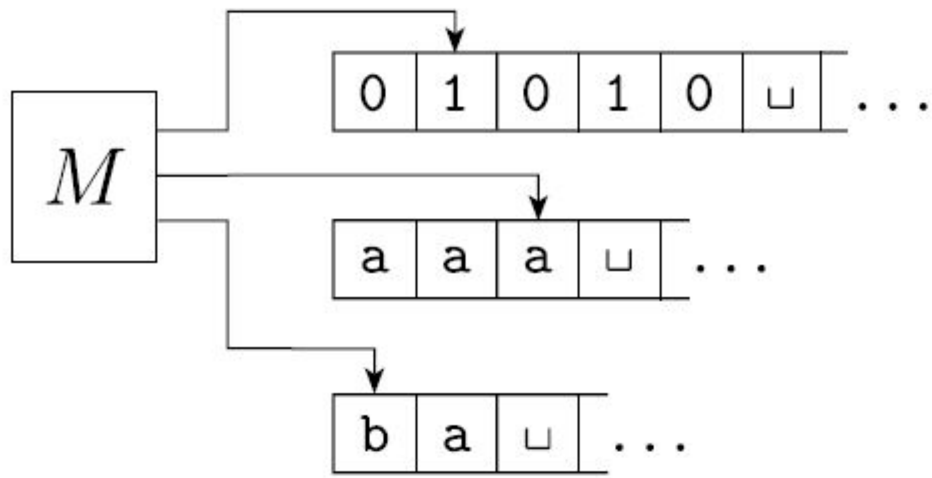
Complexity Theory …

Multi-tape, NTM, …

MULTITAPE TURING MACHINES

A k tape Turing Machine will have k tapes (a 3 tape TM is shown in the figure)



$$\delta\colon Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{\mathrm{L}, \mathrm{R}\}^k,$$

where k is the number of tapes. The expression

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, \mathrm{L}, \mathrm{R}, \ldots, \mathrm{L})$$

THEOREM 3.13

Every multitape Turing machine has an equivalent single-tape Turing machine.

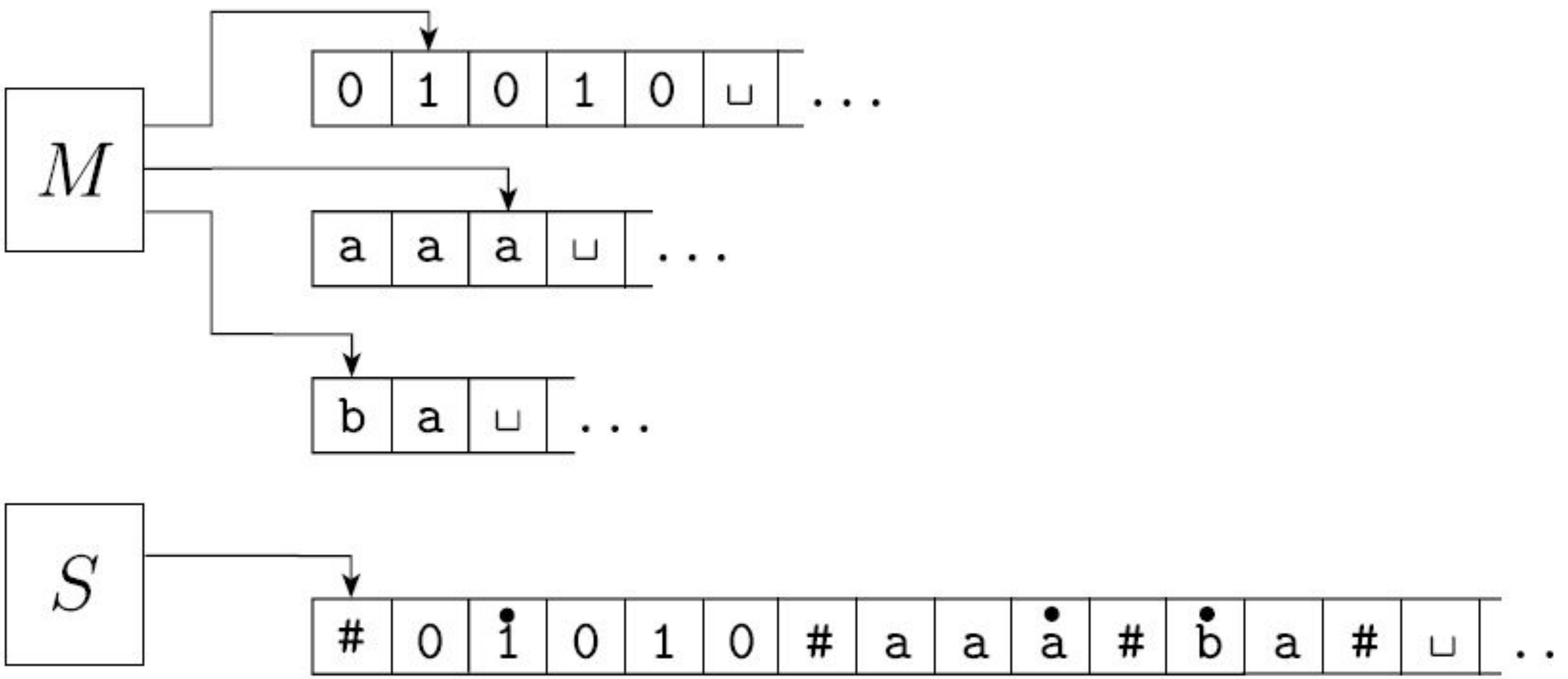


FIGURE 3.14
Representing three tapes with one

Single-Tape versus Multi-Tape

Theorem: Let $t(n)$ be a function, and $t(n) \geq n$.
Then for every $t(n)$-time TM that works with k tapes, there is an equivalent $O(k^2\, t(n)^2)$-time TM that works with 1 tape.

Proof: Let M be a k-tape TM that runs in $t(n)$ time. We construct a single-tape TM S that runs in $O(k^2\, t(n)^2)$ time.

Single-Tape vs Multi-Tape (2)

How the simulation of a k tape TM (M) can be done over a single tape TM (S):

- S uses its single tape to represent the contents of all k tapes in M
- The k tapes are stored consecutively, separated by #
- Positions of tape heads are represented by "marked" symbols

Here, S uses the same way to simulate M

Single-Tape vs Multi-Tape (3)

Recall that to perform a step in M, S will do:

- Scan the tape to collect the characters under each of the tape heads in M
- Scan the tape again, update the symbol under the tape heads of M, and update the positions of the tape heads
- Special case: when a tape head of M moves rightward onto an unread portion, we add a space in the corresponding place in S's tape (by shifting)

Now we analyze this simulation. For each step of M, machine S makes two scans over the active portion of its tape.

What is active portion of the tape?

Time required for a single scan

The length of the active portion of S's tape determines how long S takes to scan it, so we must determine an upper bound on this length

Single-Tape vs Multi-Tape (4)

Since M runs in t(n) time, each of its tape head can access only the first t(n) cells. Thus, S will use (and access) only the first $k \times t(n) + k + 1 = O(k\, t(n))$ cells.

We call these $O(k\, t(n))$ cells the active portion of S's tape

Total Time

- To simulate each of M’s steps, S performs two scans and possibly up to k rightward shifts.
- Each scan takes O(t(n)) time, so the total time for S to simulate one of M’s steps is O(t(n)).
- Now we bound the total time used by the simulation. The initial stage, where S puts its tape into the proper format, uses O(n) steps.
- Afterward, S simulates each of the t(n) steps of M, using O(t(n)) steps, so this part of the simulation uses $t(n) \times O(t(n)) = \mathbf{O(t(n)^2)}$ steps.
- Therefore, the entire simulation of M uses $O(n) + O(t(n)^2)$ steps.
- We have assumed that $t(n) \geq n$ (a reasonable assumption because M could not even read the entire input in less time). Therefore, the running time of S is $\mathbf{O(t(n)^2)}$ and the proof is complete.

Polynomial Time Bounds

If the running time $t(n)$ of a machine M is $O(n^c)$ for some fixed constant $c > 0$, the running time is called polynomial bounded, or we say M runs in polynomial time. This gives the following corollary.

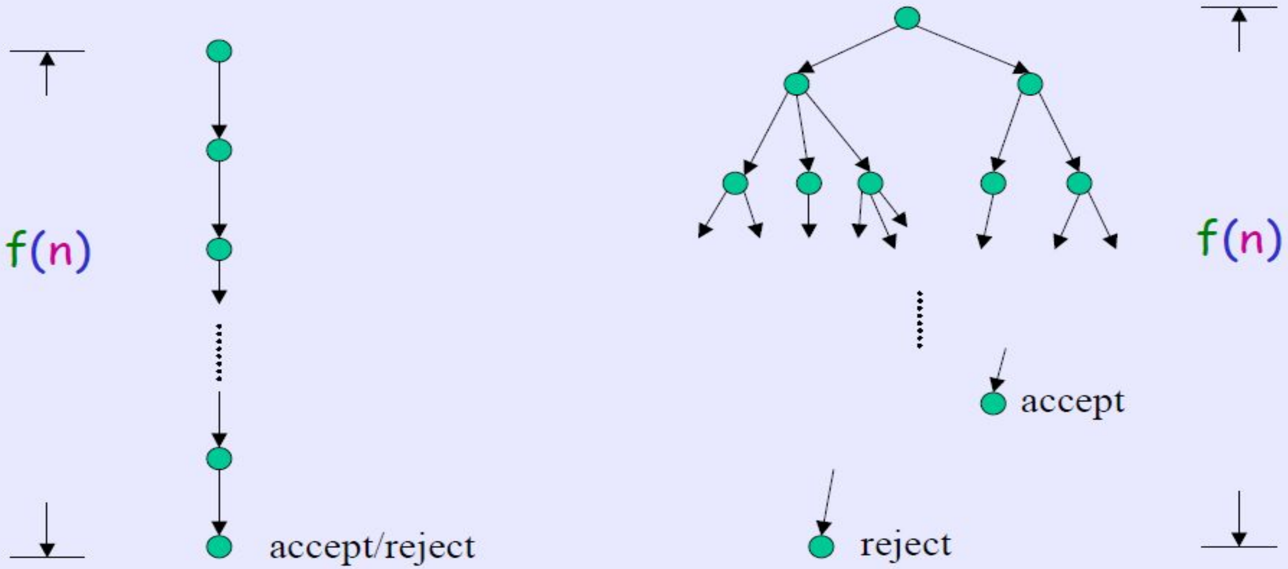
Corollary: For any k-tape TM that runs in polynomial time, it has an equivalent single-tape TM that runs in polynomial time.

NTM decider

An NTM is a decider if all its computation branches halt on all inputs.

Definition: Let M be an NTM decider. The running time of M is the function $f:N \rightarrow N$, where $f(n)$ is the maximum number of steps that M uses on any branch of its computation on any input of length n

Comparison of Running Times



Deterministic time

Non-deterministic time

DTM versus NTM decider

Theorem: Let $t(n)$ be a function, $t(n) \geq n$.
Then every $t(n)$-time single-tape NTM decider has an equivalent $2^{O(t(n))}$-time single-tape DTM

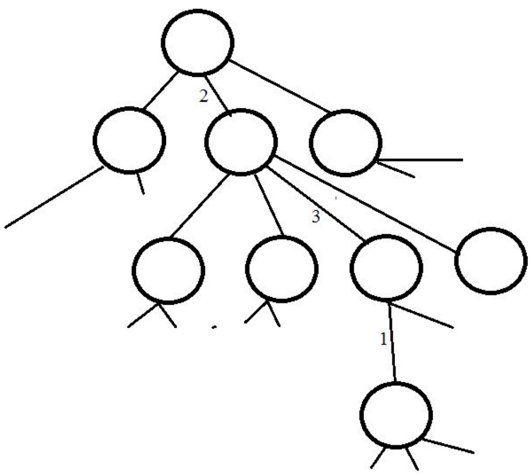
Proof: Let M be a NTM that runs in $t(n)$ time. We construct a DTM D that simulates M by searching M's computation tree, as described in Lecture 11. We now analyze D's simulation.

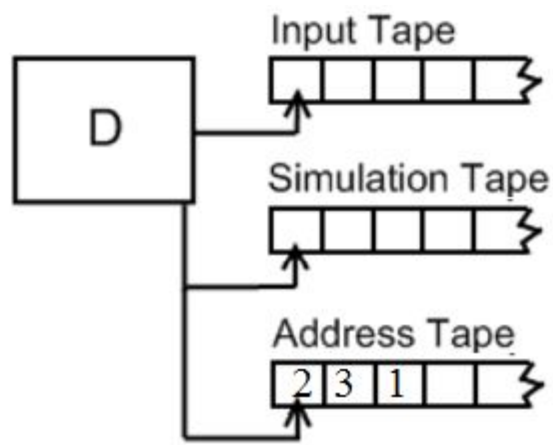
- Ref: Theorem 3.16 of Sipser…

THEOREM 3.16

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.

- The simulation is, in essence …





- Every node in the tree can have at most b children, where b is the maximum number of legal choices given by N's transition function. Thus, the total number of leaves in the tree is at most $\mathbf{b^{t(n)}}$.
- The total number of nodes in the tree is less than twice the maximum number of leaves, so we bound it by $\mathbf{O(b^{t(n)})}$**.**

The simulation proceeds by visiting the nodes (including leaves) in BFS order. Here when we always visits a node v, we always travel starting from the root.

The time it takes to start from the root and travel down to a node is O(t(n)). The time it takes to start from the root and travel down to a node is **O(t(n)).**

Time taken to visit O(t(n)) nodes , i.e.

, the running time of D is $\mathbf{O(t(n)b^{t(n)}) = 2^{O(t(n))}}$

Next Time

- P and NP
  - Two important classes of problems in time complexity theory