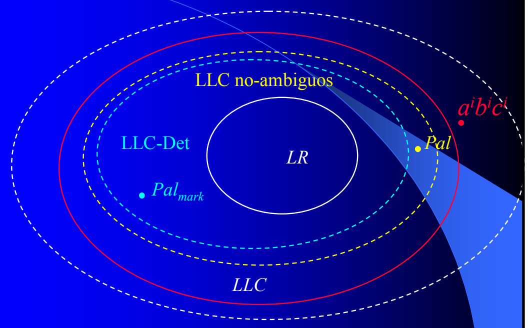
Sesión 25

Máquina de Turing,
Jerarquía de Chomsky,
Problema del paro
y
Tesis de Church

No-*LLC*

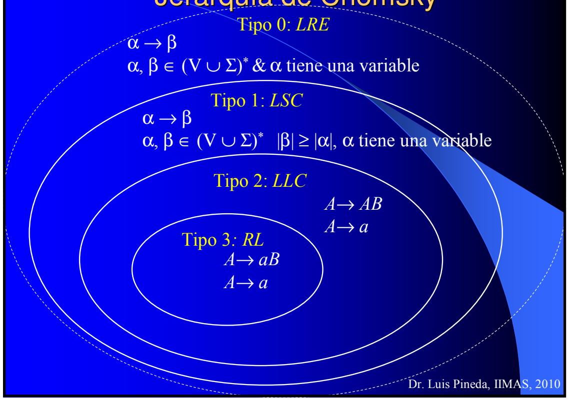


AP con dos pilas

- Proceso de $a^ib^ic^i$:
  - Push todas las a's en stack 1
  - Por cada b pop una a del stack 1& push una b en el stack 2
  - eventualmente $a^i = b^i$
  - Por cada c pop una b del stack 2
  - eventualmente $b^i = c^i$
- Pero el lenguaje no es *LLC*
- La máquina de 2 stacks no es una AP "normal"

Autómata acotado linealmente (Linear bounded automaton)

Jerarquía de Chomsky



Más allá del autómata de pila

Máquina de Turing

- Alan Turing, 1936. On computable numbers, with an application to Entscheidungsproblem, Proceedings of the London Matemathics Society. 42: 230-265 & 43:544 (1937).
- Movidas: Dependiendo del estado actual & del símbolo en la cinta:
  - Seleccionar el siguiente estado
  - Acción: escribir un símbolo o moverse una celda a la derecha o a la izquierda de la cinta

Una teoría de la computación completa

- Para todas las funciones f
- Para todos sus argumentos x
- Contar con una <u>representación</u> y con un algoritmo que permite calcular $f(x)$

¿Cuántas funciones hay?

- Sea n_0 ... n_n la lista de todos los argumentos
- Sea f_0 ... f_n la lista de todas las funciones

¿Cuántas funciones hay?

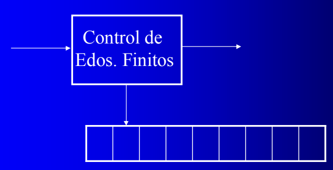
	n_0	n_1	n_2	…	n_n
f_0	$f_0(n_0)$	$f_0(n_1)$	$f_0(n_2)$	…	$f_0(n_n)$
f_1	$f_1(n_0)$	$f_1(n_1)$	$f_1(n_2)$	…	$f_1(n_n)$
f_2	$f_2(n_0)$	$f_2(n_1)$	$f_2(n_2)$	…	$f_2(n_n)$
…	…	…	…	…	…
f_n	$f_n(n_0)$	$f_n(n_1)$	$f_n(n_2)$	…	$f_n(n_n)$

$f_j(n_i)$ puede o no estar definida

Función *antidiagonal* u

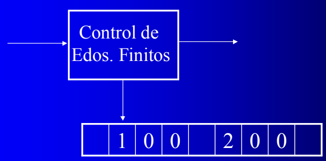
- Sea u:
  - $u(n) = 1$ si $f_n(n)$ NO está definida
  - $u(n) = f_n(n) + 1$ si $f_n(n)$ está definida
- Supongamos que $u = f_m$ (i.e. está en la lista):
  - $f_m(m) = 1$ si $f_m(m)$ NO está definida
  - $f_m(m) = f_m(m) + 1$ si $f_m(m)$ está definida
- Por lo tanto u no está en la lista!
- Ergo, las funciones no son numerables!

Máquina de Turing



Máquina de Turing

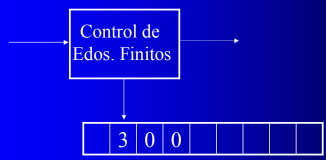
- Evaluando una función:
  - Estado inicial: los argumentos de la función



Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

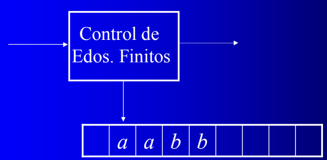
- Evaluando una función:
  - Estado final: el valor de la función para dichos argumentos



Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Aceptando un lenguaje:
  - Función característuica del lenguaje

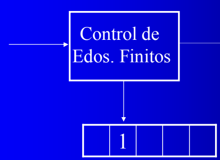


L = $\{x \,/\, |x| \text{ es par}\}$

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Aceptando un lenguaje:
  - Función característica del lenguaje

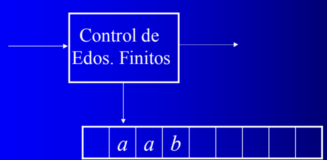


L = $\{x \,/\, |x| \text{ es par}\}$

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Aceptando un lenguaje:
  - Función característica del lenguaje

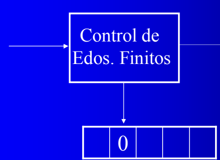


L = $\{x \,/\, |x| \text{ es par}\}$

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Aceptando un lenguaje:
  - Función característica del lenguaje



L = $\{x \,/\, |x| \text{ es par}\}$

Dr. Luis Pineda, IIMAS, 2010

Una teoría de la computación completa

- Las funciones no son contables
- Las máquinas de Turing son contables
- Hay más funciones que MT
- No todas las funciones son computables
- La función *u*, en particular, no es computable!

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Convenciones de interpretación:
  - Estado inicial: los argumentos de la función (apuntando al símbolo más izquierdo)
  - Estado final: el valor de la función para dichos argumentos (idem.)
  - Si la máquina no para, o para en una configuración no estándar (i.e. apuntando a un símbolo diferente del más izquierdo), la función no tiene valor para el argumento de entrada: es una función parcial.

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Para determinar el valor de una función, de acuerdo con las convenciones de interpretación:

Necesitamos saber si

la máquina va a parar!

Dr. Luis Pineda, IIMAS, 2010

Definición de la máquina de paro HM

- Definimos una máquina HM (máquina de paro) que recibe como argumentos el id. de la máquina bajo análisis y su argumento, y determina si dicha máquina <u>para</u> para dicho argumento:

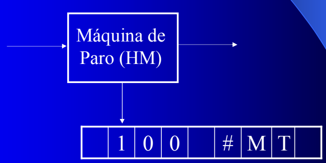
$H(n, m) = 2$ Si la máquina *m* para para el argumento *n*

$H(n, m) = 1$ en otro caso

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

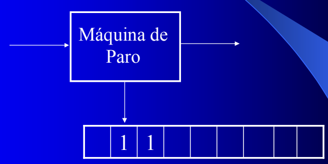
- Estado inicial:
  - Argumento
  - Id. de la función o MT correspondiente:



Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Estado final: Para (2) o no para (1)



- – La máquina *m* <u>para</u> para el argumento *n*

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

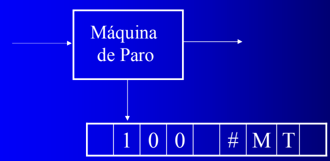
- Estado final: Para (1) o no para (2)

  - La máquina *m* NO para para el argumento *n*

Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

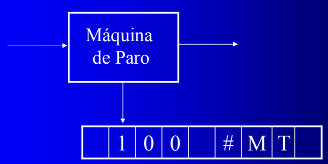
- ¿Qué hace HM?
  - Toma el argumento y la descrición de la MT indicada
  - Aplica el argumento a dicha máquina o algo así!



Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- ¿Qué hace HM?
  - Si para escribe 11 y si no para escribe 1!
  - Pero en todo caso, la máquina de paro (HM) tiene que parar para todos los argumentos y todas las MT



Dr. Luis Pineda, IIMAS, 2010

Una teoría de la computación completa

- Para todas las funciones *f*
- Para todos sus argumentos *x*
- Contar con una representación y con un algoritmo que permite calcular *f*(*x*)
- Si la representación es la máquina de Turing, el problema de la computabilidad se soluciona definiendo la máquina del paro (una MT)

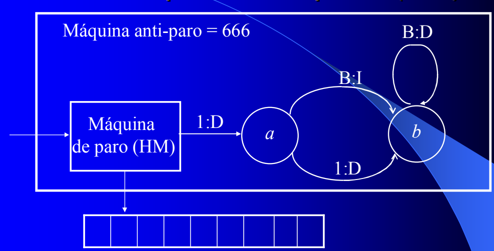
Dr. Luis Pineda, IIMAS, 2010

Una teoría de la computación completa

- Si la máquina del paro HM es una MT, siempre será posible definir una máquina anti-paro M666 tal que:
  - Si HM dice que *m* para para *n*, M666 dice dice que *m* no para para *n*
  - Si HM dice que *m* No para para *n*, M666 dice dice que *m* para para *n*

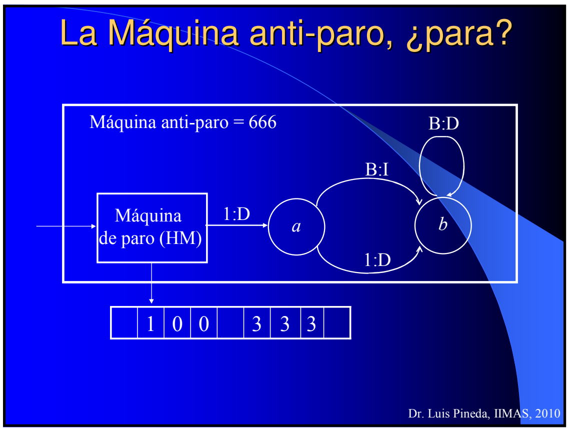
Dr. Luis Pineda, IIMAS, 2010

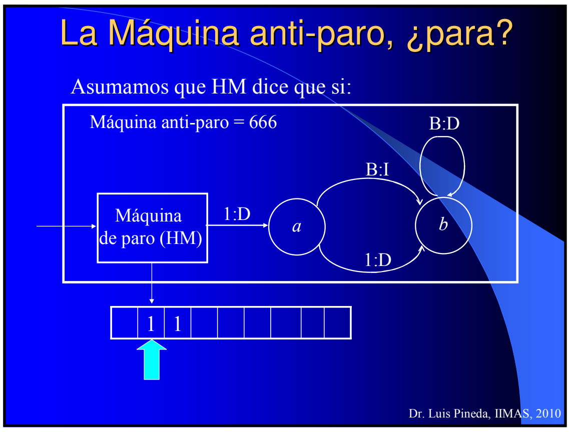
La Máquina anti-paro (AH)

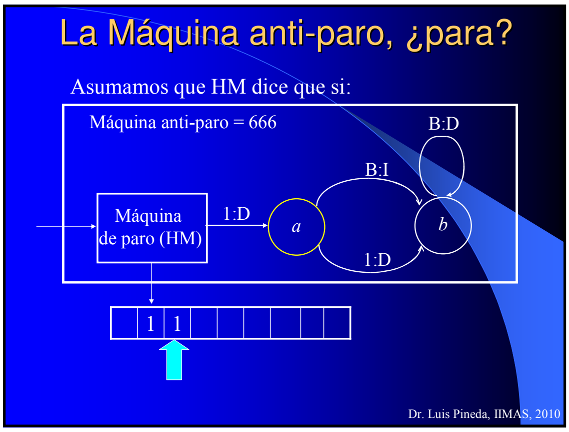


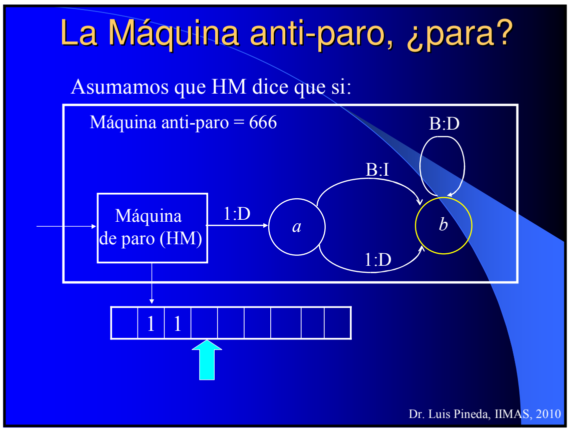
Todos los estados finales de HM se concatenan al estado inicial del segmento antiparo con una arco etiquetado 1:D

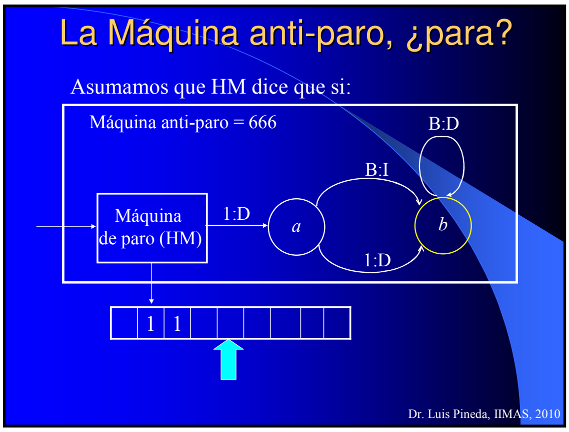
Dr. Luis Pineda, IIMAS, 2010

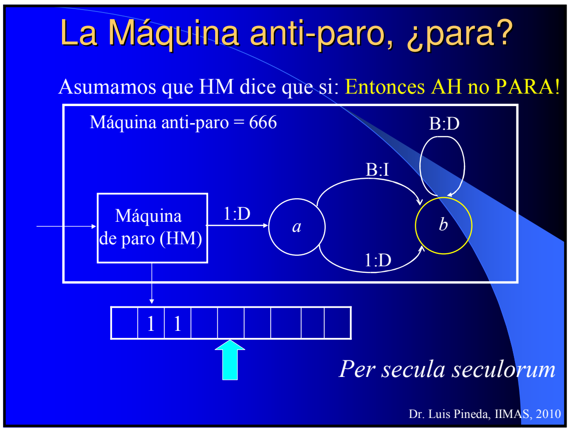

La Máquina anti-paro, ¿para?
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 0 0 3 3 3
Dr. Luis Pineda, IIMAS, 2010


La Máquina anti-paro, ¿para?
Asumamos que HM dice que si:
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 1
Dr. Luis Pineda, IIMAS, 2010


La Máquina anti-paro, ¿para?
Asumamos que HM dice que si:
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 1
Dr. Luis Pineda, IIMAS, 2010

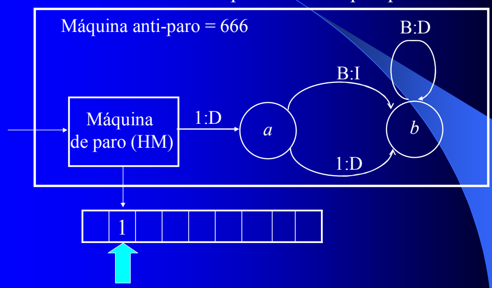

La Máquina anti-paro, ¿para?
Asumamos que HM dice que si:
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 1
Dr. Luis Pineda, IIMAS, 2010


La Máquina anti-paro, ¿para?
Asumamos que HM dice que si:
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 1
Dr. Luis Pineda, IIMAS, 2010


La Máquina anti-paro, ¿para?
Asumamos que HM dice que si: Entonces AH no PARA!
Máquina anti-paro = 666
B:D
B:I
Máquina de paro (HM)
1:D
a
b
1:D
1 1
Per secula seculorum
Dr. Luis Pineda, IIMAS, 2010

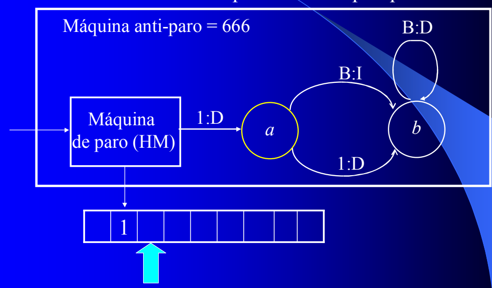
La Máquina anti-paro, ¿para?

Entonces asumamos que HM dice que que NO:



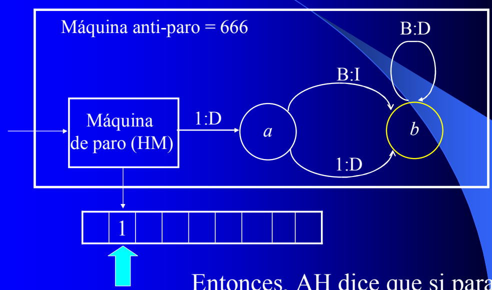
La Máquina anti-paro, ¿para?

Entonces asumamos que HM dice que que NO:



La Máquina anti-paro, ¿para?

Entonces asumamos que HM dice que que NO: AH SI PARA

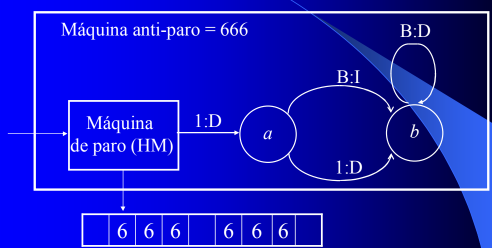


Entonces, AH dice que si para!

Una teoría de la computación completa

La máquina M666 para?

La Máquina anti-paro, ¿para?



H no es una MT

- Si HM dice que M666 para, entonces M666 no para
- Si HM dice que M666 no para, entonces M666 para
- Por lo tanto, HM, la máquina de Paro, si es que existe, no es una Máquina de Turing

El problema del paro (The halting problem)

- Si la cinta se puede reescribir, no podemos saber si la computación va a parar!
- El problema del paro no puede ser resuelto por una máquina de turing

Dr. Luis Pineda, IIMAS, 2010

Tesis de Church

- El conjunto de las funciones computables por la MT corresponde con el conjunto de funciones que los seres humanos pueden evaluar de manera intuitiva
- Cualquier mecanismo computacional que sea suficientemente general para evaluar a todas las funciones es equivalente (y puede reducirse) a la Máquina de Turing

Dr. Luis Pineda, IIMAS, 2010

Tesis de Church

- Formalismos equivalentes:
  - Máquina de Turing
  - Teoría de las funciones recursivas (Kleene)
  - Computación Abacus (Arq. de Von Newman)
  - Cálculo Lambda (Alonso Church)
  - Máquina de Post

Dr. Luis Pineda, IIMAS, 2010

Tesis de Church & el problema del paro

- Si se descubriera un mecanismo computacional que resolviera el problema del paro, dicho mecanismo no sería una máquina de Turing
- La Tesis de Church sería falsa!

Dr. Luis Pineda, IIMAS, 2010

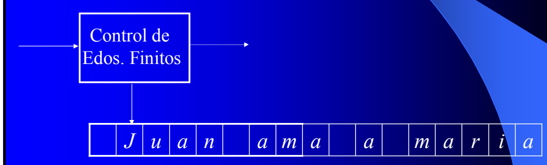
MT & Lenguaje Natural

- Interpretar el lenguaje es evaluar una función:
  - La función característica del lenguaje

Dr. Luis Pineda, IIMAS, 2010

MT & Lenguaje Natural

- Interpretar el lenguaje es evaluar una función:
  - La función característica del lenguaje



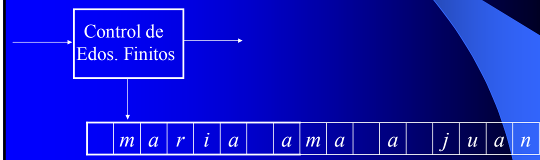
Dr. Luis Pineda, IIMAS, 2010

Máquina de Turing

- Interpretar el lenguaje es evaluar una función:

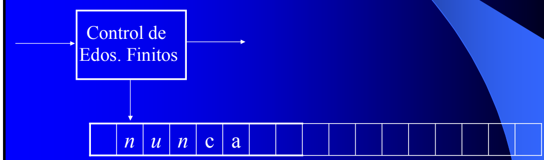
Máquina de Turing

- Interpretar el lenguaje es evaluar una función:



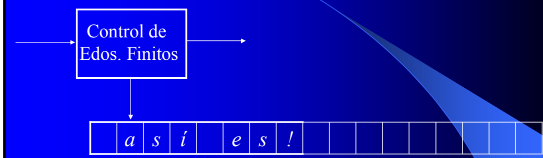
Máquina de Turing

- Interpretar el lenguaje es evaluar una función:



Máquina de Turing

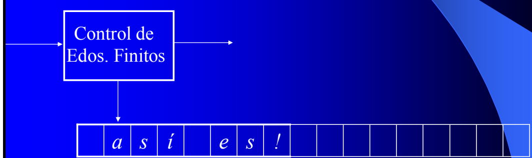
- Interpretar el lenguaje es evaluar una función:



- Hay una MT por cada función que podamos evaluar intuitivamente!
- Saber un lenguaje es saber su función característica!

El Juego de Turing (1950)

- Alan Turing, Computing machinery and Intelligence, *Mind*, Octubre, 1950, 59:433-460
- Interpretar el lenguaje es evaluar una función:



- Origen de la Inteligencia Artificial

El lenguaje natural...

- Curso: "Procesamiento del habla y del lenguaje"
  - Reconocimiento del habla
  - Interpretación del lenguaje
- Posgrado UNAM 2006-2:
  - Martes y jueves de 12:00 a 13:30 Hrs
  - Seminario
  - Evaluación: Construcción de un sistema con recursos del proyecto DIME