Tarea de Caml: Autómatas y otros animales

PROF. KIRSTEIN GÄTJENS S.

"Somos autómatas controlados totalmente por las fuerzas del medio, zarandeados como corchos en la superficie del agua, pero confundimos el resultado de los impulsos del exterior con el libre albedrío."
– Nikola Tesla

Instrucciones generales:

- Entregue un archivo de texto con el fuente del programa solicitado. Use extensión .ml.
- Debe entregarse antes del miércoles 3 de abril a la medianoche al correo kirstein.evaluaciones@gmail.com
- El nombre del .ml debe ser consistente con los estándares de clases:
  Automatas-Apellido-Nombre.ml
- El subject del correo es lo usual: TAREA: Autómatas en Caml
- El cuerpo del correo debe contener, como todo correo del curso, su nombre completo, número de carné y curso.
- La documentación a entregar debe ser en el mismo fuente entre comentarios. Tiene 3 partes mínimas, pero si requiere de más explicaciones son bienvenidas. Usualmente ayudan al asistente a poner una nota mejor.
- Debe tener una pequeña portada debe estar al inicio del mismo fuente entre comentarios. Un manual de usuario que explique como se puede probar su tarea con detalle (cuáles son las funciones de entrada y parámetros) y un análisis de resultados con el sistema de ABCDE usado en el curso.
- Al ser una tarea de un lenguaje funcional es indispensable que el código de cada función posea oportunos comentarios. Precondiciones o poscondiciones necesarias, parámetros y breve descripción.
- Esta tarea vale por 12 resúmenes ya que es más pesada de pensar, no tanto de escribir código. Por esta misma razón lo de las dos semanas y media de tiempo.
- Deben trabajar en Caml Light o en su defecto en OCaml.
- Al ser una tarea del paradigma funcional es absolutamente restringido el uso de elementos imperativos, tales como variables, iteraciones o secuencias. Salvo previo permiso del profesor.

La tarea consiste en realizar una serie de funciones que permitan manipular diferentes tipos de autómatas: Los determinísticos, los no determinísticos, los de dos vías y los con salida (máquinas de mealy y máquinas de moore).

Para cada uno de ellos debe ser posible ejecutarlo con un string particular. Es decir debe haber una función llamada ExecXXX donde XXX son las siglas del autómata que debe recibir un autómata del tipo correspondiente y un string y debe ejecutar ese autómata con el string correspondiente. La función debe retornar un valor booleano en el caso de las máquinas reconocedoras y un string en el caso de las máquinas generadoras.

Las siglas son las que se suelen usar: AEFD, AEFND, A2V, MMe y MMo.

Debe haber una función que dado un autómata nos diga verdadero o falso si es determinístico (o si no lo es). Esta solo debe funcionar para los tipos compatibles con AEFD y AEFND.

También se deben programar los algoritmos de conversión AEFND➔AEFD, MMo➔MMe y MMe➔MMo

Las respuestas a estos tres ejercicios son la máquina correspondiente, pero se debe crear un archivo de texto con una bitácora que muestre los pasos realizados. El nombre del archivo se envía como parámetro.

Al ejecutar el A2V se debe tener cuidado de que si se llega a un ciclo se rechace la tira, no que la función se pegue.

Hay que analizar la definición de tipo de cada máquina. Pensé en que las funciones de transición fueran funciones, pero al pensar mejor la solución de la tarea lo dejé en tuplas ya que manejar funciones como datos (high order) le puede subir un margen la dificultad a la tarea y ya está interesante.

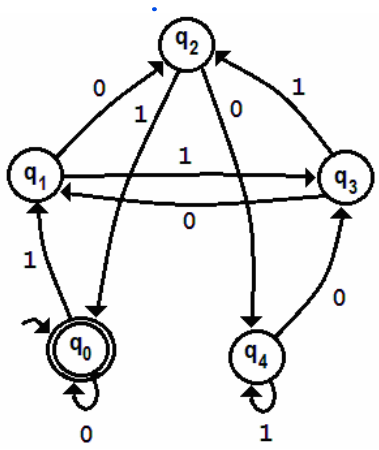
Para los AEFDs Una quíntupla con una lista de enteros (que representa los estados q0, q1, etc.). Un entero que es el estado inicial, una lista de enteros que son los estados finales, una lista de caracteres que es el alfabeto y una lista de tuplas de entero, caracter, entero que representa cada transición.

El tipo de un AEFD entonces debe ser:

```
int list * int * int list * char list * (int*char*int) list
```

Un ejemplo de un AEFD, el clásico de los números binarios divisibles por cinco de la imagen de la derecha se definiría como:

```
let AEFD1=([0;1;2;3;4],0,[0],[`0`;`1`],[(0,`0`,0);(0,`1`,1);
(1,`0`,2);(1,`1`,3);(2,`0`,4);(2,`1`,0);(3,`0`,1);(3,`1`,2);
(4,`0`,3);(4,`1`,4)]);;
```



Curiosamente con esta representación un AEFND tendría el mismo tipo que un autómata determinístico.

Un A2V solo cambiaría en la función de transición que sería una lista de tuplas de cuatro elementos para agregar la dirección como un char: `L` o `R`. Entonces quedaría el tipo de los A2V así:

```
int list * int * int list * char list * (int*char*int*char) list
```

En el caso de los autómatas con salida deben tener dos tipos diferentes ya que la función de salida es distinta entre ellos. Recuerden que no tienen estados finales al ser máquinas generadoras y que tienen dos alfabetos, el de entrada y el de salida.

Ejemplos de invocaciones a las funciones de la tarea:

```
# ExecAEFD AEFD1 "1010";;
- bool = true
# ExecAEFD AEFD1 "1111001";;
- bool = false
# Deterministicop AEFD1;;
- bool = true
```

¡Suerte!