Computing Theory

COMP 147 (4 units)

Chapter 3: The Church-Turing Thesis
Section 3.2: Variants of Turing Machines

Alonzo Church and Alan Turing

Image: http://geopolicraticus.wordpress.com/2012/11/23/the-church-turing-thesis-and-the-asymmetry-of-intuition/

The Church-Turing Thesis (1936)

"computability" ⇔ λ-calculus ⇔ Turing machines

Hilbert's 10th Problem (1900)

- Devise a process of finite steps that tests if a polynomial has an integral root
  - (Hilbert didn't actually use the word "algorithm")
- Issues at the time:
  - What is an "algorithm"?
  - Can non-existence of an algorithm be demonstrated?
  - If we don't understand precisely what an algorithm is, how can we argue about whether one exists for a particular problem?

Defining Algorithms

- Turing and Church independently came up with definitions of an algorithm

- Church: an algorithm is anything that can be expressed in the λ-calculus

- Turing: an algorithm is anything that can be performed by a Turing machine

- Gödel: an algorithm is anything that can be expressed by primitive recursive functions*

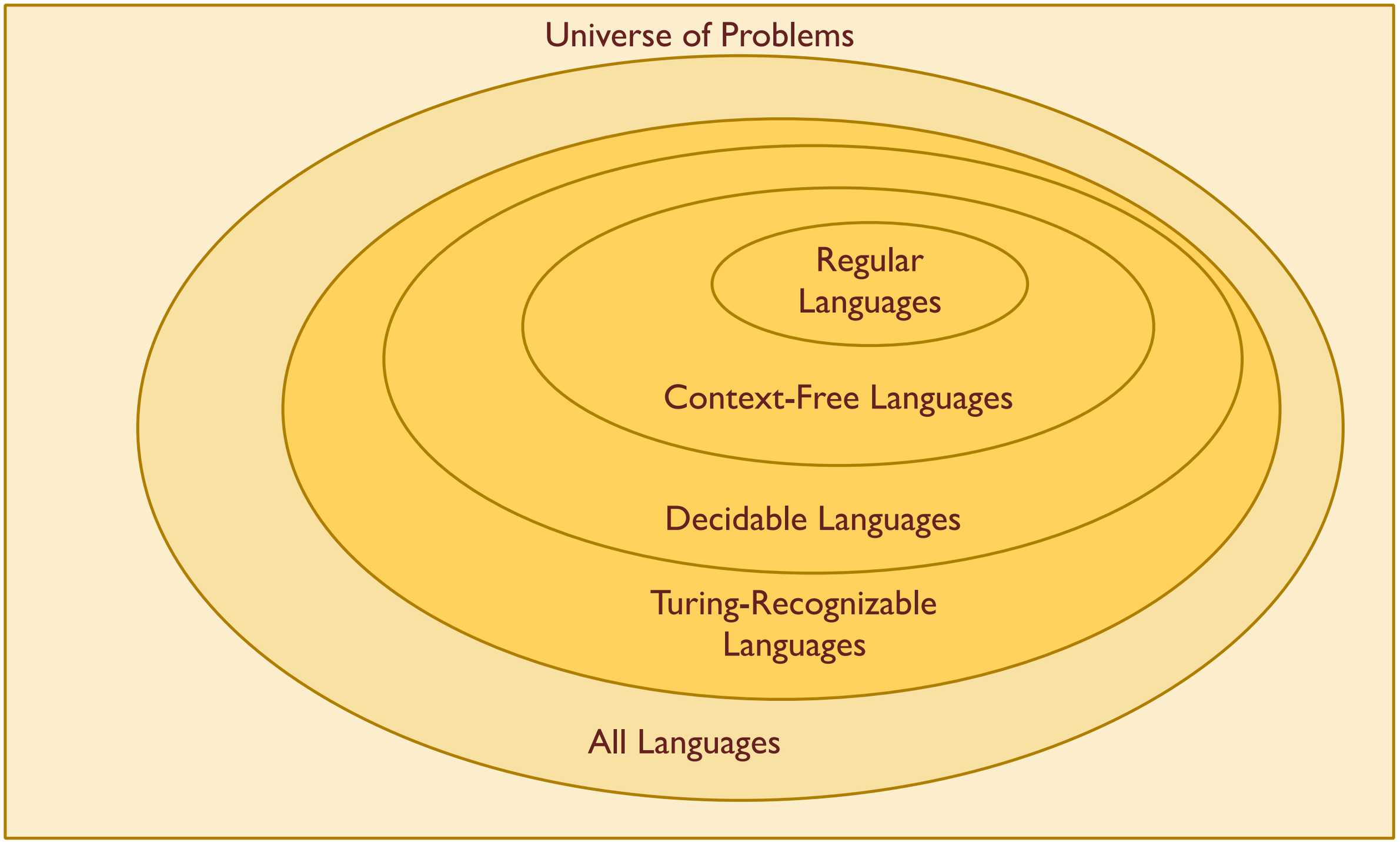
*This may be slightly inaccurate terminology.

Defining Algorithms

- Importantly, all the preceding notions of an algorithm were proven to be equivalent
  - They all state that the same class of problems is “effectively computable” in the sense that they have an algorithmic solution
  - The fact that they are all equivalent is *strong evidence* that they are correct

- Effectively computable ⇔ an algorithm exists ⇔ Turing-decidable

The Space of Problems



Recognizing vs. Deciding

Turing-recognizable: A language L is "Turing-recognizable" if there exists a TM M such that for all strings w:

- If $w \in L$: eventually M enters q_{accept}.
- If $w \notin L$: either M enters q_{reject} ***or*** M never terminates.

Turing-decidable: A language L is "Turing-decidable" if there exists a TM M such that for all strings w:

- If $w \in L$: eventually M enters q_{accept}.
- If $w \notin L$: eventually M enters q_{reject}.

More About Turing Machines

Church Turing Thesis
“Bells and Whistles”
Restrictions
Extensions

Varieties of TMs

What if we allow multiple tapes?

What if we allow nondeterminism?

What if the TM explodes?

Variants of TM models

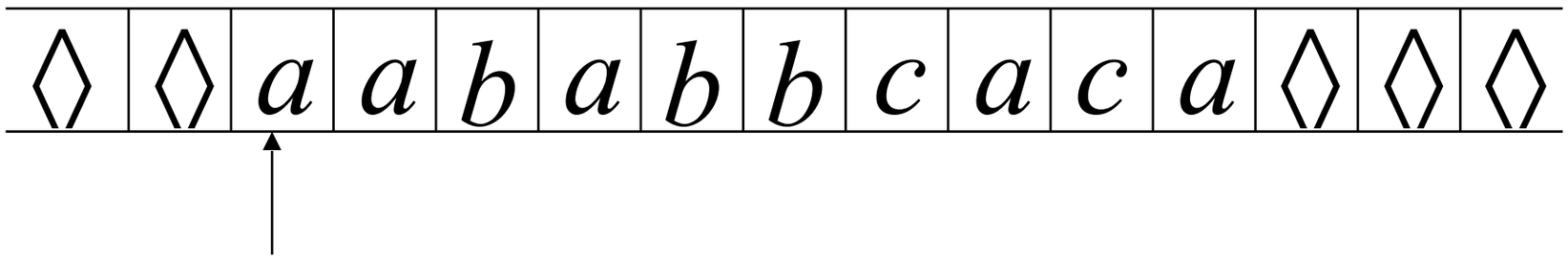
- Turing Machines that can stay (directions L, R and S)
- Multiple Tape Turing Machines
- Nondeterministic Turing Machines

Surprise!
All TMs are born equal…

- Each of the preceding TMs is equivalent to the standard TM
  - ➪ They recognize the same set of languages (the Turing-recognizable languages)
- Proof idea: Simulate the "deviant" TM using a standard TM

Turing Machines with Stay-Option

The head can stay in the same position

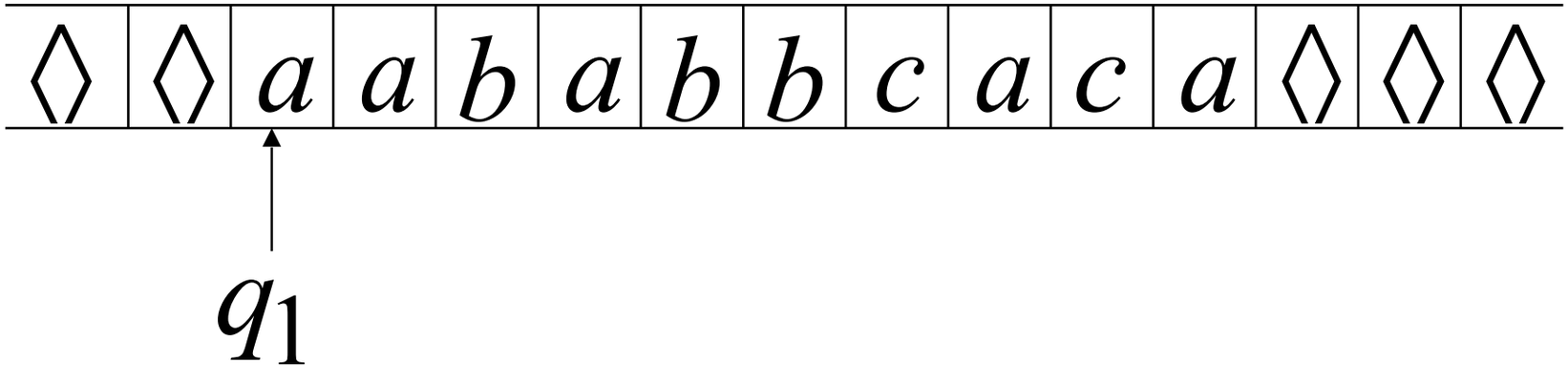


Left, Right, Stay

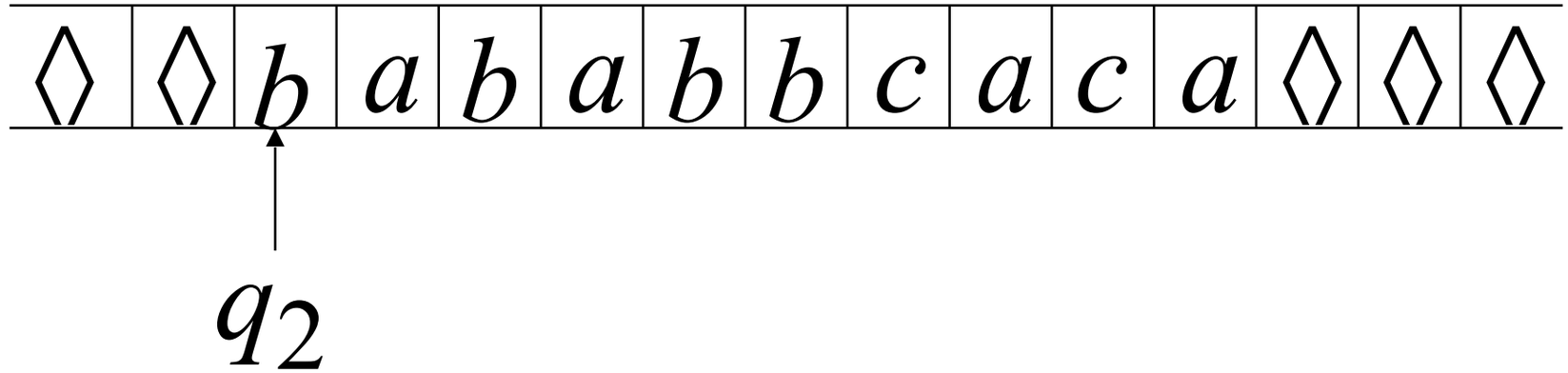
L,R,S: possible head moves

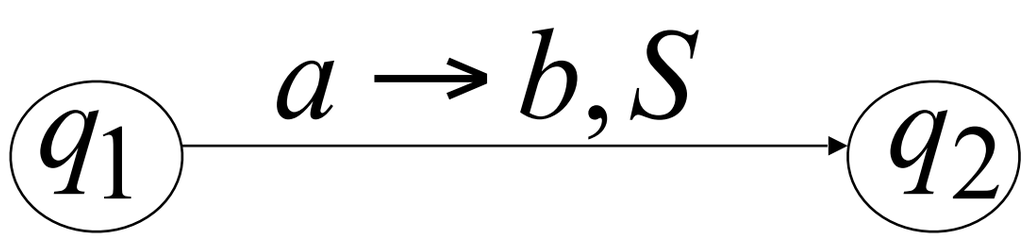
Example:

Time 1



Time 2





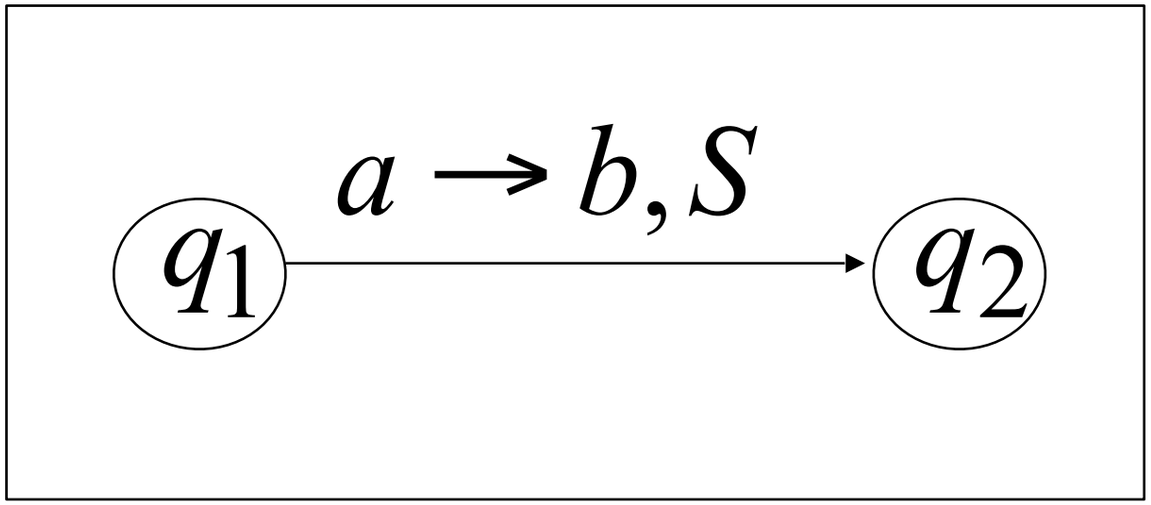
Standard Turing machines

simulate Stay-Option machines

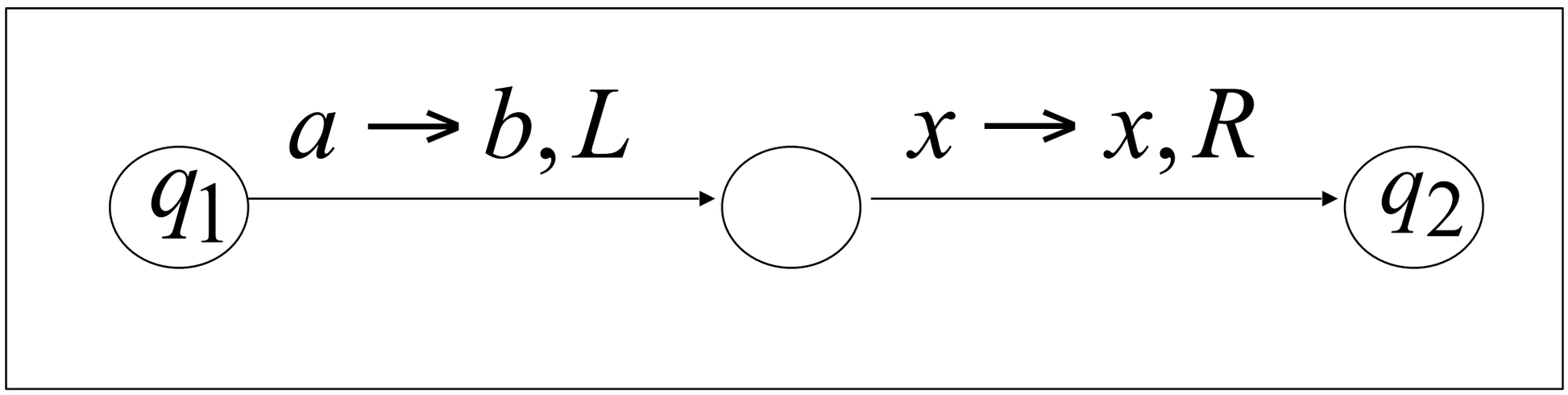
We need to simulate the stay head option

with two head moves, one left and one right

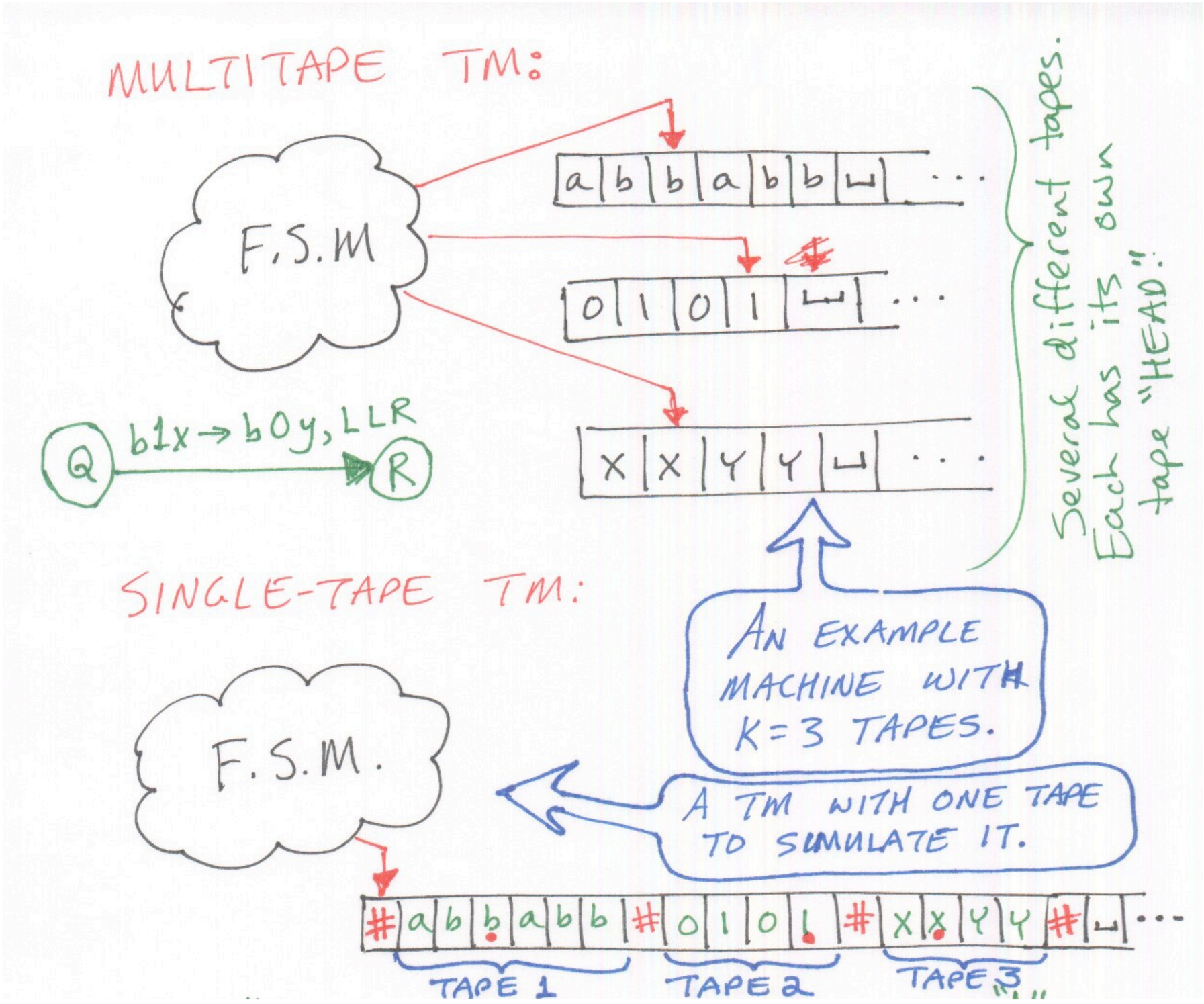
Stay-Option Machine



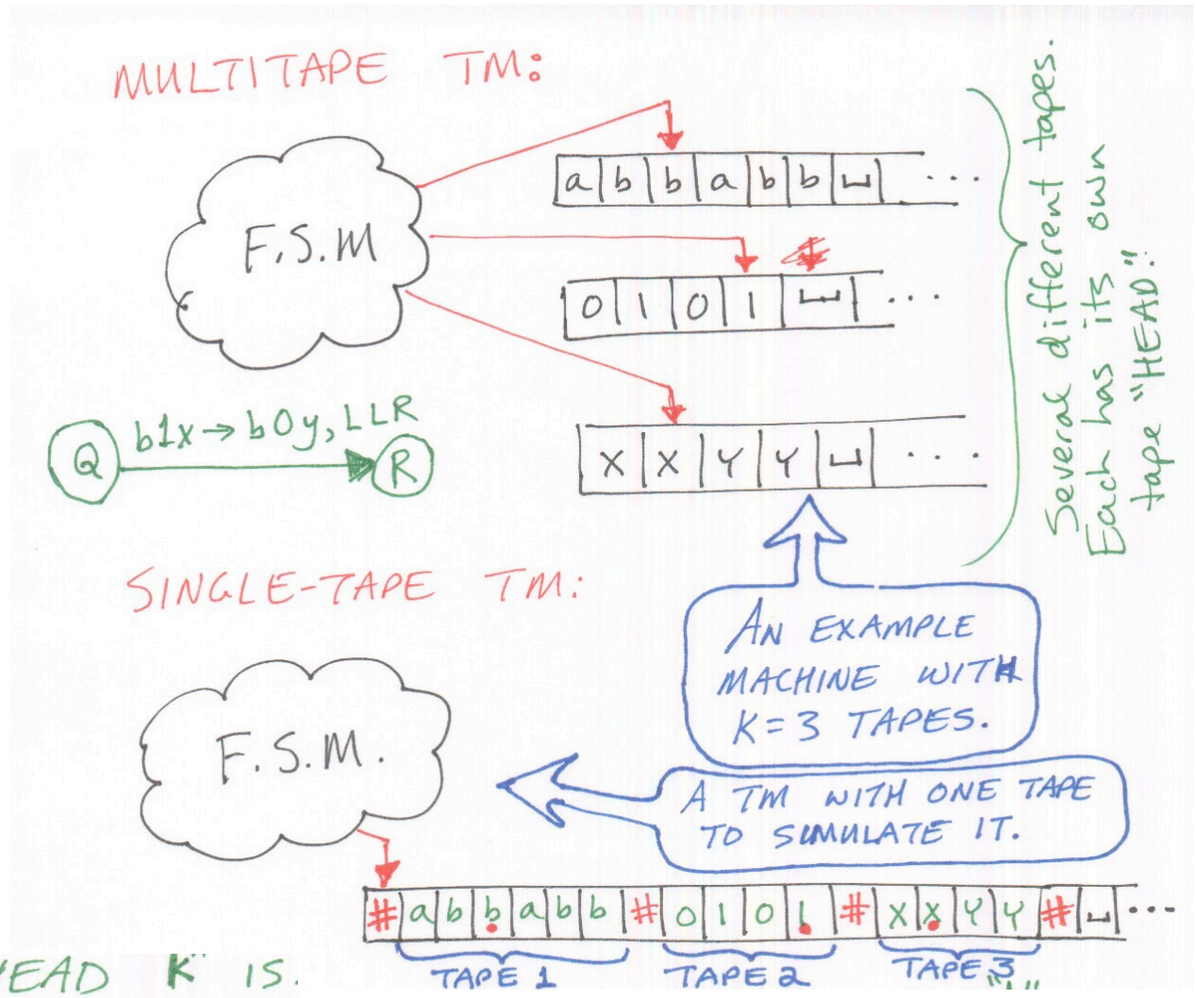
Simulation in Standard Machine



For every possible tape symbol x



slides from Harry Porter PSU



- ADD "DOTS" TO SHOW WHERE HEAD K IS.
- To simulate a transition from state Q, we must scan our tape to see which symbols are "UNDER" the K tape heads
- Once we determine this, and are ready to "MAKE" the transition, we must scan across the tape again to update the cells and move the dots.
- Whenever one head moves off the right end, we must shift our tape so we can insert a ⊔.

Variants of TM models

- Two stacks can simulate one tape.
  - One holds positions to the left of the head; the other holds positions to the right.

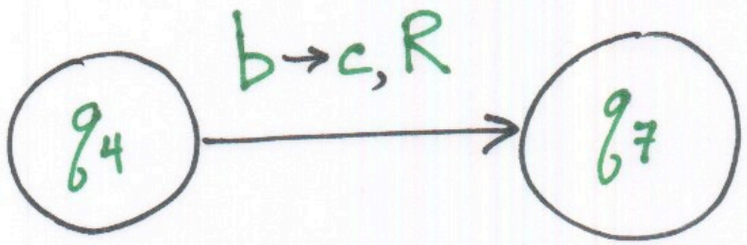
Factoid: Paper written by Pat Fischer, who was targeted by the Unabomber.

Nondeterministic Turing Machines

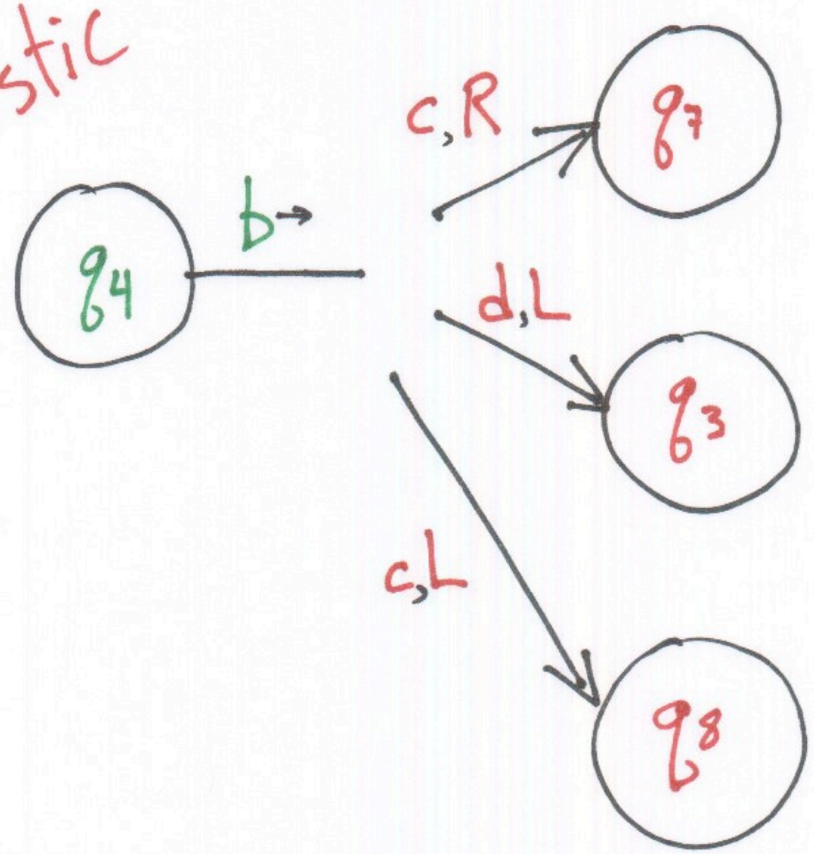
Transition Function

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

Deterministic

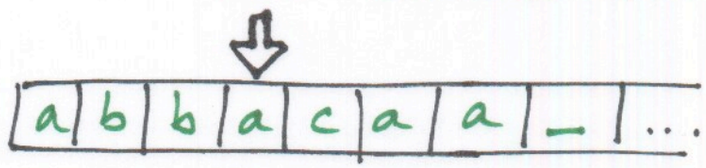


Nondeterministic



A "CONFIGURATION" is...

- A way to represent the entire state of a TM at one moment during a computation.
- A string which captures
  - THE CURRENT STATE
  - THE CURRENT POSITION OF HEAD
  - THE ENTIRE TAPE CONTENTS.

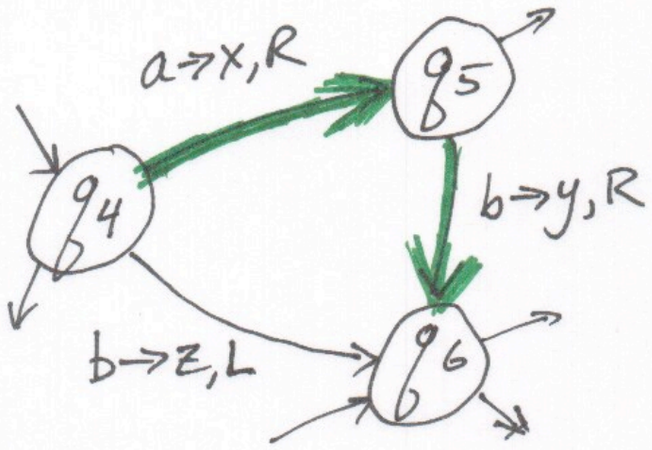


a b b q_{37} a c a a

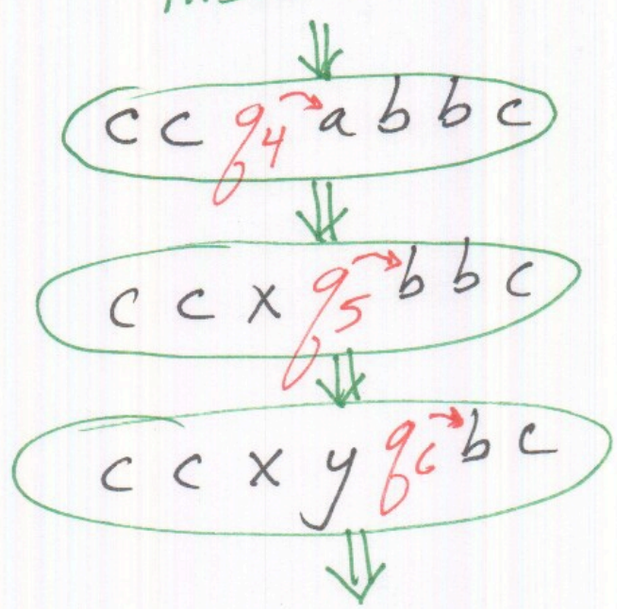
WITH NONDETERMINISM:

At each moment in the computation there can be MORE THAN ONE successor configuration.

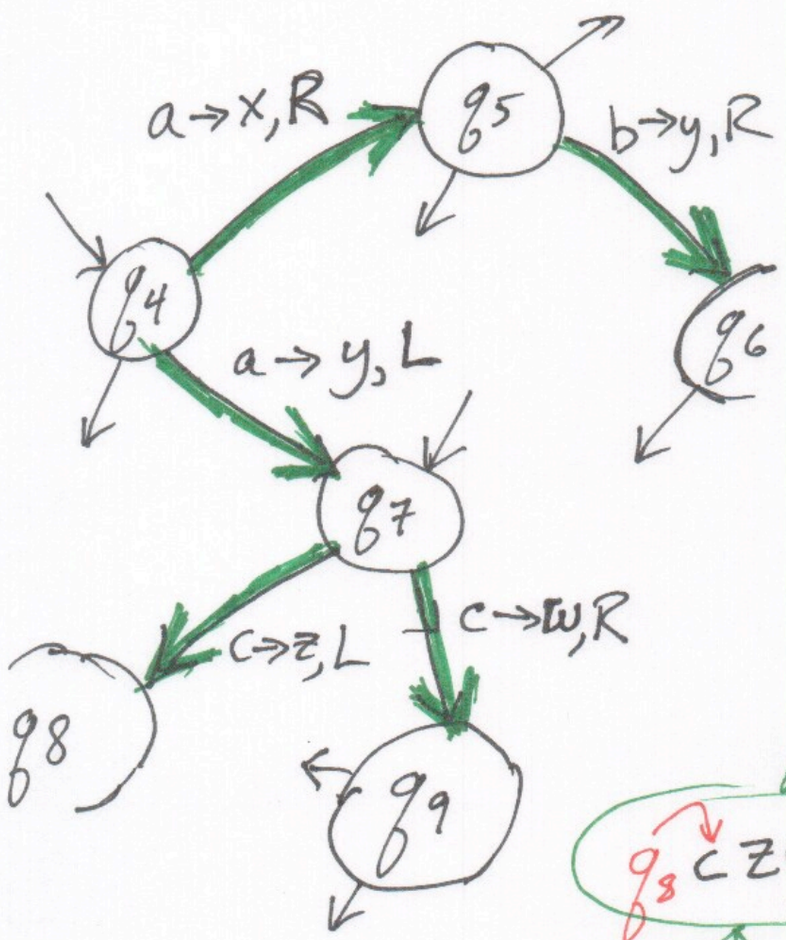
DETERMINISTIC TM

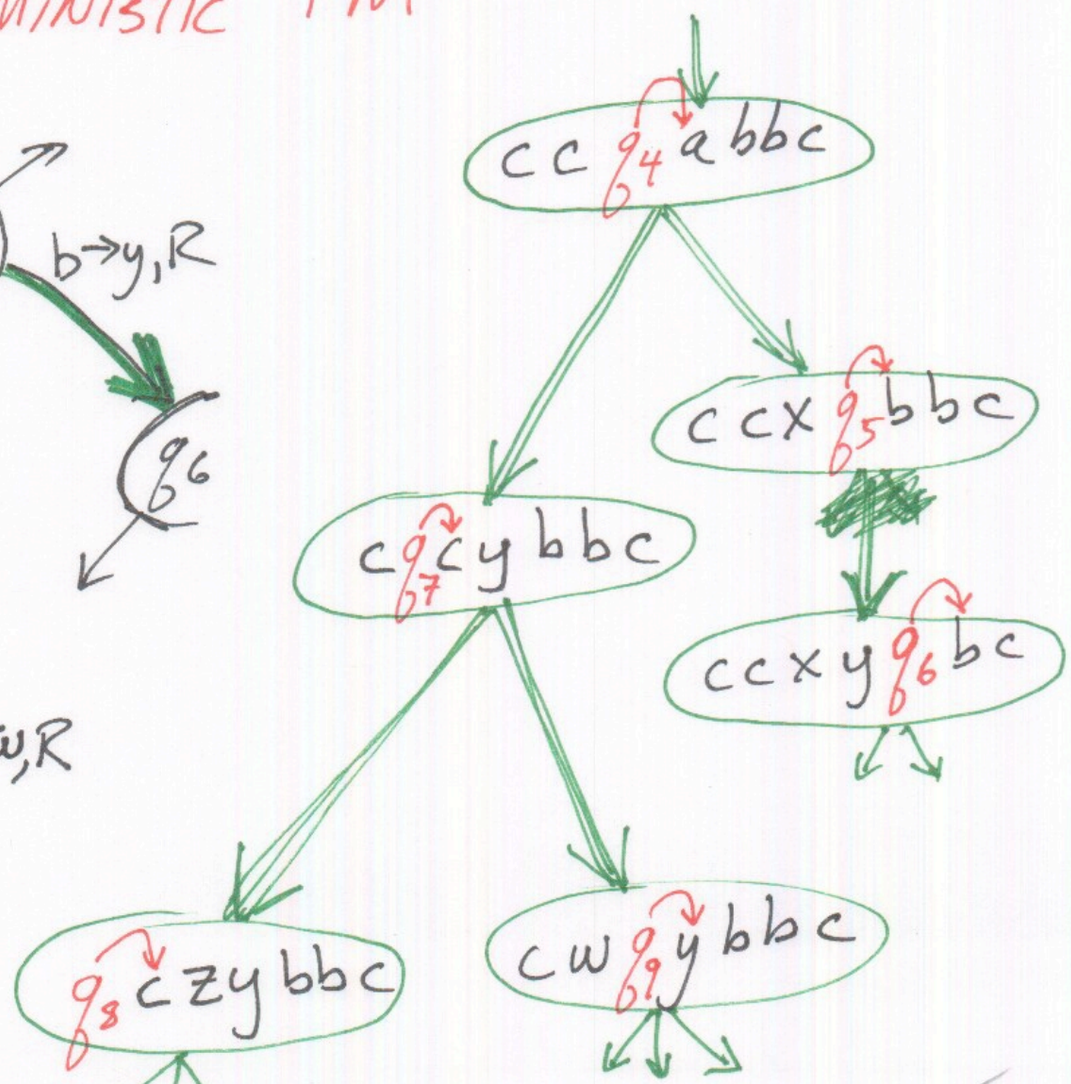


COMPUTATION HISTORY

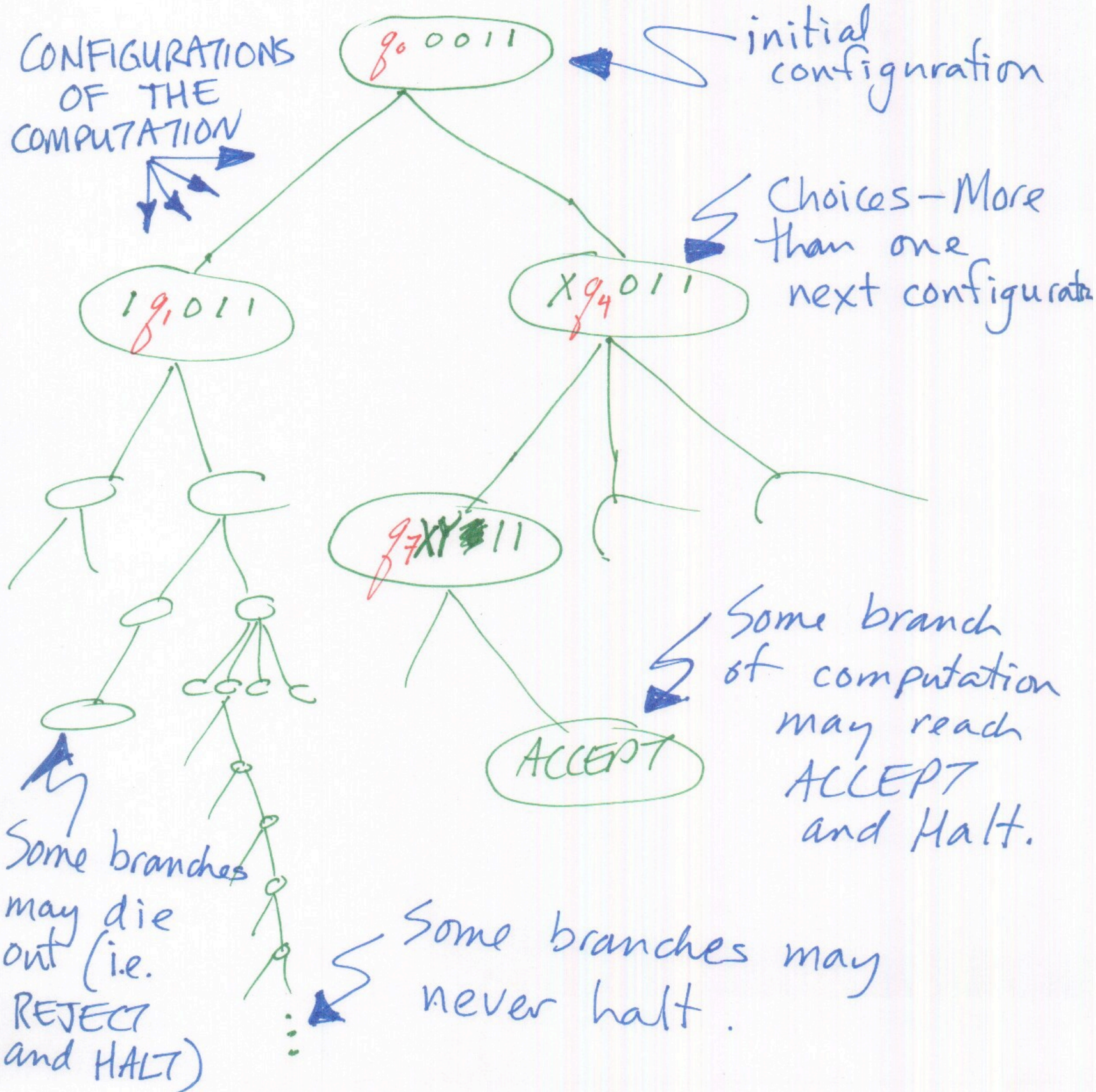


NONDETERMINISTIC TM





A TREE SHOWS THE COMPUTATION OF A NON-DETERMINISTIC TM.



Outcomes of a Nondeterministic Computation:

Accept

If *any* branch of the computation accepts, then the nondeterministic TM will accept.

Reject

If *all* branches of the computation halt and reject (i.e., no branches accept, but all computation halts), then the nondeterministic TM rejects.

Loop

Computation continues, but "accept" is never encountered.

Some branches in the computation history are infinite.

Simulating a NTM by a DTM

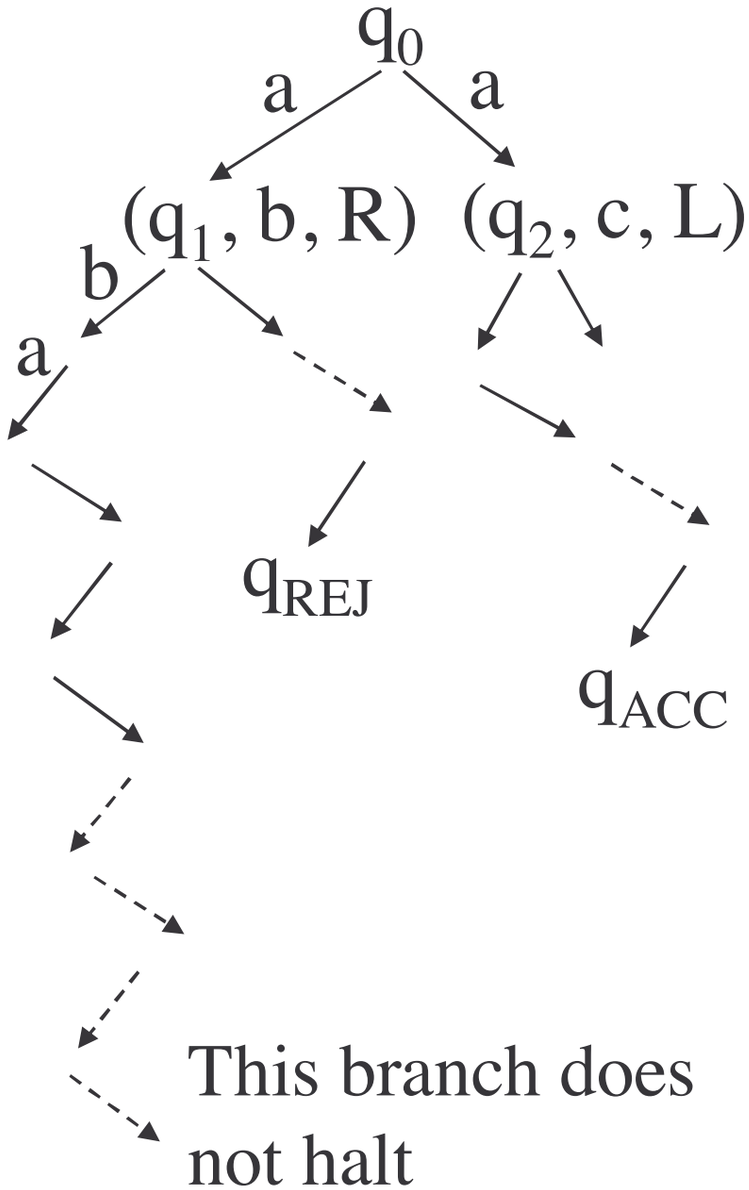
- **Theorem:** Every Nondeterministic TM has an equivalent deterministic TM

- Given a Nondeterministic TM (N) show how to construct a deterministic TM (M)

- if N accepts (on any branch) then D will accept

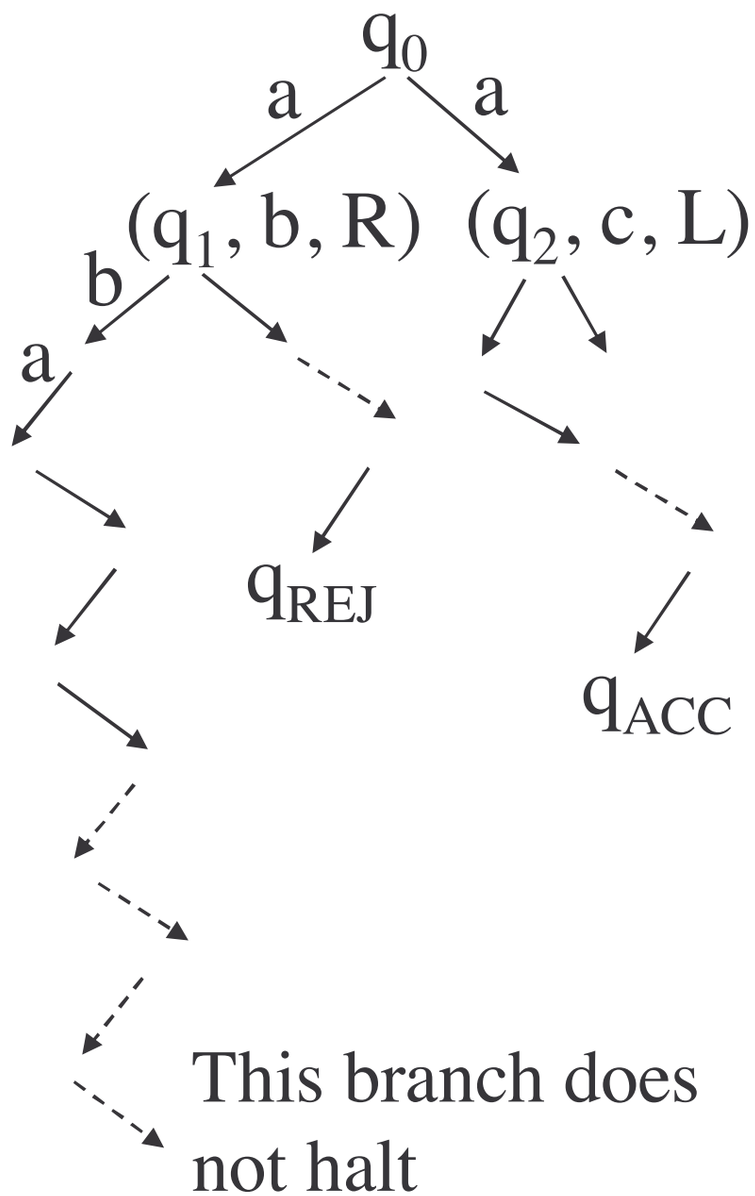
- If N halts on every branch without any “accepts” then D will halt and reject

Simulating a NTM by a DTM



Simulating a NTM by a DTM

Details in textbook



- **General proof idea**: Simulate each branch sequentially
  - Proof idea 1: Use depth first search?
    No, might go deep into an infinite branch and never explore other branches!
  - Proof idea 2: Use breadth first search
    Explore all branches at depth *n* before *n+1*

Now we can other definitions of Turing Recognizable and Turing decidable