Máquinas de Turing
Jerarquía de la Computabilidad

Ulises Jeremias Cornejo Fandos
Lucas Di Cunzolo

Ejercicio 1

Responder brevemente los siguientes incisos.

1. **¿Qué es un problema (computacional) de decisión? ¿Es el tipo de problema más general que se puede formular?**

   Dentro del universo de los problemas, se define la frontera de "lo computable", donde un problema computacional de decisión es un problema el cual se resuelve algorítmicamente para un dominio dado.

   Se entiende que este conjunto de problemas no es el conjunto más general dado que existen aquellos problemas que no pueden ser resueltos algorítmicamente, siendo estos aquellos que son no deterministicos para ciertos subconjuntos de un dominio dado, así como tambien aquellos problemas que ni siquiera son computables.

   De igual forma existen problemas que pueden ser resueltos algoritmicamente y son más generales que los problemas de decisión como por ejemplo los problemas de búsqueda.

2. **¿Qué cadenas integran el lenguaje aceptado por una MT?**

   Sea L un lenguaje en Σ^* y sea M un MT donde L(M) es el lenguaje aceptado por M, decimos que L(M)=L si y solo si M para en un estado de aceptación.

3. **En la clase teórica 1 se hace referencia al problema de satisfactibilidad de las fórmulas booleanas (se da como ejemplo la fórmula $\varphi=(x1\vee x2)\wedge(x3\vee x4)$ y la asignación A = (V, F, V, V)). Formular las tres formas del problema, teniendo en cuenta las tres visiones de MT consideradas: calculadora, aceptadora o reconocedora, y generadora.**

   Las tres formas se plantean de la siguiente forma,

   - **Calculadora**

     La MT que acepta o rechaza si la fórmula es satisfactible. (de decisión).

   - **Reconocedora**

     La MT genera una asignación de valores de verdad que hacen satisfactible a la fórmula. (si es que existe).

   - **Generadora**

     La MT que enumera todas las fórmulas satisfactibles. (no importa el input).

4. **¿Qué postula la Tesis de Church-Turing?**

   La conjetura conocida como *Tesis de Church-Turing* postula que todo lo computable puede ser llevado a cabo por una máquina de Turing.

5. **¿Cuándo dos MT son equivalentes? ¿Cuándo dos modelos de MT son equivalentes?**

   Dos Maquinas de Turing son computacionalmente equivalentes si *ambas reconocen el mismo lenguaje*. Luego, dos modelos de Maquinas de Turing son equivalentes si para toda MT de un modelo existe una MT de otro que sea equivalente.

6. **¿En qué difiere un lenguaje recursivo de un lenguaje recursivamente enumerable no recursivo?**

   Un lenguaje L es recursivamente numerable $L\in RE$ si y sólo si existe una MT M_l que lo acepta, es decir $L(M_l)=L$. Por lo tanto, para toda cadena w de Σ*:

   - Si $w\in L$, entonces M_l a partir de w para en su estado q_A.
   - Si $w\notin L$, entonces M_l a partir de w para en su estado q_R o no para.

   Un lenguaje L es recursivo, $L\in R$, si y sólo si existe una MT M_l que lo acepta y para siempre (también se puede decir directamente que lo decide). Por lo tanto, para toda cadena w de Σ*:

- Si $w \in L$, entonces M_l a partir de w para en su estado q_A.
- Si $w \notin L$, entonces M_l a partir de w para en su estado q_R.

Luego, sea $L \in RE - R$, diremos que L es un lenguaje recursivamente numerable no recursivo, es decir, que si existe una MT M_l que lo acepta, la misma no para siempre.

7. ¿En qué difiere un lenguaje recursivamente numerable de uno que no lo es?

Sea L un lenguaje computable que no es recursivamente numerable, es decir $L \in CO - RE - R$, entonces no existe una MT M_l tal que $L(M_l) = L$. Sin embargo, sabemos que existe una MT M_l talque $L(M_l) = L^c$.

Luego, si L fuese un lenguaje no computable, sabemos que existe una MT que lo reconozca.

Por lo tanto, sabemos que sea L un que no es recursivamente numerable, es decir $L \notin RE$, entonces no existe una MT M_l tal que $L(M_l) = L$.

8. Probar que $R \subseteq RE \subseteq L$.

Asumiendo un alfabeto universal de símbolos: $\Sigma = \{a1, a2, a3, ...\}$.

Σ* es el conjunto de todas las cadenas finitas formadas con símbolos de Σ. $\mathfrak{L}$ es el conjunto de todos los lenguajes formados con cadenas de Σ*: $\mathfrak{L}$ = P(Σ*), es decir que $\mathfrak{L}$ es el conjunto de partes de Σ*.

Un lenguaje L es recursivamente numerable $L \in RE$ si y sólo si existe una MT M_l que lo acepta, es decir $L(M_l) = L$. Por lo tanto, para toda cadena w de Σ*:

- Si $w \in L$, entonces M_l a partir de w para en su estado q_A.
- Si $w \notin L$, entonces M_l a partir de w para en su estado q_R o no para.

Un lenguaje L es recursivo, $L \in R$, si y sólo si existe una MT M_l que lo acepta y para siempre (también se puede decir directamente que lo decide). Por lo tanto, para toda cadena w de Σ*:

- Si $w \in L$, entonces M_l a partir de w para en su estado q_A.
- Si $w \notin L$, entonces M_l a partir de w para en su estado q_R .

Luego, por definición se cumple $R \subseteq RE \subseteq L$.

9. ¿Cuándo un lenguaje está en la clase CO-RE? ¿Puede un lenguaje estar al mismo tiempo en la clase RE y en la clase CO-RE? ¿Para todo lenguaje de la clase CO-RE existe una MT que lo acepta?

Sea CO-RE la clase de los lenguajes complemento, con respecto a Σ*, de los lenguajes recursivamente numerables. Decimos que un lenguaje está en CO-RE si su complemento pertenece a RE. Existen lenguajes que pertenecen a RE como a CO-RE, es decir, pertenecen a RE ⋂ CO-RE. Es el caso de los lenguajes pertenecientes a R. Por otro lado, no todo lenguaje de la clase CO-RE presenta una MT que lo acepte. Lo que si se sabe, es que ningun lenguaje perteniente a CO-RE – R tiene una MT que lo acepte. La única información que se tiene de dichos lenguajes es que seguro existe un "primo lejano", *su complemento*, perteneciente a RE el cual si será aceptado por una MT.

10. Justificar por qué los lenguajes Σ* y $\emptyset$ son recursivos.

Sabemos por definición de Σ* que el mismo es un lenguaje infinito de cadenas finitas formado a partir de un alfabeto Σ, también finito por definición. Además, sabemos que Σ* es un conjunto recursivamente enumerable dado que existe un MT que compute su orden canónico. Luego, se entiende que el lenguaje Σ* pertenece a RE lo cual nos dice que siempre existe una MT que al menos lo acepte. Por otro lado, sabemos que dado un input w ∈ Σ* el mismo será finito y bastará con una cantidad finita de pasos dados por una MT para recorrerlo y, a su vez, determinar si el mismo es valido o no. Finalmente se deduce que dicha MT parará, validando el input. Por lo tanto, Σ* es un lenguaje recursivo. La construcción de una MT similar a la anterior permitiría a su vez validar cualquier input perteneciente al $\emptyset$. Luego, el mismo será recursivo.

11. Si $L \subseteq \Sigma^{¿}$, ¿se cumple que $L \in R$?

No necesariamente. Dado que Σ* el conjunto de todas las palabras formadas por símbolos de un conjunto finito de caracteres, Σ, el mismo contiene no solo lenguajes pertenecientes a R, sino tambien a aquellos pertenecientes a RE-R, etc., por lo que no es verdadero que si L esta incluido en Σ*, entonces L pertenece a R.

Un ejemplo de esto es el lenguaje HP. Sabemos que Σ* pertenece a R y HP esta incluido en Σ* pero no pertenece a R.

12. Justificar por qué un lenguaje finito es recursivo.

Sea L un lenguaje finito, sabemos que L es un conjunto recursivamente enumerable dado que existe un MT que lo compute. Luego, se entiende que el lenguaje L pertenece

a RE lo cual nos dice que siempre existe una MT que al menos lo acepte. Por otro lado, sabemos que dado un input w ∈ L el mismo será finito y bastará con una cantidad finita de pasos dados por una MT para recorrerlo y, a su vez, determinar si el mismo es valido o no. Finalmente se deduce que dicha MT parará, validando el input. Por lo tanto, L es un lenguaje recursivo.

13. Justificar por qué si L1 ∈ CO-RE y L2 ∈ CO-RE, entonces (L1 ⋂ L2) ∈ CO-RE.Se

Sea L1 ∈ CO-RE y L2 ∈ CO-RE, entonces sabemos que L_1^c, L_2^c ∈ RE. De esto sabemos que para ambos lenguajes existe una MT que los acepta. Luego, se puede construir una MT M que reconozca su union, es decir:

1. L_1^c, L_2^c ∈ RE
2. (L_1^c ⋃ L_2^c) ∈ RE, por propiedad de union de lenguajes
3. L_3 ∈ RE, L_3 = (L_1^c ⋃ L_2^c)
4. L_3^c ∈ CO-RE
5. (L_1^c ⋃ L_2^c)^c ∈ CO-RE
6. (L_1 ⋂ L_2) ∈ CO-RE, por ley de De Morgan
7. **(L_1 ⋂ L_2) ∈ CO-RE**

Ejercicio 2

Dado el alfabeto Σ = {a, b, c}.

1. **Obtener el lenguaje Σ* y el conjunto de partes del subconjunto de Σ* con cadenas de a lo sumo dos símbolos. ¿Cuál es el cardinal (o tamaño) de este último conjunto?**
   - Σ* = { λ , a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...}
   - p(Σ*) = {(λ , a), (λ , b), (λ , c), ...}, dado que solo tomo las cadenas de a lo sumo 2 símbolos, la cardinalidad de dicho subconjunto es 13. Luego, la cardinalidad del conjunto de partes es |p(Σ*)| = 2^{13}.

2. **Dado el lenguaje $L=\{a^n b^n c^n : n \geq 0\}$, obtener la intersección Σ* ⋂ L, la unión Σ* ⋃ L, el complemento de L respecto de Σ*, y la concatenación Σ.L.**
   - **Σ* ⋂ L**
     Por definición de Σ*, sabemos que L está incluido o es igual a Σ*. Luego, por definición de inclusión, todo w perteneciente a L pertenece a Σ*, y además, sabemos, por la definición de L, que existen palabras pertenecientes a Σ* que no pertenecen a L. Por lo tanto, Σ* ⋂ L = L.
   - **Σ* ⋃ L**

Por definición de Σ*, sabemos que L está incluido o es igual a Σ*. Luego, por definición de inclusión, todo w perteneciente a L pertenece a Σ*. Por lo tanto, Σ* ⋃ L = Σ*.

- **el complemento de L respecto de Σ***
  El conjunto resultante se define como $\{w \in \Sigma^{¿} : w \notin L\}$ que es igual a $\{w \in \Sigma^{¿} : w \neq a^n b^n c^n\}$.
- **la concatenación Σ.L**
  $\Sigma . L = \{w_1 w_2 : w_1 \in \Sigma \wedge w_2 \in L\}$

Ejercicio 3

Construir una MT (puede tener varias cintas) que acepte de la manera más eficiente posible el lenguaje L = {$a^n b^n c^n$ | n ≥ 0}. Plantear primero la idea general.

- Idea general

  Por cada símbolo *a* que lee, la MT M lo reemplaza con un blanco en la primer cinta y lo copia en la segunda cinta. Luego, por cada símbolo *b* que lee, lo reemplaza por un blanco y lo copia en la tercer cinta. Finalment, por cada símbolo *c* que lee, la saltea hasta llegar a un blanco. Cada uno de los pasos mencionados se realiza sin modificar aquellas cintas que no tengan participación. Finalmente, se recorre a la izquierda en las 3 cintas mientras las tuplas leidas sean de la forma (c, b, a), aceptando cuando la misma llegue a tres blancos. Si al final del proceso no quedan símbolos por reemplazar, la MT M se detiene en un estado de F, porque significa que la entrada tiene la forma $a^n b^n c^n$, con n ≥ 0. En caso contrario, M se detiene en un estado de (Q-F).

- Construcción de la MT M

  La MT $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ es tal que:

  - Q = {q_a, q_c, q_r, q_F}. El estado q_a es el estado en que M verifica el input antes de comenzar el proceso de copiado en las demas cintas. El estado q_c es el estado en el que se ejecuta el proceso de copiado de simbolos en cada cinta recorriendo el input hacia la derecha. El estado q_r es el estado en el que M recorre las cintas hacia la izquierda en las 3 cintas validando las tuplas. El estado q_F es el estado final.
  - $\Sigma = \{a, b, c\}$
  - $\Gamma = \{a, b, B\}$
  - $q_0 = q_a$
  - F = {q_F}

- La función de transición δ se define de la siguiente manera:

1. $\delta(q_a, B, B, B) = (q_F, (B, S), (B, S), (B, S))$
2. $\delta(q_a, a, B, B) = (q_c, (B, R), (a, R), (B, S))$
3. $\delta(q_c, a, B, B) = (q_c, (B, R), (a, R), (B, S))$
4. $\delta(q_c, b, B, B) = (q_c, (B, R), (B, S), (b, R))$
5. $\delta(q_c, c, B, B) = (q_c, (c, R), (B, S), (B, S))$
6. $\delta(q_c, B, B, B) = (q_r, (B, L), (B, L), (B, L))$
7. $\delta(q_r, c, a, b) = (q_r, (B, L), (B, L), (B, L))$
8. $\delta(q_r, B, B, B) = (q_F, (B, S), (B, S), (B, S))$

Ejercicio 4

Explicar (informal pero claramente) cómo simular una MT por otra que en un paso no pueda simultáneamente modificar un símbolo y moverse.

Para simular este tipo de MT, lo que se puede hacer es descomponer aquellas funciones de transición para un estado en el que simultaneamente se modifique un símbolo y además se mueva, por dos definiciones de la misma en las cuales la primera modifique el simbolo y la segunda efectue unicamente el movimiento. Entonces, una función del estilo $\delta(q, a) = (q, B, R)$ quedaría de la forma $\delta(q, a) = (q_r, B, S)$ donde q_r es el estado en el que se mueve a la derecha y quedaría definido para el símbolo *a* de la siguiente forma $\delta(q_r, a) = (q, B, R)$. El mismo razonamiento puede ser aplicado para una MT de k cintas.

Ejercicio 5

Explicar (informal pero claramente) cómo simular una MT por otra que no tenga el movimiento S (es decir el no movimiento).

Para explicar la idea general de como simular una MT con otra, tomamos como ejemplo el caso de dos maquinas de una cinta con estados finales q_A y q_R. Cabe aclarar que esta idea puede ser aplicada a cualquier MT de *k* cintas. Supongamos una MT M_1 cuya función de transición sea de la forma $\delta: Q \times \Gamma \rightarrow (Q \cup \{q_A, q_R\}) \times \Gamma \times \{L, R, S\}$ y una MT M_2 cuya función de transición es de la forma $\phi: Q \times \Gamma \rightarrow (Q \cup \{q_A, q_R\}) \times \Gamma \times \{L, R, S\}$. Luego, una función de transición definida en M1 puede definirse para M2 de la siguiente forma:

- Para aquellas funciones cuyo movimiento pertenezca a {L, R}, la función será igual en ambas maquinas, dado que no presenta problema alguno.
- Para aquellas funciones cuyo movimiento del cabezal sea S (es decir el no movimiento), se deberá cambiar dicha función por dos en las que primero se mueva

a la izquierda y luego a la derecha, y vicebersa. Podemos observar esto con un ejemplo simple:

Supongamos una cinta cuyo estado actual es la palabra: “aaabbb”, y la maquina con el cabezal en el principio del input. Luego, la función $\delta(q,a)=(q,B,S)$ sería equivalente a $\phi(q,a)=(q_s,B,R)$, $\phi(q_s,a)=(q,a,L)$, $\phi(q_s,b)=(q,b,L)$.

Ejercicio 6

Explicar (informal pero claramente) cómo simular una MT por otra que no tenga el movimiento L sino el movimiento JUMP, cuyo efecto es posicionar el cabezal en el símbolo de más a la izquierda.

Para simular una maquina de este tipo, planteamos una MT de k cintas. La idea general es disponer en principio de una cinta reservada para el input y otra destinada a ser utilizada como contador. En cuanto al procesamiento de la MT, no tenemos ningun problema respecto de aquellos estados en los que la función de transición mueve el cabezal a la derecha o no movimiento. Respecto del caso contrario, es decir, moviendose con el movimiento JUMP, el contador de la segunda cinta permitiría saber cuantos movimientos hacia la derecha deben hacerse para simular un movimiento a la izquierda luego de un JUMP.

Ejercicio 7

Supongamos que tenemos la fórmula $p(x)=(x_1\wedge x_2)\vee x_3$ tal que no p(x) pertenece a USAT.

La ejecución en cada paso se daría de esta manera:

1. La formula es *correcta sintácticamente*, asi que la máquina *no rechaza*.
2. Se generan no deterministicamente una asignacion de valores de verdad, A. Supongamos que al generar el valor de verdad genera $(F\wedge F)\vee V$, esto satisface la formula asi que la máquina no rechaza.
3. Se generan no deterministicamente una asignacion de valores de verdad de A´ diferente de A. Supongammos que al generar el valor de verdad genera $(F\wedge F)\vee F$. A´ no satisface la formula, por lo cual la maquina acepta.

En las MTN se acepta una entrada w si a partir de w al menos una computacion es aceptada. Por lo tanto la maquina aceptaria un $(x_1\wedge x_2)\vee x_3$, que no pertenece a USAT.