# Teoría de Autómatas y Lenguajes Formales

## Práctica 4 - Máquinas de Turing

**Miguel Martín Fernández-Corugedo**
NIA: 100432072
100432072@alumnos.uc3m.es

**Iker Tejero Merino**
NIA: 100432078
100432078@alumnos.uc3m.es

**Grupo 80**
Curso 2020/2021

uc3m | Universidad Carlos III de Madrid

# Índice

1. **Diseñar y construir una máquina de Turing capaz de recibir movimientos rectilíneos (solo en direcciones norte y suroeste) y modificar los valores de los motores para una única iteración. (3p)**
    a. **Maquina1.jff**

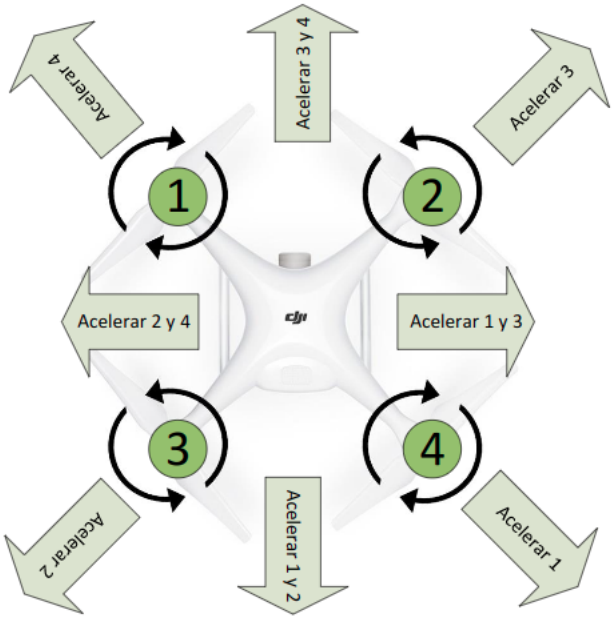
| □ | Maniobra | $ | 1 | 0 | 0 | $ | 1 | 0 | 0 | $ | 1 | 0 | 0 | $ | 1 | 0 | 0 | □ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Ver 2.1 | Sep | Motor 1 | | | Sep | Motor 2 | | | Sep | Motor 3 | | | Sep | Motor 4 | | | |

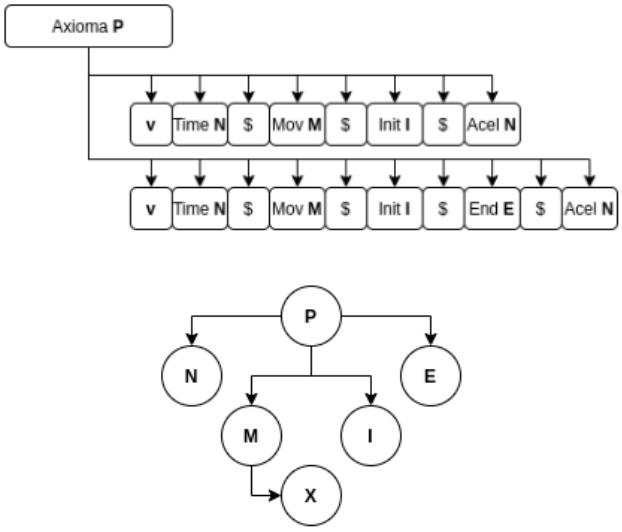
Variante 1:

| □ | 1 | 1 | 0 | $ | 0 | 0 | 1 | $ | n | e | $ | s | o | $ | 1 | 0 | 1 | □ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Tiempo | | | Sep | Movimiento | | | Sep | Dir ini | | Sep | Dir end | | Sep | Aceleración | | | |

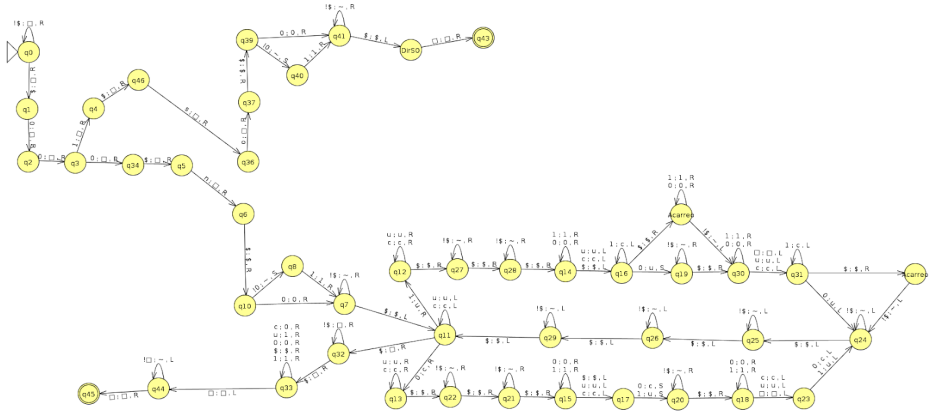
Variante 2:

| □ | 1 | 1 | 0 | $ | 0 | 0 | 1 | $ | n | e | $ | 1 | 0 | 1 | □ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Tiempo | | | Sep | Movimiento | | | Sep | Dir ini | | Sep | Aceleración | | | |

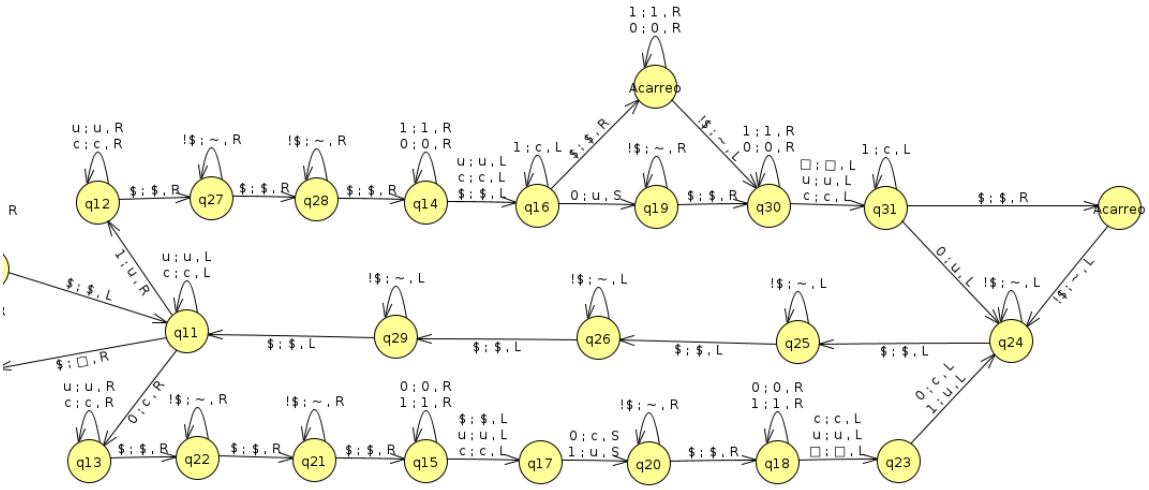


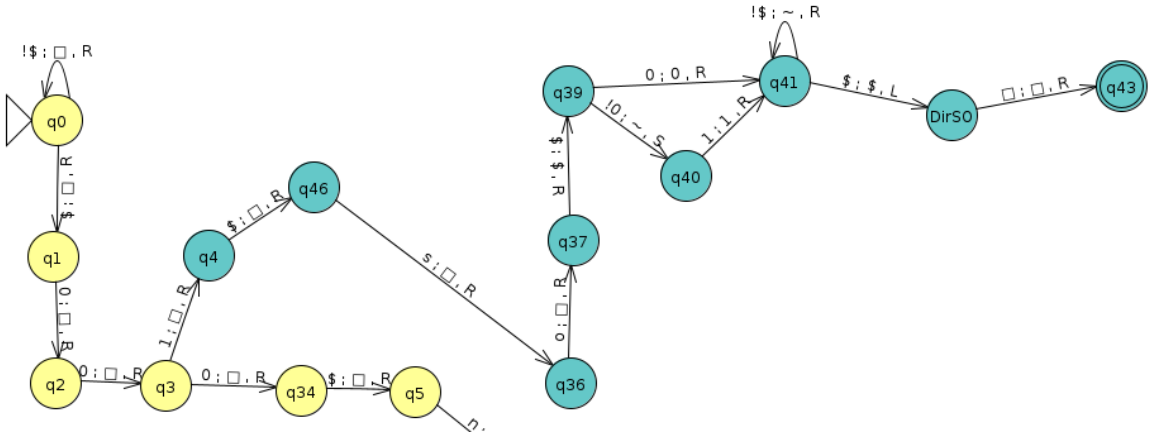

**El primer autómata** (precursor para hacer el segundo y que nos resulte más fácil) nos ha quedado de la siguiente manera:



En **q11**, podemos ver cómo comienza la lógica que da paso a la ejecución de la instrucción del vuelo del dron cuando la **dirección introducida es Norte**. Aquí vamos sumando bit por bit de la aceleración a los motores 3 y 4. Si el bit a sumar es un 1, nos vamos por la rama de **q12**, donde manejamos el acarreo (en caso de que haya). Si el bit de aceleración es un 0, nos vamos por la rama de **q13**, donde simplemente sumamos a los motores mencionados.
Una vez finalizada la suma, simplemente borramos los elementos de la cinta que no sean los motores para que saque el output correctamente.




De la misma manera, si la **dirección es Suroeste**, nos vamos por la rama de **q4** y seguimos una lógica prácticamente igual. La única diferencia es que el motor que hay que acelerar es el número 2, por lo tanto hemos hecho cambios en consecuencia.

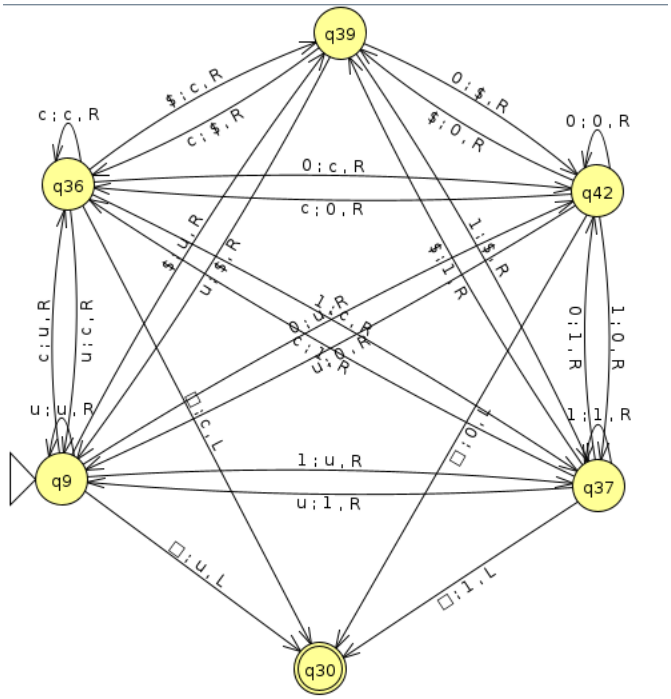



Y aquí acabaría el funcionamiento de esta máquina, ya que sólo hay que hacer una iteración.

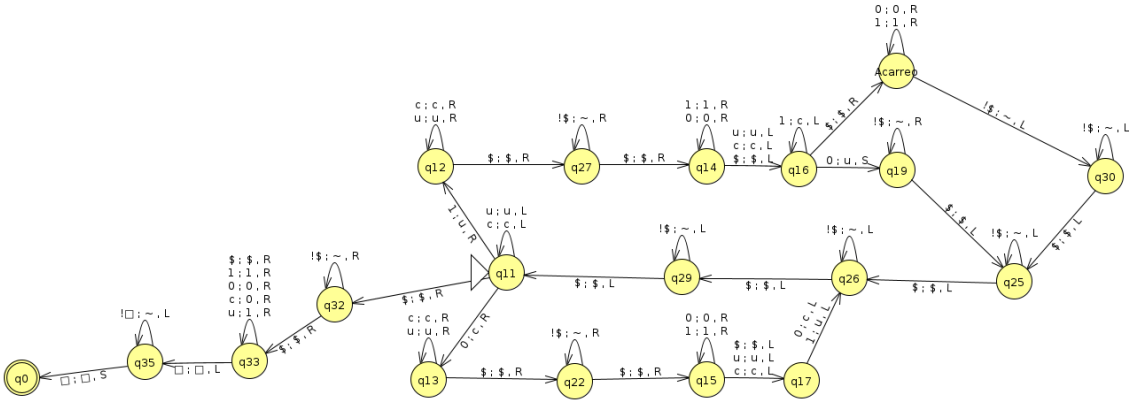
**Cabe destacar el uso de 2 *building blocks***. El primero llamado "**Acarreo**" que es el encargado de que al sumar la aceleración a los motores, no nos de ningún tipo de error por overflow. El segundo es "**DirSO**" que se encarga de acelerar los motores respectivos para que vaya en la dirección Suroeste.

El *building block* **Acarreo** nos ha quedado tal que así:

Lo que se hace aquí es entrar por **q9** e independientemente del elemento que reciba, **siempre el primero se va a convertir en un 1**. Por lo tanto, tenemos que "echar hacia la derecha" todos los demás elementos. Y eso es lo que se hace al entrar en cada uno de los estados de este Building Block.




Mientras que "**DirSO**" nos queda de la siguiente manera:

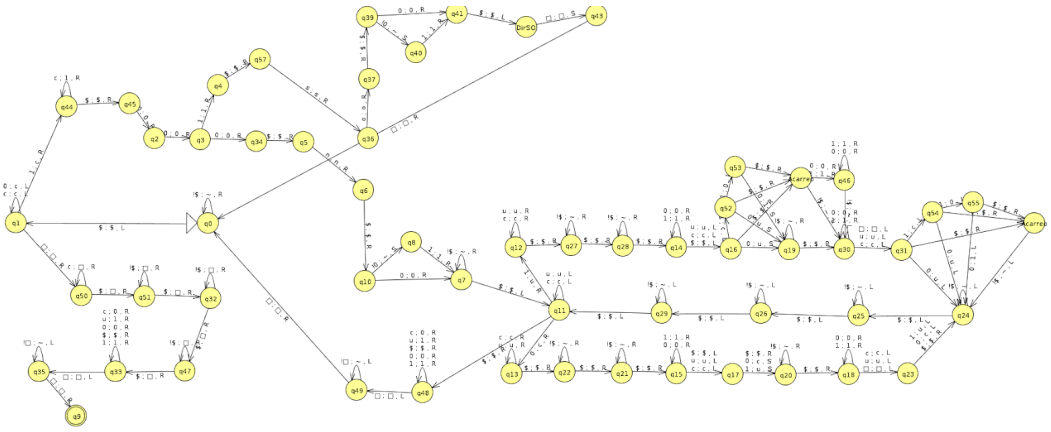



En este bloque, seguimos la **misma lógica** que la que se hace cuando el usuario introduce la dirección norte, con la diferencia de que en el caso de Suroeste tenemos que acelerar únicamente el segundo motor.

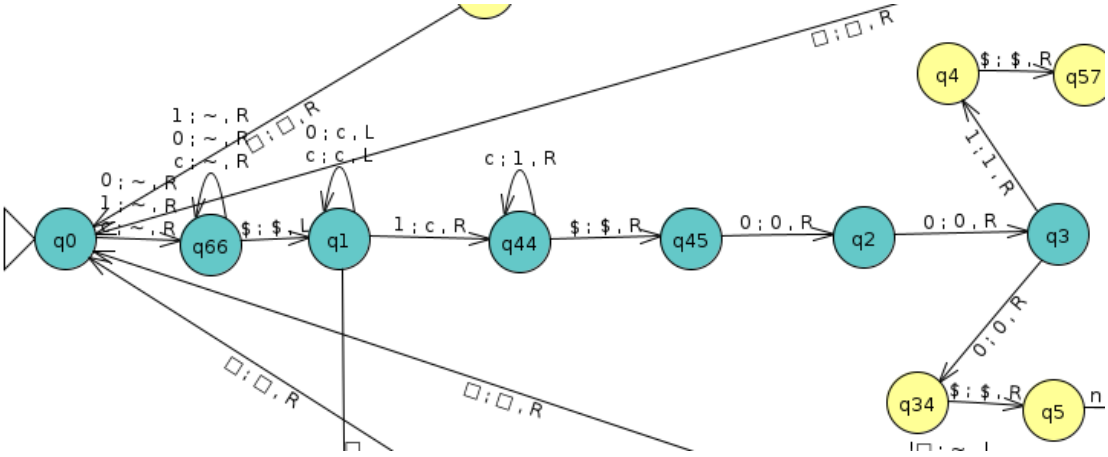
2. **Realizar y explicar la modificación sobre la máquina de Turing previa para hacerla funcionar en bucle (agregando la aceleración, tantas veces como instantes de tiempo se indiquen en el valor correspondiente) y completar la orden de vuelo. Si se desea se puede realizar este paso únicamente, explicando y entregando una única MT, aunque se recomienda hacerlo en dos pasos para simplificar el problema. (4p)**
    a. **Maquina2.jff**

Las modificaciones necesarias para hacer que este autómata funcionase se basan en añadir un **bucle que tuviese en cuenta el tiempo** que se introducía en la instrucción de vuelo. De manera que cada vez que se haga una iteración de suma de la aceleración, el tiempo **se reduce en 1 segundo**.

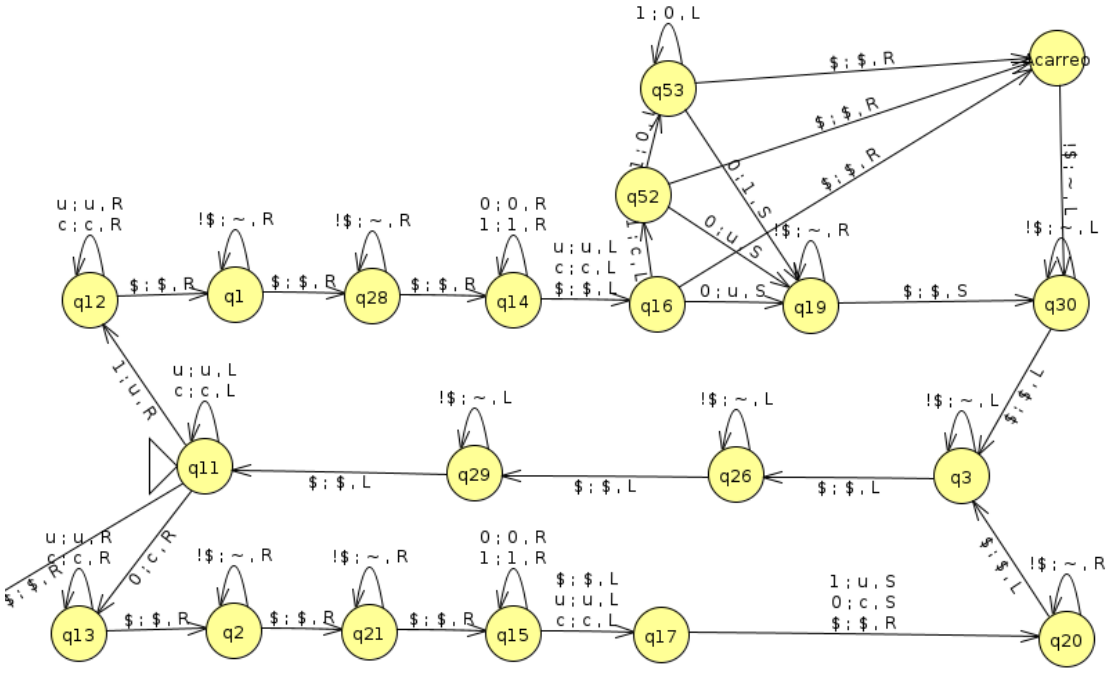
Una vez hemos sumado todos los bits de aceleración, **volvemos a cambiar las letras por los números** y procedemos a chequear el tiempo. Si el tiempo no está a cero, restamos un segundo y reiteramos de la misma manera (ya sea dirección suroeste o norte)



Concretamente, la parte en la que se van contando las iteraciones comienza en el estado q0 hasta q3, donde se lee los siguientes elementos de la cinta y se mira qué dirección ha introducido.
Aquí, nos encargamos de mirar si el tiempo ya está a 000 (o ccc). Si encontramos un 1 (o u), entonces le restamos para ponerlo a 0 y que en la siguiente iteración no lo leamos.



Por último, hemos añadido la **funcionalidad para ir marcha atrás** (dirección *re*) porque si no una de las variantes del movimiento de vuelo no tenía sentido. Para ello, hemos creado otros dos nuevos Building Blocks: **BBnoroeste** y **BBsur**, los cuales son **la marcha atrás de suroeste y de norte, respectivamente**.




Estos funcionan de la **misma manera que los anteriores**, exceptuando que suman el número de la aceleración a los motores correspondientes a la marcha atrás.

3. **Diseñar, describir y validar con JFLAP un conjunto de pruebas para verificar el correcto funcionamiento de la(s) MT construida(s). Es importante que las pruebas busquen las limitaciones de la máquina y no sean elementos introducidos al azar. Una prueba fallida es una prueba útil si se busca dicho resultado. Por ejemplo, ser capaces de localizar una entrada de la cinta incorrecta. (2p)**
    a. Pruebas.txt

    Aquí hacemos una prueba únicamente para la MT construida en el ejercicio 2 ya que en ella se incluye implícitamente a la del ejercicio 1.

| Input | Output Esperado | Output Obtenido |
|---|---|---|
| **010$000$n$010$100$100$100$100** | **Aceptado**: 100$100$1000$1000 | **Aceptado**: 100$100$1000$1000 |
| **010$001$so$010$100$100$100$100** | **Aceptado**: 100$1000$100$100 | **Aceptado**: 100$1000$100$100 |

| | | |
|---|---|---|
| **$010$001$so$010$100$100$100$100** | **Rechazado** | **Rechazado**: Debido a un signo distinto de 1 o 0 al principio. |
| **010$001$me$010$100$100$100$100** | **Rechazado** | **Rechazado**: Debido a una dirección no reconocida (*n*, *so* o *re*) |
| **010$001$n$010$100$100$100$100 $100** | **Rechazado** | **Rechazado**: Hay un motor de más en la entrada |
| **000$001$so$010$100$100$100$100** | **Aceptado**: 100$100$100$100 | **Aceptado**: 100$100$100$100 |
| **010$000$so$010$100$100$100$100** | **Rechazado** | **Rechazado**: El movimiento 000 nos indica dos motores, y para la dirección *so* sólo se usa 1. |
| **010$001$n$010$100$100$100$100** | **Rechazado** | **Rechazado**: El movimiento 001 nos indica dos motores, y para la dirección *n* se usan 2. |
| **010$001$so$111$100$111$100$100** | **Aceptado**: 100$10101$100$100 | **Aceptado**: 100$10101$100$100 |
| **010$001$so$111111$100$111111$100$100** | **Aceptado**: 100$1011101$100$100 | **Aceptado**: 100$1011101$100$100 |
| **001$001$so$re$010$100$100$100 $100** | **Aceptado**: 100$100$110$100 (con marcha atrás) | **Aceptado**: 100$100$100$100 |
| **001$001$re$so$010$100$100$100 $100** | **Rechazado** | **Rechazado**: No se puede empezar en la dirección marcha atrás |