COMS3261:
Computer Science Theory

Fall 2013

Mihalis Yannakakis

Lecture 18, 11/11/13

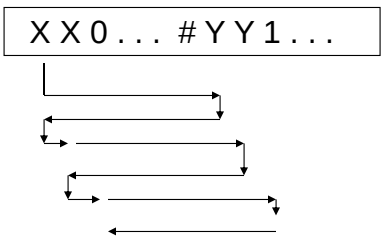
Turing Machine Programming Tricks

1. Structure in state: can have any finite data structure in state, for example a tuple.

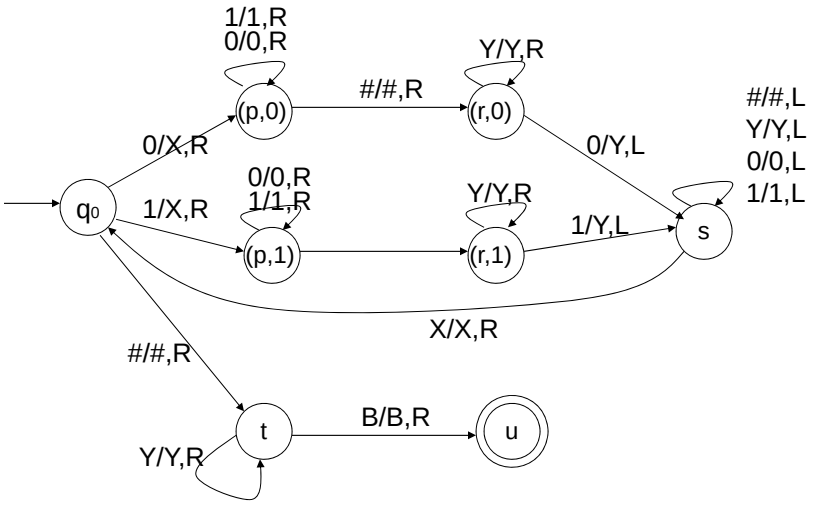
Example: {w#w | w $\in${0,1}* }

Remember in state first uncrossed symbol, travel right, check with first non-Y symbol in 2nd half.

States: (. ,0), (. ,1) remember if bit = 0 or 1



TM for { w#w | w ∈ {0,1}* }



Turing Machine Programming Tricks

2. Structure in tape symbols & tape: can have any finite data structure in tape cell, for example a tuple,

   e.g. Can think of tape as having multiple tracks

Example: {w#w | w ∈{0,1}* } but leave input at the end as is.

Remember in state first uncrossed symbol, travel right, check with first non-Y symbol in 2nd half. States: (q,0), (q,1) …

At the end, restore the original input

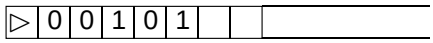
0	0	0	. . .	#	0	0	1	. . .
X	X		. . .		Y	Y		

↑
(0,X)

(0,B) same a 0

Many variations in TM definitions

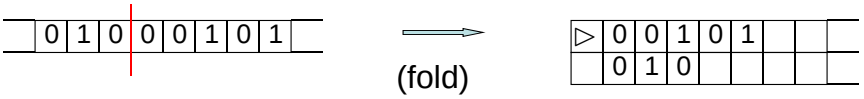
- Different authors often use slightly different definitions
- Some variants:
- Head moves left or right but does not stay in place
- Machine does not ever write the blank symbol B
- Semi-infinite tape instead of two-way infinite tape (with a left endmarker ▷ so head does not fall off tape)



- All equivalent

2-way Infinite tape → semi –infinite tape

- Fold the 2-way infinite tape in two → semi-infinite tape with 2 tracks for the two halfs of the 2-way infinite tape.



Include in the state a bit indicating whether the head of the original TM with the 2-way infinite tape is on the right or the left half of the tape (i.e., whether to read the symbol from the first or the second track)

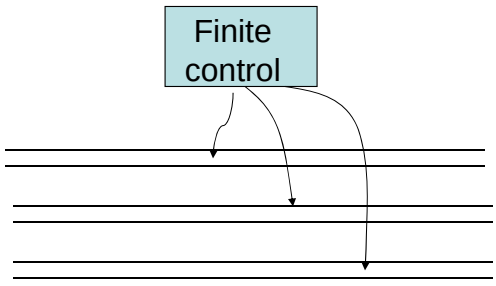
Robustness of Turing machines

- Many other models, more general or less, equally powerful: all recognize the same languages & can compute the same functions
- Multitape TMs
- Nondeterministic TMs
- Multistack machines
- Counter machines
- "Real" computers

Multitape TMs

- k tapes, each with own head
- Initially one has input, other blank
- Transition: $\delta(q,X_1,\ldots,X_k) = (p,Y_1,\ldots Y_k, D_1,\ldots,D_k)$

  $\delta: Q\times\Gamma^k \rightarrow Q\times\Gamma^k\times\{L,R,S\}^k$

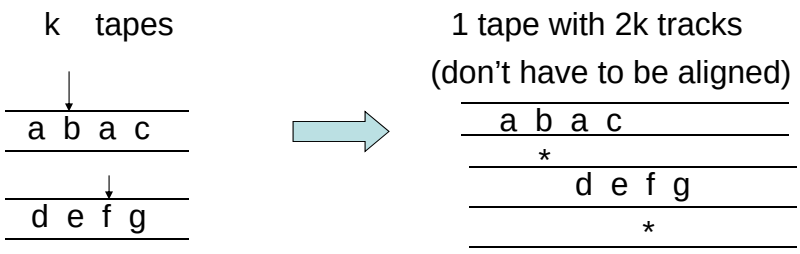


Example of Multitape TM

- L = {x#y | x=y ∈{0,1}* }
- 2-tape TM: Much simpler than 1 tape
  - Traverse x and copy to second tape.
  - After # compare y with x.
- One pass, O(n) steps where n=length of input
- Instead of the 1-tape TM that goes back and forth and spends $O(n^2)$ time.

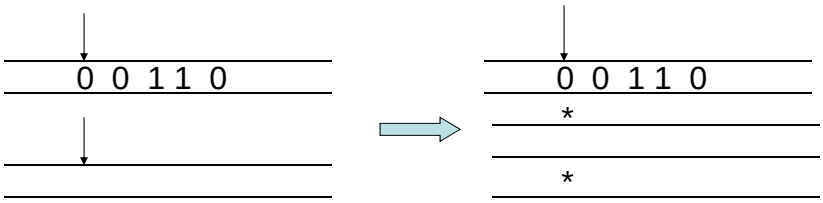
Simulation of multitape TM by 1-tape TM

- M1 = k-tape TM ⇒ M2 = 1-tape TM with 2k tracks
- For each tape of M1, one track of M2 holds tape contents, another track marks the position of the tape head



New tape symbols: 2k-tuples

Initialization



Simulation

- Simulation of 1 move of M1 by M2:

  M2 has head at leftmost * in the current state

1. Move head right and record all tape symbols $X_1,\ldots,X_k$ over the marks (i.e. symbols read by M1) in the finite control (state) of M2
2. Determine move of M1
3. Move left and change for each tape the symbol and the position of * (can do all tapes in one pass, or can do one separate pass for each tape)
4. Leave head at leftmost * and in the next state

Time of Simulation

- Suppose n=input length, and k-tape TM runs for t steps
- Non-blank portion of tape has length $\leq$ n+2t
- Each step of original multitape TM is simulated in time proportional to the length of the non-blank portion, i.e. in time O(n+2t) = O(n+t)

- All the t steps of the multitape TM are simulated in $O(t(n+t)) = O(tn + t^2)$ time by the 1-tape TM

- Quadratic time penalty is unavoidable:

  the language {w#w | w $\in${0,1}* } can be recognized in O(n) time by a 2-tape TM, but 1-tape TM require $O(n^2)$ time