IIC2213 — Lógica para Ciencia de la Computación

Notas Semana 4

1. No determinismo

1.1. Maquinas de Turing No-deterministas

Recuerda que usamos la notación 2^S para denotar a todos los subconjuntos de S (el conjunto potencia de S).

Definición. Una máquina de turing no-determinista es una tupla $M = (Q, \mathbf{A}, B, q_0, F, \delta)$, donde

- Q es un conjunto de estados
- $\mathbf{A}$ es un alfabeto
- $q_0 \in Q$ es un estado inicial
- $F \subseteq Q$ son los estados finales
- δ es la función de transición, una función parcial definida como

$$\delta : Q \times (\mathbf{A} \cup B) \to 2^{Q \times (\mathbf{A} \cup B) \times \{\leftarrow, \rightarrow\}}$$

Notarás que la única diferencia es que la función δ ahora asigna, a cada par (estado, símbolo), un conjunto de triples en $Q \times (\mathbf{A} \cup B) \times \{\leftarrow, \rightarrow\}$. Cada uno de estos triples representa una posible transición de la máquina, y la idea es que podemos elegir qué transición usar.

Las máquinas no deterministas funcionan casi igual que las deterministas:

- Comenzamos con la cinta de trabajo con la palabra input w escrita y el resto de las celdas vacías. La cabeza lectora apuntando al primer símbolo de la palabra w y la máquina en un estado inicial q_0
- En cada paso, la máquina observa el símbolo s que está leyendo la cabeza, y el estado q de la máquina, y determina el conjunto $\delta(q, s)$ de posibles movimientos.
- Si $\delta(q, s)$ es vacío, la máquina se detiene y termina. De lo contrario, la máquina elige uno de los triples en $\delta(q, s)$ y de acuerdo a eso escribe un símbolo, pasa a otro estado y se mueve a la derecha o a la izquierda.

- De acuerdo al símbolo s leído y al estado q actual, el conjunto de instrucciones ahora ofrece una o más posibilidades para hacer. *La máquina elige uno de los triples en $\delta(q, s)$, y de acuerdo a eso se mueve*
- Se dice que la máquina *acepta* a la palabra w si el estado en el que se detuvo la máquina es uno de los estados designados como *finales*.

El lenguaje de las máquinas no deterministas se define ahora un poco distinto: la idea es que nos basta con que uno de los posibles caminos que podríamos haber tomado se detenga en un estado final. Más específicamente, definimos el lenguaje $L(M)$ de una máquina no-determinista M sobre alfabeto $\mathbf{A}$ como

$$L(M) = \{w \in \mathbf{A}^* \mid \text{ existe una ejecución de la máquina } M \text{ con input } w \text{ que se detiene en un estado final.}\}$$

Vamos a formalizar esta noción un poco más abajo. Por mientras, un ejercicio.

Ejercicio. La siguiente es una máquina no-determinista que acepta todas las palabras binarias cuyo largo es divisible por 2 o que solo tienen símbolos 1. Definimos $M = (Q, \mathbf{A}, q_0, F, \delta)$, con $Q = \{q_0, q^1, q^s, q_p, q_i\}$, $\mathbf{A} = \{0, 1\}$, $F = \{q_p, q^1\}$ y donde la función de transición se define como:

$$\begin{array}{lcl} \delta(q_0, B) & \to & (q_p, B, \to) \\ \delta(q_0, 0) & \to & (q_i, 0, \to) \\ \delta(q_0, 1) & \to & \{(q_i, 1, \to), (q^1, 1, \to)\} \end{array} \qquad \begin{array}{lcl} \delta(q_i, 0) & \to & (q_p, 0, \to) \\ \delta(q_i, 1) & \to & (q_p, 1, \to) \end{array}$$

$$\begin{array}{lcl} \delta(q_p, 0) & \to & (q_i, 0, \to) \\ \delta(q_p, 1) & \to & (q_i, 1, \to) \end{array} \qquad \begin{array}{lcl} \delta(q^1, 1) & \to & (q^1, 1, \to) \\ \delta(q^1, 0) & \to & (q^s, 0, \to) \end{array}$$

La idea es que el estado q_0 es un estado donde puedo adivinar si la palabra va a ser de largo par, o si va a ser una palabra compuesta solo por símbolos 1. Los estados q_p, q_i simulan una máquina que cuenta el largo de las palabras, alternando entre q_p (por *par*, y estado final) y q_i (por *impar*, y estado no final). Los estados q^1 y q^s simulan una máquina que se mantiene en q^1 hasta que ve un 0, y en ese entonces pasa a un estado no final q^s que no transiciones.

Entonces. Si leo un blanco en el estado inicial q_0, la palabra es de largo par, y por eso avanzo inmediatamente a q_p. Si leo un 0, entonces la palabra debe ser de largo par para aceptar (por que ya vi un 0), y por tanto avanzo a q_p. Pero si leo un 1, tengo dos opciones: me paso a q_p si creo que la palabra va a ser de largo par, o me paso a q^1 si creo que la palabra solo va a tener 1s.

Para este ejercicio, comprueba cómo opera la máquina, paso por paso, con inputs $w_1 = 0010$, $w_2 = 111$ y $w_3 = 0$. Cuantas ejecuciones hay para cada una de estas palabras? Recuerda que la máquina acepta si alguna de las ejecuciones posibles termina en un estado final.

2. Semántica formal: Configuraciones, pasos y ejecuciones

Veamos ahora la definición formal de la semántica de una máquina de turing. Mientras lees, nota como todas estas definiciones son casi las mismas que para el caso de las máquinas deterministas (de hecho, una máquina determinista es un caso especial de las máquinas no deterministas, en donde cada transición en δ tiene una sola posibilidad). Sólo cambia el hecho de que ahora puede haber más de una ejecución por cada palabra.

Definición[configuraciones - igual que antes]. Sea $M = (Q, \mathbf{A}, B, q_0, F, \delta)$ una máquina de Turing. Una *configuración* de M es una tupla $c = (q, i, t)$, en donde $q \in Q$, i es un número entero, y $t : \mathbb{Z} \to (\mathbf{A} \cup \{B\})$ es una función que asigna a cada número entero un caracter de $\mathbf{A} \cup \{B\}$. Nota que una configuración $c = (q, i, t)$ representan la foto de la ejecución de una máquina en un punto en particular: q es el estado actual de la máquina, i la posición donde está la cinta lectora, y t representa la cinta como un arreglo infinito en ambas direcciones. Dada una palabra $w = a_0 \cdots a_{n-1}$, la *configuración inicial* de M con w es la tupla $(q_0, 0, t_{\mathrm{w}})$, con t_{w} la función definida como

$$t_{\mathrm{w}}(j) = \begin{cases} a_j, & \text{si } 0 \leq j \leq n-1 \\ B & \text{en otro caso.} \end{cases}$$

De la misma forma, una *configuración final* de M es cualquier configuración $c = (q, i, t)$ para la cual $\delta(q, t(i))$ no está definida, y además q pertenece a F.

Definición[pasos, ejecuciones - casi igual, ¿puedes ver la diferencia?]. Decimos que podemos ir de una configuración $c = (q, i, t)$ a otra $c' = (q', i', t')$ en *un paso* si cualquiera de las siguientes opciones es verdadera:

1. Se tiene que $i' = i - 1$, t y t' solo difieren en $t(i)$[1], y la tupla $(q', t'(i), \leftarrow)$ está en $\delta(q, t(i))$.

2. Se tiene que $i' = i+1$, t y t' solo difieren en $t(i)$, y la tupla $(q', t'(i), \rightarrow)$ está en $\delta(q, t(i))$.

Una *ejecución* de M con una palabra w es una secuencia de configuraciones $c_0, \ldots, c_n$ en donde c_0 es la configuración inicial de M con w y para $0 \leq j \leq n-1$ podemos ir desde c_j a c_{j+1} en un paso. Si además c_n es una configuración final, entonces la ejecución es una *ejecución válida*.

Semántica de una máquina de turing. Ahora si, considerando que pueden haber múltiples ejecuciones para una palabra, la definición del lenguaje de una máquina debe cambiar. Sea M una maquina de turing no-determinista sobre un alfabeto $\mathbf{A}$. Luego, definimos que M acepta a una palabra w si y solo si existe una ejecución válida de M con w. Finalmente, el lenguaje $L(M)$ de una máquina no-determinista se define como el conjunto de todas las palabras sobre $\mathbf{A}$ que son aceptadas por M.

[1]Es decir, $t'(j) = t(j)$ para $j \neq i$

3. No-determinismo y poder expresivo

La meta de esta sección es probar lo siguiente:

Teorema. Para cada maquina de Turing no-determinista N, existe una máquina de Turing determinista M tal que $L(N) = L(M)$

Vamos a empezar por un ejercicio simple.

Ejercicio. Escribe una máquina no-determinista sin estados finales que con input ε tenga 2^n ejecuciones distintas y al cabo de cada ejecución la cinta termine con un string binario distinto.

3.1. No determinismo y árbol de ejecuciones

Para las máquinas deterministas siempre existe una sola configuración a la que puedo pasar desde la configuración inicial, y así sucesivamente: siempre hay a lo más una configuración a la que llego desde otra configuración en un paso.

En el caso de las máquinas no-deterministas, eso no es cierto: ahora puedo tener varias opciones por cada configuración.

Definición[árbol de ejecuciones]. El *árbol de ejecuciones* para una máquina M con una palabra w es un árbol cuyos nodos son configuraciones de M. La raíz del árbol es la configuración inicial de M con w, y los hijos de cada nodo c son todas las configuraciones c' a las cuales puedo llegar de un paso desde c.

Observación. M acepta a una palabra w si y solo si el árbol de ejecuciones de M con w tiene al menos una hoja que es una configuración final. Cada camino del árbol de configuraciones representa una ejecución; los caminos a hojas cuyos estados son finales representan ejecuciones válidas.

A continuación vamos a demostrar el teorema de que las máquinas no-deterministas pueden ser simuladas con una máquina determinista. Eso sale directo del siguiente resultado:

Teorema[máquina universal determinista]. Dada una máquina M, es posible construir una máquina de Turing determinista U_M que con input una palabra w, simula una búsqueda por amplitud del árbol de ejecuciones de M con w.

(Idea de la demostración) Para construir la máquina U_M, la idea es ir simulando primero todas las configuraciones a las que puedo alcanzar con un paso desde la configuración inicial c_0. Sabemos exactamente cuantos hijos tiene c_0, pues depende de la función de transición de M. Luego, se puede simular el primer hijo, ver si es final. Borrar todo, simular el segundo hijo, y así sucesivamente hasta que ya hemos simulado todas las opciones posibles. Luego de eso, vamos a simular todas las configuraciones que puedo alcanzar con dos pasos. Borramos todo, simulamos desde cero todas los nodos alcanzables en el árbol de largo 2: vamos al primer hijo del primer hijo de c_0. luego al segundo hijo del primer hijo de c_0, y así sucesivamente.