemos que probar es de la forma “X si y sólo si Y”. Esto requiere hacer dos cosas:

1. Probar la parte “si”: Suponer Y y probar X.
2. Probar la parte “y-sólo-si”: Suponer X y probar Y.

Equivalencia de Conjuntos

Muchos hechos importantes en la teoría del lenguaje son de la forma de dos conjuntos de cadenas, descritas de dos maneras diferentes y que realmente son el mismo conjunto. Para probar que los conjuntos S y T son los mismos, probar:

- $x \in S$ si y sólo si $x \in T$. Esto es:
  - Suponer que $x \in S$; probar que $x \in T$.
  - Suponer que $x \in T$; probar que $x \in S$.

Otro tipo de Pruebas

- Existen muchos otros tipos de pruebas, por ejemplo, por contra-ejemplos, ...
- Este último puede ser muy fácil de probar, por ejemplo: "*todos los primos son impares*", lo cual se puede probar que es falso porque 2 es par y es primo.
- Nos vamos a concentrar más en pruebas por inducción.

Pruebas por Inducción

- Para objetos definidos recursivamente.
- La mayoría maneja enteros, pero en autómatas se trabaja con árboles y expresiones de varios tipos (expresiones regulares).

Ejemplo

- Retomando el problema anterior: *Si* $x \geq 4$ *Entonces* $2^x \geq x^2$.
- *Base*: si $x = 4$, los dos son 16 y se cumple.
- *Inducción*: probar para $x + 1$, esto es: $2^{x+1} \geq (x + 1)^2$.
- $2^{x+1} = 2 \times 2^x$, para el paso de inducción suponemos que $2^x \geq x^2$ y probamos para el siguiente entero.
- Por lo que $2^{x+1} \geq 2x^2$ y queremos probar entonces que: $2x^2 \geq (x + 1)^2$.
- $2x^2 \geq (x + 1)^2 \Rightarrow x^2 \geq 2x + 1$, que al dividir entre x nos da: $x \geq 2 + \frac{1}{x}$.
- Como $X \geq 4$ sabemos que $\frac{1}{x} \leq \frac{1}{4}$, lo cual es cierto.

Ejemplos ...

- Probar que $1 + 3 + 5 + \ldots + (2n - 1) = n^2$.
- *Base*: $P(1) \rightarrow 1 = 1^2$
- *Inducción*: Suponemos
  $P(k) : 1 + 3 + 5 + \ldots + (2k - 1) = k^2$. Para probar que $P(k + 1)$ es verdadera:
  $P(k + 1) \leftarrow 1 + 3 + 5 + \ldots + [2(k + 1) - 1] = (k + 1)^2$
  y esto se puede escribir como:
  $1 + 3 + 5 + \ldots + (2k - 1) + [2(k + 1) - 1] = (k + 1)^2$
  y la parte izquierda es igual a:
  $k^2 + [2(k + 1) - 1] = k^2 + 2k + 1 = (k + 1)^2$
  por lo tanto,
  $1 + 3 + 5 + \ldots + [2(k + 1) - 1] = (k + 1)^2$
  esto verifica $P(k + 1)$ y prueba que la ecuación es verdadera para cualquier entero positivo n.

Ejemplos ...

- Probar que $1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$ para toda $n \geq 1$.
- *Base*: $P(1) \leftarrow 1 + 2 = 2^{1+1} - 1, 3 = 2^2 - 1$, lo cual es verdadero.
- *Inducción*: Suponemos que:
  $P(k) : 1 + 2 + 2^2 + \ldots + 2^k = 2^{k+1} - 1$, es verdadero e intentamos probar que $P(k+1)$ también lo es:
  $P(k+1) : 1 + 2 + 2^2 + \ldots + 2^{k+1} = 2^{(k+1)+1} - 1$

Ejemplos ... (cont.)

Rescribiendo la ecuación anterior tenemos:

$$\begin{aligned} 1 + 2 + 2^2 + ... + 2^k + 2^{k+1} &= 2^{k+1} - 1 + 2^{k+1} \\ &= 2(2^{k+1}) - 1 \\ &= 2^{k+1+1} - 1 \end{aligned}$$

Por lo tanto:

$$1 + 2 + 2^2 + \ldots + 2^{k+1} = 2^{(k+1)+1} - 1$$

Ejemplos ...

- Probar que $x^0 + x^1 + x^2 + \ldots + x^n = (x^{n+1} - 1)/(x - 1)$
- *Base*: $P(1) : x^0 + x^1 = x + 1$, y
  $(x^2 - 1)/(x - 1) = ((x + 1)(x - 1))/(x - 1) = (x + 1)$
- *Inducción*: Suponemos que:
  $P(k) : x^0 + x^1 + x^2 + \ldots + x^k = (x^{k+1} - 1)/(x - 1)$ se cumple.
  Probamos que también $P(k + 1)$ se cumple:
  $P(k + 1) : x^0 + x^1 + x^2 + \ldots + x^k + x^{k+1} = \frac{x^{k+1+1}-1}{(x-1)}$
  Demostramos que el lado izquierdo es igual al lado derecho:
  $\frac{x^{k+1}-1}{(x-1)} + x^{k+1} = \frac{(x^{k+1}-1)+(x-1)(x^{k+1})}{(x-1)} = \frac{x^{k+2}-1}{x-1}$

Ejemplos ...

- Probar que $n^2 > 3n$ para $n \geq 4$.
- *Base*: $P(4) : 4^2 > 3(4)$ ó $16 > 12$, lo cual es cierto
- *Inducción*: Suponemos $P(k) : k^2 > 3k$ para $k > 4$ y tratamos de probar que $P(k+1) : (k+1)^2 > 3(k+1)$

$$
\begin{aligned}
(k+1)^2 &> 3(k+1) \\
k^2 + 2k + 1 &> 3k + 3 \\
3k + 2k + 1 &> 3k + 3 \text{ (por la hipótesis de Inducción)} \\
2k + 1 &> 3 \text{ lo cual es cierto para } k \geq 4
\end{aligned}
$$

Ejercicios a Resolver en Clase:

1. $2 + 6 + 10 + \ldots + (4n - 2) = 2n^2$
2. $1 + 5 + 9 + \ldots + (4n - 3) = n(2n - 1)$
3. $1 + 3 + 6 + \ldots + n(n + 1)/2 = \frac{n(n+1)(n+2)}{6}$
4. $5 + 10 + 15 + \ldots + 5n = \frac{5n(n+1)}{2}$
5. $1^2 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}$
6. $1^3 + 2^3 + \ldots + n^3 = \frac{n^2(n+1)^2}{4}$
7. $n^2 > n + 1$ para $n > 2$

Ejercicios de tarea para la siguiente clase:

1. $2 + 4 + 6 + ... + 2n = n(n + 1)$
2. $1^2 + 3^2 + ... + (2n - 1)^2 = \frac{n(2n-1)(2n+1)}{3}$
3. Probar que $2^n < n!$, para $n \geq 4$

Inducciones Estructurales

- Existen varios problemas que se pueden representar por estructuras (en lugar de solo números) y que se utilizan en teoría de autómatas (e.g., árboles o gramáticas).
- Se hace básicamente lo mismo: Probar el enunciado $S(X)$ respecto a una familia de objetos X (i.e., enteros, árboles) en dos partes:
  1. *Base*: Probar directamente para uno o varios valores pequeños de X.
  2. *Paso de Inducción*: Suponer $S(Y)$ para Y "más pequeña que" X; probar $S(X)$ a partir de lo que se supuso.

Ejemplo

Un árbol binario con n hojas tiene $2n - 1$ nodos.

- Formalmente, $S(T)$: Si T es un árbol binario con n hojas, entonces T tiene $2n - 1$ nodos.
- Inducción respecto al tamaño $=$ número de nodos de T.

Base: Si T tiene 1 hoja, se trata de un árbol de un nodo.
$1 = 2 \times 1 - 1$, está bien.

Ejemplo (cont.)

Inducción: Suponer $S(U)$ para árboles con menos nodos que T. En particular, suponer para los sub-árboles de T:

- T debe consistir de una raíz más dos subárboles U y V.
- Si U y V tienen u y v hojas, respectivamente, y T tiene t hojas, entonces $u + v = t$.
- Por la hipótesis de inducción, U y V tienen $2u - 1$ y $2v - 1$ nodos, respectivamente.
- Entonces T tiene $1 + (2u - 1) + (2v - 1)$ nodos.
  - $= 2(u + v) - 1$.
  - $= 2t - 1$, probando con el paso de inducción.

Teoría de Autómatas

Los autómatas finitos se utilizan como modelos para:

- Software para diseñar circuitos digitales.
- Analizador léxico de un compilador.
- Buscar palabras clave en un archivo ó en el web.
- Software para verificar sistemas de estados finitos, como protocolos de comunicaciones.

Conceptos de Teoría de Autómatas

- Alfabeto = conjunto finito de símbolos (Σ)
  - $\Sigma = \{0, 1\}$, alfabeto binario.
  - $\Sigma = \{a, b, c, d, \ldots, z\}$, alfabeto del abecedario en minúsculas.
  - Conjunto de los caracteres ASCII.

Conceptos (cont.)

- Cadena = secuencia finita de símbolos elegidos de un alfabeto.
  - 01101.
  - *abracadabra*.
  - La cadena vacía se denota como: ϵ,
  - Todas las cadenas de un alfabeto Σ de longitud k se denotan como Σ^k. E.g., $\Sigma^0 = \{\epsilon\}$, si $\Sigma = \{0, 1\}$, entonces, $\Sigma^1 = \{0, 1\}, \Sigma^2 = \{00, 01, 10, 11\}$, etc.
  - El conjunto de todas las cadenas de un alfabeto Σ se denota como Σ^*. $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$. Sin la cadena vacía: Σ^+. Por lo que: $\Sigma^* = \Sigma^0 \cup \Sigma^+$.

Conceptos (cont.)

- Concatenación de x y $y = xy$. Por lo mismo $\epsilon x = x\epsilon = x$.
- Lenguaje = conjunto de cadenas elegidas de algún alfabeto.
  - Nota: El lenguaje puede ser infinito, pero existe algún conjunto finito de símbolos de los cuales se componen todas sus cadenas.
  - El conjunto de todas las cadenas binarias que consisten de algún número de 0's seguidos por un número igual de 1's; esto es: $\{\epsilon, 01, 0011, 000111, \ldots\}$.
  - C (el conjunto de programas compilables en C).
  - Español.

Problemas comunes en Autómatas

- Si σ es una alfabeto y L es un lenguaje sobre σ se tiene que decidir si una cadena $w \in \sigma^*$ también $w \in L$.
- Por ejemplo, si tenemos un autómata que acepta números primos en forma de cadenas binarias, lo podríamos usar para ver si una cadena de un nuevo número es primo o no.
- Este es el típico problema de los *parser*s, se tiene un autómata que acepta programas escritos en “C” (por ejemplo) y tiene que decidir, si el programa cumple con la sintaxis válida de “C”.

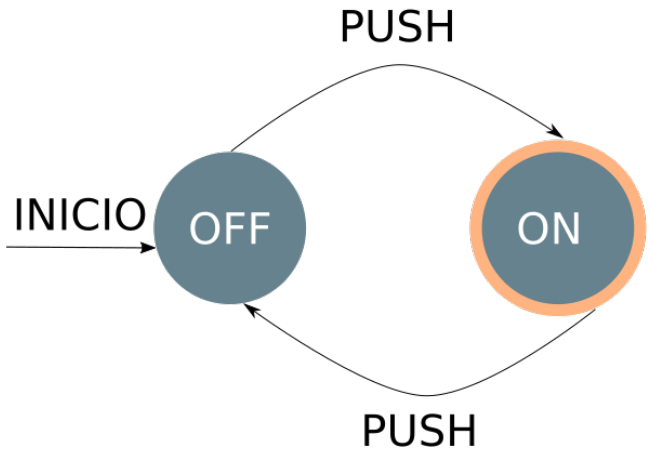
Representación Gráfica

Los autómatas se pueden representar de forma gráfica:

- Un grafo con un número finito de nodos, llamados *estados*.
- Los arcos se etiquetan con uno o más símbolos de algún alfabeto.
- Un estado es designado como el *estado de comienzo* ó *estado inicial*.
- Algunos estados son *estados finales* o *estados de aceptación*.

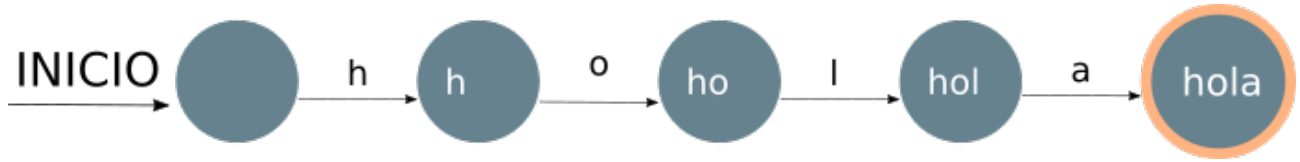
Ejemplos de Autómatas Finitos

Autómata finito para modelar un switch de encendido/apagado:



Ejemplos de Autómatas Finitos II

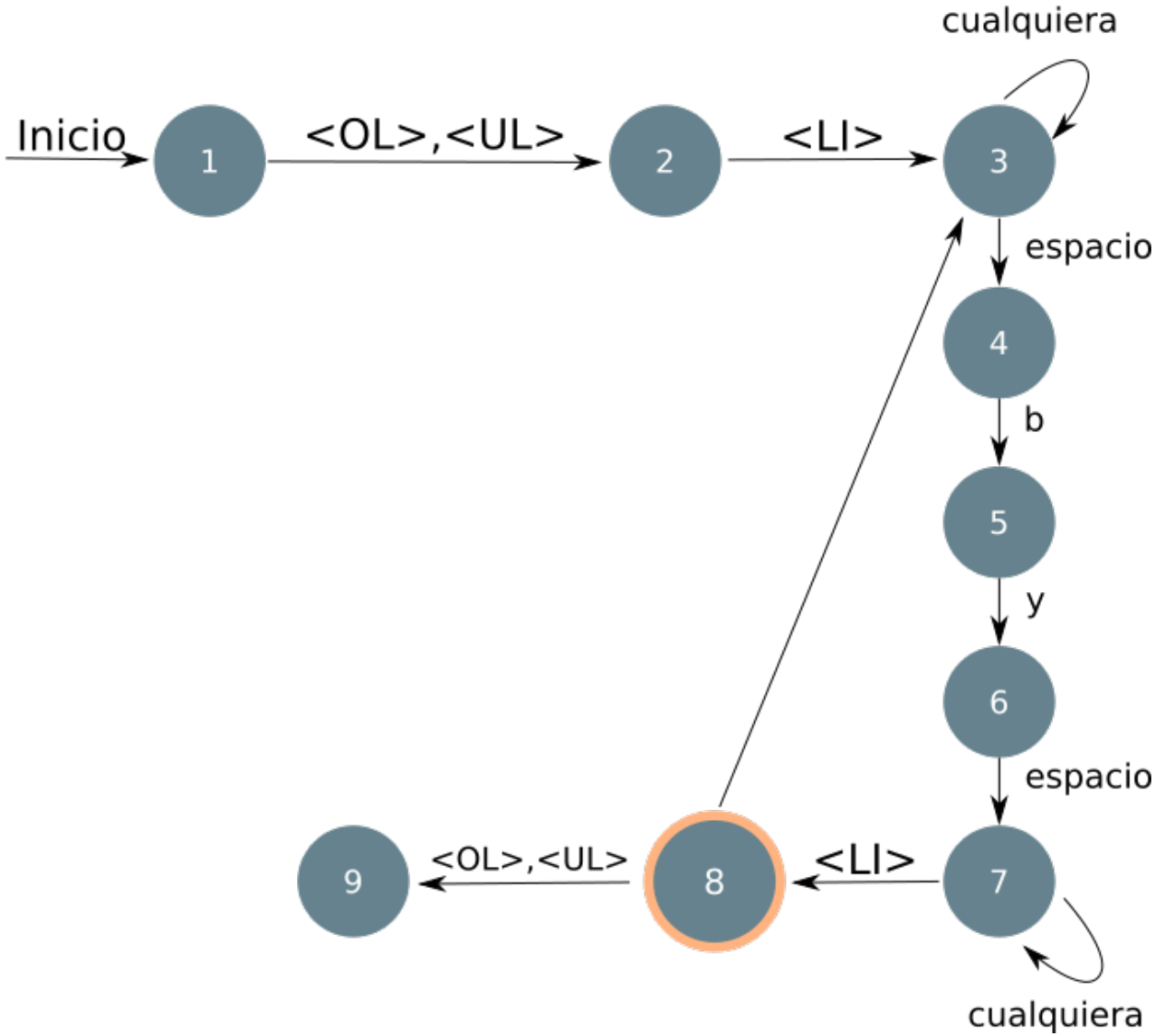
Autómata Finito para reconocer la cadena “hola”:



Lenguajes Regulares

- El lenguaje de los **AF**s (Autómata Finitos) es el conjunto de cadenas que etiquetan rutas que van desde el estado inicial a algún estado de aceptación.
- Abajo, el **AF** explora documentos HTML, busca una lista de lo que podrían ser los pares de título-autor, quizás en una lista de lectura para algún curso de literatura.

INICIO


cualquiera
Inicio
1
<OL>,<UL>
2
<LI>
3
espacio
4
b
5
y
6
espacio
7
<LI>
8
<OL>,<UL>
9
cualquiera

Ejemplo anterior ...

- Acepta cuando encuentra el final de un elemento de la lista.
- En una aplicación, las cadenas que casan con el título (antes de “by”) y autor (después) serían almacenadas en una tabla de pares de título-autor que son acumulados.

Representaciones Estructurales

Otra forma común de representar autómatas es por medio de gramáticas.

- Gramáticas: Modelos útiles para diseñar SW que procesan datos con una estructura recursiva.
  - Un parser de un compilador.
  - Expresiones aritméticas, condicionales, etc.
  - Una regla: $E \Rightarrow E + E$.
- Expresiones Regulares: Denotan la estructura de los datos, especialmente cadenas de texto.

Representaciones Estructurales

- Autómatas pueden describir lo mismo que puede describir un lenguaje regular.
- El estilo difiere del de las gramáticas.
- Estilo de expresiones regulares en UNIX:
  - $[A-Z][a-z]*[\ ][A-Z][A-Z]$
  - Representa palabras que inician con mayúscula seguidas de un espacio y dos letras mayúsculas (e.g., Ithaca NY)
  - No incluye ciudades con nombres compuestos ...
  - $([A-Z][a-z]*[\ ])*[A-Z][A-Z]$

Ejercicios en clase:

1. Modelar un AF para cadenas de 0's y 1's que terminen siempre en 010.
2. Modelar un AF para las cadenas que terminan con tres 0's consecutivos.
3. Describir con palabras el conjunto aceptado por el autómatas finitos del siguiente diagrama de transiciones:

Autómatas y Complejidad

Los autómatas son esenciales para el estudio de los límites de la computación.

- Qué puede hacer una computadora?
  - Decidibilidad (*decidability*).
  - Es decidible o no un problema.
- Qué puede hacer una computadora eficientemente?
  - Intractabilidad (*intractability*).
  - Polinomial vs. Exponencial.