Non-Deterministic Turing Machines

A NDTM can choose which of a number of instructions to execute at points in a run..

no program control at run time.

..so a run is not determined in advance - no control via δ-function,

..can have more than 1 entry for (state q_i, symbol read a)

This can give a "free' exhaustive search for a possible solution which is then checked.

We would like to replace this non-deterministic choice with a clever search strategy to select a suitable possible solution, and then check it in p-time.

The choice is done by **multiple entries for $(q,a) \in Q \times \Sigma$.**

Formal definition of NDTM

$$N = (Q, \Sigma, I, q_0, \delta, F), \text{ with}$$

$$\delta : (Q \setminus F) \times \Sigma \to 2^{Q \times \Sigma \times \{\pm 1, 0\}}$$

- $: 2^{Q \times \Sigma \times \{\pm 1, 0\}}$: the set of all subsets of $Q \times \Sigma \times \{\pm 1, 0\}$

ie. the function value for (q,a) is a set of the alternatives

If there is no applicable instruction for (q,a) then $\delta(q,a)=\emptyset$ (empty set).

if δ(q,a) contains only one (q',a',d) or is empty, we have an ordinary deterministic TM.

Input and Output for NDTMs

Input: tape contents before the first ^ as before

Output: can have many different results on tape for an input w - for each successful (H & S) run

$\Rightarrow$ a set of possible outputs.

$\Rightarrow$ simplify by considering only yes/no problems

- **acceptance** for NDTMs:
  N **accepts** $w \in I^*$ if there exists a successful run with w as input:
  -..if at least one run on w Halts and Succeeds.

- **rejection** for NDTMs
  -N **rejects** $w \in I^*$ if all runs on input w Halt and Fail.

Speed of NDTMs

For any input $w \in I^*$ the NDTM can

1) Halt and Succeed (i.e. accept w)

2) Halt and Fail (i.e. reject w)

(3) Never halt..excluded when N ***solves*** the problem).

Run-time function for NDTMs:

$time_N$ (n) = length of longest run for any $w \in I^*$ of length n.

$time_N (n) \leq$.

We say that a NDTM N runs in polynomial time if

$\exists$ a polynomial p such that

$$\mathbf{time_N(n) \leq p(n), \text{ all } n > 0}$$

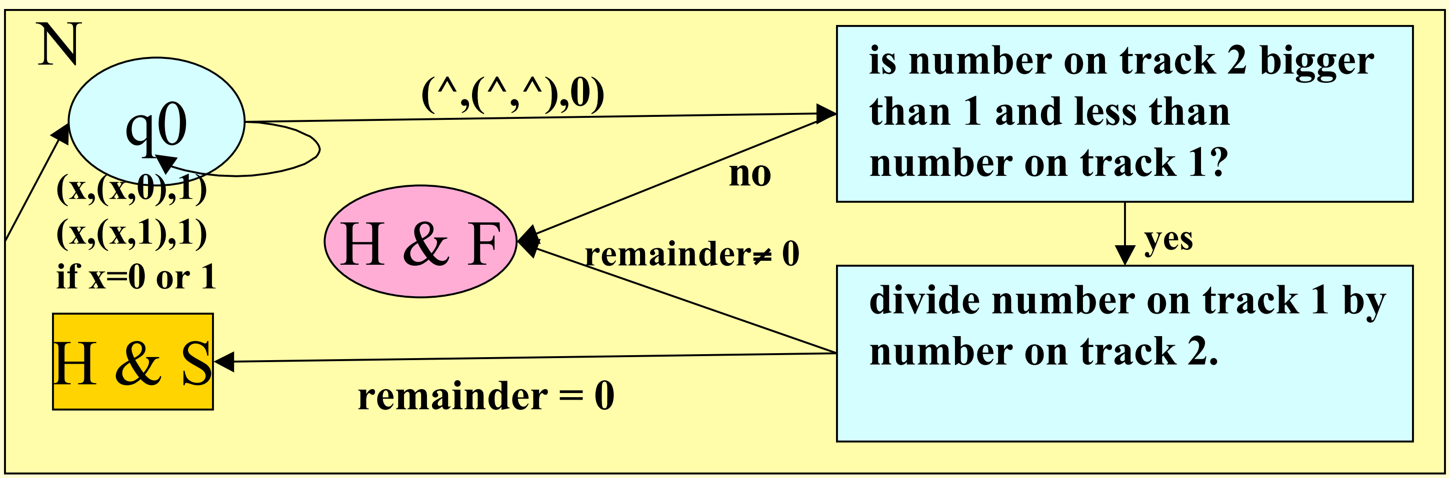
p(n) is an upper bound on time for inputs of length n.

Example of a Non-Deterministic Turing Machine

Non-primality testing. Given a number n, is n composite?

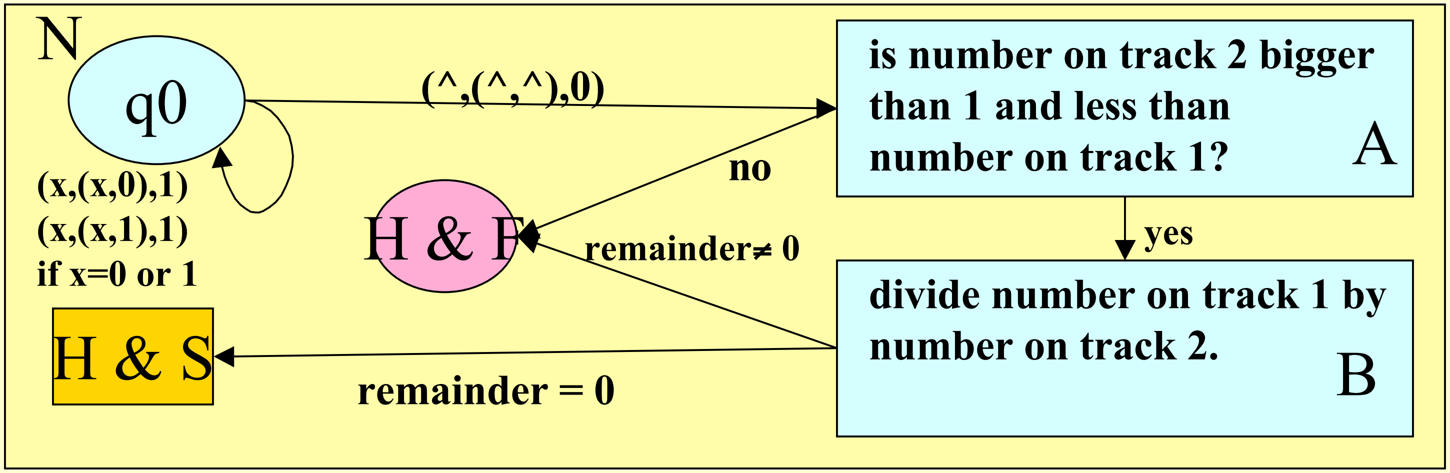
We build a 2-track NDTM which:

1. Guesses a number m, where 1 < m < n.
2. Divides n by m (deterministically in p-time)
3. If there is no remainder (m divides n) then it Halts and Succeeds: “yes” otherwise it Halts and Fails: “no”.

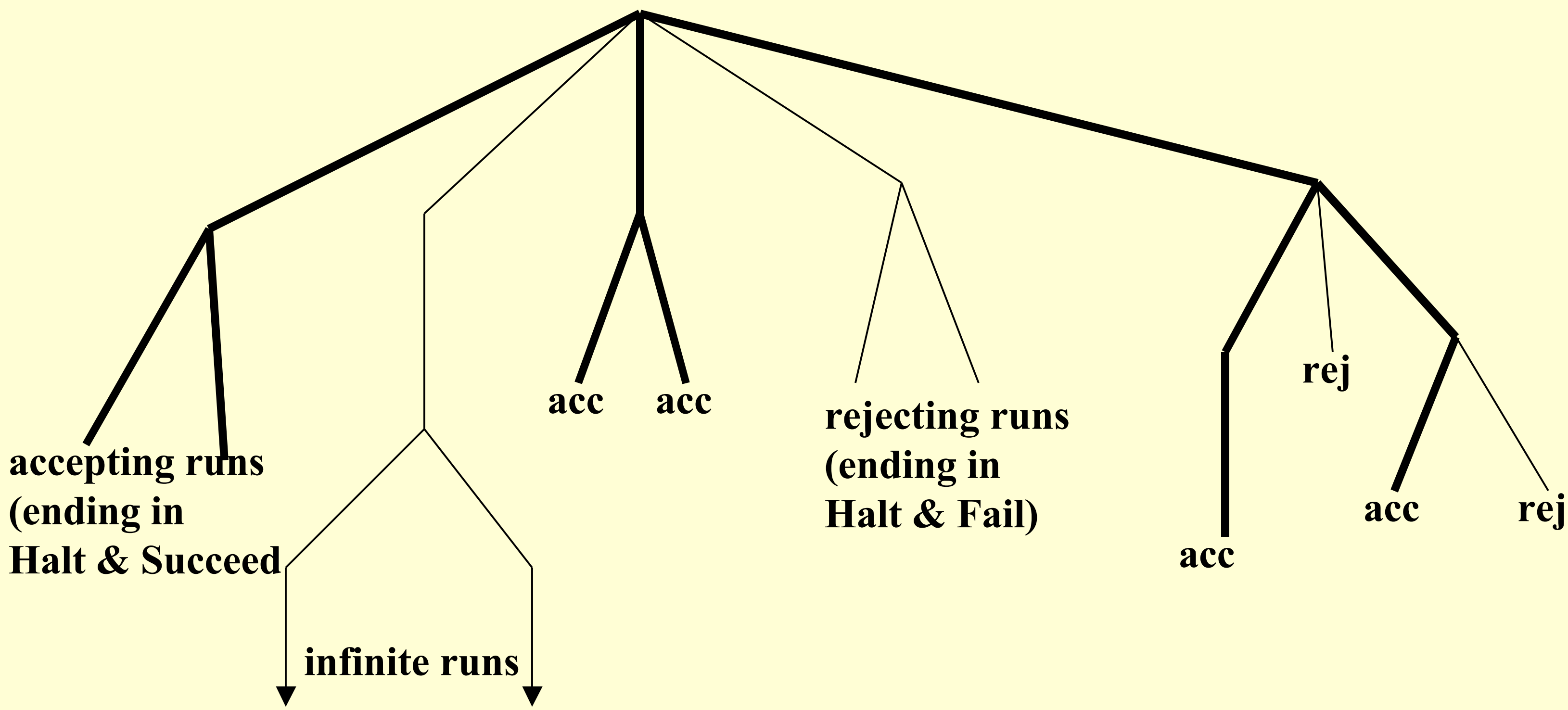


Example of a Non-Deterministic Turing Machine

- the input is a binary number on a single track
- N non-deterministically writes a binary number, m on track 2.
  ending at the end of n.
- N checks whether m = 0, or m= 1 or m ≥ n: if so N Halts and Fails
  if not, N divides n by m and checks whether the remainder> 0
  ..if not, N Halts and Succeeds
  otherwise N Halts and Fails



NDTM N



tree representation of the possible runs of NDTM, N on some input

The Class NP of problems

NP consists of all yes/no problems A such that there is some **NDTM N that runs in p-time and solves A;**

N accepts all the yes-instances of A
rejects all the no-instances of A.

This is the class of ($\exists$) type problems that would be in P if they had a clever search strategy.

P $\subseteq$ NP as p-time deterministic TMs are a special case of p-time non-deterministic TMs.

P = NP? .. yes/no problem not yet answered

Simulation of NDTMs by ordinary TMs.

..we demonstrate that a NDTM is no more powerful than an ordinary TM..further evidence for the Church-Turing Thesis.

..we have seen the non-determinism as a way of guessing from the search space of possible solutions..and then checking deterministically whether this ***is*** a solution..

Equivalence of ordinary TMs and NDTMs:

1) given an ordinary TM $M = (Q, \Sigma, I, q_0, \delta, F)$ we can construct an equivalent NDTM $N = (Q, \Sigma, I, q_0, \delta', F)$ such that:

$$\delta'(q, a) = \{\delta(q, a)\} \text{ if } \delta(q, a) \text{ is defined}$$
$$\varnothing \text{ otherwise.}$$

N behaves exactly as M - deterministically,
but it is a valid NDTM.

Equivalence

2)any yes/no problem solvable by a NDTM can also be solved by an ordinary deterministic Turing Machine.

Given a NDTM N □ construct a deterministic TM to solve the same Problem.

We will simulate N with a 3-tape ordinary TM, M:

M rejects/accepts the same input words

M does breadth-first traversal of the tree of possible runs of N for given input w.

M Halts and Succeeds when it finds a Halting state of N.

why a breadth-first tree traversal?

-we want to find any H & S

- we don't want to go down infinite (non-terminating) branches.

At each node of the search tree, N has a **configuration:**

- current state of N
- current contents of N’s tape
- position of read head of N

which determines what N does next..

..the possible nodes at the next level of the tree

Representation of the configuration of N:

s * t

word representing the State of N

A “ 2-track word” giving the Contents of N’s tape and the current Position of N’s head.

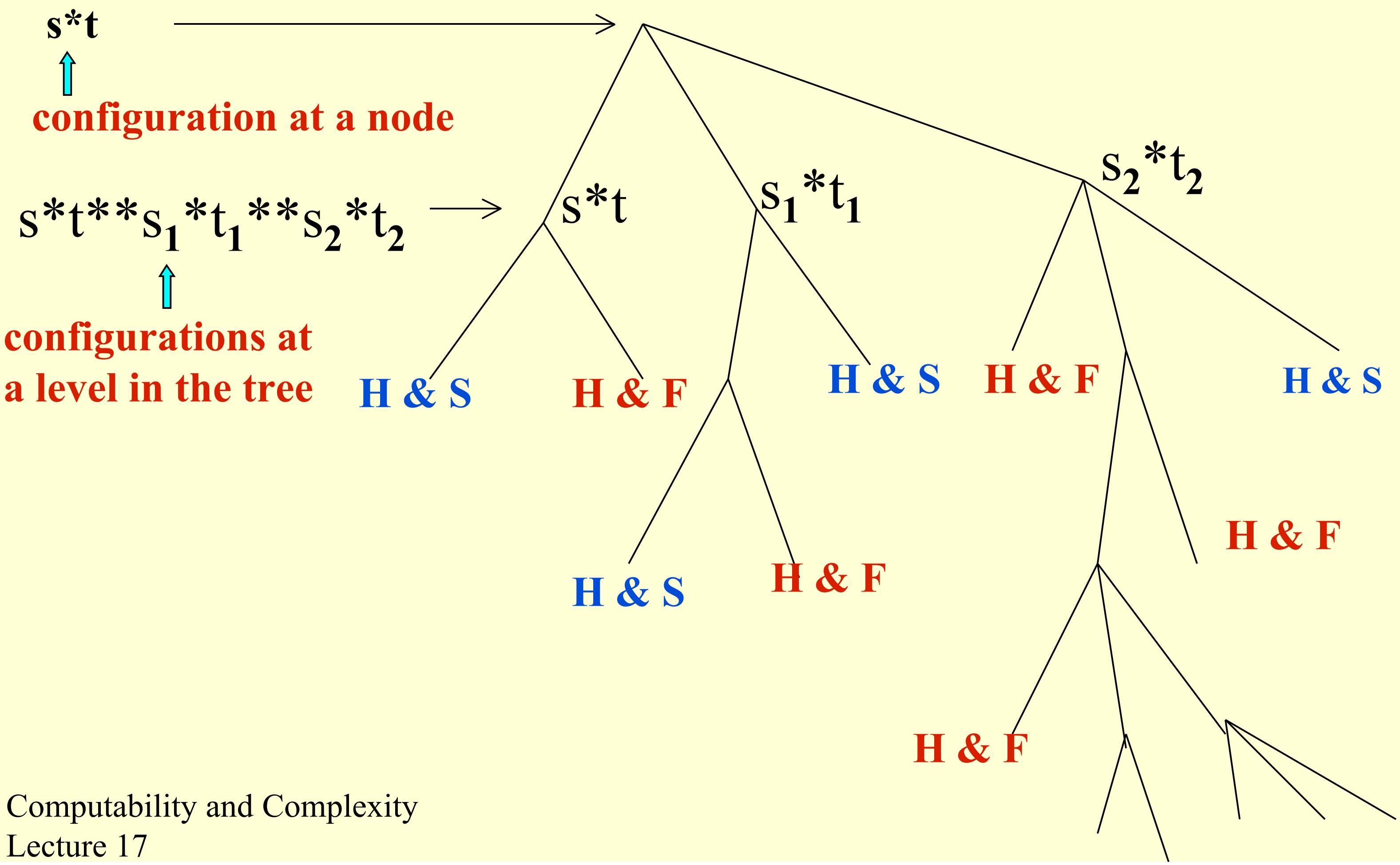
current position of N’s head

Contents of N’s tape^^^

X

We can describe a level of the run-tree of N by the set of possible configurations after the corresponding number of steps in N's run.

N's run tree has levels:



M simulates N by building up the possible configurations of N, level-by-level: at first N has input word w $\Rightarrow$ M starts with w on tape 1

$\Rightarrow$ replace w by config(q_0, w, 0)

tape 2 is used for scratch work

Breadth-first search for Halt and Succeed:

after n cycles (levels of the run tree):

M has $s*t**s_1*t_1**s_2*t_2**s_3*t_3..**s_r*t_r$ on tape 1

configurations for nodes at level n

M follows N:

- look for a Halting State of N among the configurations at level n
  it there is one $\Rightarrow$ M Halts and Succeeds
- check each s_i*t_i ..each node at level n, calculate possible nodes at level n+1..(ignore if this involves move left from sq 0)
  ..add the new configuration to the end of tape 2 of M
  if no child nodes, no change to tape 2.

M now has:

configurations of nodes at level n

Tape 1 $s*t**s_1*t_1**s_2*t_2**s_3*t_3**s_4*t_4**$

configurations of nodes at level n+1

Tape 2 $u*v**u_1*v_1**u_2*v_2**u_3*v_3**u_4*v_4**u_5*v*_5*u_6*v_6$

scratch work

Tape 3

When all level n+1 configurations are on tape 2,
copy Tape 2 to Tape 1 and repeat for level n+1 on tape 1, building Level n+2 on Tape 2.

The cycle is repeated for level n+1.
if Tape 2 is empty at the end of a cycle

no child nodes

Halt and Fail

Success/ Failure:
N accepts w □ □ some successful run of N on w
□ somewhere in the tree □ config(q, w, m), **q □ F.**
M will find this, and Halt & Succeed.

N rejects w □ every run of N on w □ Halt and Fail
□ tree has finite depth, say n
level n+1 is empty □ Halt and Fail.

N and M solve the same yes/no problem.

NDTMs are equivalent to ordinary TMs

Summary

We have defined **Nondeterministic Turing Machines**

..and shown that they are equivalent to ordinary deterministic TMs, by showing that

1) An ordinary TM is a special case of a NDTM
   (δ(q, a) has one entry or is the empty set)

2) We can build an ordinary TM, M which searches the run-tree of a NDTM N and: Halts and Succeeds when it finds the first (w.r.t distance in levels from the start of the run) possible halting state(there is an accepting run of N on w)
   or: Halts & Fails if it has explored the whole tree without finding a halting state.

The complexity class NP:
those yes/no problems which can be solved in p-time by a NDTM.