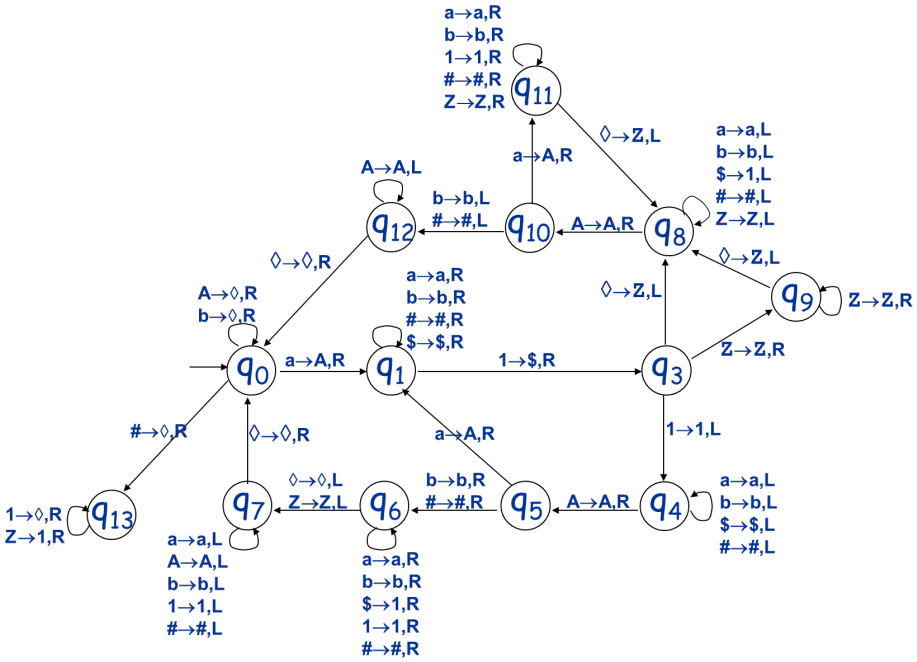
Tarea #4 - Parte 1 (Preguntas 1, 2 y 3) - Resolución

1. **[25 pt]** Construya una máquina de Turing **estándar** que, dado un input de la forma $w\#n$, donde $w \in \{a,b\}^*$ y $n \geq 1$ está escrito en unario, calcule $f(w,n) = |w|_{a^n}$ (es decir, la cantidad de veces que w contiene n a's consecutivas. Por ejemplo, $f(abba,1) = 2$, $f(abba,2) = 0$, $f(abbaab,2) = 1$, $f(abaaaa,3) = 2$. Por lo tanto, en la cinta para esos inputs el resultado final tendría que ser 11, ε (o sea, sólo $\Diamond$'s), 1 y 11 respectivamente.

   R:

   Hay varias formas de plantearse una estrategia para esto (y seguramente varias de ellas aparecerán al corregir). Uno podría ir viendo los grupos de a, y partir a buscar los 1 para comparar. O ir viendo los 1 y buscando las primeras a. Y si el grupo de a es más grande que n, uno podría altiro agregar el excedente a la respuesta (o esperar a contar el nuevo grupo tras avanzar en el traslape). Y la respuesta uno la puede ir guardando hacia la izquierda o hacia la derecha... O marcar primero las posiciones en que parten n a's, y contarlas al final. Por mencionar algunas alternativas solamente. Yo me la jugué por esta:



   Es de 13 estados (del 0 al 13, pero no hay q_2 porque recién vi que sobraba y me dió flojera cambiar todos los números). La respuesta se va escribiendo al final del área de trabajo, con letras Z; además, cada vez que se detecta un bloque de n a's, se aprovecha altiro de agregar más Z por cada a extra que venga después del bloque. Y el input se va borrando a medida que se procesa. Veamos como opera. En q_0 borramos posibles b iniciales (que no interesan) y posibles a ya procesadas (que serán las A). Si encontramos una a, puede ser el comienzo de un bloque útil, así que nos vamos a q_1, avanzamos hasta donde está n, y le marcamos el primer 1... Y espiamos lo que sigue.

   Si hay más 1, significa que nuestras A aún no son un bloque completo, así que nos vamos a q_4 a rebobinar hasta donde estaba la A que pusimos, y miramos a su derecha. Si hay otra a, alargamos el bloque e iteramos volviendo a q_1. Si no, significa que el bloque estaba incompleto (eran menos de n letras a) así que no cuenta; nos vamos al extremo derecho para pasar a

desmarcar los 1 (pasando los \$ a 1) y nos devolvemos al extremo izquierdo, volviendo a q_0 para borrar esas A fallidas y seguir avanzando.

Si en q_3 notamos que *no* hay más 1, significa que nuestras A son un bloque completo, así que agregamos una Z al final. Luego nos devolvemos, en q_8, aprovechando de desmarcar los 1, hasta llegar al final del bloque de A. Si a su derecha hay otra a, nos vamos al final en q_{11} a agregar otra Z, e iteramos (pues esa a indicaría que hay otro bloque de n a's, traslapado con el anterior, y así sucesivamente por cada a extra que pillemos ahí). Si no hay más a, nos vamos en q_{12} al extremo izquierdo y volvemos a iniciar (lo cual borrará todo este bloque de A ya procesadas).

Finalmente, cuando al avanzar nos topamos con el $\#$, lo borramos y pasamos al estado q_{13}, que borra también los 1, convierte las Z en 1, y hará que la máquina se detenga cuando llegue al extremo derecho, dejando la respuesta y nada más en la cinta.

Ok, admito que era más engorroso el ejercicio de lo que pensé, pero 13 estados no es taaaanto tampoco.

2. **[25 pt]** Dada una palabra no nula $w = w_1 w_2 \dots w_n$, definiremos su ensanchamiento $e(w)$ como $e(w) = w_1 w_1 w_2 w_2 \dots w_n w_n$. O sea: cada letra se duplica, de modo que $e(a) = aa$, $e(aca) = aaccaa$, $e(hola) = hhoollaa$, etcétera. Construya una máquina que decida el lenguaje

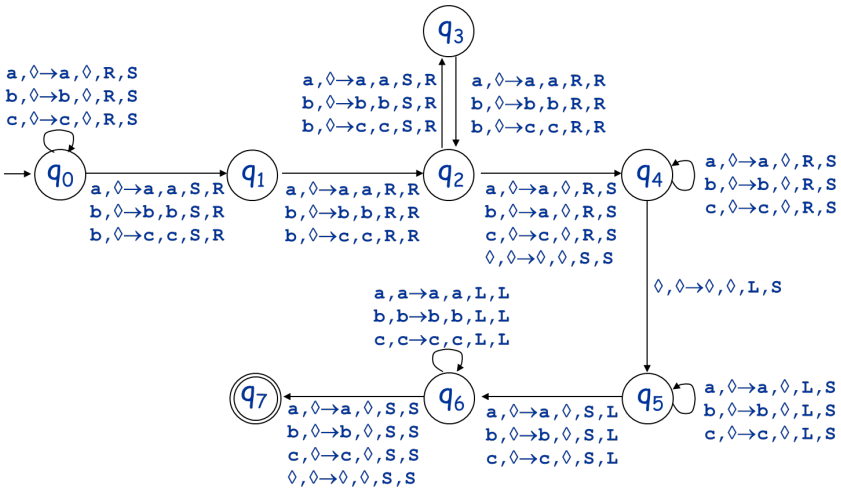
$$\{w \in \{a,b,c\}^* : \exists u \in \{a,b,c\}^+ \text{ tal que } u \prec w, e(u) \prec w\}$$

donde el símbolo $\prec$ denota la relación de subpalabra (no necesariamente propia: $e(u) = w$ estaría ok). Así que el lenguaje consiste en palabras que incluyen en alguna parte a una palabra y además a su versión ensanchada. Por ejemplo, aca no estaría, pero $acbaccbba$ sí (pues contiene cb y también $e(cb) = ccbb$). Ojo que puede estar traslapadas: $aabbb$ estaría en el lenguaje, pues contiene a ab y también $e(ab) = aabb$.

Esta no necesita ser estándar; pueden usar cualquiera de las extensiones que vimos en clases.

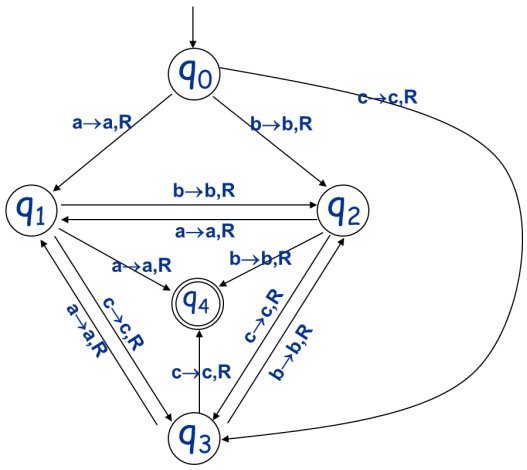
R:

Acá pasó al revés: el ejercicio es más *fácil* de lo que pensé inicialmente. Yo pensaba en una solución del tipo

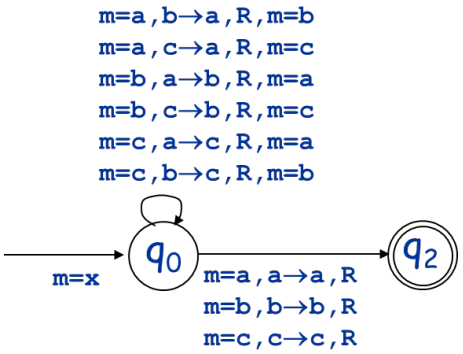


que usa dos cintas, no-determinismo y stay, pero que dentro de todo no es muy grande ni muy compleja. En q_0 saltamos parte del input, luego copiamos una parte hacia la segunda cinta (ensanchándola de paso), luego nos vamos al final, retrocedemos una parte del input, y comprobamos (en q_6) que esa palabra ensanchada que pusimos en la segunda cinta efectivamente está presente en el input.

Pero: como varios me hicieron notar (muchos en realidad, así que supongo que se popularizó el dato), si un input cumple la propiedad para alguna palabra w cualquiera, entonces también la cumple si tomamos simplemente su primera letra, en lugar de tomar todo w. Y viceversa: si se cumple para una palabra de una letra, entonces se cumple! Así que todo lo que hay que chequear es acaso el input contiene aa, bb o cc... ¡lo cual podría hacerse con un autómata finito! Pero queríamos una máquina de Turing. Una máquina *estándar* podría ser



pero ya que están permitidas las extensiones, puede incluso ser



que utiliza una variable interna m con valores $\{a, b, c, x\}$ para ir recordando la última letra leída y así compararla con la siguiente.

3. **[27 = 3×9 pt]** ¿Decidible o no? Para cada problema, determine acaso es decidible, y demuestre que su respuesta es correcta.

   (a) Diremos que una máquina de Turing estándar es *exigente* si, para cualquier palabra que acepta, no acepta ninguna de sus subpalabras. El problema: dada una MT, determinar acaso es exigente.

   (b) Dados una máquina de Turing M y un símbolo cualquiera $\alpha \neq \Diamond$, determinar acaso al ejecutar la máquina (partiendo con cinta vacía) en algún momento la máquina escribirá α en la cinta.

   (c) Dados una máquina de Turing M, determinar acaso al ejecutar la máquina (partiendo con cinta vacía) en algún momento se escribirá en la cinta algún símbolo $\alpha \neq \Diamond$.

   R:

(a) INDECIDIBLE. Es una propiedad del lenguaje (en el fondo, se trata de determinar acaso el lenguaje de la MT es exigente), así que aplicamos Rice. No hace falta construir MT para el "sí" y el "no" (aunque no cuesta mucho tampoco). Basta evocar los ejemplos que vimos de lenguajes regulares exigentes y no exigentes. Como regular es caso particular de decidible, existen máquinas para esos lenguajes.

(b) INDECIDIBLE. Notemos primero que, dada una máquina, siempre podemos modificarla para que opere igual pero *no* escriba jamás α en la cinta: es cosa de reemplazar en sus flechas todos los α que escriba por, digamos, α', y para todos los α que lea agregar una flecha entre los mismos estados que lea α' en lugar de α.

Hecho eso, podemos resolver el alto (versión sin input, o sea, partiendo con cinta vacía). Dada una máquina para la cual quiero resolver el alto, la modifico como dijimos arriba, y después todas las transiciones indefinidas (las que la habrían hecho detenerse) las defino como transiciones que escriben α (da lo mismo a que estado apunten; podría ser q_0, podrían ser loops). Luego aplicamos a esta máquina modificada el supuesto algoritmo que resuelve nuestro problema, determinando acaso α se escribe, y la respuesta nos estará dando la respuesta al problema del alto de la máquina original.

(c) DECIDIBLE. El algoritmo para contestar consiste en: calcular la cantidad de estados de la máquina ($|Q|$) y ejecutarla durante $|Q|+1$ pasos. Si la vimos escribir algo en la cinta, contestamos que sí. Si no, contestamos que no, pues la máquina habrá entrado en un loop así que jamás escribirá nada.

¿Por qué bastan $|Q|+1$ pasos? Pues porque, mientras no se escriba en la cinta, lo único que distingue a una configuración de otra es el estado de la máquina (y la posición del cabezal, pero como la cinta esta vacía esa posición da lo mismo). No podrá dar más de $|Q|+1$ pasos sin caer en un ciclo.

4. PREGUNTA 4: Pendiente (hasta después de que la entreguen!).