Multiple tracks on single tape

input

%	0	1	1	0	$	
⊔						
⊔						

↑

Each symbol is a k-tuple (here, k = 3)

Ex. **Construct M to accept if input is a binary representation of a prime number n**

(n is prime if has no other factors exept n and 1 that divide it evenly)

Procedure: Check to see if 2, 3, ... n-1 divide n; if any do, reject. if none, accept.

How use tracks?

Multiple tracks on single tape

input

1	%	0	1	1	0	$	
2	⊔						
3	⊔						

↑

On track 2, cycle through numbers m, starting with 2 and ending with n-1.
For each such m, use track 3 to see if n is divisible by m: start by copying track 1 to track 3. Then repeatedly subtract m on track 2 from track 3, until you get a remainder on track 3 which is less than m. If remainder = 0, then m divides n, and so M rejects. If remainder is not 0, add 1 to m to obtain the new m, and continue, till get to n.

How do subtraction? test if one number less than another?

Shifting Symbols

TM can make space on its tape by shifting all non-blank symbols to right. To do so, use state to store symbol that is being moved, till get to deposit location (e.g., $\sqcup$) deposit the symbol, and move left again.

Ex. TM routine to move symbols 2 places to right uses states {q1, q2} x Γ x Γ, new symbol X

1. δ **((q1,$\sqcup$, $\sqcup$) A) = ((q1,$\sqcup$, A), X, R)**

(Store first symbol read in last state component, replacing it by X, and move Right)

2. δ **((q1,$\sqcup$, A) B) = ((q1, A ,B) , X, R)**

(Store new symbol read in last component, replace by X, symbol in last component to the middle, and move Right)

Shifting Symbols

3. δ **((q1,A ,B) , C) = ((q1,B ,C), A, R)**

(Store symbol read in last state component, shifting B to middle, and deposits A 2 spaces to right)

4. δ **((q1,A ,B),$\sqcup$) = ((q1,B ,$\sqcup$), A, R)**

(When get to blank, stored symbols are deposited in order)

5. δ **((q1,A ,$\sqcup$),$\sqcup$) = ((q2, $\sqcup$,$\sqcup$), A, L)**

(When all symbols deposited, got to state q2 and move left to find X, the rightmost vacated space)

6. δ **((q2,$\sqcup$, $\sqcup$) X) = ((q2, $\sqcup$,$\sqcup$) , X, L)**

(T moves left, until a X is found. At that point, T will transfer control to a different state)

Ex. 010

Simulation

Suppose M with input w goes from C1, C2, ..., Cn

M' *simulates* **M if on input w, M' goes through a sequence of configurations representing C1,...,Cn**

(Note that M' can enter other configurations in between)

M' must be able to

1. **calculate rep. of Ci+1 from rep. of Ci**
2. **determine if M accepts on Ci using rep. of Ci.**

Ex. M from M1, M2 with $L(M) = L(M1) \cap L(M2)$

Simulate M1 and M2 on track 2.

%	0	1	1	0	$

1. *Copy input onto track 2. Simulate M1 on track 2. if M1 accepts, then M goes to step 2 ; ow, reject.*
2. *Copy input onto track 2. Simulate M2 on track 2. if M2 accepts, then M accepts; ow, reject.*

Subroutines

Somewhat like simulation, but "separate" machines

M' *subroutine* **of M:**

M', M should have differents states

To call M', M should enter the start state of M' and follow the transitions of M'

From a halting state of M', M reenters state of M, and proceed

input parameters to M' should be in fixed place; output of M' as well.

Ex. Shifting symbols as subroutine

Ex. arithmetic subroutines

Variants of TM's

Many different ways to define TM model:

multiple tapes, 2-way tapes, nondeterminism

Our model, and all reasonable variants, have same power (i.e., accept same class of languages)

So the model is *robust*

Ex. $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R,S\}$ *(S for stay)*

Is this class of TM more powerful? NO!

Given any TM M' in extended class, can define

a regular TM M that does 2 moves (R,L)

for every S move of M', and acts exactly like

M' otherwise

Two-way tapes

Lemma: If L is accepted by a TM M with a 2-way tape, then L is accepted by a TM M' with 1-way tape.

Proof idea: Use 2 tracks:

A_0	A_1	
\$	A_{-1}	

for single 2-way:

... A_{-1} A_0 A_1 ...

M′ must keep track of whether scanning symbol on top or bottom track, build into state:

$Q' = \{q1\} \cup \{Q \times \{U,L\}\}$ $\Gamma' = \{ [x,y] \mid x, y \text{ in } \Gamma \text{ or } y = \$\}$

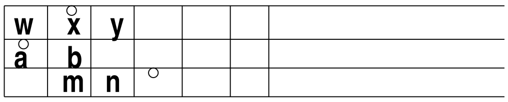
$\Sigma' = \{ [a,\sqcup] \mid a \text{ in } \Sigma \}$

Multiple tapes

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

Lemma: If L is accepted by a multitape TM, then L is accepted by a single tape TM.

Proof idea: use tracks, with special markers° for each tape head



M′ will simulate M by making multiple passes over tape, 1 for each track. First, it needs to determine all the symbols being read, in order to determine the next move. So k symbols must be stored in state. Then, once the next move is determined, it must update each track with new symbol and new tape head position.

Nondeterministic TM's

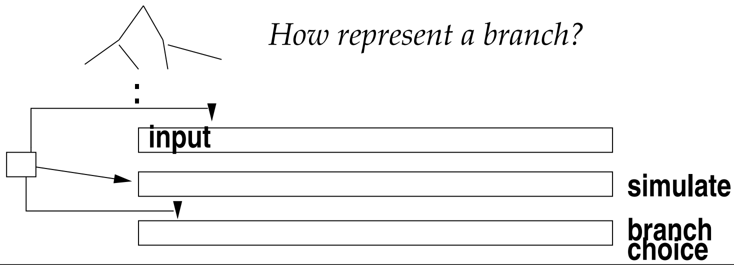
$\delta: Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L,R\})$

Lemma: If L is accepted by a nondet. TM N, then L is accepted by some deterministic TM D.

Proof idea: D will simulate N by trying all possible branches of N's computation on input w. If D finds an accept state on some branch, it accepts. OW, D does not accept.

QU: How should D search the computation tree??

How represent a branch?



Robustness of TM's

Is this surprising??
TM's ⟷ "Idealized" programming languages

Programming languages, if sufficiently general, can do the same computations

Ex. C, Lisp, Java

TM's make formal our informal notion of algorithm

Church-Turing Thesis: The Turing machine model (and any reasonable variant of it) embodies our informal notion of algorithm.

Can't be proved!