AUTÓMATAS Y LENGUAJES FORMALES MÁQUINAS DE TURING Y DECIDIBILIDAD

Francisco Hernández Quiroz

Departamento de Matemáticas
Facultad de Ciencias, UNAM
E-mail: fhq@ciencias.unam.mx
Página Web: www.matematicas.unam.mx/fhq

Posgrado en Ciencia e Ingeniería de la Computación

Entscheidungsproblem I

Las máquinas de Turing se inventaron para contestar una pregunta abierta en lógica matemática:

- Sean $\alpha_1, \ldots, \alpha_n$ fórmulas del cálculo de predicados a las que llamaremos *axiomas*.
- Sea θ otra fórmula a la que llamaremos *candidato a teorema*.
- Queremos un método para poder contestar la pregunta

$$¿\alpha_1, \ldots, \alpha_n \models \theta?$$

(Que se lee "¿θ es una consecuencia lógica de $\alpha_1, \ldots, \alpha_n$?")

- Éste es el *problema clásico de la decisión* o *Entscheidungsproblem*.

Procedimiento efectivo

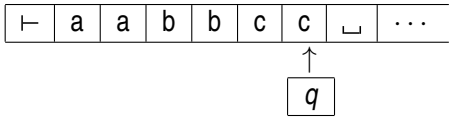
El método que buscamos deberá tener las siguientes características:

1. Deberá expresarse en términos tan formales como las instancias del problema.
2. Deberá ser "mecánico": su aplicación se basará en reglas que se siguen sin necesidad de interpretación o de "inspiración".
3. El método deberá especificarse de manera finita.
4. Los recursos utilizados deben ser finitos tanto espacial como temporalmente y, de preferencia, deben poder acotarse con anticipación.

Church y Turing demostraron que no existe un método para resolver el *Entscheidungsproblem* que cumpla con lo anterior.
De paso, inventaron dos de los modelos más populares de computación.

Máquinas de Turing (TM)

Una *máquina de Turing* es un mecanismo muy simple que consiste en una cinta de lectura y escritura, una cabeza de lectura y escritura que recorre esta cinta (y que puede estar en diferentes estados) y una función que regula los movimientos de la cinta.



Definición formal

Una máquina de Turing está formada por los siguientes elementos:

- un conjunto de estados Q;
- un alfabeto de entrada Σ;
- un alfabeto de cinta, con $\Sigma \subseteq \Gamma$;
- una función de transición $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\rightarrow, \leftarrow\}$
- los estados inicial, de aceptación y de rechazo $s, t, r \in Q$, respectivamente;
- el símbolo de espacio en blanco $\llcorner \in \Gamma - \Sigma$;
- el símbolo de inicio de la cinta $\vdash \in \Gamma$.

Es imposible moverse a la izquierda del símbolo de inicio.
Los estados de aceptación y rechazo no se pueden abandonar.

Aceptación y rechazo I

Una configuración es una terna

$$(q, \vdash \alpha \llcorner^{\infty}, n),$$

donde $q \in Q$, $\alpha \in \Gamma$ y $n \in \mathbb{N}$. La configuración inicial es

$$(s, \vdash \alpha \llcorner^{\infty}, 0) \qquad \alpha \in \Sigma^*.$$

Definiremos una relación entre configuraciones. Sean

$$\alpha_{[n]}$$

el n-ésimo símbolo de la cadena α y

$$\alpha_{[n/x]}$$

Aceptación y rechazo II

la cadena resultante de sustituir ese símbolo por x.
Entonces:

$$(q, \alpha, n) \Rightarrow (r, \beta, m) \quad \text{sii} \quad \delta(q, \alpha_{[n]}) = (r, b, \rightarrow), \beta = \alpha_{[n/b]}, m = n + 1$$
$$\text{o bien} \quad \delta(q, \alpha_{[n]}) = (r, b, \leftarrow), \beta = \alpha_{[n/b]}, m = n - 1$$

$\Rightarrow^*$ es la cerradura transitiva y reflexiva de esta relación.
Una máquina de Turing *M se detiene* con una entrada $\alpha \in \Sigma^*$ sii

$$(s, \vdash \alpha \llcorner^{\infty}, 0) \Rightarrow^* (q, \vdash \beta \llcorner^{\infty}, n) \qquad \text{con } q = t \text{ o bien } q = r.$$

El lenguaje aceptado por M es

$$L(M) = \{\alpha \in \Sigma^* \mid (s, \vdash \alpha \llcorner^{\infty}, 0) \Rightarrow^* (t, \vdash \beta \llcorner^{\infty}, n)\}.$$

Ejemplos

Una máquina que acepta $\{a^n b^n c^n\}$, con $Q = \{s, q_1, \ldots, q_{10}, t, r\}$, $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b, c, \vdash, \llcorner, \dashv\}$ y δ definida por la tabla

	$\vdash$	a	b	c	$\llcorner$	$\dashv$
s	$(s, \vdash, \rightarrow)$	$(s, a, \rightarrow)$	$(q_1, b, \rightarrow)$	$(q_2, c, \rightarrow)$	$(q_3, \dashv, \leftarrow)$	
q_1		$(r, -, -)$	$(q_1, b, \rightarrow)$	$(q_2, c, \rightarrow)$	$(q_3, \dashv \leftarrow)$	
q_2		$(r, -, -)$	$(r, -, -)$	$(q_2, c, \rightarrow)$	$(q_3, \dashv, \leftarrow)$	
q_3	$(t, -, -)$	$(r, -, -)$	$(r, -, -)$	$(q_4, \llcorner, \leftarrow)$	$(q_3, \llcorner, \leftarrow)$	
q_4	$(r, -, -)$	$(r, -, -)$	$(q_5, \llcorner, \leftarrow)$	$(q_4, c, \leftarrow)$	$(q_4, \llcorner, \leftarrow)$	
q_5	$(r, -, -)$	$(q_6, \llcorner, \leftarrow)$	$(q_5, b, \leftarrow)$		$(q_5, \llcorner, \leftarrow)$	
q_6	$(q_7, \vdash, \rightarrow)$	$(q_6, a, \leftarrow)$			$(q_6, \llcorner, \leftarrow)$	
q_7		$(q_8, \llcorner, \rightarrow)$	$(r, -, -)$	$(r, -, -)$	$(q_7, \llcorner, \rightarrow)$	$(t, -, -)$
q_8		$(q_8, a, \rightarrow)$	$(q_9, \llcorner, \rightarrow)$	$(r, -, -)$	$(q_8, \llcorner, \rightarrow)$	$(r, -, -)$
q_9			$(q_9, b, \rightarrow)$	$(q_{10}, \llcorner \rightarrow)$	$(q_9, \llcorner, \rightarrow)$	$(r, -, -)$
q_{10}				$(q_{10}, c, \rightarrow)$	$(q_{10}, \llcorner, \rightarrow)$	$(q_3, \dashv, \leftarrow)$

Variantes de máquinas de Turing

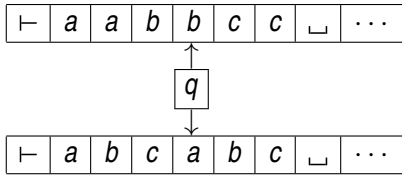
Existen variantes de las TM que no añaden poder computacional (aunque sí ventajas prácticas):

- Máquinas con varias cintas (se presentará el caso particular de dos cintas).
- Máquinas con cintas infinitas en ambos sentidos.
- Máquinas no deterministas.

El último caso tiene interés especial en la teoría de la complejidad.
Todas estas variantes pueden reducirse al modelo original.

Varias cintas I

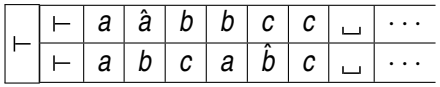
Una máquina de Turing con dos cintas tiene una función de transición $\delta : Q \times \Gamma^2 \to Q \times \Gamma^2 \times \{\rightarrow, \leftarrow\}^2$



Esta máquina se puede simular con una máquina de una sola cinta con alfabeto

$$\Gamma_1 = \{\vdash\} \cup \Sigma \cup (\Gamma \cup \{\hat{a} \mid a \in \Gamma\})^2.$$

La cinta de esta máquina se ve así



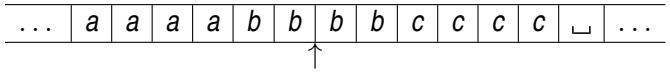
Varias cintas II

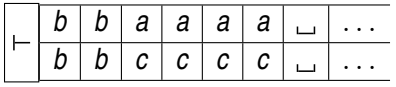
La nueva máquina utiliza con símbolos del tipo $\hat{a}$ para indicar la posición en cada una de las cintas de la máquina original y simula de esta forma una transición original con una serie de transiciones en la nueva máquina.

Cinta infinita en ambos sentidos

Una máquina con cinta infinita en ambos sentidos se puede simular por medio de una máquina con dos cintas:

- Se elige un punto arbitrario en la cinta original.
- Se divide la cinta original a partir de este punto.
- Se copia la información de la cinta original a las dos cintas nuevas, invirtiendo el orden de una de ellas para simular la parte infinita hacia la izquierda.





TM no deterministas I

Una máquina de Turing no determinista (NTM) es similar a una TM, excepto por

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{\to, \leftarrow\}).$$

La aceptación se produce sii existe una secuencia de elecciones que lleve al estado de aceptación.
Para reducir una NTM a un TM se utilizarán tres cintas:

- la primera cinta tendrá la entrada original;
- la segunda tendrá secuencias de los dígitos 1, . . . , r, separadas por $\#$ y ordenadas lexicográficamente:

1	#	2	#	···	#	r	#	11	#	12	#	···	#	rr	#	111	#	112	#	···

TM no deterministas II

- en la tercera cinta se copiará la entrada original y se "procesará" de acuerdo con las instrucciones de la máquina original, elegidas de acuerdo con las secuencias de la cinta 2:

donde r es el número máximo de opciones de la función δ original.
Esta máquina aceptará la entrada sii en alguna de las simulaciones de la máquina original acepta.

Lenguajes recursivos y recursivamente enumerables

Sea $M \in$ TM. Dada una cadena $\alpha \in \Sigma^*$, tenemos tres posibilidades:

1. M acepta α, es decir, $\langle s, \vdash\alpha\sqcup^\omega, 0\rangle \Rightarrow_M^* \langle t, \vdash\beta\sqcup^\omega, n\rangle$;
2. M rechaza α, es decir, $\langle s, \vdash\alpha\sqcup^\omega, 0\rangle \Rightarrow_M^* \langle r, \vdash\beta\sqcup^\omega, n\rangle$;
3. M no hace ninguna de las dos cosas anteriores pues entra en un *ciclo* infinito.

Diremos que M es *total* si para toda $\alpha \in \Sigma^*$, M acepta o rechaza α.
Definiremos ahora dos tipos de lenguajes. Sea $L \subseteq \Sigma^*$.

- L es *recursivo* (rec) sii existe una $M \in$ TM *total* tal que $L = L(M)$;
- L es *recursivamente enumerable* (r.e.) sii existe una $M \in$ TM tal que $L = L(M)$;

Como se verá más adelante, algunos lenguajes que son r.e. *no* son rec.

Propiedades decidibles, indecidibles y semidecidibles

Sea P una propiedad de cadenas en Σ^*, i.e., $P(\alpha) = V$ sii P es verdadera de α. Por ejemplo, la propiedad

$$\text{Primo}(\alpha) = V \qquad \text{sii} \qquad |\alpha| = n \text{ y } n \text{ es primo.}$$

Una propiedad P es

decidible	sii	$\{\alpha \mid P(\alpha) = V\}$ es recursivo
indecidible	sii	$\{\alpha \mid P(\alpha) = V\}$ no es recursivo
semidecidible	sii	$\{\alpha \mid P(\alpha) = V\}$ es r.e.

Indecidibilidad

- De acuerdo con la definición,todas las propiedades decidibles son semidecidibles.
- Sin embargo, algunas propiedades semidecidibles son indecidibles.
- Peor todavía, existen también propiedades indecidibles que no son siquiera semidecidibles.

Para demostrar lo anterior, construiremos máquinas de Turing que se basan en la idea de universalidad.

Codificación de máquinas de Turing I

El primer paso es representar de manera adecuada una máquina de Turing adecuada. Un esquema posible es el siguiente. Sea
$M = \langle Q, \Sigma, \Gamma, \delta, s, t, r, \vdash, \sqcup \rangle$ una TM con

$$
\begin{array}{rcl}
|Q| & = & i \\
|\Sigma| & = & j \\
|\Gamma| & = & k \\
s & & \text{el } m\text{-ésimo estado} \\
t & & \text{el } n\text{-ésimo estado} \\
r & & \text{el } p\text{-ésimo estado} \\
\vdash & & \text{el } q\text{-ésimo carácter de } \Gamma \\
\sqcup & & \text{el } r\text{-ésimo carácter de } \Gamma
\end{array}
$$

Codificación de máquinas de Turing II

Entonces la cadena binaria siguiente codifica esta información

$$0^i 1 0^j 1 0^k 1 0^m 1 0^n 1 0^p 1 0^q 1 0^r 1$$

Y una transición en delta $\delta(s_u, a_v) = (s_w, a_x, \rightarrow)$ está representada por la cadena

$$0^u 1 0^v 1 0^w 1 0^x 1 0 1$$

Mientras que la transición $\delta(s_u, a_v) = (s_w, a_x, \leftarrow)$ queda codificada en la cadena

$$0^u 1 0^v 1 0^w 1 0^x 1 1 1.$$

Finalmente una cadena en Γ de la forma

$$a_{n_1} a_{n_2} \cdots a_{n_z}$$

Codificación de máquinas de Turing III

queda codificada por la cadena

$$0^{n_1} 1 0^{n_2} 1 \cdots 0^{n_z} 1$$

En adelante, designaremos con $\langle M \rangle$ y $\langle \alpha \rangle$ las codificaciones de la TM M y la cadena α, respectivamente.
Si las queremos combinar en una cadena usaremos el símbolo $\#$ para separarlas:

$$\langle M \rangle \# \langle \alpha \rangle$$

Máquina de Turing universal

- Dada esta codificación, se puede crear un máquina de Turing que, para todas $M \in$ TM y $\alpha \in \Sigma_M$, *simule* la ejecución de las instrucciones de M con la entrada α a partir de la cadena

$$\langle M \rangle \# \langle \alpha \rangle.$$

- Llamaremos *máquina de Turing universal* (o para abreviar TMU) a esta máquina y, con algunas leves modificaciones propias de cada caso, la usaremos como base para construir TM especiales.
- Por ejemplo, podemos modificar la TMU de tal forma que al terminar de simular M con α acepte la cadena $\langle M \rangle \# \langle \alpha \rangle$ sii M aceptó α.

Problema de la detención HP

- Con las codificaciones anteriores podemos definir dos conjuntos:

$$\begin{aligned} \text{HP} &= \{\langle M \rangle \# \langle \alpha \rangle \mid M \text{ se detiene con la entrada } \alpha\} \\ \text{MP} &= \{\langle M \rangle \# \langle \alpha \rangle \mid \alpha \in L(M)\}. \end{aligned}$$

- El primer conjunto debe su nombre a que describe lo que se conoce como el *problema de la detención* (halting problem); y el segundo, el *problema de la pertenencia* (membership problem).
- Si HP fuera rec, la pregunta de si una TM dada se detiene con una entrada particular sería decidible. Algo análogo se puede plantear para MP.
- En cambio, es muy fácil ver que ambos son r.e., pues una leve adaptación de la TMU nos lleva a tener TM que aceptan HP y MP (véase la lámina anterior).

HP no tiene solución total I

- HP no es rec, i.e. la pregunta de si una TM se detiene con una entrada no es decidible. Si aceptamos la tesis de Church-Turing, esto equivale a decir que no existe un *procedimiento efectivo* para responder siempre esta pregunta.
- Antes de demostrar lo anterior, véase cómo la codificación de las TM nos permite ordenarlas lexicográficamente.
- Se puede hacer lo mismo con las cadenas y, gracias a esto, construir una tabla como la siguiente:

TM/cad.	ϵ	0	1	...	α	...
M_ϵ	s	s	n	...	n	...
M_0	s	s	s	...	n	...
...	...	...	...	...	...	...
M_β	n	n	s	...	n	...

HP no tiene solución total II

donde tenemos s en la entrada (M_β, α) si la TM codificada por la cadena β se detiene con la entrada α; y n, en caso contrario.

- Por reducción al absurdo, supongamos que existe una TM total H tal que

$$L(H) = \text{HP} = \{\langle M \rangle \# \langle \alpha \rangle \mid M \text{ se detiene con } \alpha\}$$

- H nos permitiría construir la tabla anterior sin error, pues

$$\langle M \rangle \# \langle \alpha \rangle \in L(H)$$

sii M se detiene con la entrada α.

- Considérese a hora la siguiente TM, $\overline{H}$: para toda entrada $\langle M \rangle \# \langle \alpha \rangle$, $\overline{H}$ la acepta si H la rechaza, y entra en un ciclo trivial en caso contrario.
- Obsérvese cómo $\overline{H}$ es muy fácil de construir, suponiendo que contamos ya con H.

HP no tiene solución total III

- Por otro lado, la lista de TM codificada es exhaustiva (pero veremos que $\overline{H}$ no está ahí).
- Nos podemos preguntar ahora si

$$\langle \overline{H} \rangle \# \langle \overline{H} \rangle \in \text{HP},$$

lo que equivale a decir que $\overline{H}$ se detiene cuando recibe como entrada la cadena que la codifica.

- Pero en ese caso, por la definición de $\overline{H}$, esta TM debe entrar en un ciclo trivial cuando tiene como entrada su propia descripción, lo cual contradice nuestra hipótesis.
- Si por el contrario suponemos que

$$\langle \overline{H} \rangle \# \langle \overline{H} \rangle \notin \text{HP},$$

llegamos a una contradicción análoga.

HP no tiene solución total IV

- En conclusión, la suposición de que H existe nos lleva una contradicción.
- La demostración anterior se basa en el método de diagonalización de Cantor, pues $\overline{H}$ es la negación de la diagonal de la tabla anterior, lo que la hace diferir de todas los renglones de la lista.
- Esto contradice el supuesto de que la lista es exhaustiva.

Problemas indecidibles o que no son semidecidibles

- El problema de la detención no es el único indecidible; el problema de la pertenencia tampoco es decidible.
- Sin embargo, ambos problemas son semidecidibles, i.e., HP y MP son r.e.
- Para demostrarlo, se necesitan adaptaciones muy simples de la TMU. Dadas $M \in$ TM y $\alpha \in \Sigma^*$:
  - se simula M con α y se acepta $\langle M \rangle \# \langle \alpha \rangle$ si M acepta α (el lenguaje aceptado es MP);
  - se simula M con α, pero se acepta $\langle M \rangle \# \langle \alpha \rangle$ si M termina, ya sea con aceptación o rechazo (el lenguaje aceptado es HP).
- Otros problemas no son siquiera semidecidibles, e.g. determinar si una TM *no* se detiene con una entrada:

$$\sim HP = \{\langle M \rangle \# \langle \alpha \rangle \mid M \text{ no se detiene con } \alpha\}.$$

Complementos de lenguajes I

El caso de $\sim$HP se puede demostrar por medio de un teorema muy general.

Definición

Sea $L \subseteq \Sigma^*$. Designaremos su complemento con $\sim L$. Si $\sim L$ es rec (r.e.), entonces diremos que L es co-rec. (co-r.e.).

Teorema

(a) Si tanto L como $\sim L$ son r.e., entonces ambos son rec.
(b) Si L es rec, entonces es co-rec.

Dem. (a) Si L y $\sim L$ son r.e., existen $M, \overline{M} \in TM$ tales que

$$L(M) = L \qquad L(\overline{M}) = \sim L.$$

Complementos de lenguajes II

La TMU se puede adaptar muy para obtener $N \in TM$ tal que

$$L(N) = L \text{ y } N \text{ es total.}$$

N funciona así:

- N tiene dos cintas y $\forall \alpha \in \Sigma^*$
- N simula M en la primera cinta
- N simula $\overline{M}$ en la segunda cinta
- las simulaciones se realizan alternando pasos (para evitar la posibilidad de que N entre en un ciclo infinito si M o $\overline{M}$ lo hacen)
- si M acepta α, N acepta
- si $\overline{M}$ acepta α, N rechaza.

Complementos de lenguajes III

Dado que $\alpha \in L = L(M)$ o bien $\alpha \in \sim L = L(\overline{M})$, N siempre se detiene y, por tanto, L es recursivo.
Se puede construir una TM total para $\sim L$ de manera análoga.
Para (b) la hipótesis implica la existencia de una TM total que acepta L.
Basta modificar esta TM invirtiendo los estados de aceptación y rechazo para obtener otra TM total que acepta $\sim L$.

Corolario

Corolario. $\sim$HP no es r.e.

Dem. HP es r.e, y (a) implicaría que si $\sim$HP también fuera r.e., ambos serían rec y ya se demostró que HP no es rec.

Funciones computables

- Hasta ahora hemos visto las TM como mecanismos para decidir problemas (i.e., responder preguntas con sí o no).
- Pero también podríamos verlas como formas de calcular funciones.
- Sea $M \in TM$, con Σ y Γ como alfabetos de entrada y de la cinta, respectivamente. Entonces, podemos definir una función $f_M : \Sigma^* \to \Gamma^*$ como

$$\{(\alpha, \beta) \mid \langle s_M, \alpha, 0\rangle \Rightarrow_M^* \langle t, \beta, n\rangle\}.$$

- Obsérvese que f_M no tiene por qué ser total.
- Por otro lado, diremos que la función $\sigma : \Sigma^* \to \Delta^*$ es:
  (a) *Turing-computable* sii existe una $M \in TM$ tal que $\sigma = f_M$;
  (b) *computable* sii existe una $M \in TM$ *total* tal que $\sigma = f_M$.

Reducción $\leq_m$ I

Definición

Sean $A \subseteq \Sigma^*$ y $B \subseteq \Delta^*$. Una reducción (no inyectiva) de A a B es una función computable $\sigma : \Sigma^* \to \Delta^*$ tal que para toda $\alpha \in \Sigma^*$

$$\alpha \in A \quad \text{sii} \quad \sigma(\alpha) \in B.$$

En general, escribiremos $A \leq_m B$ sii existe una reducción de A a B.

El siguiente teorema será de gran utilidad en adelante:

Teorema

Sean $A, B \in \Sigma^*$ y $A \leq_m B$. Entonces:
(a) Si B es r.e (rec), entonces A también es r.e (rec).
(b) Si A no es r.e. (rec), entonces B tampoco es r.e. (rec).

Reducción $\leq_m$ II

Dem. Por hipótesis, existe σ computable tal que para toda $\alpha \in \Sigma^*$

$$\alpha \in A \qquad \text{sii} \qquad \sigma(\alpha) \in B.$$

En el caso (a) existe una $M \in TM$ (total) tal que $L(M) = B$. Entonces, podemos construir una $N \in TM$ (total) tal que $L(N) = A$.
N procede así:

- para toda α, N calcula $\sigma(\alpha)$;
- simula M con $\sigma(\alpha)$;
- acepta si M acepta $\sigma(\alpha)$.

Entonces

$$N \text{ acepta } \alpha \text{ sii } M \text{ acepta } \sigma(\alpha) \text{ sii } \sigma(\alpha) \in B \text{ sii } \alpha \in A.$$

En cuanto a (b), es fácil ver que la reducción σ nos permitiría decidir la pertenencia a A por medio de la pertenencia a B si B fuera r.e. (rec).

Ejemplos

Sea

$$FIN = \{\langle M \rangle \mid L(M) \text{ es finito}\}$$

Entonces

$$HP \leq_m MP$$

$$\sim HP \leq_m FIN.$$

El primer ejemplo se resuelve con una $K \in TM$ total que, a partir de una cadena $\langle M \rangle \# \langle \alpha \rangle$, genera otra $M' \in TM$ que, para toda entrada β, simula M con α y, si esta se detiene, acepta β.
M' acepta todas las cadenas sii M se detiene con α.
K computa la σ deseada y la pregunta ¿$\langle M \rangle \# \langle \alpha \rangle \in HP$? se convierte en la pregunta ¿$\langle M' \rangle \# \langle \gamma \rangle \in MP$?, para cualquier γ arbitraria.
El segundo caso se puede demostrar con una construcción similar.

Propiedades no triviales de lenguajes r.e.

Sea Σ un alfabeto y consideremos ahora sólo los subconjuntos de Σ^* que sean lenguajes r.e.
Nos interesan propiedades de lenguajes r.e. como las siguientes:

- ser regular;
- ser un CFL;
- ser infinito;
- incluir a ϵ.

Sea P una propiedad y L un lenguaje. Si P es cierta de L escribiremos $P(L) = V$. En caso contrario, $P(L) = F$.
Una propiedad que es cierta de algunos lenguajes r.e. pero no de otros es *no trivial*. Todos los ejemplos anteriores son no triviales.

Propiedades decidibles

Hay que observar que hablamos de propiedades de lenguajes y no de cadenas. Sin embargo, los conceptos de *decidibilidad, semidecibilidad e indecibilidad* se pueden extender a estas propiedades.
Sea P una propiedad y sea L un r.e. Entonces P es *decidible* sii

$$\exists M, N \in \text{TM} \,.\, L(N) = L \wedge M \text{ es total } \wedge (\langle N \rangle \in L(M) \Leftrightarrow P(L) = V).$$

Indecibilidad y semidecibilidad se definen de manera análoga.

Teorema de Rice I

Sea P una propiedad no trivial de lenguajes r.e. Entonces P es indecidible.
Demostración. Reduciremos HP al lenguaje

$$\{\langle M\rangle \mid P(L(M)) = V\}.$$

Supongamos que $P(\emptyset) = F$. Como P no es trivial, existe A r.e. tal que

$$P(A) = V.$$

Sea $M_A \in \text{TM}$ tal que $L(M_A) = A$.
Dada una $N \in \text{TM}$ y una $\alpha \in \Sigma_N^*$, construimos una $K = \sigma(\langle N\rangle\#\langle\alpha\rangle)$ tal que para toda $\beta \in \Sigma_K^*$

1. preserva una copia de β en una cinta separada;
2. escribe α en su cinta;
3. simula N con α;

Teorema de Rice II

4. si N se detiene con α, simula M_A con β y acepta si M_A acepta.

Si $\langle N\rangle\#\langle\alpha\rangle \notin \text{HP}$, $L(K) = \emptyset$. En caso contrario, $L(K) = L(M_A) = A$. Es decir,

$$\begin{array}{lllll} \langle N\rangle\#\langle\alpha\rangle \in \text{HP} & \Rightarrow & L(K) = A & \Rightarrow & P(L(K)) = P(A) = V \\ \langle N\rangle\#\langle\alpha\rangle \notin \text{HP} & \Rightarrow & L(K) = \emptyset & \Rightarrow & P(L(K)) = P(\emptyset) = F \end{array}$$

es decir,

$$\text{HP} \leq_m \{\langle M\rangle \mid P(L(M)) = V\}.$$

Como HP no es rec, $\{\langle M\rangle \mid P(L(M)) = V\}$ tampoco lo es y P es indecidible.

Segundo teorema de Rice I

Sea P una propiedad de lenguajes r.e. Diremos que P es monótona si $\forall A, B$ r.e.

$$A \subseteq B \wedge P(A) = V \Rightarrow P(B) = V.$$

Por ejemplo, la propiedad de contener a ϵ es monótona, lo mismo que la propiedad de ser infinito, pero las propiedades de ser finito o regular no son monótonas.
Segundo teorema de Rice. Sea P una propiedad de lenguajes r.e. no monótona. Entonces P no es siquiera semidecidible.
Demostración. Por reducción de $\sim$ HP al conjunto

$$\{\langle M\rangle \mid P(L(M)) = V\}.$$

Para esto, $\forall M \in \text{TM}$, $\forall \alpha \in \Sigma_M^*$ construimos una $K \in \text{TM}$ tal que

$$\langle M\rangle\#\langle\alpha\rangle \in \sim \text{HP} \qquad \text{sii} \qquad \langle K\rangle \in \{\langle M\rangle \mid P(L(M)) = V\}.$$

Segundo teorema de Rice II

Como P no es monótona, existen A, B r.e. tales que

$$A \subseteq B \qquad P(A) = V \qquad P(B) = F.$$

Sean $M_A, M_B \in \text{TM}$ tales que

$$L(M_A) = A \qquad L(M_B) = B.$$

K contará con tres cintas y $\forall\beta$ procederá de la siguiente forma

1. escribe β en las cintas 1 y 2;
2. escribe α en la cinta 3;
3. simula M_A en la cinta 1, M_B en la cinta 2 y M en la cinta 3;
4. acepta β si ocurre cualquiera de los siguientes casos:
   1. M_A acepta β o bien
   2. M_B acepta β y M se detiene con α.

Segundo teorema de Rice III

Entonces

$$\langle M\rangle\#\langle\alpha\rangle \in \text{HP} \quad \Rightarrow \quad L(K) = L(M_B) \quad \Rightarrow \quad P(L(K)) = P(L(M_B)) = F$$
$$\langle M\rangle\#\langle\alpha\rangle \in\sim \text{HP} \quad \Rightarrow \quad L(K) = L(M_A) \quad \Rightarrow \quad P(L(K)) = P(L(M_A)) = V$$

Incomputabilidad y sus grados

- Hasta ahora parece que los conjuntos incomputables se reducen a dos tipos: los que no son recursivos y los que ni siquiera son recursivamente enumerables.
- En realidad, el panorama puede ser más complejo: los conjuntos no r.e. son clasificables en función de su grado de incomputabilidad.
- El método básico es determinar qué problema necesitamos resolver para que un conjunto se vuelva rec o r.e.
- Las soluciones a problemas de decisión se pueden clasificar por medio de *oráculos*.

Máquinas de Turing con oráculo

- Un oráculo es una cinta adicional donde se codifica la *función característica* de un conjunto.
- Supongamos que ordenamos las cadenas en un alfabeto Σ y que la función $\nu : \Sigma^* b \to \mathbb{N}$ nos dice la posición que le corresponde a una cadena en el orden lexicográfico.
- Sea $A \subseteq \Sigma^*$. El oráculo de A, denotado por O_A es una cinta infinita que en cada celda guarda 0 o 1 de acuerdo con la regla

$$\alpha \in A \qquad \text{sii} \qquad O_A[n] = 1.$$

- Una máquina de Turing con oráculo (TMO) es una TM que durante un momento de su ejecución puede entrar en un estado tal que puede consultar el valor de una celda de un oráculo.

Reductibilidad-Turing

- Sean $A, B \subseteq \Sigma^*$. A es *recursivamente enumerable en B* sii existe una $M \in$ TMO con oráculo O_B tal que

$$A = L(M).$$

- Si M es total, entonces A es *recursiva en B* o *Turing-reductible a B*. En símbolos

$$A \leq_T B.$$

- Ejemplos: MP $\leq_T$ HP y HP $\leq_T$ MP.
- Obsérvese que HP $\leq_T\sim$ HP y viceversa, pero $\sim$ HP $\not\leq_m$ HP.

Jerarquía aritmética

La relación $\leq_T$ nos permite clasificar lenguajes no r.e.:

$$\begin{aligned}
\Sigma_1^0 &= \{\text{lenguajes r.e.}\} \\
\Delta_1^0 &= \{\text{lenguajes rec}\} \\
\Sigma_{n+1}^0 &= \{\text{lenguajes r.e. en algún } L \in \Sigma_n^0\} \\
\Delta_{n+1}^0 &= \{\text{lenguajes rec en algún } L \in \Sigma_n^0\} \\
\Pi_n^0 &= \{\text{complementos de lenguajes en } \Sigma_n^0\}
\end{aligned}$$

Alternativamente, un lenguaje L es r.e. sii existe una propiedad R *decidible* de pares de cadenas tal que $L = \{\alpha \mid \exists\beta \,.\, R(\alpha, \beta)\}$. Por ejemplo

$$\begin{aligned}
HP &= \{\langle M\rangle\#\langle\alpha\rangle \mid \exists x \,.\, M \text{ se detiene con } \alpha \text{ en } x \text{ pasos}\} \\
MP &= \{\langle M\rangle\#\langle\alpha\rangle \mid \exists x \,.\, M \text{ acepta } \alpha \text{ en } x \text{ pasos}\}
\end{aligned}$$

Versión lógica de la jerarquía aritmética

Las observaciones anteriores se pueden generalizar por medio del siguiente teorema:

1. Un lenguaje L está en Σ_n^0 sii una propiedad R $(n+1)$-aria decidible tal
$$L = \{\alpha \mid \exists\beta_1\forall\beta_2 \ldots \,.\, R(\alpha, \beta_1, \ldots, \beta_n)\}$$
2. Un lenguaje L está en Π_n^0 sii una propiedad R $(n+1)$-aria decidible tal
$$L = \{\alpha \mid \forall\beta_1\exists\beta_2 \ldots \,.\, R(\alpha, \beta_1, \ldots, \beta_n)\}$$
3. $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$.

Completitud

Definición. Sea $\mathcal{C}$ un nivel de la jerarquía aritmética y sea L un lenguaje. Diremos que L es

- $\mathcal{C}$-difícil sii para todo $A \in \mathcal{C}$, $A \leq_m L$.
- $\mathcal{C}$-completo sii L es $\mathcal{C}$-difícil y $L \in \mathcal{C}$.

Por ejemplo, HP y MP son Σ_1^0-completos.