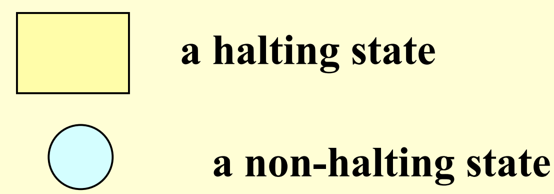
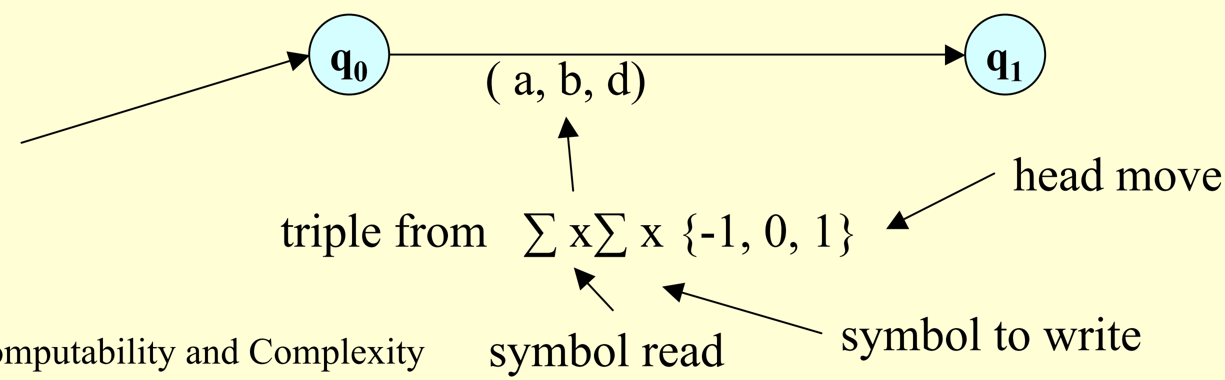
Turing Machine Representation

1. List all the entries in the **δ-function** (as in previous examples)
   but • cumbersome for more than 2 or 3 states
   • difficult to see the structure or pattern in the algorithm

2. Draw a **State Diagram** -
   states are represented by nodes
   rectangles for halting states
   circles for other states
   δ-function entries as arrows from current state to new state

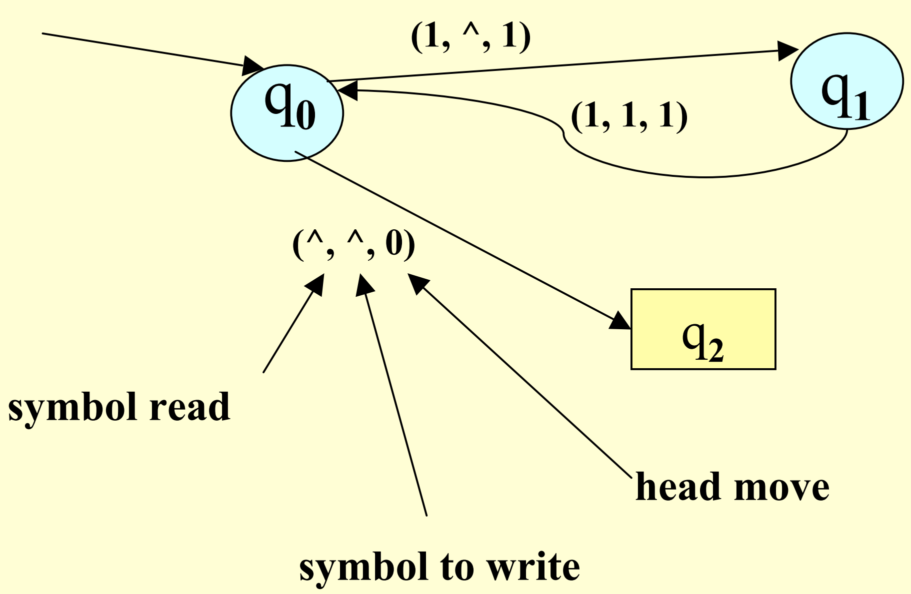




for example, the odd/even Turing Machine

$\delta\,(q_0, 1) = (q_1, \wedge, 1)$
$\delta\,(q_0, \wedge) = (q_2, \wedge, 0)$
$\delta\,(q_1, 1) = (q_0, 1, 1)$

If $\delta(q, a) = (q', b, d)$ the graph has an arrow labeled (a, b, d) from q to q'



3. as **Pseudocode**

basic operations: TM read
TM write
head movement

\+ control structures: if..then..else
while..do

pseudocode feels like “proper programming” but ..

it is easy to work at too high a level and to forget that

- the TM cannot address its storage (tape), just read the symbol on the current-square
- it cannot do additional operations ”on the side” eg counting,

an example: the **Head** function: build TM, M such that

$$f_M(w) = \text{head}(w)\ ,\ w \in I^*$$

Input: a word w of I^*, starting at square 0

Output: a word consisting of the first symbol of w, followed by ^.

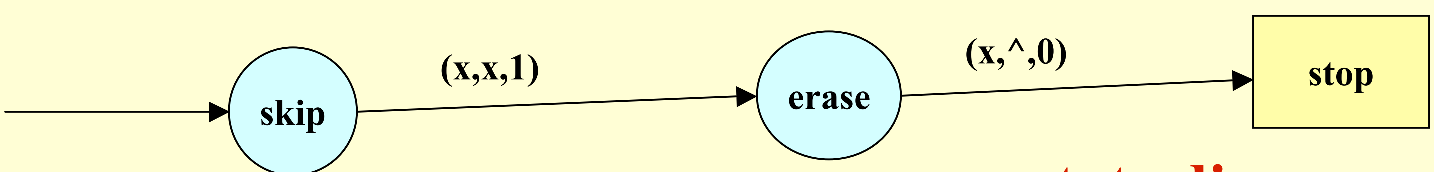
$\sum = I \cup \{\wedge\}$

set of states, Q = {skip, erase, stop}

δ-function: $\delta(\text{skip}, x) = (\text{erase}, x, 1)$, all $x \in I$

$\delta(\text{erase}, x) = (\text{stop}, \wedge, 0)$, all $x \in I$

so $M = \{Q, \sum, I, \text{skip}, \delta\ \{\text{stop}\}\}$

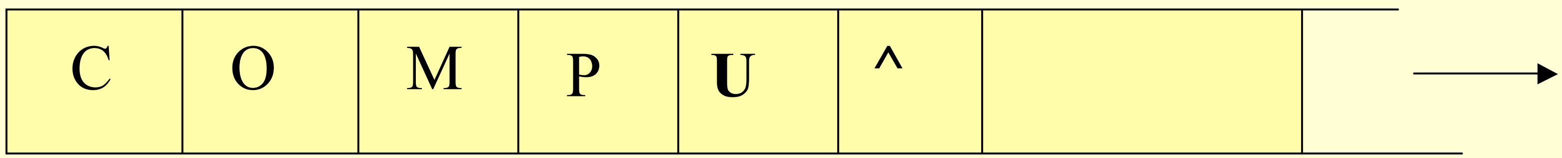


state diagram

pseudocode:

read symbol from current square
if symbol = ‘^’ then Halt and Fail
move right
write ‘^’ (whatever symbol is read)
Halt and Succeed.

eg.



Input - the string of symbols up to but not including the first ‘^’

Output - the string of symbols up to but not including the first ‘^’
..rest of tape ignored

pseudocode:

read symbol from current square
if symbol = ‘^’ then Halt and Fail
move right
write ‘^’ (whatever symbol is read)
Halt and Succeed.

eg.

C	^	M	P	U	^		

Input - the string of symbols up to but not including the first ‘^’

Output - the string of symbols up to but not including the first ‘^’
..rest of tape ignored

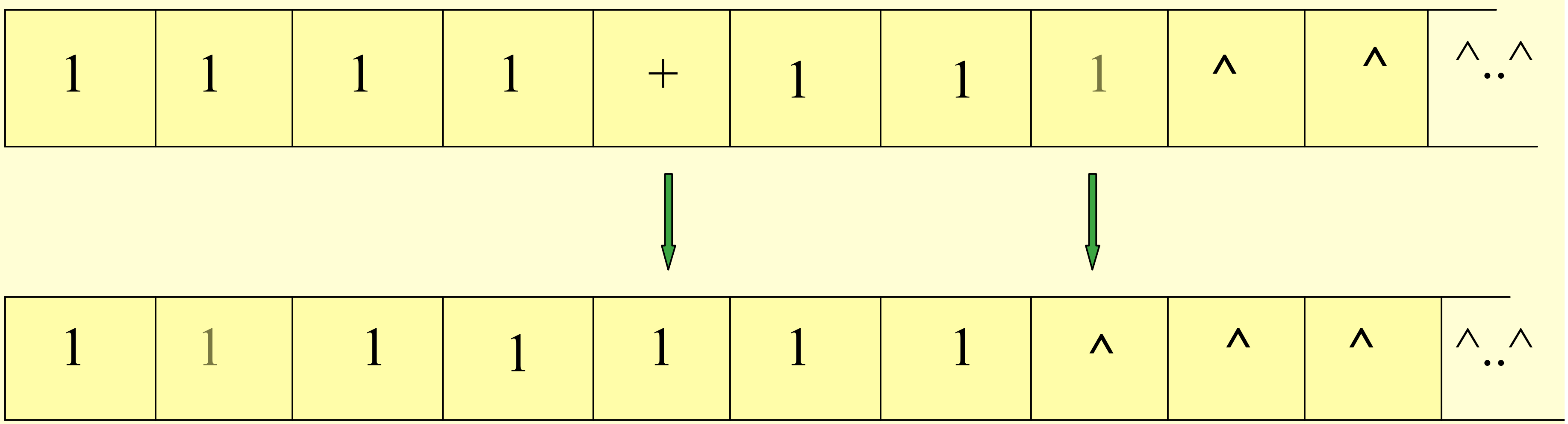
Example - Unary addition

Unary notation: represent n by 1111..11 (n 1s)..written as 1^n

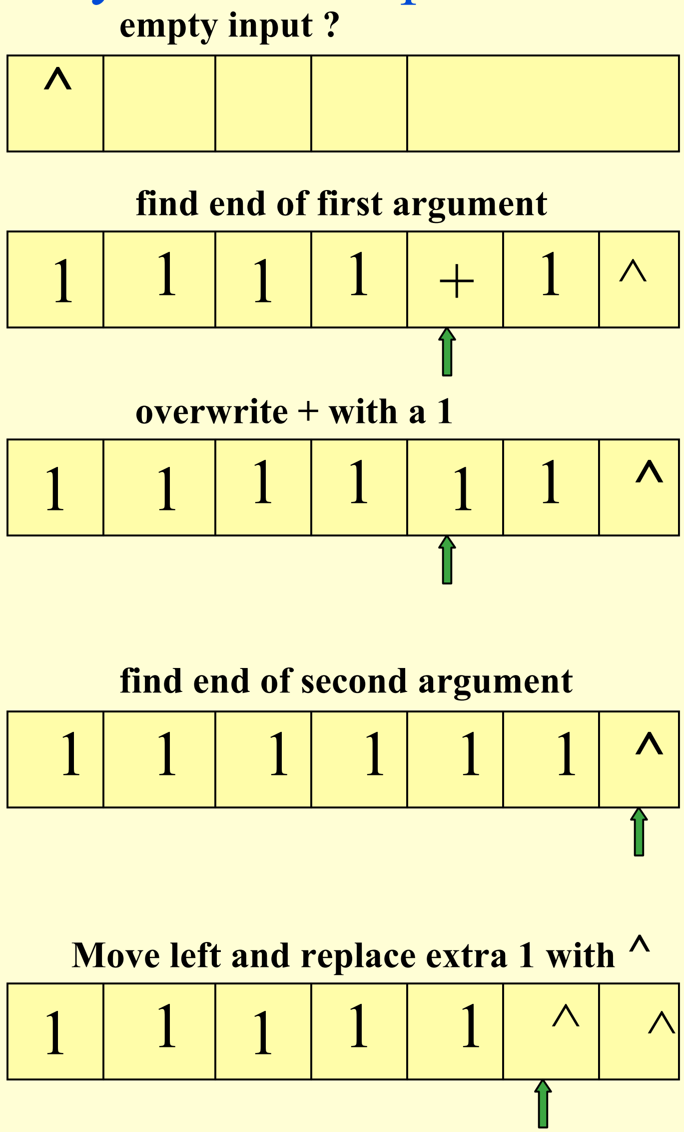
Design TM, M, such that $f_M(1^n.+.\ 1^m) = 1^{\ n+m}$

I = {1,+}

$\sum$ = {1, +, ^}

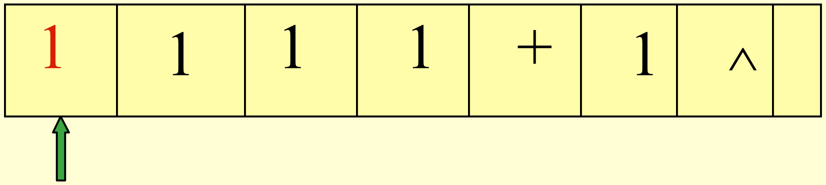


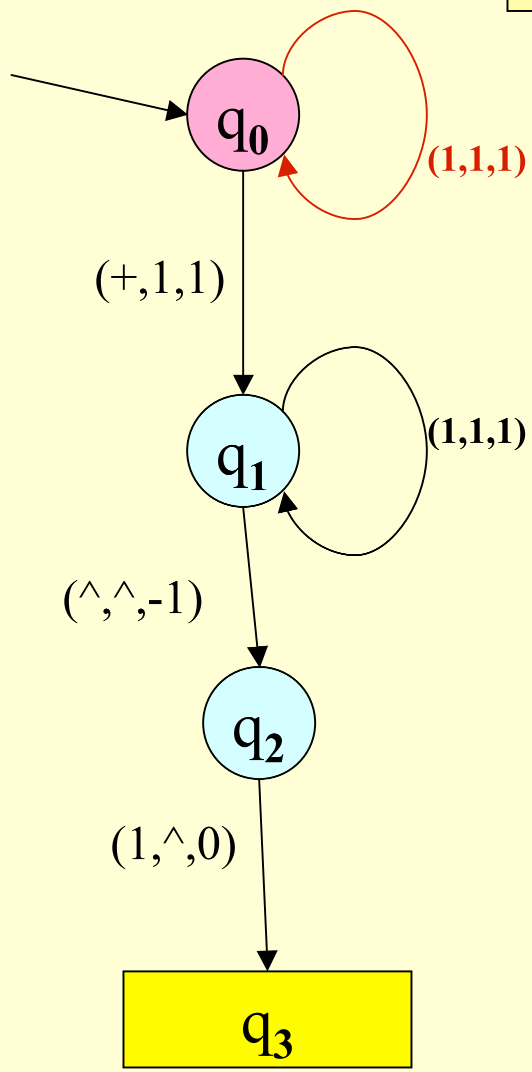
Unary addition - pseudocode




```
if current symbol = ^ then Halt & Fail
while current-symbol = 1
                    write 1
                    move right
endwhile
if current-symbol =  +
                    write 1
                    move right
          else Halt & Fail
endif
while current-symbol = 1
                    write 1
                    move right
end while
if current-symbol = ^
                    write ^
                    move left
          else Halt & Fail
end if
write ^
Halt & Succeed
```

Unary addition - state diagram:





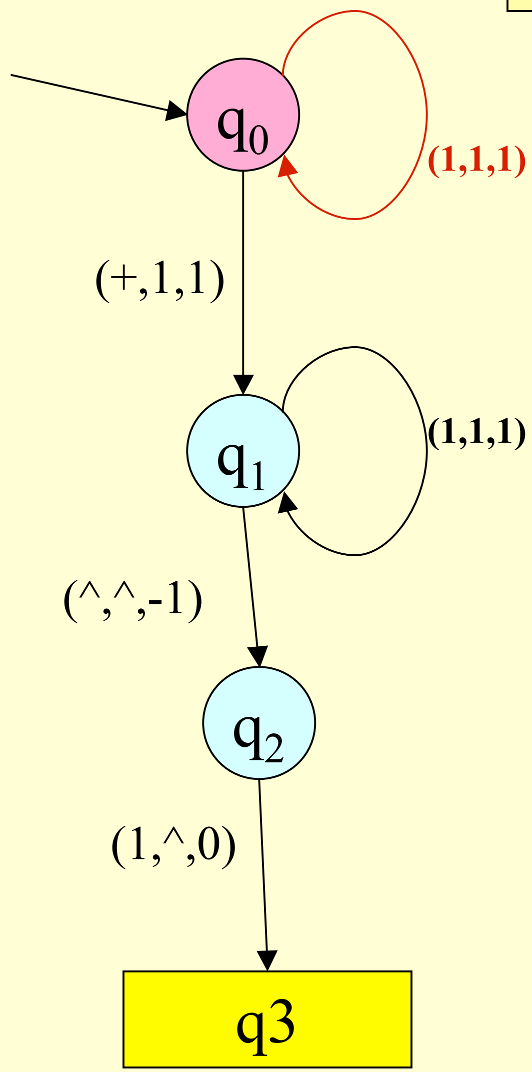
$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$
$\delta(q_2,1) = (q_3,\wedge,0)$

Unary addition - state diagram:

1	1	1	1	+	1	^	

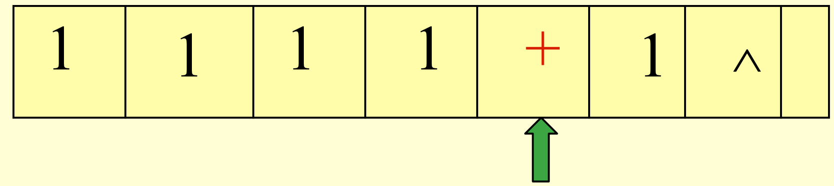


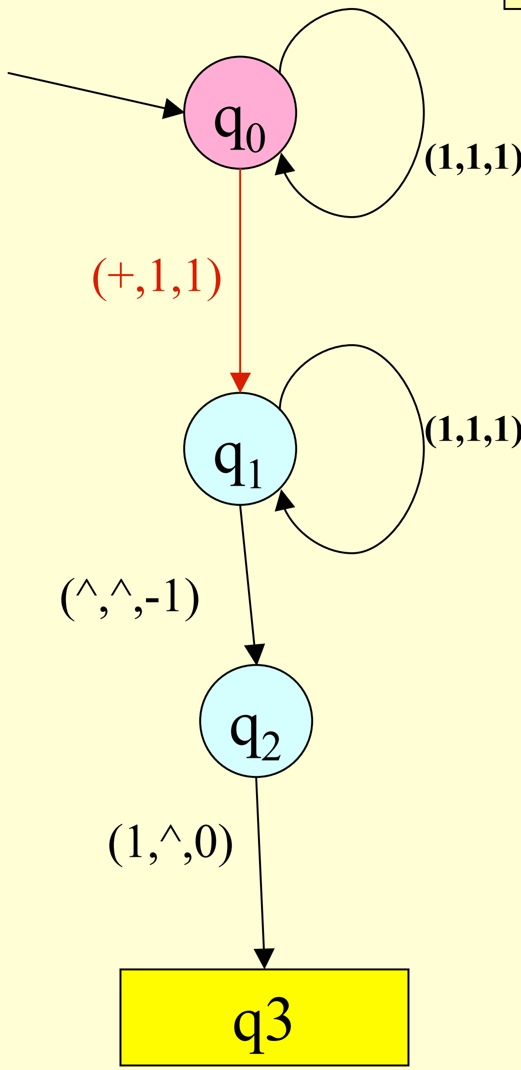
$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$
$\delta(q_2,1) = (q_3,\wedge,0)$

Unary addition - state diagram:



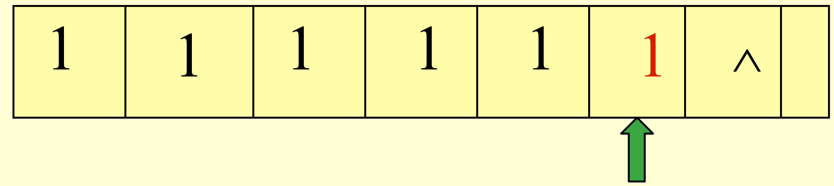


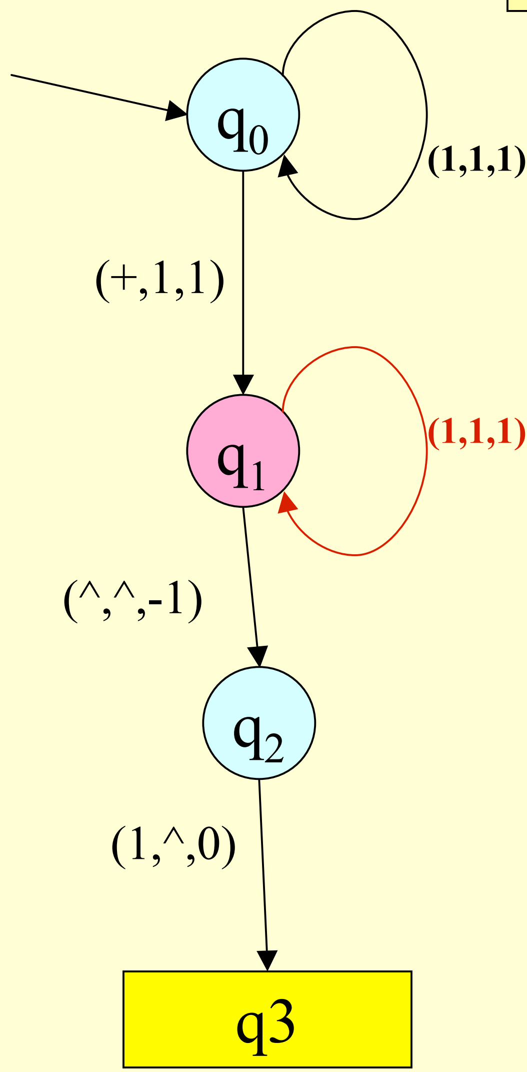
$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$
$\delta(q_2,1) = (q_3,\wedge,0)$

Unary addition - state diagram:



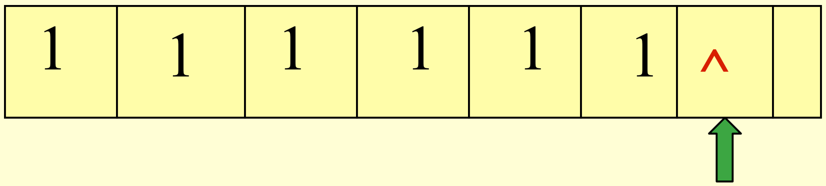


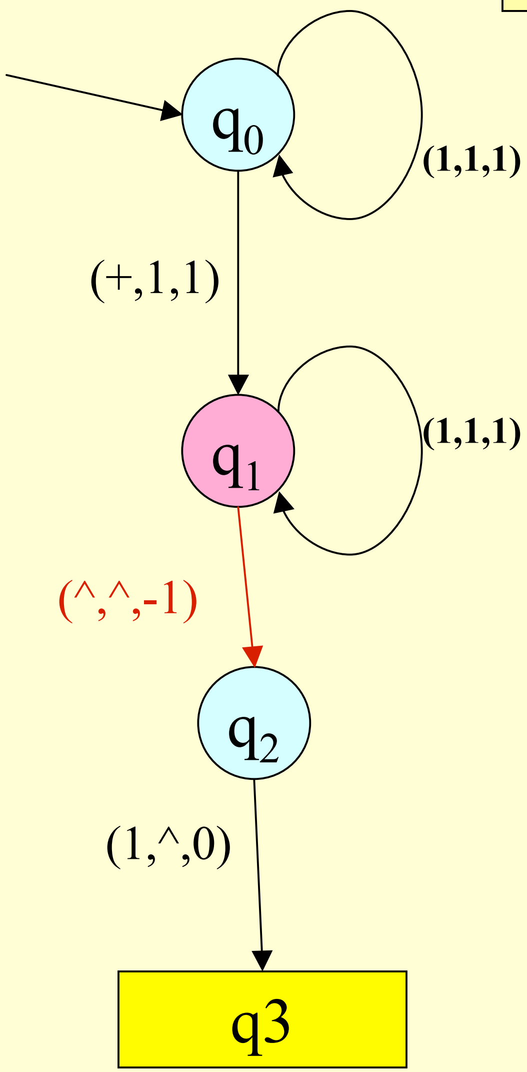
$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0, 1) = (q_0, 1, 1)$
$\delta(q_0, +) = (q_1, 1, 1)$
$\delta(q_1, 1) = (q_1, 1, 1)$
$\delta(q_1, \wedge) = (q_2, \wedge, -1)$
$\delta(q_2, 1) = (q_3, \wedge, 0)$

Unary addition - state diagram:



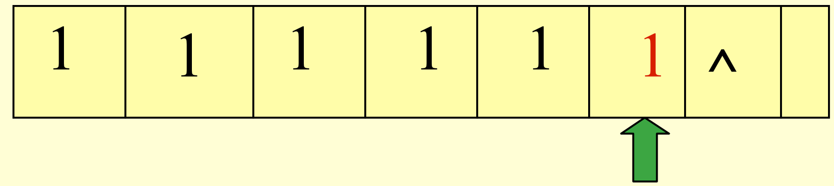


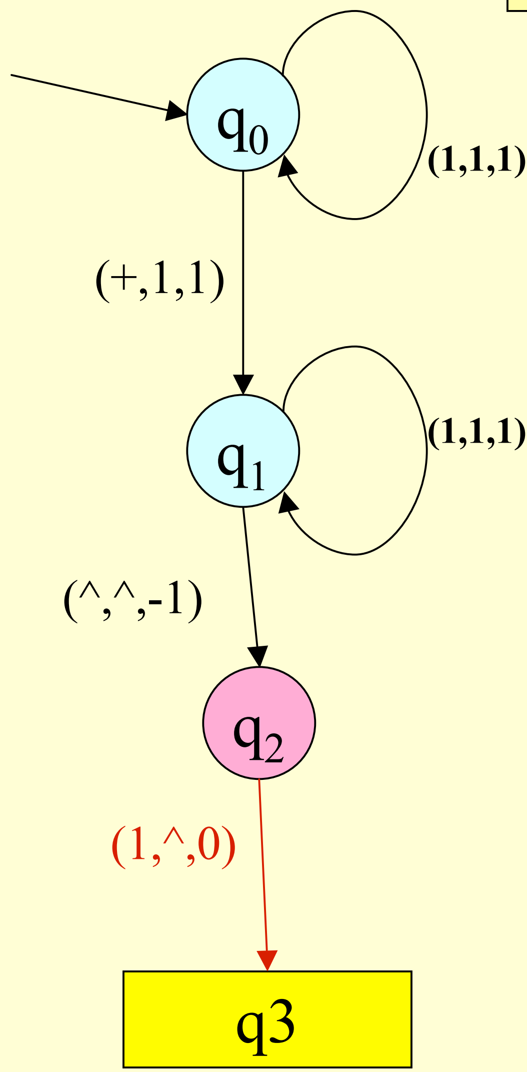
$Q = \{q_0,q_1,q_2,q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$
$\delta(q_2,1) = (q_3,\wedge,0)$

Unary addition - state diagram:



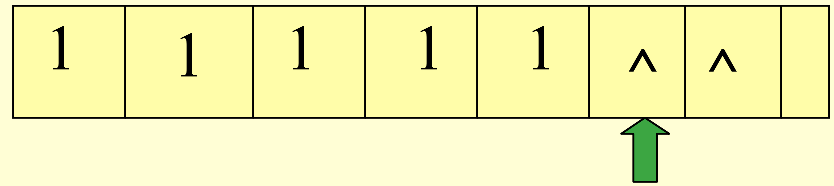


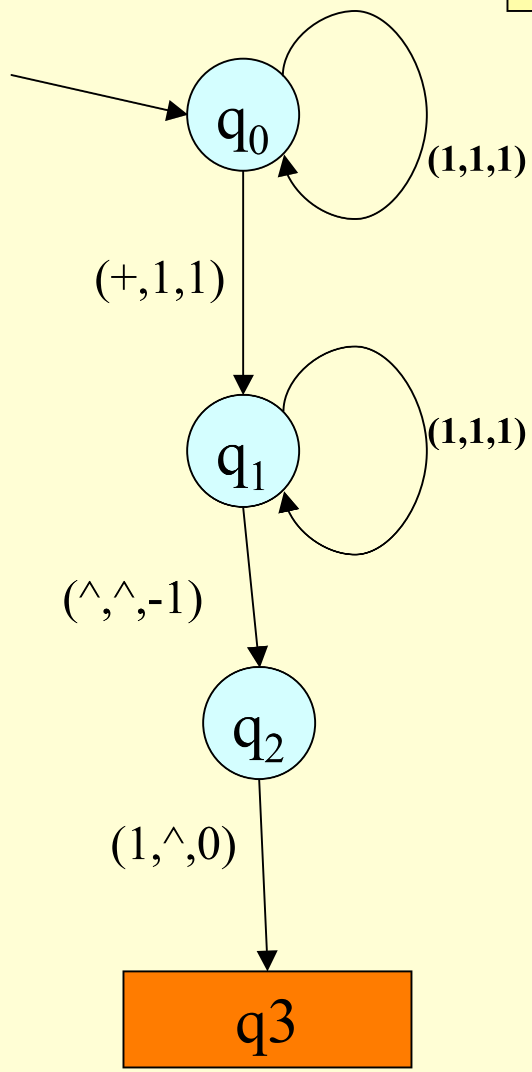
$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$
$\delta(q_2,1) = (q_3,\wedge,0)$

Unary addition - state diagram:





$Q = \{q_0, q_1, q_2, q_3)$
$F = \{q3\}$

δ-function:

$\delta(q_0,1) = (q_0,1,1)$
$\delta(q_0,+) = (q_1,1,1)$
$\delta(q_1,1) = (q_1,1,1)$
$\delta(q_1,\hat{}) = (q_2,\hat{},-1)$
$\delta(q_2,1) = (q_3,\hat{},0)$

Design a Turing Machine to implement the Tail function

$Q = \{q_0, \ ..\ \ \ \ \ \ \ \ \ \}$, $F = \{\ \ \}$, $\sum = \{a,b,\wedge\}$.

a	**b**	**a**	**b**	^	^ ^ ^ ^

Method: • if current symbol = ^, then move left. (gives Halt & Fail where input word = ε)
else leave current symbol unchanged
move right
• repeatedly { • Let current symbol be ‘s’; leave s unchanged and move left
• write ‘s’
• move right
• if current-symbol = ^ then Halt & Succeed
else move right}

Design a Turing Machine to implement the Tail function

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $F = \{q_6\}$, $\sum = \{a,b,\wedge\}$.

Algorithm 1- “oscillating”

a	**b**	**a**	**b**	^	^ ^ ^ ^

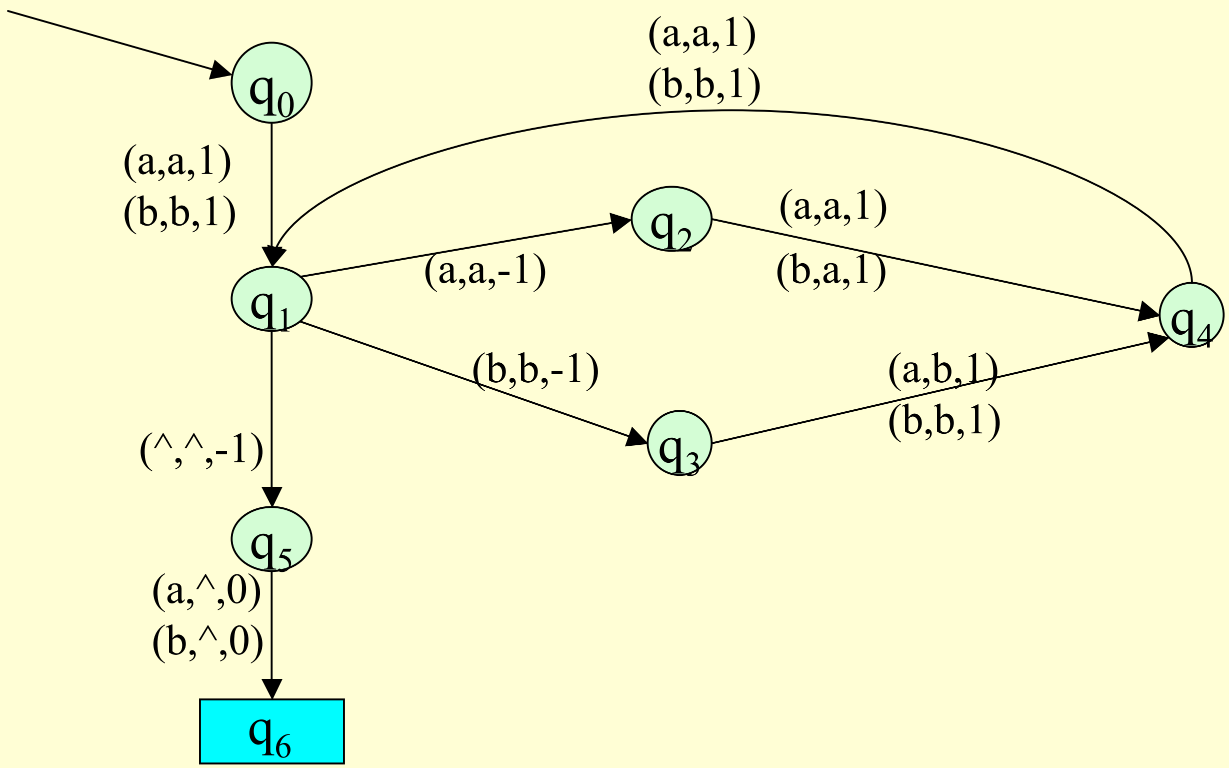
$\delta(q_0,a) = (q_1,a,1)$
$\delta(q_0,b) = (q_1,b,1)$

$\delta(q_1,a) = (q_2,a,-1)$
$\delta(q_1,b) = (q_3,b,-1)$
$\delta(q_1,\wedge) = (q_5,\wedge,-1)$

$\delta(q_2,a) = (q_4,a,1)$
$\delta(q_2,b) = (q_4,a,1)$

$\delta(q_3,a) = (q_4,b,1)$
$\delta(q_3,b) = (q_4,b,1)$

$\delta(q_4,a) = (q_1,a,1)$
$\delta(q_4,b) = (q_1,b,1)$
$\delta(q_5,b) = (q_6,\wedge,0)$
$\delta(q_5,a) = (q_6,\wedge,0)$



Design a Turing Machine to implement the Tail function

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $F = \{q_5\}$, $\sum = \{a,b,\wedge\}$.

Algorithm 2 - “shifting”

a	**b**	**a**	**b**	**^**	**^ ^ ^ ^**

Method 2:

If current symbol = ^ Halt & Fail.
Repeat {move right}
until current symbol = ^.
Move left;
Let ‘s’ be current symbol; write ^.
Repeatedly: {move left write ‘s’, let ‘s’ be current symbol ; until reach square 0}**
Halt & Succeed

** the **TM squares are not marked or individually addressable**, but the machine must be able to recognise when, moving left, it has reached square 0.
This is done to avoid causing Halt & Fail by a further move left.

One method is to write a special symbol in square 0 which can be recognised later - often ^ can be used.

Design a Turing Machine to implement the Tail function

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, $F = \{q_5\}$, $\sum = \{a,b,\wedge\}$.

Algorithm 2 - “shifting”

a	**b**	**a**	**b**	^	^ ^ ^ ^

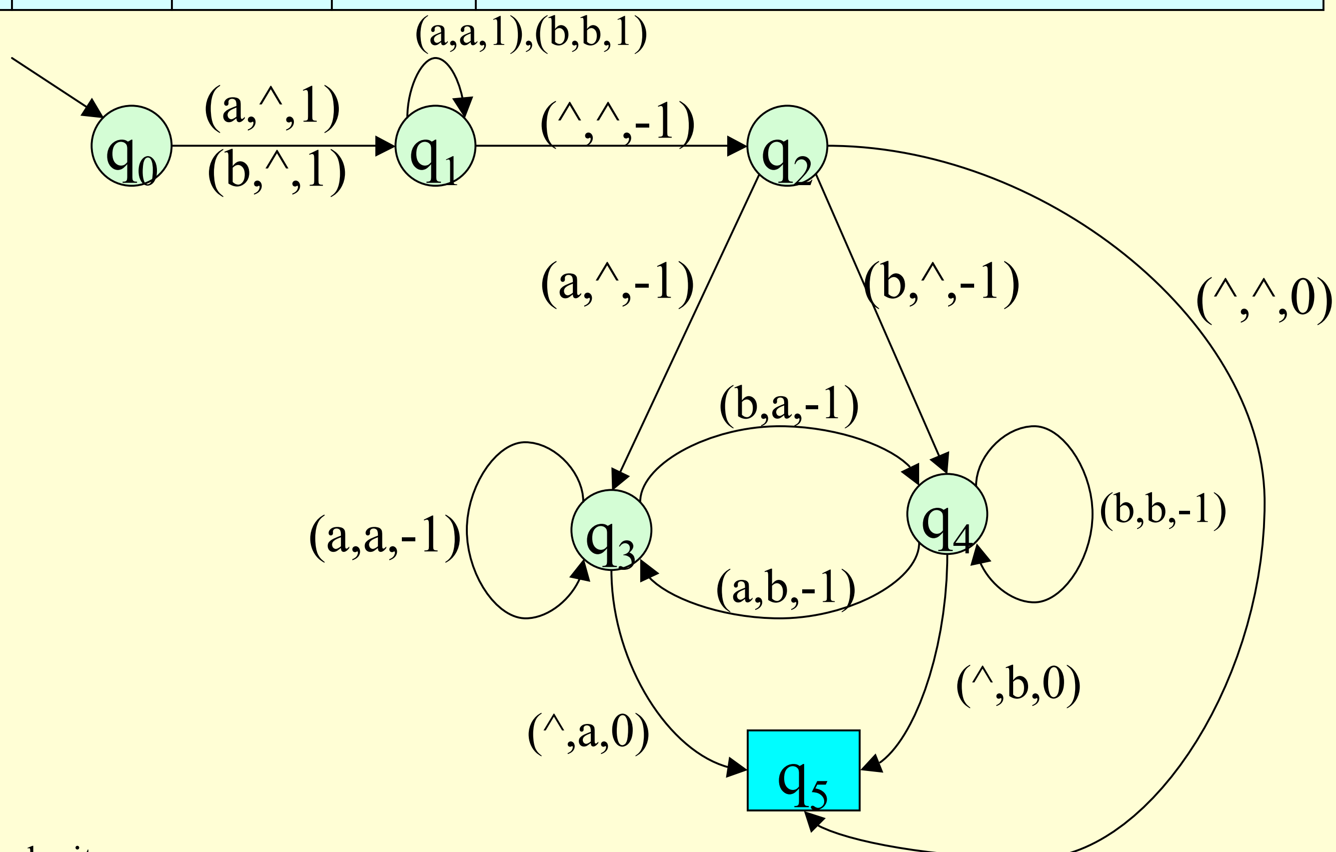
$\delta(q_0,a) = (q_1,\wedge,1)$
$\delta(q_0,b) = (q_1,\wedge,1)$

$\delta(q_1,a) = (q_1,a,1)$
$\delta(q_1,b) = (q_1,b,1)$
$\delta(q_1,\wedge) = (q_2,\wedge,-1)$

$\delta(q_2,\wedge) = (q_5,\wedge,0)$
$\delta(q_2,a) = (q_3,\wedge,-1)$
$\delta(q_2,b) = (q_4,\wedge,-1)$

$\delta(q_3,a) = (q_3,a,-1)$
$\delta(q_3,b) = (q_4,a,-1)$
$\delta(q_3,\wedge) = (q_5,a,0)$

$\delta(q_4,a) = (q_3,b,-1)$
$\delta(q_4,b) = (q_4,b,-1)$
$\delta(q_4,\wedge) = (q_5,b,0)$



Summary

We have seen δ-function
state diagram
pseudo-code
representations for Turing Machines and examples of these for odd/even
Head
unary addition
Tail functions.

State diagrams permit visualisation of the TM and the structure of an algorithm. They correspond directly to the δ-function representation.