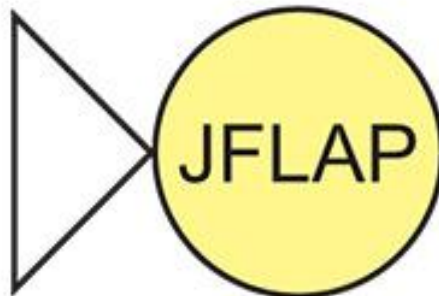


UC3M Campus de Colmenarejo



Teoría de Autómatas y Lenguajes Formales Práctica Final



Lucas González de Alba | 100383228 | 100383228@alumnos.uc3m.es

Víctor Alonso López | 100383276 | 100383276@alumnos.uc3m.es

Diciembre | 2018-2019

Escuela Politécnica Ingeniería Informática
UC3M, Colmenarejo.

Índice:

| | |
|--|---|
| Explicación del funcionamiento de la máquina final | 1 |
| Detalles sobre los Buildings Blocks utilizados | 2 |
| -Bloque cociente | 2 |
| -Restador | 3 |
| -Comparador Signo Resta | 4 |
| -Positivo | 4 |
| -Negativo | 4 |
| -Precisión | 4 |
| -Sumador | 2 |
| -Decrementador | 5 |
| -Batería de pruebas | 6 |

En primer lugar, vamos a comentar por encima el funcionamiento de la máquina de Turing final que hemos creado, y después pasaremos a describir con breves pinceladas cada una de las máquinas utilizadas para el correcto cálculo de las soluciones de una familia concreta de polinomios de orden 2.

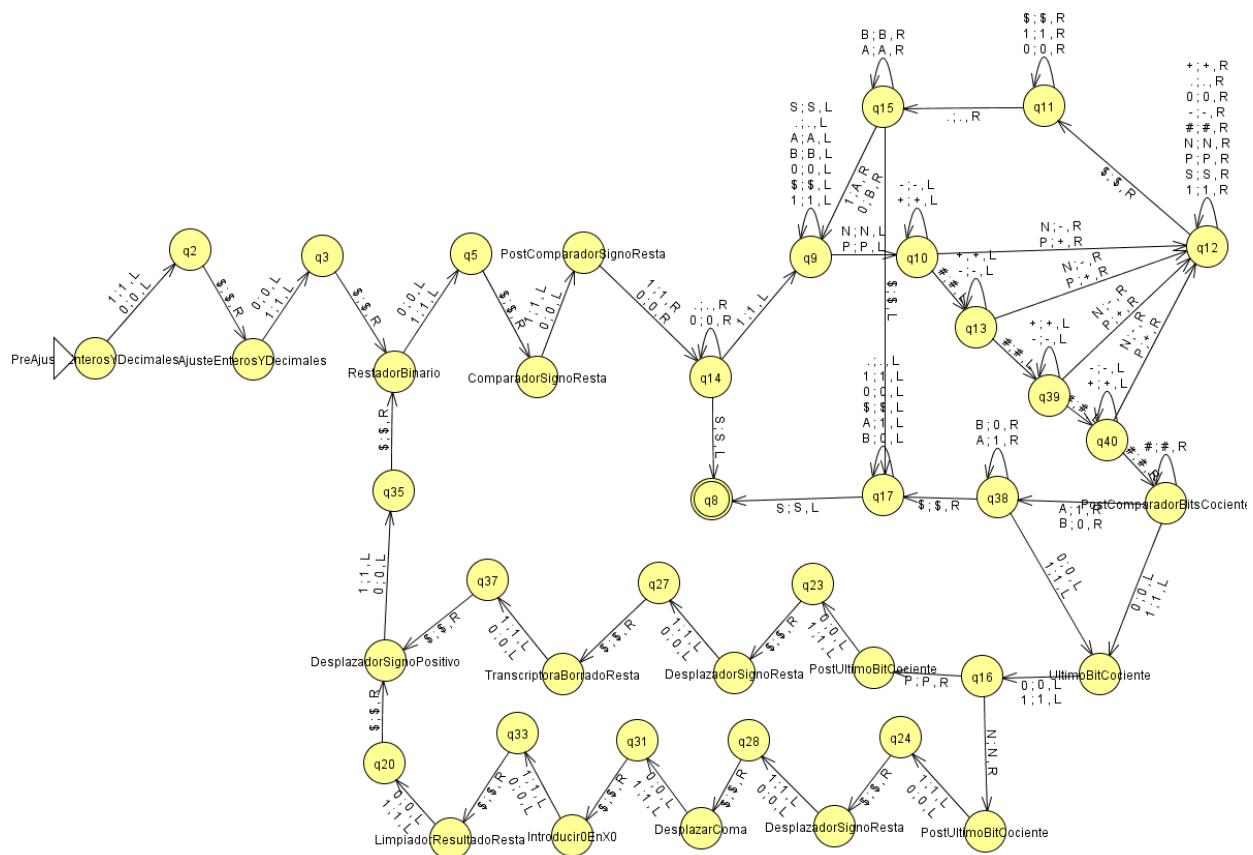
Explicación del funcionamiento de la máquina final

Empezaremos por el aparatado de la división. La primera acción que llevamos a cabo es la resta del numerador y denominador para extrapolarlo a la comprobación de si efectivamente puede realizarse la división. Dicha comprobación la basamos en que, si el número resultante de la resta es negativo, determinamos que el numerador no cabe en el denominador por lo que procedemos a aumentar el número de bits del numerador para volver a realizar la resta (haciendo los arreglos oportunos para que los dos números consten del mismo número de bits). Este paso se llevará a cabo de forma reiterada tantas veces como sea necesario para poder efectuar la división entre el numerador y denominador introducidos. Una vez comprobado que el valor resultante de la resta es positivo, tendremos luz verde para seguir con el siguiente paso, que sería sustituir el numerador por el número resultante de la resta anteriormente hecha. Con esto continuaríamos a seguir con la ejecución descrita en los pasos anteriores hasta que lleguemos a tener la misma precisión decimal descrita por el primer término de la sucesión. La clave en plantear el procedimiento de esta manera es que cada vez que, en la comprobación de la resta, sale que el número es negativo, se apuntara en el cociente un 0, mientras que, si la comprobación es positiva, añadiremos un 1. Esto hace que al final de las sucesivas comprobaciones a lo largo de la división, se nos haya ido formando una sucesión de 1s y 0s en el cociente, los cuales conforman la solución de la división pedida.

Una vez que ya tenemos determinado el funcionamiento del apartado $1/x_i$, procedemos a sumarlo con el respectivo valor de A. En nuestro razonamiento y debido a la mención en el enunciado de diversas propiedades, concretamente de esta, “otra propiedad que puede ser útil es que $1/x_i < 1, \forall i > 0, A > 0$ ”, decidimos orientar nuestro planteamiento de tal manera, en la que nuestra decisión de cálculo se basaba en determinar como solución, la asignación del número resultante de la división como parte decimal del valor A. En resumidas palabras sería algo tal que “ValorDeA, Valor $1/x_i$ “. Si por ejemplo el valor de la división fuera 0,25, solo debería asignarse su respectivo valor en binario en la parte decimal de A, que por poner un ejemplo sería el 32, resultando así la cifra 32,25, es decir, el 100000 como 32, y a la derecha de la coma, el 0100000 como 0.25, obteniendo el 100000.010000. Tras esta organización del planteamiento decidimos abordar esta estrategia en cada una de las iteraciones de X_i . Finalmente deberá ejecutarse de forma iterativa las operaciones descritas anteriormente, hasta que el valor inicial asignado a N llegase a cero, concluyendo así con un método, que converge hacia una aproximación de la raíz buscada en la ecuación polinómica, pedida en un principio.

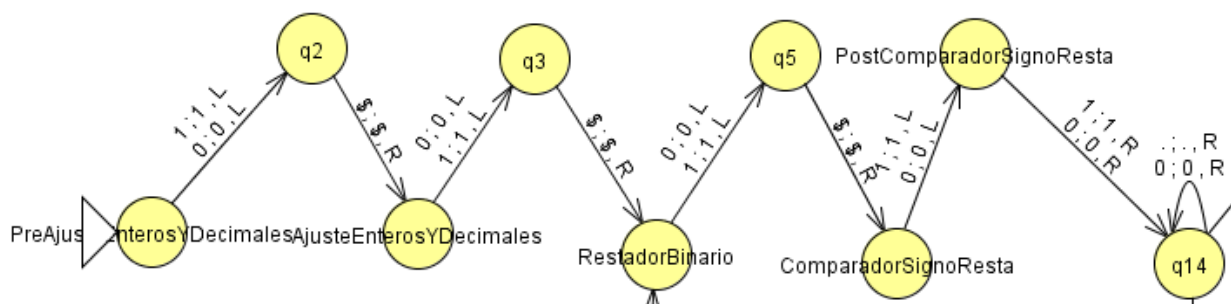
Detalles sobre los Buildings Blocks utilizados

En este apartado pasaremos a mencionar tanto la mayoría de los Buildings Blocks utilizados para la correcta ejecución de la práctica, como aquellos componentes en los que recae una gran importancia del funcionamiento.



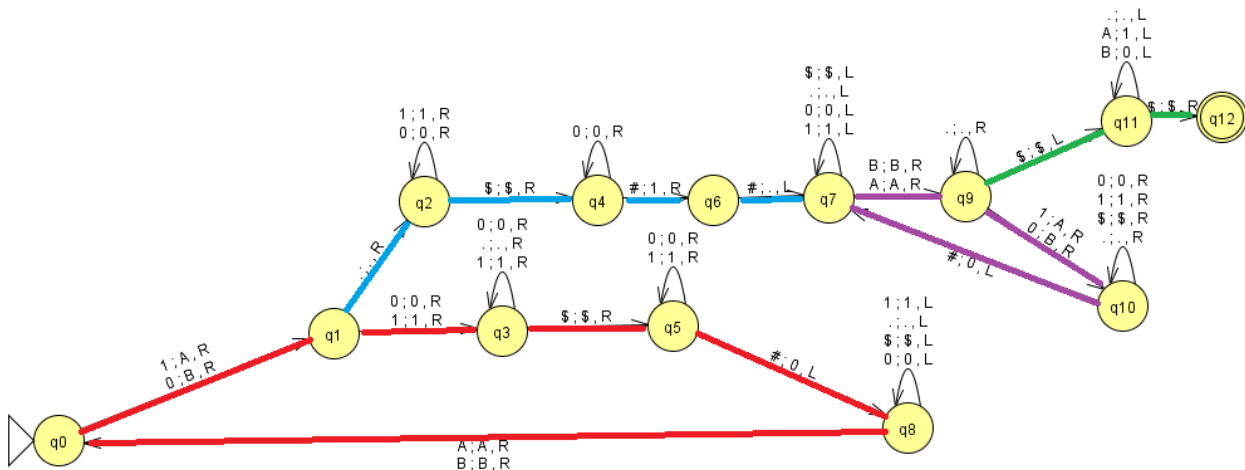
Bloque Cociente:

Para explicar este Building Block vamos a dividirlo en tres partes.



En primer lugar, los Building Block AjusteEnterosYDecimales y su predecesor se encargan de añadir el 1 en la cinta. Puesto que nuestro restador operará con un mismo número de bits que el termino X_n .

Ajuste_enteros_y_decimales:



Este bloque se encarga de preparar la cinta para dividir. Puesto que nuestra división es $\frac{1}{X_n}$ tenemos que añadir en la cinta la siguiente estructura:

| | | | | | | | | | | | |
|---|---|----|---|----|---|----|-----------|----|---|---|---|
| # | # | \$ | A | \$ | N | \$ | X_{n-1} | \$ | 1 | # | # |
|---|---|----|---|----|---|----|-----------|----|---|---|---|

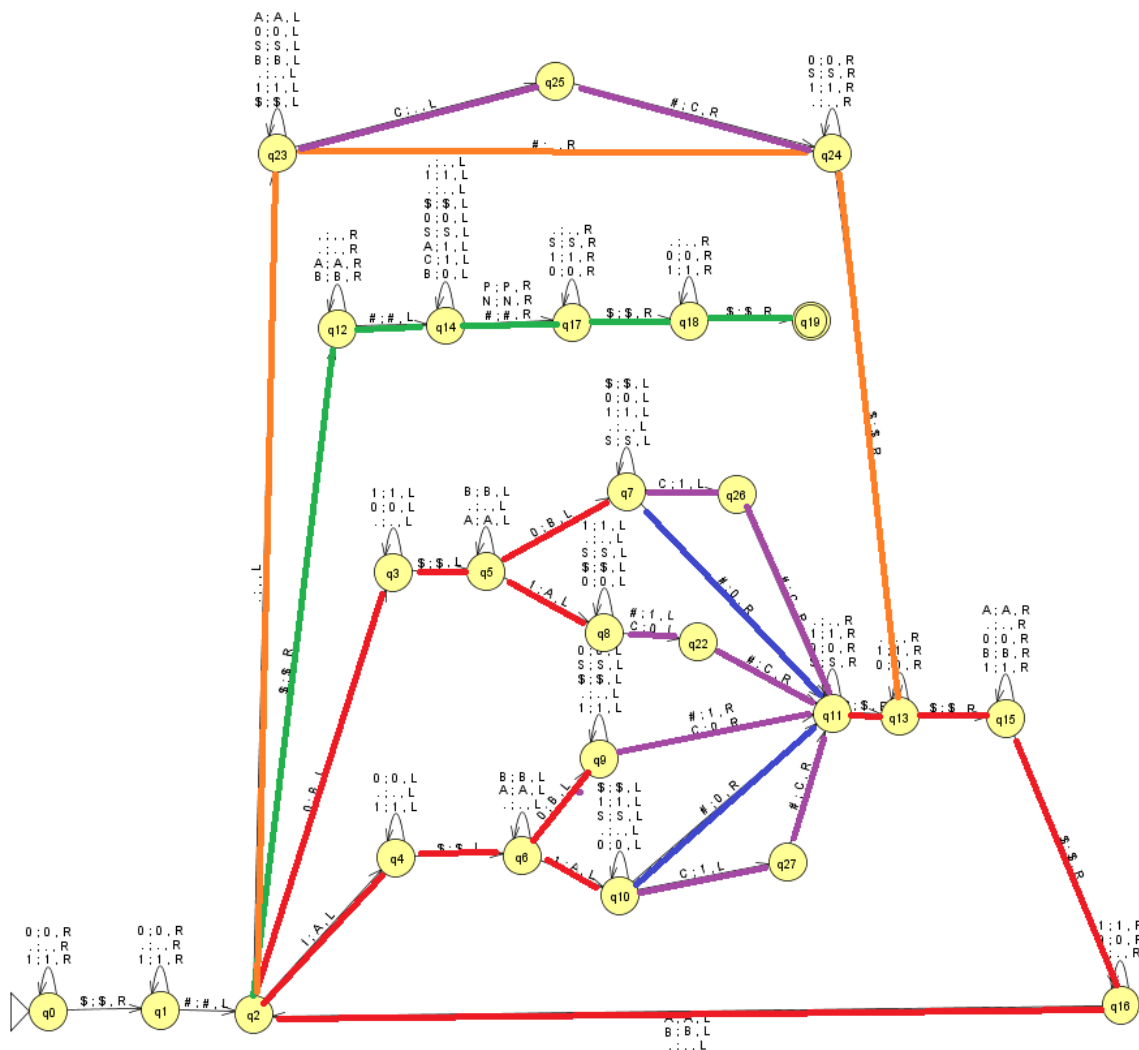
Es importante recordar que este ajuste debe ser genérico, es decir, también debe contemplar el siguiente formato:

| | | | | | | | | | | | | | | | |
|---|---|----|---|----|---|----|-----------|----|---|---|---|---|-----|---|---|
| # | # | \$ | A | \$ | N | \$ | X_{n-1} | \$ | 1 | . | 0 | 0 | ... | # | # |
|---|---|----|---|----|---|----|-----------|----|---|---|---|---|-----|---|---|

Por tanto, el autómata actuará de la siguiente forma:

Lee y marca el ultimo bit de X_{n-1} . Luego escribe un '0' → Cuando lea un '.' pasará a escribir un '1' → A continuación le seguirán tantos '0's como bits decimales tenga → Condición de salida

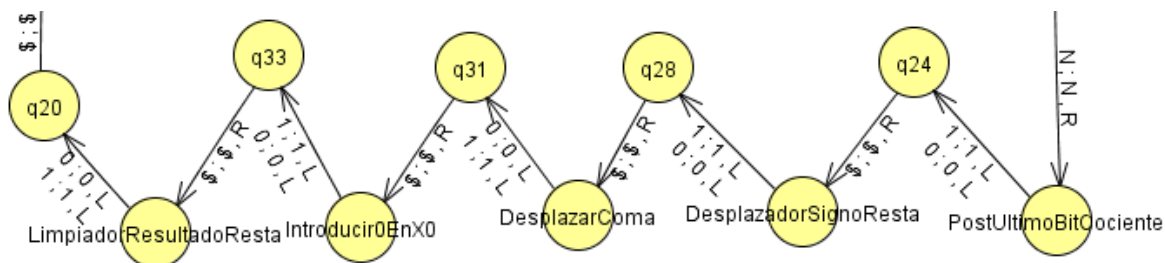
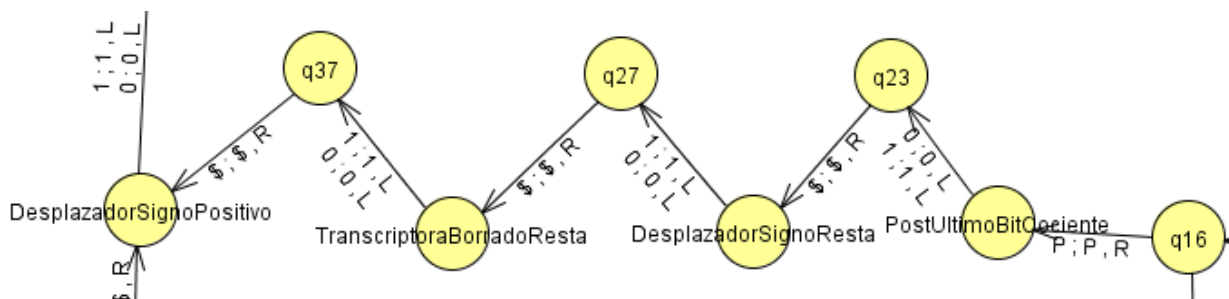
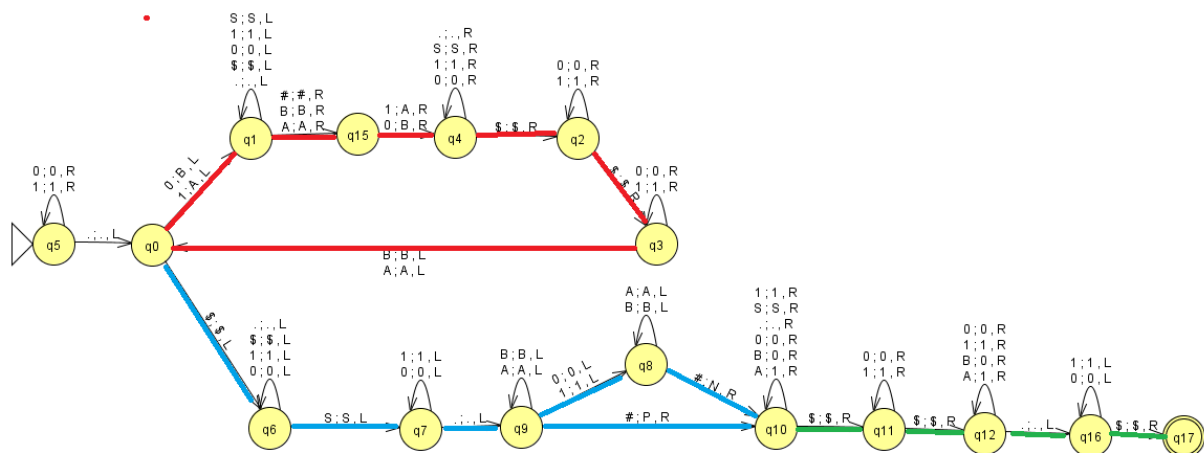
A continuación (una vez ajustado el tamaño de los bits a uno adecuado) se procede a restar.



El funcionamiento del restador se basa en contemplar todas las posibilidades dentro de la operación de la resta y teniendo en cuenta los posibles acarreos que surgen en diversos casos. Tras haber ejecutado la máquina, se nos indicará si el número es negativo, en el caso en el que este tenga overflow. Para realizar la operación empezará con la resta de los pares de bits de los dos números empezando por el lado derecho. Cabe mencionar que los dos números a restar tendrán el mismo número de bits (esto está regulado por otro Building Block).

Realización de la operación de la resta entre el par de bits actual y transición al camino morado en función de si se produce acarreo o no → Aplica el acarreo → No aplica el acarreo → Alcanza el punto, y en función de la existencia de acarreo pasa al morado, y sino simplemente coloca el punto → Condición de salida y retorno mientras desmarca

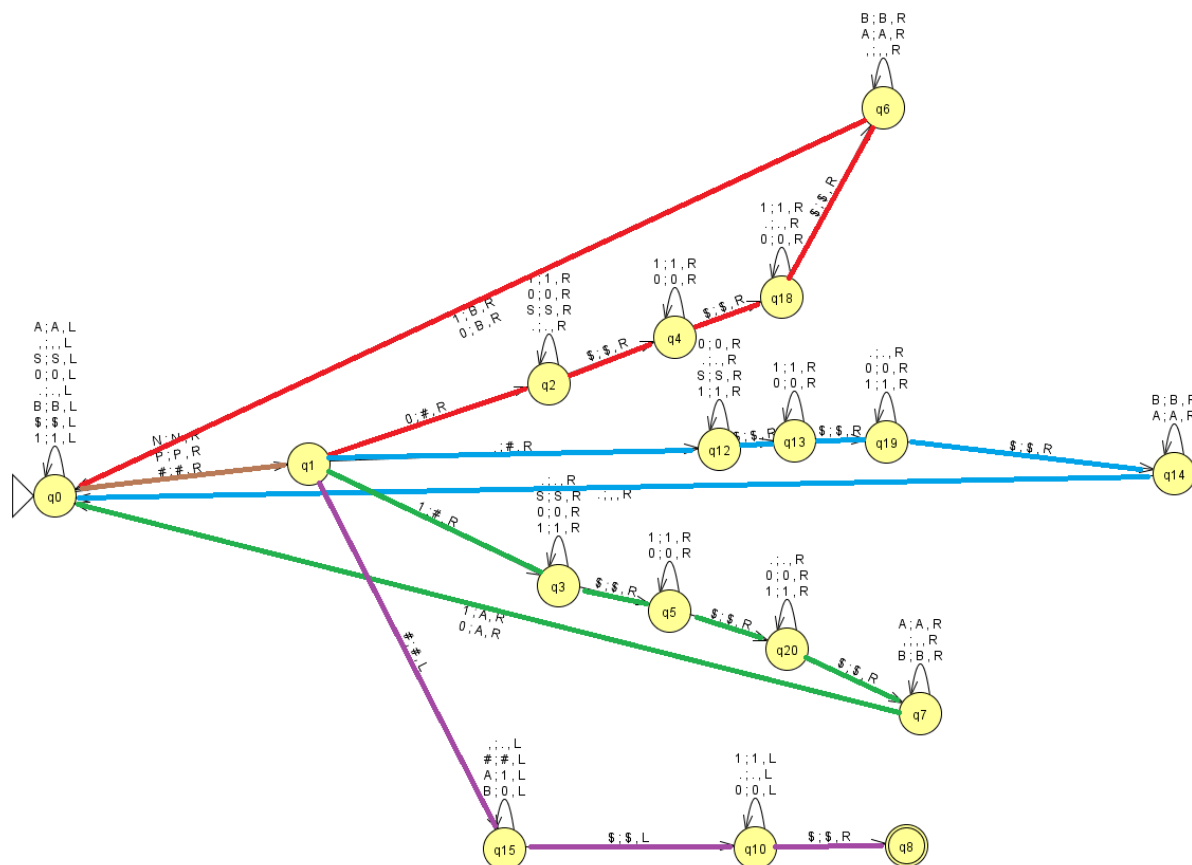
Una vez realizada la resta en complemento A2 pasaremos a comprobar el signo de dicha operación



POSITIVO:

En esta rama del autómata tendremos que introducir el resultado de la resta como nuevo numerador. Para ello hay que eliminar el anterior numerador y transcribir el nuevo.

Transcriptora_borrado_resta

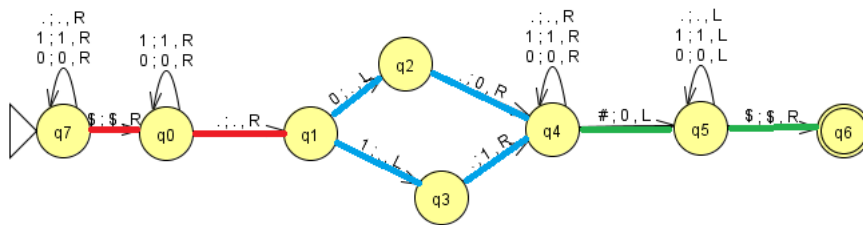


La funcionalidad de este bloque consiste en transcribir el resultado del cociente en el espacio iterativo X0. Para ello debemos retraer el cabezal de la máquina de Turing hasta llegar al primer bit del cociente. Entonces según sea un '.', '0' o '1' el bit que leamos tendremos una bifurcación. En cada bifurcación la operación a ejecutar será la misma, es decir, dependiendo de la entrada, por poner un ejemplo, el 1, se desplazará hasta el final de la cadena y lo sustituirá por el valor del último bit. Este proceso se realizará iterativamente hasta transcribir al completo el resultado del cociente.

Condición de entrada → Bit leído correspondiente a un '.' → Bit leído correspondiente a un '1'
→ Bit leído correspondiente a un '0' → Desplazamiento del cabezal hacia de la izquierda hasta el siguiente '\$'.

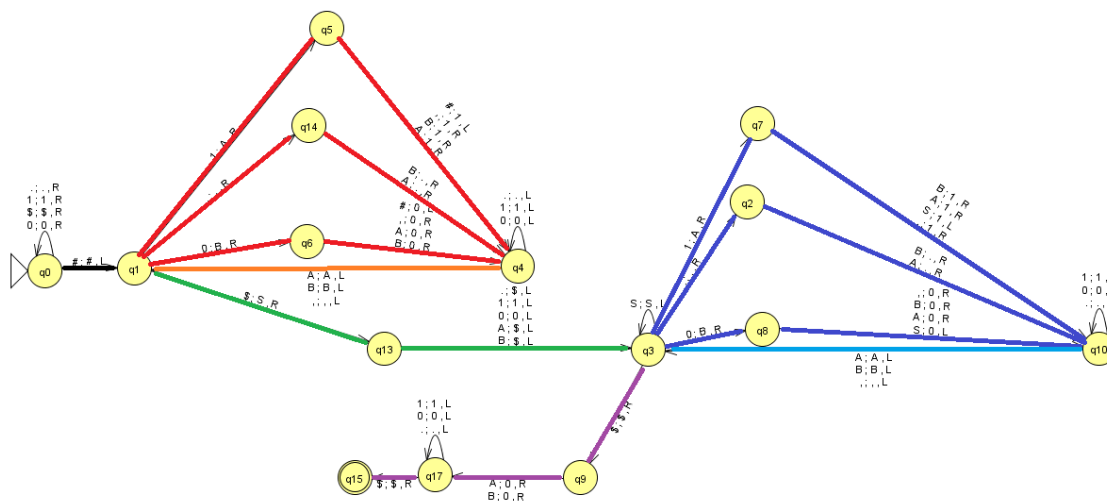
NEGATIVO:

Desplazar_coma:



Localización de la coma ‘.’ → Retroceso y modificación de la coma por el valor posterior y viceversa → Entrada de un cero adicional por la derecha y finalización del proceso

Introducir_0_en_x0:



Para poder insertar un 0 en X_n necesitamos desplazar todos los valores de la cinta desde el numerador una posición a la derecha. Es decir, el siguiente movimiento:

| | | | | | | | | | | | | | | | |
|-----------------|---|----|---|----|---|----|-----------------|---|------------------------|---|---|-----|----|---|---|
| # | # | \$ | A | \$ | N | \$ | 0 | 1 | . | 1 | 0 | ... | \$ | # | # |
| Permanece igual | | | | | | | Permanece igual | | Desplazar una posición | | | | | | |

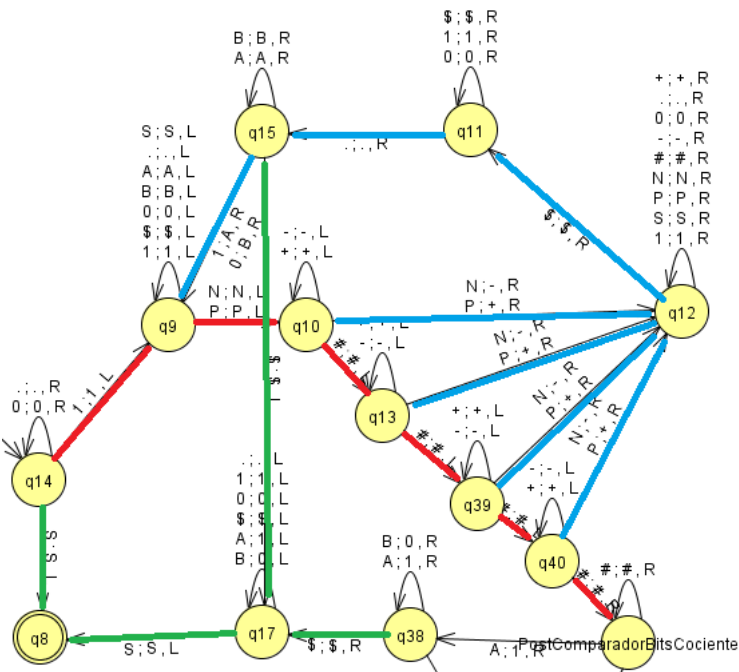
| | | | | | | | | | | | | | | | |
|-----------------|---|----|---|----|---|----|-----------------|---|--------------------------------|---|---|---|-----|----|---|
| # | # | \$ | A | \$ | N | \$ | 0 | 1 | . | 1 | B | 0 | ... | \$ | # |
| Permanece igual | | | | | | | Permanece igual | | Escribe cero y lo marca como B | | | | | | |

| | | | | | | | | | | | | | | | |
|-----------------|---|----|---|----|---|----|-----------------|---|-------------------------------|---|---|---|-----|----|---|
| # | # | \$ | A | \$ | N | \$ | 0 | 1 | . | A | 1 | 0 | ... | \$ | # |
| Permanece igual | | | | | | | Permanece igual | | Escribe uno y lo marca como A | | | | | | |

| | | | | | | | | | | | | | | | |
|-----------------|---|----|---|----|---|----|-----------------|---|---------------------------------|---|---|---|-----|----|---|
| # | # | \$ | A | \$ | N | \$ | 0 | 1 | , | . | 1 | 0 | ... | \$ | # |
| Permanece igual | | | | | | | Permanece igual | | Escribe ‘.’ y lo marca como ‘.’ | | | | | | |

| | | | | | | | | | | | | | | | |
|-----------------|---|----|---|----|---|--------------------------------|---|---|-----------------|---|---|---|-----|----|---|
| # | # | \$ | A | \$ | N | \$ | 0 | A | 1 | . | 1 | 0 | ... | \$ | # |
| Permanece igual | | | | | | Escribe uno y lo marca como A | | | Permanece igual | | | | | | |
| # | # | \$ | A | \$ | N | \$ | B | 0 | 1 | . | 1 | 0 | ... | \$ | # |
| Permanece igual | | | | | | Escribe cero y lo marca como B | | | Permanece igual | | | | | | |
| # | # | \$ | A | \$ | N | \$ | 0 | 0 | 1 | . | 1 | 0 | ... | \$ | # |
| Permanece igual | | | | | | Añade 0 | | | Permanece igual | | | | | | |

PRECISIÓN:



Este bloque determinará la precisión del cálculo de decimales. En otras palabras, iremos marcando los bits del cociente calculado y los bits de nuestro denominador.

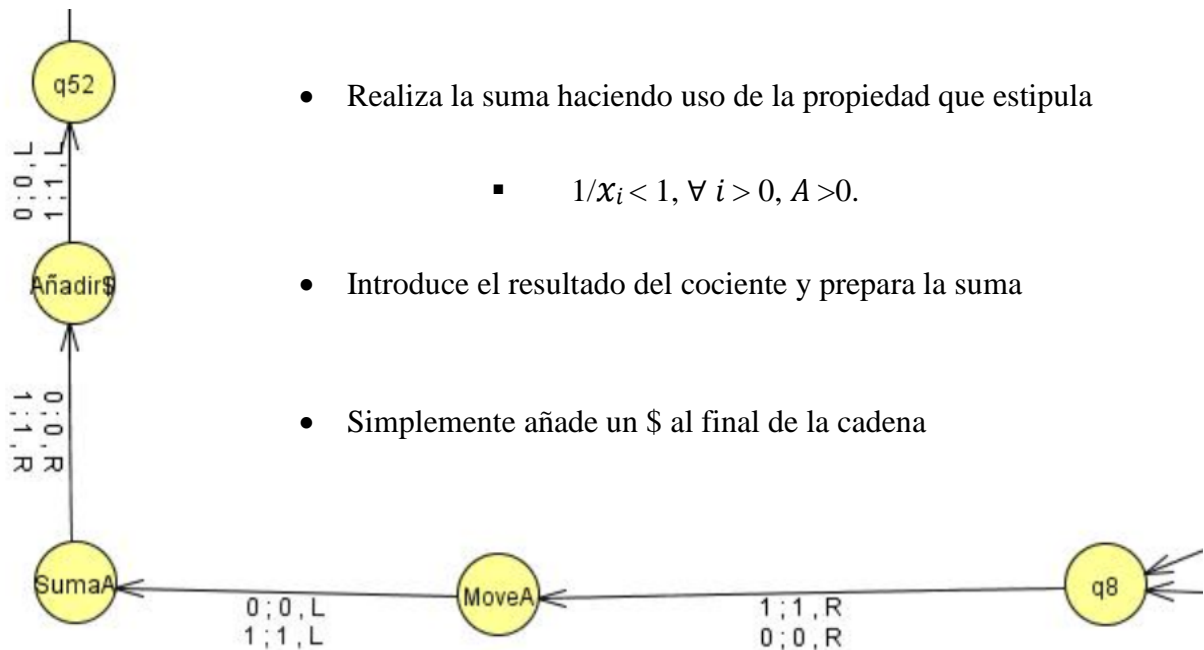
Este bloque determinará la precisión del cálculo de decimales. En otras palabras, iremos marcando los bits del cociente calculado y los bits de nuestro denominador. La ejecución de este planteamiento es totalmente variable, esto quiere decir, que en función de la entrada que tengamos, los bits de precisión serán más o menos (esto habiendo sido regulado por otro Building Block que se encarga de tener el mismo número de bits).

Acceso al cociente y comienzo de marcado → Búsqueda de otro bit en el denominador y marcado del mismo → Condición de salida

Si hemos marcado todos los bits del denominador (lectura de un \$) procedemos a desmarcarlos. Una vez hecho finaliza el proceso. Cabe recalcar que este último camino puede comprobarse dos puntos (cociente y denominador) así que disponemos de dos accesos, lo cual optimiza el proceso. De esta forma su precisión consistirá en marcar el mismo número de bits cociente.

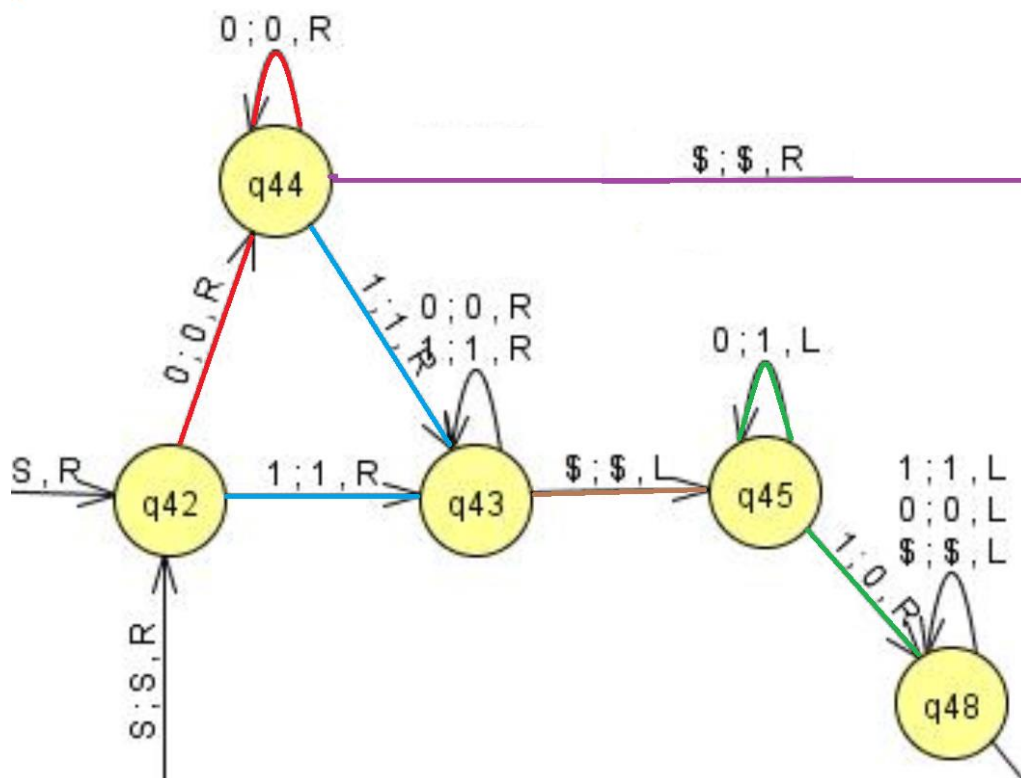
| | | | | | | | | | | | | | | | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|-----|----|---|---|---|---|---|---|---|---|---|
| Entrada: | # | N | N | P | N | P | # | # | N | S | ... | \$ | 1 | 0 | . | 1 | 0 | 0 | 1 | 1 | # |
| Salida: | # | - | - | + | - | + | # | # | - | S | ... | \$ | A | B | , | A | B | B | A | A | # |

Sumador:



- Realiza la suma haciendo uso de la propiedad que estipula
 - $1/x_i < 1, \forall i > 0, A > 0.$
- Introduce el resultado del cociente y prepara la suma
- Simplemente añade un \$ al final de la cadena

Decrementador:



Proceso que controla si N vale 0 → Condición de entrada en el decrementador (Si N no vale 0)

→ Condición de salto al siguiente proceso → Reduce en una unidad a N

→ Condición de salida (Si N vale 0)

Batería de pruebas:

En cuanto a las pruebas seleccionadas, hemos decidido pasar diferentes cadenas que van desde los casos normales, hasta casos especiales para así probar como de robusto es la máquina de Turing creada.

La primera cadena se trata de una cadena totalmente normalizada y la cual no se pondría como ningún caso especial. La segunda cadena de prueba tiene la peculiaridad de que $N=0$ mientras que la tercera se caracteriza porque $A=0$. En cuanto a la cuarta, consta de $X0$ con decimales iguales a cero y una alta precisión, sin embargo, la quinta se basaría en lo mismo, pero haciendo uso de una baja precisión. Como última cadena se lleva a cabo la prueba con $X0$ como entero igual a cero.

Formato: -Entrada / -Salida

Aciertos:

Cadena normal.

```
#####S11$11$10.01110#####  
#####11.01001#####
```

Cadena con $N=0$.

```
#####S00$10$01.01110#####  
#####10.10110#####
```

Cadena con $A=0$.

```
#####S10$00$11.01101#####  
#####0.01001#####
```

Cadena con $X0$ =con decimales iguales a cero y alta precisión.

```
#####S110$1$10.00000000#####  
#####1.100111100#####
```

Cadena con $X0$ con decimales iguales a cero y baja precisión.

```
#####S110$1$10.00#####  
#####1.10#####
```

Cadena con $X0$ con entero igual a cero.

```
#####S1$1110$00.0101010#####  
#####1110.0000111#####
```

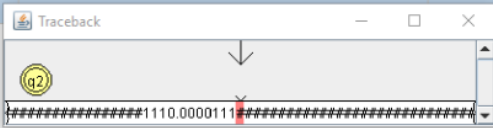
Fallos:

- División entre cero
- Dividir entre un entero (sin especificar decimales)

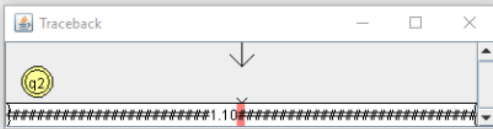
Tabla de pruebas

| Input | Output | Result |
|----------------------------------|--------|--------|
| #####S11\$11\$10.01110\$##### | ##### | Accept |
| #####S00\$10\$01.01110\$##### | ##### | Accept |
| #####S10\$00\$11.01101\$##### | ##### | Accept |
| #####S110\$1\$10.00000000\$##### | ##### | Accept |
| #####S110\$1\$10.00\$##### | ##### | Accept |
| #####S1110\$00.0101010\$##### | ##### | Accept |

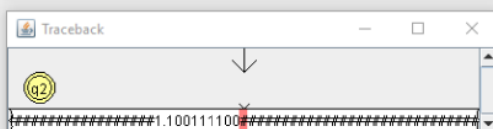
Traceback



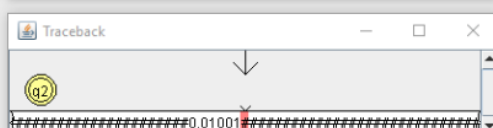
Traceback



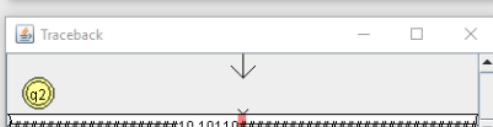
Traceback




Traceback



Traceback



Traceback



Load Inputs Run Inputs Clear Enter Lambda View Trace

Máquina de Turing

