Máquinas de Turing y otros artilugios

1er Encuentro de Educación en Ciencia de la Computación

Luis Sierra

Instituto de Computación

1 de Noviembre del 2012

Plan

- Máquinas de Turing

Plan

- Máquinas de Turing
- Otros artilugios

Plan

- Máquinas de Turing
- Otros artilugios
  1. m. Mecanismo, artefacto, sobre todo si es de cierta complicación. U. m. en sent. despect.
  2. m. Ardid o maña, especialmente cuando forma parte de algún plan para alcanzar un fin.
  3. m. Herramienta de un oficio.

Máquina P

Borra | Constante (C3, un arg.) | Constante (PI2, dos arg.) | Sucesor

Programa

```
1  ?  2  6
2  0  3
3  R  4
4  ?  3  5
5  I  6
```

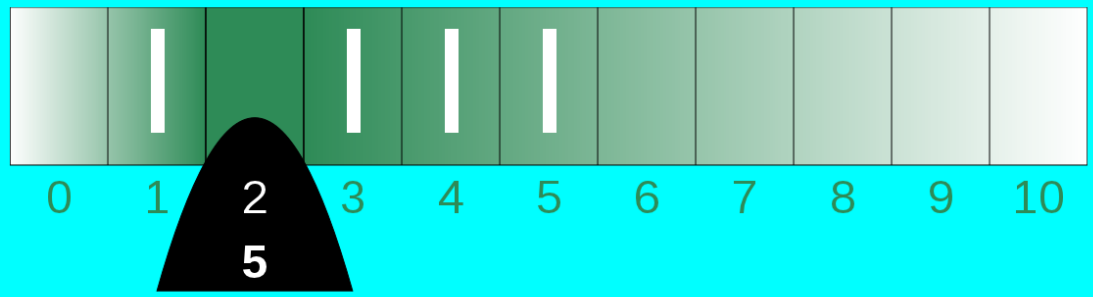
Cinta

Entrada

2 3

Salida

...



Procesos combinatorios finitos (Journal of Symbolic Logic, Vol.1, Iss.3, 1936)

Tenemos en mente un *problema general* formado por una clase de *problemas especificos*. Una solución al problema general proporcionará una respuesta a cada problema específico.

En la siguiente formulación de la solución participan dos conceptos: el de un *espacio de símbolos* en el que se lleva a cabo el trabajo de pasar del problema a la respuesta, y el de un *conjunto de instrucciones* fijo e inalterable que controla tanto las operaciones en el espacio de símbolos como el orden en que dichas instrucciones se deben aplicar.

Procesos combinatorios finitos

En esta formulación el espacio de símbolos consiste en una secuencia de espacios o casillas, que es infinita en ambos sentidos; es decir, ordinalmente similar a la serie de enteros . . . , -3, -2, -1,0,1,2,3 , . . . El solucionador de problemas, o trabajador, debe moverse y trabajar en este espacio de símbolos, pudiendo ver y operar sobre una única casilla a la vez. Aparte de la presencia del trabajador, una casilla admite sólo una de dos condiciones posibles; estar vacía, o tener una marca única, por ejemplo un trazo vertical.

Se debe elegir una casilla llamada el punto de partida. Suponemos que un problema específico debe darse en forma simbólica por medio de un número finito de casillas marcadas. Del mismo modo, la respuesta debe darse en forma simbólica por una configuración de casillas marcadas. Para ser específicos, la respuesta es la configuración de casillas marcadas que quedan luego del proceso de resolución.

Procesos combinatorios finitos

Se supone que el trabajador es capaz de realizar las siguientes acciones primitivas:

(*a*) *Marcar la casilla en que se encuentra (suponiéndola vacía)*

(*b*) *Borrar la marca de la casilla en que se encuentra (suponiéndola marcada)*

(*c*) *Moverse a la casilla de la derecha*

(*d*) *Moverse a la casilla de la izquierda*

(*e*) *Determinar si la casilla en que se encuentra está marcada o no*

Procesos combinatorios finitos

El conjunto de instrucciones que, sea dicho, es el mismo para todos los problemas específicos y por tanto corresponde al problema general, ha de ser de la siguiente forma. Debe comenzar

Comience en el punto de partida y siga la instrucción 1

Luego, consta de un número finito de instrucciones numeradas 1, 2, 3, . . . , n . La instrucción i-ésima debe tener alguna de las siguientes formas:

(A) Realice la operación O_i $[O_i = (a), (b), (c),$ o $(d)]$ y luego siga la instrucción j_i

(B) Realice la operación (e) y, según la respuesta es sí o no, seguir la instrucción j_i' o j_i''

(C) Deténgase

Procesos combinatorios finitos

El conjunto de instrucciones que, sea dicho, es el mismo para todos los problemas específicos y por tanto corresponde al problema general, ha de ser de la siguiente forma. Debe comenzar

Comience en el punto de partida y siga la instrucción 1

Luego, consta de un número finito de instrucciones numeradas 1, 2, 3, . . . , n . La instrucción i-ésima debe tener alguna de las siguientes formas:

(A) Realice la operación O_i [$O_i = (a), (b), (c),$ o (d)] y luego siga la instrucción j_i

(B) Realice la operación (e) y, según la respuesta es sí o no, seguir la instrucción j_i' o j_i''

(C) Deténgase

Pero ...

...esta no es la máquina de Turing.

Post vs Turing

- **Finite Combinatory Processes - Formulation 1**, Emil L. Post, *The Journal of Symbolic Logic*, Vol. 1, No. 3, pp. 103-105 (Sep., 1936)
- **On Computable Numbers, with an application to the Entscheidungsproblem**, Alan Turing, *Proceedings of the London Mathematical Society* (2) 42 pp 230-265 (1936-37)

Post vs Turing

- **Finite Combinatory Processes - Formulation 1**, Emil L. Post, *The Journal of Symbolic Logic*, Vol. 1, No. 3, pp. 103-105 (Sep., 1936)
- **On Computable Numbers, with an application to the Entscheidungsproblem**, Alan Turing, *Proceedings of the London Mathematical Society* (2) 42 pp 230-265 (1936-37)

¿En qué se parecen o diferencian estas propuestas?

Máquina de Post

Borra | Constante (C3, un arg.) | Constante (PI2, dos arg.) | Sucesor

Programa

```
1  ?  2  6
2  0  3
3  R  4
4  ?  3  5
5  I  6
```

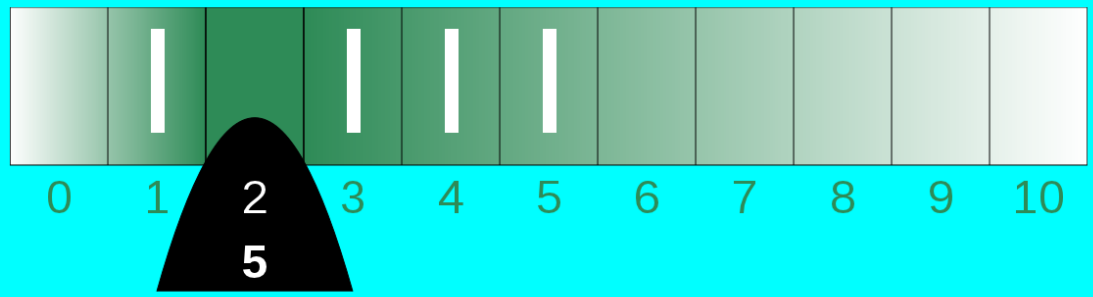
Cinta

Entrada

2 3

Salida

...



Máquina de Turing

Borra | Constante (C3, un arg.) | Constante (PI2, dos arg.) | Sucesor

Programa

```
start 0 stop 1 R
start 1 q 0 R
q 1 q 1 R
q 0 stop 1 R
```

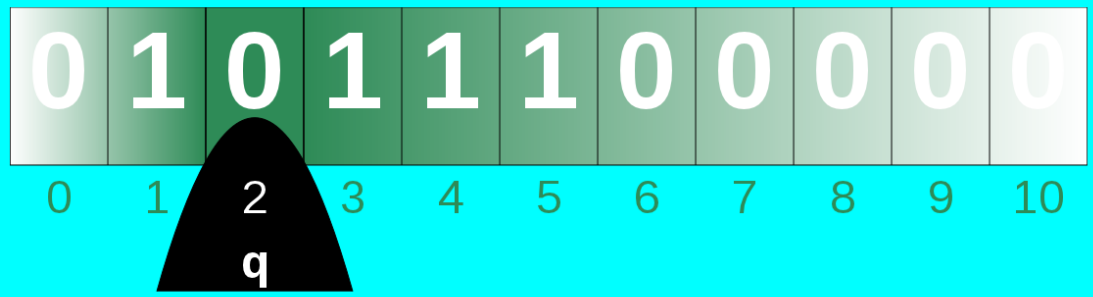
Cinta

Entrada

2 3

Salida

...



Se diferencian en ...

el largo Post: pp. 103-105, Turing: pp 230-265

las marcas En Turing hay marcas de primera clase (0, 1) y marcas de segunda, auxiliares para el cómputo, pero que no se consideran resultado del cómputo

las instrucciones En Turing son de largo fijo (à la RISC), en Post de largo variable (à la CISC)

Se parecen en la implementación

	Post	Turing
Biblioteca	18	18
Despliegue	161	161
Motor	87	87
Lo diferente	41	32
Total	307	298

El código coincide en un 90%.

Se parecen sus programas

Las instrucciones son tiras finitas.
Los programas son tiras finitas de instrucciones.

Se parecen sus programas

Las instrucciones son tiras finitas.
Los programas son tiras finitas de instrucciones.

Podemos numerar los programas; asignarle un nombre, o índice a cada programa.

Se parecen sus programas

Las instrucciones son tiras finitas.
Los programas son tiras finitas de instrucciones.

Podemos numerar los programas; asignarle un nombre, o índice a cada programa.

1	?	2	4
2	O	3	
3	R	1	
4	X		

Se parecen sus programas

Las instrucciones son tiras finitas.
Los programas son tiras finitas de instrucciones.

Podemos numerar los programas; asignarle un nombre, o índice a cada programa.

1	?	2	4
2	O	3	
3	R	1	
4	X		

```
1 ?  11 1111 11 0 111 111 R 1 1111 X
```

Se parecen sus programas

Las instrucciones son tiras finitas.
Los programas son tiras finitas de instrucciones.

Podemos numerar los programas; asignarle un nombre, o índice a cada programa.

1	?	2	4
2	O	3	
3	R	1	
4	X		

```
1 ?  11 1111 11 0 111 111 R 1 1111 X

1 1 11 1111 11 11 111 111 111 1 1111 1111
```

Se parecen en lo que pueden hacer

Pueden sumar y borrar.

- Y además calculan funciones constantes, proyecciones, y sucesor
- Y pueden programar `if x=y then w else z`

Se parecen en lo que pueden hacer

Pueden sumar y borrar.

- Y además calculan funciones constantes, proyecciones, y sucesor
- Y pueden programar `if x=y then w else z`

O sea, son artilugios que cumplen las siguientes propiedades de Hennie

- la Propiedad 1, de las funciones básicas
- la Propiedad 4, de selección

También cumplen la propiedad 2 de Hennie

Propiedad de clausura

Si tengo un programa P que computa la función f, y un programa Q que computa la función g, hay un programa $P;Q$ que computa la función $g.f$.

También cumplen la propiedad 2 de Hennie

Propiedad de clausura

Si tengo un programa P que computa la función f, y un programa Q que computa la función g, hay un programa $P;Q$ que computa la función $g.f$.

Algo así como ...

1. Si computo P con la entrada a
2. y obtengo la salida b
3. que uso de entrada al programa Q
4. que computa la salida c,
5. entonces el programa $P;Q$ computa c a partir de la entrada a

También cumplen la propiedad 2 de Hennie

Propiedad de clausura

Si tengo un programa P que computa la función f, y un programa Q que computa la función g, hay un programa $P;Q$ que computa la función $g.f$.

Algo así como ...

1. Si computo P con la entrada a
2. y obtengo la salida b
3. que uso de entrada al programa Q
4. que computa la salida c,
5. entonces el programa $P;Q$ computa c a partir de la entrada a

Si prefieren, computa la secuencia o la composición de dos programas.

También cumplen la propiedad 3 de Hennie

Propiedad de enumeración

Hay un programa universal U que interpreta cualquier otro programa P.

También cumplen la propiedad 3 de Hennie

Propiedad de enumeración

Hay un programa universal U que interpreta cualquier otro programa P.

Algo así como ...

1. Si quiero computar P con la entrada a
2. y n es el índice de P
3. invoco al programa intérprete U con dos argumentos;
4. el índice n y la entrada a

Se parecen en lo que NO pueden hacer

Consideremos un programa `Halt` que recibe dos argumentos, i y x, y devuelve 1 si el programa con índice i termina al ejecutarse con la entrada x, y 0 en caso contrario.

Se parecen en lo que NO pueden hacer

Consideremos un programa `Halt` que recibe dos argumentos, i y x, y devuelve 1 si el programa con índice i termina al ejecutarse con la entrada x, y 0 en caso contrario.

el programa con índice i termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 1$

el programa con índice i NO termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 0$

Se parecen en lo que NO pueden hacer

el programa con índice i
termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 1$

el programa con índice i
NO termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 0$

Considere el programa siguiente, con índice k:

```
P(x):
    if Halt.x.x = 1:  NO TERMINA
    else:  TERMINA
```

Se parecen en lo que NO pueden hacer

el programa con índice i
termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 1$

el programa con índice i
NO termina con la entrada x
$\Rightarrow$
$\texttt{Halt}.i.x = 0$

Considere el programa siguiente, con índice k:

```
P(x):
    if Halt.x.x = 1:  NO TERMINA
    else:  TERMINA
```

Consideremos la situación que resulta de invocar `P(x)` con el argumento `k`.

Se parecen en lo que NO pueden hacer

P termina con la entrada k	P NO termina con la entrada k
$\Rightarrow$	$\Rightarrow$
$\texttt{Halt}.k.k = 1$	$\texttt{Halt}.k.k = 0$

Considere el programa siguiente, con índice k:

```
P(x):
     if Halt.x.x = 1:  NO TERMINA
     else:  TERMINA
```

Consideremos la situación que resulta de invocar `P(x)` con el argumento `k`.

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

ENTSCHEIDUNGSPROBLEM

- Problema de decisión de los teoremas de la lógica de primer orden

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

ENTSCHEIDUNGSPROBLEM

- Problema de decisión de los teoremas de la lógica de primer orden
- Planteado por David Hilbert en 1928

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

ENTSCHEIDUNGSPROBLEM

- Problema de decisión de los teoremas de la lógica de primer orden
- Planteado por David Hilbert en 1928
- Resuelto por Alonzo Church en 1936

Diferencias en los artículos de Post y Turing

treinta carillas

- Turing construye la máquina universal (el primer intérprete).
- Turing muestra el problema de la parada.
- Turing lo aplica al Entscheidungsproblem.

ENTSCHEIDUNGSPROBLEM

- Problema de decisión de los teoremas de la lógica de primer orden
- Planteado por David Hilbert en 1928
- Resuelto por Alonzo Church en 1936
- Y por Alan Turing en 1936

Otras tres diferencias

- La máquina de Post modela a un trabajador que sigue instrucciones; la máquina de Turing hace consideraciones acerca de la memoria humana

Otras tres diferencias

- La máquina de Post modela a un trabajador que sigue instrucciones; la máquina de Turing hace consideraciones acerca de la memoria humana
- La máquina de Post es determinista; Turing se concentra en las máquinas *automáticas*, pero también presenta las *choice machines* en las que el no determinismo es resuelto por un *external operator*.

Otras tres diferencias

- La máquina de Post modela a un trabajador que sigue instrucciones; la máquina de Turing hace consideraciones acerca de la memoria humana
- La máquina de Post es determinista; Turing se concentra en las máquinas *automáticas*, pero también presenta las *choice machines* en las que el no determinismo es resuelto por un *external operator*.
- La máquina de Turing que mostré ...

Otras tres diferencias

- La máquina de Post modela a un trabajador que sigue instrucciones; la máquina de Turing hace consideraciones acerca de la memoria humana
- La máquina de Post es determinista; Turing se concentra en las máquinas *automáticas*, pero también presenta las *choice machines* en las que el no determinismo es resuelto por un *external operator*.
- La máquina de Turing que mostré ...
- ... tampoco es la del artículo

Tesis de Church

La noción de calculabilidad efectiva

Definimos ahora la ya discutida noción de una función *efectivamente calculable* de los naturales identificándola con la noción de función recursiva sobre los naturales. (o sobre las funciones λ-definibles sobre naturales). Pensamos que esta definición queda justificada por las siguientes consideraciones ...

Tesis de Church

La noción de calculabilidad efectiva

Definimos ahora la ya discutida noción de una función *efectivamente calculable* de los naturales identificándola con la noción de función recursiva sobre los naturales. (o sobre las funciones λ-definibles sobre naturales). Pensamos que esta definición queda justificada por las siguientes consideraciones ...

Algunos modelos

- Emil Post, y su máquina

Tesis de Church

La noción de calculabilidad efectiva

Definimos ahora la ya discutida noción de una función *efectivamente calculable* de los naturales identificándola con la noción de función recursiva sobre los naturales. (o sobre las funciones λ-definibles sobre naturales). Pensamos que esta definición queda justificada por las siguientes consideraciones ...

Algunos modelos

- Emil Post, y su máquina
- Alan Turing, y su máquina

Tesis de Church

La noción de calculabilidad efectiva

Definimos ahora la ya discutida noción de una función *efectivamente calculable* de los naturales identificándola con la noción de función recursiva sobre los naturales. (o sobre las funciones λ-definibles sobre naturales). Pensamos que esta definición queda justificada por las siguientes consideraciones ...

Algunos modelos

- Emil Post, y su máquina
- Alan Turing, y su máquina
- Alonzo Church, y su λ-cálculo

Tesis de Church

La noción de calculabilidad efectiva

Definimos ahora la ya discutida noción de una función *efectivamente calculable* de los naturales identificándola con la noción de función recursiva sobre los naturales. (o sobre las funciones λ-definibles sobre naturales). Pensamos que esta definición queda justificada por las siguientes consideraciones ...

Algunos modelos

- Emil Post, y su máquina
- Alan Turing, y su máquina
- Alonzo Church, y su λ-cálculo
- Stephen Kleene, y las funciones generales recursivas de Herbrand y Gödel

En breve ...

En los treinta, antes de las computadoras modernas, ya sabíamos de los límites de la informática.

En breve ...

En los treinta, antes de las computadoras modernas, ya sabíamos de los límites de la informática.
En el siglo veintiuno, luego del uso masivo de computadoras, se cree que no hay límites para la informática.

En breve ...

En los treinta, antes de las computadoras modernas, ya sabíamos de los límites de la informática.
En el siglo veintiuno, luego del uso masivo de computadoras, se cree que no hay límites para la informática.

Hace algunos años me preguntaba si era pertinente introducir los temas de computabilidad en la enseñanza media y primaria.

En breve ...

En los treinta, antes de las computadoras modernas, ya sabíamos de los límites de la informática.
En el siglo veintiuno, luego del uso masivo de computadoras, se cree que no hay límites para la informática.

Hace algunos años me preguntaba si era pertinente introducir los temas de computabilidad en la enseñanza media y primaria.
Hoy me pregunto cómo habría que hacerlo, o al menos, cómo convencer a la gente que los límites de la informática van más allá de los costos y el procesador que se tenga.

Herramientas

HTML	http://www.w3.org/TR/html4/
CSS	http://www.w3.org/TR/CSS2/
JavaScript	https://developer.mozilla.org/en/JavaScript/
SVG	http://www.w3.org/TR/SVG/
Raphaël	Raphaël-JavaScript Library http://raphaeljs.com/

Bibliografía

- **Finite Combinatory Processes - Formulation 1**, Emil L. Post, *The Journal of Symbolic Logic*, Vol. 1, No. 3, pp. 103-105 (Sep., 1936)
- **On Computable Numbers, with an application to the Entscheidungsproblem**, Alan Turing, *Proceedings of the London Mathematical Society* (2) 42 pp 230-265 (1936-37)
- **An Unsolvable Problem of Elementary Number Theory**, Alonzo Church, *The American Journal of Mathematics* 58 pp 345-363 (1936)
- **The undecidable**, Martin Davis (Ed.), Dover, 2004
- **Introduction to Computability**, Fred Hennie, Addison-Wesley Longman Publishing Co., 1977