Non-Deterministic TM:

Language Recognition

(i) NFA / FA : Same Class.

(ii) DPDA / NPDA : NPDA is powerful.
eg. PAL can not be accepted by any DPDA.

(iii) DTM / NTM : Same class.

- A Non-Deterministic TM is as powerful as a Deterministic TM.

- Definition of NTM is same as a TM except a change in the δ fn.

Deterministic TM – δ: $Q \times (\Gamma \cup \{\Delta\})$
$\rightarrow (Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{L, S, R\}$

NTM – δ: $Q \times (\Gamma \cup \{\Delta\})$
$\rightarrow$ subsets of $(Q \cup \{h\}) \times (\Gamma \cup \{\Delta\}) \times \{L, S, R\}$

Theorem

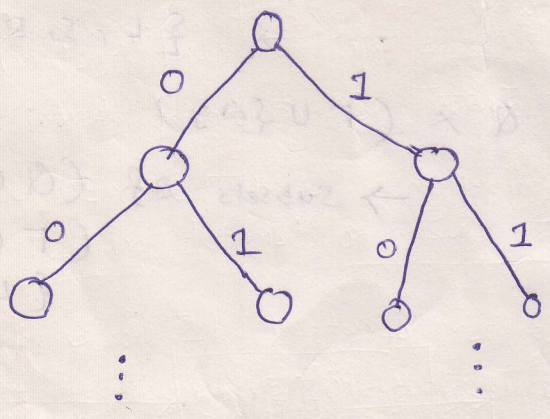
Let $T_1 = (Q_1, \Sigma, \Gamma_1, q_1, \delta_1)$ be a Non-Deterministic Turing machine. Then, there is an ordinary (Deterministic) TM $T_2 = (Q_2, \Sigma, \Gamma_2, q_2, \delta_2)$ with $L(T_2) = L(T_1)$.

- Idea is to create a DTM which mimics the activity of NDTM.

- Suppose for some i/p x, T_1 halts, then T_2 must also halt.

- If T_1 Crash or Loops, then T_2 should try to search for any alternative path in the transition diagram of T_1 such that it leads to acceptance of string, i.e. T_2 should discard current sequence of moves and try some other moves possible on T_1.

- Without loss of generality, we assume that at every point there are exactly two moves ~~two~~ available to T_1, which we arbitrarily number as '0' and '1'. We represent this choice in the form of a 'Computational Tree' which looks like as follows;



- In the tree, the root node corresponds to the initial configuration, Leaf nodes correspond to configuration where T_1 halts or crashes.

- T_2 does BFS traversal of the tree, i.e. it tries out paths of length 1 and if they fail, then paths of length 2 and so on. [This is to ensure that T_2 never get stuck in

Construction of T_2

- T_2 is built out of a no. of sub-TM which will do a unique job.
- T_2 is assumed to have 3 tapes.

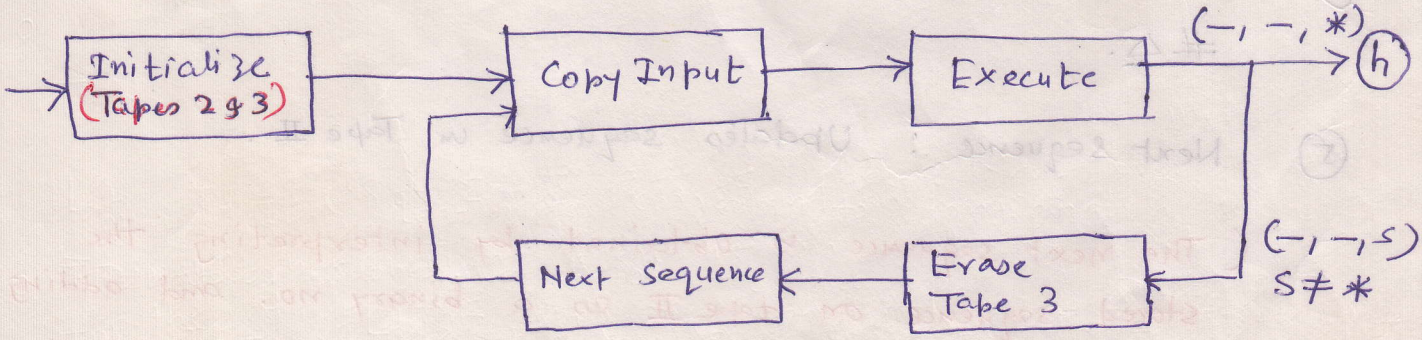
Tape I : Used to save the original i/p string and its content are never changed.

Tape II : It is used to keep track of the sequence of moves of T_1 that T_2 is currently attempting to execute (Here strings are arranged in Cannonical Order. e.g. 0, 1, 00, 01, 10, 11, 000 ...)

Tape III : It's the "working tape", corresponding to T_1's tape, where T_2 actually carries out the steps specified by the current string on Tape II.

- Every time T_2 begins trying a new sequence, the third tape is erased and i/p from Tape I copied onto it.

Structure of T_2 in terms of Sub TMs



Sub - TMs

(1) Initialize

- It will prepare tapes II and III.
- Writes symbol '0' in square 1 of tape II.

Tape II
i.e. Δ0

This is the first move to be tried. Square 0 is 'Δ'.

- Writes '#' in square ϕ of tape III. This will prevent T_2 from crashing, as no movement from cell ϕ to the left is allowed.

Tape III
#Δ

(2) Copy Input

Copies I/P string on tape I to tape III, so that tape III ends up with the contents #Δx.

(3) Execute

It simulates action of T_1 for current sequence of moves in tape II.

If T_1 halts, then T_2 halts and puts symbol '*' in current head position in Tape III. Otherwise, it continues with the Erase Tape 3 component.

(4) Erase Tape 3 : It restores tape III to the configuration #Δ.

(5) Next sequence : Updates sequence in Tape II.

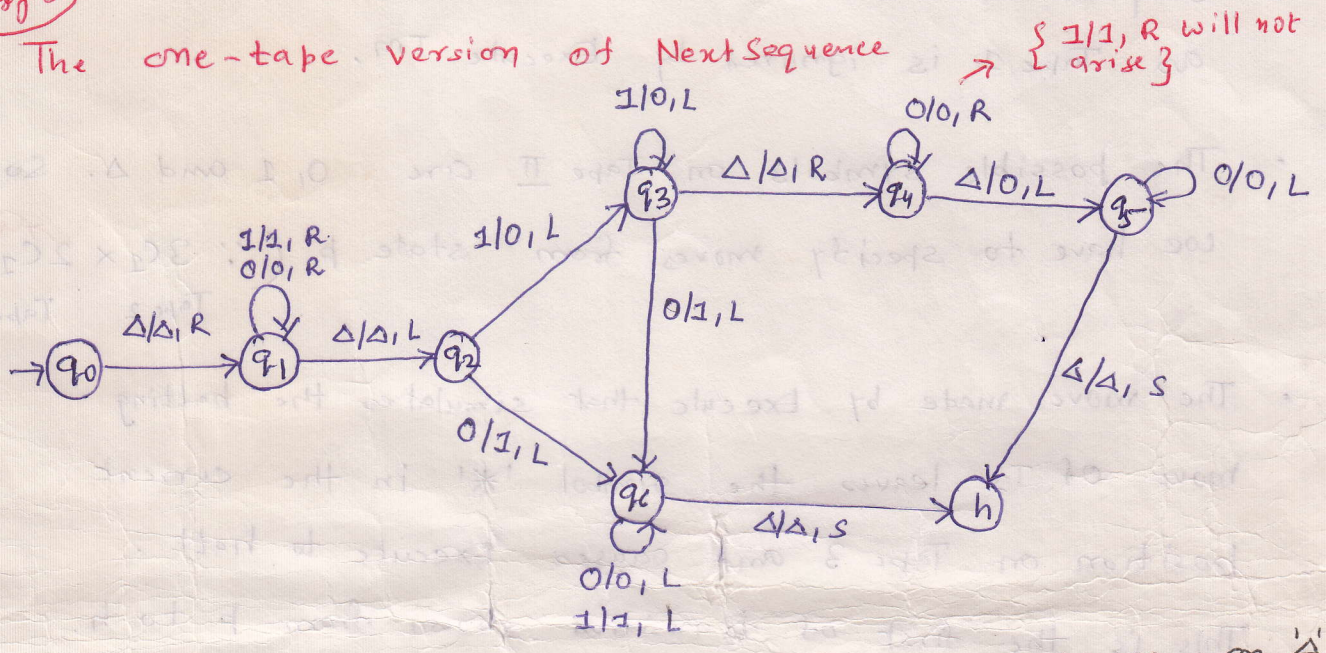
(The next sequence is obtained by interpreting the stored sequence on tape II as a binary no. and adding 1 if the string is not consisting of all 1's. If it is so, then next no. is a string of n+1 0's where n is no. of 1's in previous string)

So, first it is checked if after the move is over, if the current symbol on Tape 3 is #. If it is other than '#', then a move to state q is safe. Otherwise, T_2 halts as in that case T_1 crashes.

page 6B (Continue after last page)

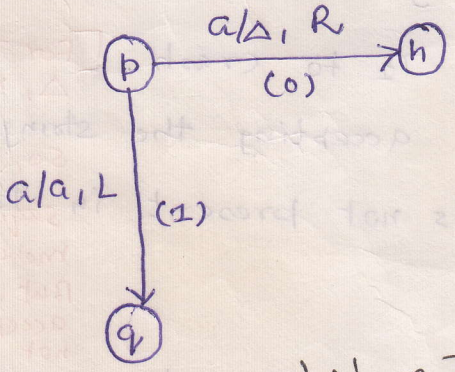
Page 5

The one-tape version of Next Sequence



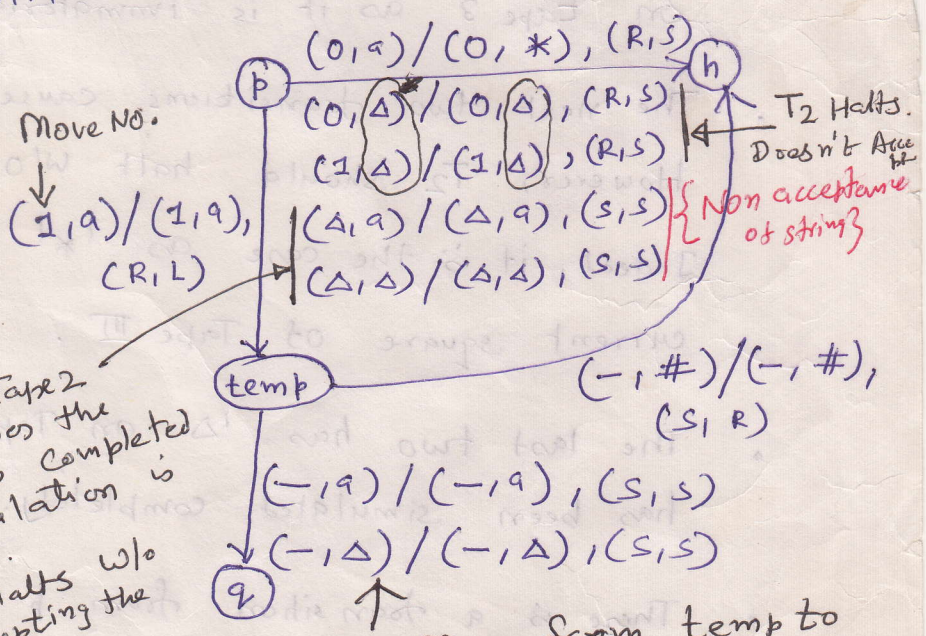
A small portion of Execute

Transition Diagram for NTM



No move on 'Δ' in NTM. ∴ string not accepted.

Transition Diagram for DTM



Here,
For simplicity, we have assumed that

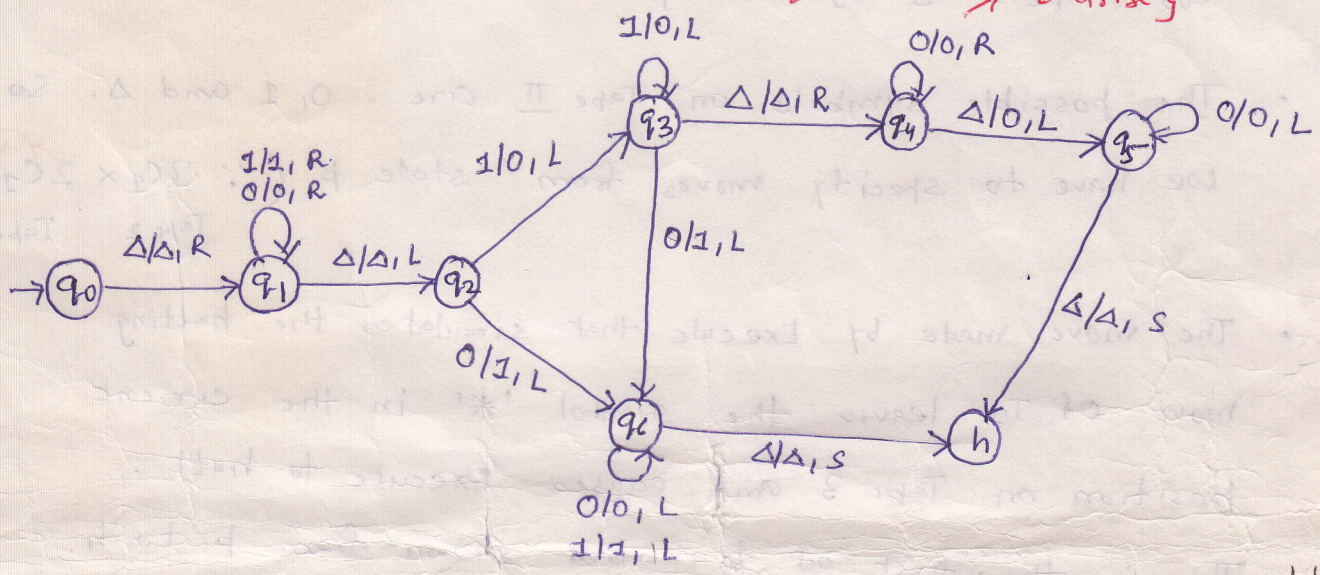
- $\Gamma_1 = \{a\}$, so only alphabets on T_2's tape are 'a' and 'Δ'.
- Only contents ~~on~~ of Tape 2 and Tape 3 of T_2 is shown as Tape 1 is ignored by Execute TM.
- The possible symbols on Tape II are 0, 1 and Δ. So, we have to specify moves from state p ($\because {}^{3}C_1 \times {}^{2}C_1 = 6$; Tape 2: a, Δ; Tape 3)
- The move made by Execute that simulates the halting move of T_1 leaves the symbol '*' in the current position on Tape 3 and causes Execute to halt. This is the first of transition shown from p to h.
  {It ignores the instruction to move head to the right on tape 3 as it is immaterial.}
- The next two transitions cause T_1 to crash. However, T_2 should halt W/O accepting the string. Indeed, it is the case as '*' is not present in current square of Tape III.
  [For the symbol Δ, no move by T_1. So, current sequence of move is simulated. But string is not accepted as '*' is not present on Tape III.]
- The last two has 'Δ' on Tape 2, means the move has been simulated completely.
- There is a transition from p to temp state because T_1 may crash when head is moved from Zeroth position.

So, first it is checked if after the move is over, if the current symbol on Tape 3 is #. If it is other than '#', then a move to state q is safe. Otherwise, T_2 halts as in that case T_1 crashes.

page 6B (continue after last page)

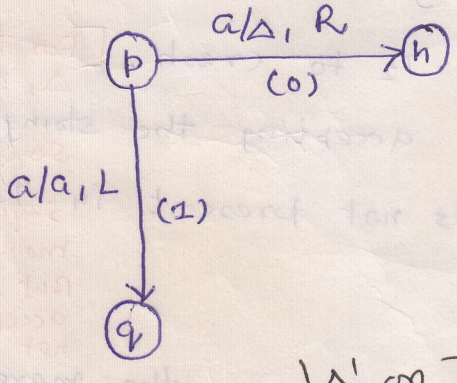
Page 5

The one-tape version of NextSequence



A small portion of Execute

Transition Diagram for NTM



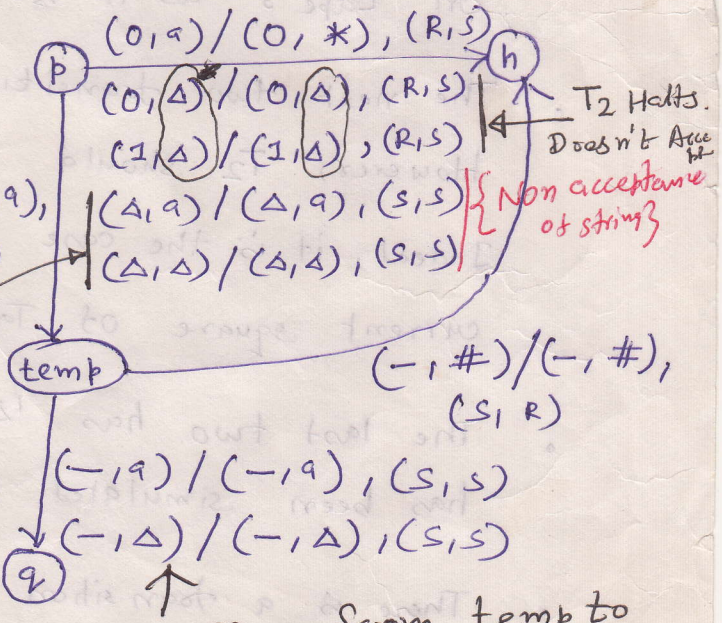
Move No.

(1,a)/(1,a), (R,L)

'Δ' on Tape 2 indicates the move is completed & simulation is over. ∴ T_2 Halts w/o accepting the string.

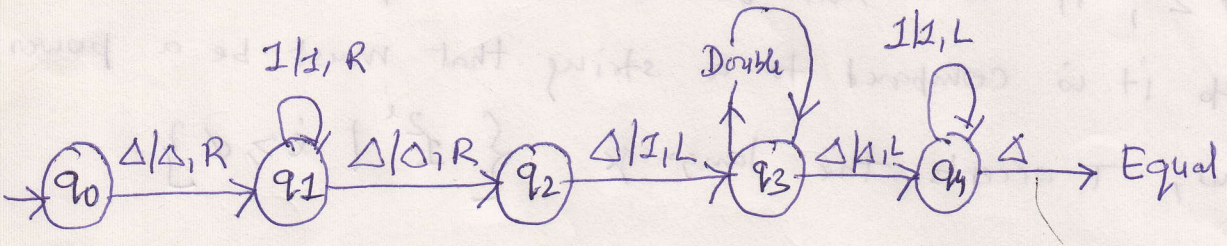
No Move on 'Δ' in NTM. ∴ String not accepted.

Transition Diagram for DTM



Two moves from temp to 'q' as it depends on content of Tape III, which is 'a' and 'Δ' only.

An NTM for accepting strings of length 2^i.



Working of Double :

- It's modified copy for one-symbol Input alphabet.
- It's make a copy of numerical input.
- It then deletes the blank separating the original input from the copy and returns the tape head to square 0.
- In short, it doubles the value of Input, w/o a separating blank.

Working of Equal

Starting with tape contents $\Delta x \Delta y$, where x and y are strings of 1's, Equal accepts if and only if $x = y$.

- The non-determinism lies in the indeterminate no. of times Double is executed.
- When Equal is finally executed, the string following the input on the tape represents some arbitrary power of 2.
- If the original Input string happens to represent a power of 2, say 2^i, then there is a sequence of choices T can make that will cause the input to be accepted - namely, the sequence in which Double is executed

exactly i times.

On the other hand, if the input string is not a power of 2, it will fail to be executed, because the last step it is compared to a string that must be a power of 2.

Thus, T accepts the language $\{1^{2^i} \mid i \geq 0\}$

(Using Deterministic approach, it can be solved. Here, Division will be performed instead of multiplication. Check first, if it is 1 and if not then perform a sequence of divisions by 2.

If at any step before reaching 1, we get a non-zero remainder, we answer no. If we finally obtain the quotient 1 without any of the divisions producing a remainder, the answer is ~~no~~ yes. However, a Multiplication by a 2 is an easier approach than dividing by a 2.

An easier approach therefore might be to start with 1 an perform a sequence of multiplications by 2. We could compare the result of each of these to the original input, accepting if we eventually obtained a no. equal to it, and either rejecting if we eventually obtained a no. larger than the input, or simply letting the iterations continue forever.