Universidad de Huelva

Tema 5

Decidibilidad

- Se dice que una máquina de Turing M ***acepta*** una cadena de entrada w si es capaz de procesar la cadena y alcanzar un estado de parada, generando como salida la aceptación de la cadena.
- Se dice que una máquina de Turing M ***rechaza*** una cadena de entrada w si es capaz de procesar la cadena y alcanzar un estado de parada, generando como salida el rechazo de la cadena.
- Dada una máquina de Turing M y una entrada w, la salida generada por la máquina puede ser ***aceptar***, ***rechazar*** o ***no parar***, es decir, quedar en un buble infinito (***accept***, ***reject***, ***loop***).

- Se define como ***lenguaje reconocido por la máquina de Turing M***, $L(M)$, como el conjunto de cadenas aceptadas por la máquina.
- Se dice que un lenguaje L es ***reconocible*** si existe una máquina de Turing que puede reconocerlo.
- Si $w \in L$, entonces la máquina de Turing debe aceptarla, pero si $w \notin L$ la máquina de Turing puede rechazarla o no parar.
- Los lenguajes reconocibles-Turing se denominan también ***lenguajes recursivamente enumerables***.

- Se dice que un lenguaje L es ***decidible*** si existe una máquina de Turing que puede decidirlo, es decir, que ante cualquier entrada la máquina debe alcanzar la parada y decidir si la cadena se acepta o rechaza.
- Si $w \in L$, entonces la máquina de Turing debe aceptarla. Si $w \notin L$ la máquina de Turing debe rechazarla. En ningún caso puede quedar la máquina en un bucle infinito.
- Los lenguajes decidibles-Turing se denominan también ***lenguajes recursivos***.
- Todo lenguaje decidible-Turing es reconocible-Turing.

- Un lenguaje regular es un lenguaje que puede ser reconocido por medio de un Autómata Finito.
- Un Autómata Finito Determinista puede ser codificado de una forma parecida a la codificación de una máquina de Turing determinista. En un AFD las transiciones sólo tienen en cuenta el estado final (no hay que codificar la acción de escribir o moverse a izquierda o derecha).
- El comportamiento de un AFD consiste en partir del estado inicial y realizar las transiciones asociadas a cada símbolo de la cadena de entrada. Si al llegar al final de la entrada el autómata se encuentra en un estado final la cadena se acepta. En caso contrario se rechaza.

- Sea A_{DFA} el lenguaje formado por las cadenas <B,w> tales que B es la codificación de un DFA y w es una cadena aceptada por dicho DFA.
- TEOREMA: **El lenguaje A_{DFA} es decidible**
- Demostración: Se puede desarrollar una máquina de Turing que simule el comportamiento del autómata B sobre la entrada w. Si la simulación termina en un estado final la cadena se acepta; si no la cadena se rechaza. Como el tamaño de la entrada w es finito, la simulación siempre alcanzará el final de la entrada y podrá tomar la decisión correspondiente.

- Sea A_{NFA} el lenguaje formado por las cadenas <B,w> tales que B es la codificación de un autómata finito no determinista NFA y w es una cadena aceptada por dicho NFA.
- TEOREMA: **El lenguaje A_{NFA} es decidible**
- Demostración: Se puede desarrollar una máquina de Turing que cree el autómata finito determista C equivalente al autómata finito no determinista B y simule la cadena w sobre el autómata C.

- Sea A_{REX} el lenguaje formado por las cadenas $<R,w>$ tales que R es la codificación de una expresión regular y w es una cadena aceptada por dicho expresión.
- TEOREMA: **El lenguaje A_{REX} es decidible**
- Demostración: Se puede desarrollar una máquina de Turing que transforme la expresión regular R en un autómata finito no determinista N y simular N sobre la entrada w.

- Sea E_{DFA} el lenguaje formado por las cadenas <A> tales que A es la codificación de un autómata finito determinista que no reconoce ninguna entrada, es decir, cuyo lenguaje es el lenguaje vacío:

$$L(A) = \emptyset$$

- TEOREMA: **El lenguaje E_{DFA} es decidible**
- Demostración: Se puede desarrollar una máquina de Turing que desarrolle el siguiente algoritmo:

1. Crear una lista de estados marcados L1 y una lista de estados estudiados L2.
2. Incluir el estado inicial en L1.
3. Mientras L1 no esté vacía, desplazar un estado E hasta L2 y estudiar sus transiciones incluyendo en la lista L1 cualquier estado destino que no estuviera ya en L2 ni en L1.
4. Si L2 no contiene ningún estado final, aceptar; en otro caso, rechazar.

Nota: La lista L2 contiene los estados alcanzables desde el estado inicial

- Sea EQ_{DFA} el lenguaje formado por las cadenas <A,B> tales que A *y* B son codificaciones de autómatas finitos deterministas que reconocen el mismo lenguaje:

$$L(A) = L(B)$$

- TEOREMA: **El lenguaje EQ_{DFA} es decidible**
- Demostración: Se puede construir un nuevo autómata C que acepte solo las cadenas aceptadas por A o B pero no por ambos autómatas. C se conoce como la diferencia simétrica. Si C reconoce el lenguaje vacío ($C \in E_{DFA}$), entonces aceptar; en otro caso rechazar.
- Nota: C se puede construir creando un estado de C por cada par (a,b) de estados de A y B. Los estados finales de C son los que tienen un estado final de A o B (pero no los dos).

- Sea A_{CFG} el lenguaje formado por las cadenas <G,w> tales que G es la codificación de una gramática libre de contexto (CFG) y w es una cadena generada por dicha gramática.
- TEOREMA: **El lenguaje A_{CFG} es decidible**
- Demostración: Toda CFG puede expresarse en Forma Normal de Chomsky. Para derivar una cadena w de longitud n a partir de una gramática en Forma Normal de Chomsky son necesarios $2n$-1 pasos. Dada una gramática G se puede construir una máquina de Turing que genera la gramática equivalente en Forma Normal de Chomsky y, a continuación, genere todas las derivaciones posibles con 2n-1 pasos. Si alguna de estas derivaciones es w entonces se acepta; en caso contrario se rechaza.

- Sea E_{CFG} el lenguaje formado por las cadenas <G> tales que G es la codificación de una gramática libre de contexto (CFG) que solo reconoce el lenguaje vacío:

$$L(G) = \emptyset$$

- TEOREMA: **El lenguaje E_{CFG} es decidible**
- Demostración: Para que una gramática reconozca el lenguaje vacío es necesario que a partir del símbolo inicial no sea posible derivar ninguna cadena de símbolos terminales (una sentencia):
  1. Marcar todos los símbolos terminales
  2. Si un símbolo no terminal A no está marcado y tiene una regla en la que todos los símbolos de la derecha estén marcados, marcar A.
  3. Repetir el paso 2 hasta que no se marque ningún nuevo símbolo:
  4. Si el símbolo inicial no está marcado, aceptar; si no rechazar.

- TEOREMA: **Todo lenguaje libre de contexto es decidible**
- Demostración: Sea L un lenguaje libre de contexto. Esto quiere decir que existe una gramática libre de contexto, G, que reconoce dicho lenguaje. Dada una entrada w, se puede ejecutar la máquina de Turing descrita para el lenguaje A_{CFG} sobre la entrada $\langle G,w \rangle$. Si la máquina acepta $\langle G,w \rangle$ entonces se acepta; en caso contrario se rechaza.

- Sea A_{TM} el lenguaje formado por las cadenas $<M,w>$ tales que M es la codificación de una máquina de Turing y w es una cadena aceptada en dicha máquina.
- TEOREMA: **El lenguaje A_{TM} es indecidible**
- Demostración (por contradicción): Supongamos que A_{TM} es decidible. En tal caso, debe existir una máquina de Turing, H, que decida el lenguaje, es decir:

$$H(<M,w>) = \begin{cases} aceptar & \text{si } M \text{ acepta } w \\ rechazar & \text{si } M \text{ no acepta } w \end{cases}$$

- TEOREMA: **El lenguaje A_{TM} es indecidible**
- Demostración (sigue):
- A partir de la máquina H podemos definir una máquina D tal que $D(M)$ consiste en ejecutar $H(<M,M>)$ y devolver lo contrario, es decir:

$$D(<M>) = \begin{cases} aceptar & \text{si } H \text{ rechaza } <M,M> \\ rechazar & \text{si } H \text{ acepta } <M,M> \end{cases} = \begin{cases} aceptar & \text{si } M \text{ no acepta } M \\ rechazar & \text{si } M \text{ acepta } M \end{cases}$$

- Si ejecutamos D sobre la codificación de D

$$D(<D>) = \begin{cases} aceptar & \text{si } D \text{ no acepta } D \\ rechazar & \text{si } D \text{ acepta } D \end{cases}$$

- Si ejecutamos D sobre la codificación de D

$$D(< D >) = \begin{cases} aceptar & \text{si } D \text{ no acepta } D \\ rechazar & \text{si } D \text{ acepta } D \end{cases}$$

- Hemos encontrado una contradicción. Esto quiere decir que es imposible construir D. Pero esto implica que es imposible construir H, lo que significa que A_{TM} es indecidible.
- Nota: A_{TM} es reconocible porque podemos construir una máquina de Turing que lo acepta (la máquina de Turing universal).

(https://www.youtube.com/watch?v=92WHN-pAFCs)

- Dado un lenguaje L definido sobre un alfabeto Σ, se define el ***complemento de L*** como el lenguaje formado por las cadenas de Σ que no pertenecen a L.
- Decimos que un lenguaje L es ***co-reconocible-Turing*** si es el complemento de un lenguaje reconocible-Turing.

- TEOREMA: **Un lenguaje *L* es decidible si y solo si es reconocible y co-reconocible**.
- Demostración (→):
  - Si L es decidible, entonces existe una máquina de Turing tal que

$$M(w) = \begin{cases} aceptar & \text{si } w \in \mathrm{L} \\ rechazar & \text{si } w \notin \mathrm{L} \end{cases}$$

  - Por tanto, podemos crear una nueva máquina que reconozca el lenguaje complementario

$$N(w) = \begin{cases} aceptar & \text{si M rechaza } w \\ rechazar & \text{si M acepta } w \end{cases}$$

- TEOREMA: **Un lenguaje *L* es decidible si y solo si es reconocible y co-reconocible**.
- Demostración (←):
  - Si L es reconocible y co-reconocible, entonces existen dos máquinas de Turing tal que

$$M_1(w) = \begin{cases} aceptar & \text{si } w \in \mathrm{L} \\ no-parar & \text{si } w \notin \mathrm{L} \end{cases}$$

$$M_2(w) = \begin{cases} aceptar & \text{si } w \notin \mathrm{L} \\ no-parar & \text{si } w \in \mathrm{L} \end{cases}$$

  - Podemos ejecutar las dos máquinas en paralelo por medio de una máquina con dos cintas en la que en cada paso se ejecute un paso de cada máquina. Si M1 acepta la entrada, entonces hay que aceptar. Si M2 acepta la entrada, entonces hay que rechazar.

- COROLARIO: El complemento de A_{TM} es no reconocible.

- A_{TM} es reconocible. Si su complemento también lo fuera, entonces A_{TM} sería decidible.

- Sea $HALT_{TM}$ el lenguaje formado por las cadenas <*M*,*w*> tales que *M* es la codificación de una máquina de Turing y *w* es una cadena que hace que dicha máquina termine (ya sea aceptando o rechazando).
- TEOREMA: **El lenguaje $HALT_{TM}$ es indecidible**
- Demostración: (por reducción): Supongamos que $HALT_{TM}$ es decidible. En tal caso, debe existir una máquina de Turing, *R*, que decida el lenguaje, es decir:

$$R(< M, w >) = \begin{cases} aceptar & \text{si } M \text{ para ante } w \\ rechazar & \text{si } M \text{ no para ante } w \end{cases}$$

- TEOREMA: **El lenguaje $HALT_{TM}$ es indecidible**
- Demostración (sigue):
  - A partir de R podemos construir una máquina, S, que resuelve el problema A_{TM} . Puesto que es A_{TM} indecidible, S no puede existir. Por tanto R no puede existir.
  - Descripción de S (<M,w>):
    1. Ejecutar R(<M,w>).
    2. Si R rechaza (es decir, M no para ante w), *rechazar*.
    3. Si R acepta (es decir, M para ante w), simular M sobre w con una Máquina de Turing Universal.
    4. Si M acepta, *aceptar*; si M rechaza, *rechazar*.

- Sea E_{TM} el lenguaje formado por las cadenas <M> tales que M es la codificación de una máquina de Turing que no reconoce ninguna entrada, es decir, cuyo lenguaje es el lenguaje vacío:

$$L(M) = \emptyset$$

- TEOREMA: **El lenguaje E_{TM} es indecidible**
- Demostración (por reducción):
  - Dada una máquina M y una cadena w podemos construir una máquina M_1 definida como

$$M_1(x) = \begin{cases} \text{si } x \neq w & \textit{rechazar} \\ \text{si } x = w & \text{ejecutar } M(w) \text{ y aceptar si } M \text{ acepta} \end{cases}$$

- TEOREMA: **El lenguaje E_{TM} es indecidible**
- Demostración (por reducción):
  - Si E_{TM} es decidible, existe una máquina R que lo decide. A partir de R y de M_1 podríamos construir una máquina S de la siguiente forma:
  - Descripción de S ($\langle M,w \rangle$):
    1. Construir M_1 a partir de M y w.
    2. Aplicar R sobre M_1.
    3. Si R acepta, *aceptar*; si R rechaza, *rechazar*.
  - La máquina S resuelve el problema A_{TM}, que es indecidible. Por tanto, si S no puede existir , R tampoco.

- Sea EQ_{TM} el lenguaje formado por las cadenas $<M_1,M_2>$ tales que M_1 y M_2 son codificaciones de máquinas de Turing que reconocen el mismo lenguaje, es decir:

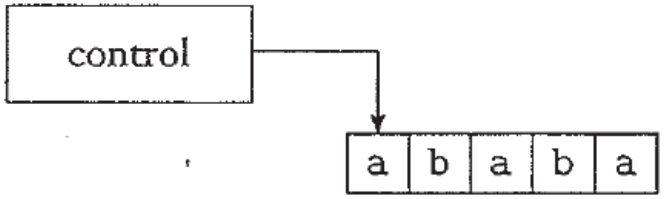
$$L(M_1) = L(M_2)$$

- TEOREMA: **El lenguaje EQ_{TM} es indecidible**
- Demostración (por reducción):
  - Es fácil crear una máquina M_1 que rechace todas las entradas. Si existe la máquina R que reconozca el lenguaje EQ_{TM}, entonces podemos construir S tal que

$$S(< M >) = \begin{cases} aceptar & \text{si } R(< M, M_1 >) \text{ acepta} \\ rechazar & \text{si } R(< M, M_1 >) \text{ rechaza} \end{cases}$$

  - La máquina S no puede existir ya que resolvería el problema E_{TM} . Por tanto R tampoco puede existir.

- Autómata Linealmente Acotado (*linear bounded automaton* – LBA)
  - Un Autómata Linealmente Acotado es una Máquina de Turing que trabaja sobre una cantidad de memoria limitada al tamaño de la entrada que debe estudiar.
  - Utilizando un alfabeto de escritura en la cinta que sea mayor que el alfabeto de entrada se puede simular una cinta de un tamaño n veces mayor.
  - Por tanto, un LBA es una Máquina de Turing con una cantidad de memoria que depende linealmente del tamaño de entrada.



- Autómata Linealmente Acotado (*linear bounded automaton* – LBA)
  - A pesar de la restricción, los LBA son bastante potentes. De hecho, los lenguajes regulares y los lenguajes libres de contexto pueden ser decididos mediante LBAs.

- Configuraciones
  - Las configuraciones son una forma de representar el estado instantáneo de una Máquina de Turing o de un Autómata Linealmente Acotado.

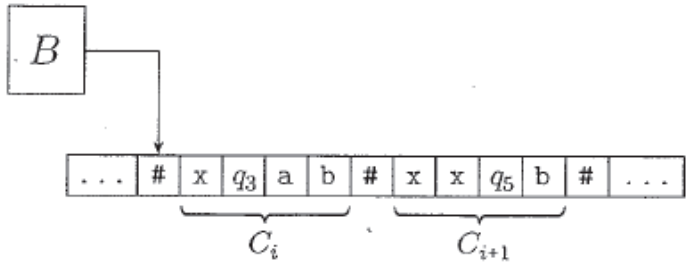
$$[x_1\, x_2 \ldots \mathbf{p}\, x_j \ldots x_n]$$

- Número de configuraciones diferentes en un LBA
  - Dado un LBA con q estados, un alfabeto de cinta de g símbolos y una cadena de entrada de longitud n, el número de configuraciones diferentes es

$$q \cdot n \cdot g^n$$

- Sea A_{LBA} el lenguaje formado por las cadenas <B,w> tales que B es la codificación de un Autómata Linealmente Acotado y w es una cadena aceptada por dicho autómata.
- TEOREMA: **El lenguaje A_{LBA} es decidible**
  - Podemos construir una Máquina de Turing que simule el funcionamiento del Autómata Linealmente Acotado sobre la entrada w. El resultado de la simulación será que el autómata acepta la cadena, la rechaza o no para (***accept***, ***reject***, ***loop***).
  - La única forma de que el Autómata Linealmente Acotado no pare es que entre en un bucle en el que se repita una configuración. Como el número de configuraciones distintas es finito ($C = q \cdot n \cdot g^n$), si el número de pasos simulados es mayor que C, entonces es seguro que el LBA está repitiendo configuraciones y, por tanto, se encuentra en un bucle infinito.
  - Es posible construir una Máquina de Turing que simule C pasos del Autómata Linealmente Acotado. Si en esos C pasos el autómata acepta o rechaza la cadena, la Máquina de Turing también la acepta o rechaza. Si se alcanzan los C pasos, la Máquina de Turing rechaza la cadena.

- Sea E_{LBA} el lenguaje formado por las cadenas <*B*> tales que *B* es la codificación de un Autómata Linealmente Acotado que no reconoce ninguna entrada, es decir, cuyo lenguaje es el lenguaje vacío ($L(B) = \emptyset$)
- TEOREMA: **El lenguaje E_{LBA} es indecidible**
- Demostración (por reducción a A_{TM})
  - Dada una Máquina de Turing, *M*, y una entrada *w*, se puede construir un Autómata Linealmente Acotado, *B*, cuya entrada sea una lista de configuraciones de *M* sobre *w* y cuyo funcionamiento consiste en verificar que la lista de configuraciones corresponde a una secuencia de configuraciones de *M* que conduzca a la aceptación de *w*.



- TEOREMA: **El lenguaje E_{LBA} es indecidible (sigue)**
  - El Autómata Linealmente Acotado debe verificar:
    a) Que la primera configuración corresponde a la configuración inicial de M sobre w, $[q_0\, w_1\, w_2 \ldots w_n\, \beta]$.
    b) Que la última configuración corresponde a aceptar, $[h\beta 1\, \beta]$.
    c) Que cada configuración corresponde a una transición correcta de M.
  - Todas estas instrucciones pueden realizarse mediante marcas y procesos de zig-zag en la cinta, pero no requieren consumir más espacio en la cinta de entrada. Por tanto, se pueden realizar mediante un Autómata Acotado Linealmente.
  - Si el problema E_{LBA} tiene solución, existe una Máquina de Turing, R, que puede comprobar que B reconoce el lenguaje vacío. Es decir, que no existe ninguna lista de configuraciones que permita a M reconocer w. Es decir, permite detectar que una Máquina de Turing, M, no va a detenerse al intentar reconocer w. Por tanto, si R existiera podríamos resolver el problema de la parada. Luego R no existe.

- Problema de correspondencia de Post
  - Dado un alfabeto, Σ, definimos una "pieza de dominó" como una pareja formada por dos cadenas del alfabeto

a
ab

  - Dada colección de piezas,

a	b	ca	abc
ab	ca	a	c

  - el problema de correspondencia consiste en encontrar una secuencia de piezas (se pueden repetir) que construya la misma cadena en la parte superior e inferior.

a	b	ca	a	abc
ab	ca	a	ab	c

- Problema de correspondencia de Post
  - El Problema de Correspondencia de Post, PCP, consiste en aceptar las colecciones de piezas que permitan construir una secuencia repetida y rechazar las colecciones que no lo permitan.
- TEOREMA: El problema PCP es indecidible
  - La demostración consiste en construir una colección de piezas que permita generar la secuencia de configuraciones de una Máquina de Turing, M, reconociendo una cadena w.

- TEOREMA: El problema PCP es indecidible
  - Primera pieza:

#
q_0 w_1 w_2 … w_n

  - Por cada transición δ(p,a) = (q, R) añadir

p a
a **q**

  - Por cada transición δ(p,a) = (q, b) añadir

p a
p b

  - Por cada transición δ(p,a) = (q, L) añadir

b **p** a	c **p** a
q b a	**q** c a

…

  - Para cada símbolo *a* añadir

a $\mathbf{q}_{accept}$	$\mathbf{q}_{accept}$ a
$\mathbf{q}_{accept}$	$\mathbf{q}_{accept}$

  - Última pieza

$\mathbf{q}_{accept}$ # #
#

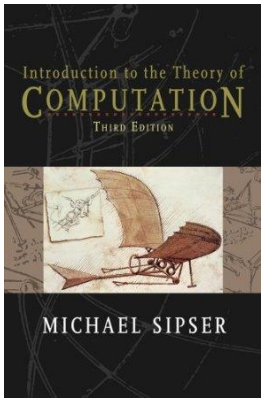
- TEOREMA: El problema PCP es indecidible
  - Encontrar una secuencia repetida con estas piezas equivale a encontrar una secuencia de configuraciones de la Máquina de Turing, M, que reconozca la cadena w.
  - Si existiera una solución para el Problema de Correspondencia de Post, podríamos saber si una Máquina de Turing, M, acepta la cadena w (si la colección de piezas puede producir una cadena repetida) o no la acepta (si la colección de piezas no puede producir una cadena repetida). Es decir, si existiera una solución podríamos resolver el problema A_{TM}, que sabemos que es indecidible.

- Definición de reducción
  - Una función f: $\Sigma^* \rightarrow \Sigma^*$, se dice que es ***computable*** si existe una Máquina de Turing que para cualquier entrada $w \in \Sigma^*$ alcance la parada y genere como salida $f(w)$.
  - Un lenguaje A es ***reducible por mapeo*** a un lenguaje B, ($A \leq_m B$), si existe una función computable f tal que

$$w \in A \Leftrightarrow f(w) \in B$$

  - La función f se denomina ***reducción de A a B***
  - **TEOREMA:** Si $A \leq_m B$ y B es decidible entonces A es decidible.
  - **TEOREMA:** Si $A \leq_m B$ y A es indecidible entonces B es indecidible.

Bibliografía



- Michael Sipser (2005). Introduction to the Theory of Computation (2nd Edition) Thompson. Capítulos 4 y 5.