Chomsky Normal Form

and

TURING MACHINES

TUESDAY Feb 4

CHOMSKY NORMAL FORM

A context-free grammar is in Chomsky normal form if every rule is of the form:

$A \to BC$	B and C aren't start variables
$A \to a$	a is a terminal
$S \to \varepsilon$	S is the start variable

Any variable A that is not the start variable can only generate strings of length > 0

CHOMSKY NORMAL FORM

A context-free grammar is in Chomsky normal form if every rule is of the form:

$A \to BC$	B and C aren't start variables
$A \to a$	a is a terminal
$S \to \varepsilon$	S is the start variable

$S \to 0S1$
$S \to TT$
$T \to \varepsilon$

$S_0 \to TU \mid TV \mid \varepsilon$
$T \to 0$
$U \to SV$
$S \to TU \mid TV$
$V \to 1$

Theorem: If G is in CNF, $w \in L(G)$ and $|w| > 0$, then any derivation of w in G has length $2|w| - 1$

Proof (by induction on $|w|$):

Base Case: If $|w| = 1$, then any derivation of w must have length 1 $(A \to a)$

Inductive Step: Assume true for any string of length at most $k \geq 1$, and let $|w| = k+1$

Since $|w| > 1$, derivation starts with $A \to BC$

So $w = xy$ where $B \Rightarrow^* x$, $|x| > 0$ and $C \Rightarrow^* y$, $|y| > 0$

By the inductive hypothesis, the length of any derivation of w must be

$$1 + (2|x| - 1) + (2|y| - 1) = 2(|x| + |y|) - 1$$

Theorem: Any context-free language can be generated by a context-free grammar in Chomsky normal form

"Can transform any CFG into Chomsky normal form"

Theorem: **Any context-free language can be generated by a context-free grammar in Chomsky normal form**

Proof Idea:

1. Add a new start variable
2. Eliminate all $\mathbf{A \rightarrow \varepsilon}$ rules ($\boldsymbol{\varepsilon}$ rules). Repair grammar
3. Eliminate all $\mathbf{A \rightarrow B}$ rules (unit productions). Repair

4. Convert $\mathbf{A \rightarrow u_1 u_2 \ldots u_k}$ to $\mathbf{A \rightarrow u_1 A_1, A_1 \rightarrow u_2 A_2, \ldots}$

If $\mathbf{u_i}$ is a terminal, replace $\mathbf{u_i}$ with $\mathbf{U_i}$ and add $\mathbf{U_i \rightarrow u_i}$

1. Add a new start variable S_0 and add the rule $S_0 \rightarrow S$

2. Remove all $A \rightarrow \varepsilon$ rules (where A is not S_0)

For each **occurrence** of A on right hand side of a rule, add a new rule with the occurrence deleted

If we have the rule $B \rightarrow A$, add $B \rightarrow \varepsilon$, unless we have previously removed $B \rightarrow \varepsilon$

3. Remove unit rules $A \rightarrow B$

Whenever $B \rightarrow w$ appears, add the rule $A \rightarrow w$ unless this was a unit rule previously removed

$S_0 \rightarrow S$
$S \rightarrow 0S1$
$S \rightarrow T\#T$
$S \rightarrow T$
$T \rightarrow \varepsilon$
$S \rightarrow T\#$
$S \rightarrow \#T$
$S \rightarrow \#$
$S \rightarrow \varepsilon$
$S \rightarrow 01$
$S_0 \rightarrow 0S1$
$S_0 \rightarrow \varepsilon$

4. Convert all remaining rules into the proper form:

$S_0 \rightarrow 0S1$

$S_0 \rightarrow A_1A_2$

$A_1 \rightarrow 0$

$A_2 \rightarrow SA_3$

$A_3 \rightarrow 1$

$S_0 \rightarrow 01$

$S_0 \rightarrow A_1A_3$

$S \rightarrow 01$

$S \rightarrow A_1A_3$

$S_0 \rightarrow \varepsilon$
$S_0 \rightarrow 0S1$
$S_0 \rightarrow T\#T$
$S_0 \rightarrow T\#$
$S_0 \rightarrow \#T$
$S_0 \rightarrow \#$
$S_0 \rightarrow 01$
$S \rightarrow 0S1$
$S \rightarrow T\#T$
$S \rightarrow T\#$
$S \rightarrow \#T$
$S \rightarrow \#$
$S \rightarrow 01$

Convert the following into Chomsky normal form:

$$A \rightarrow BAB \mid B \mid \varepsilon$$
$$B \rightarrow 00 \mid \varepsilon$$

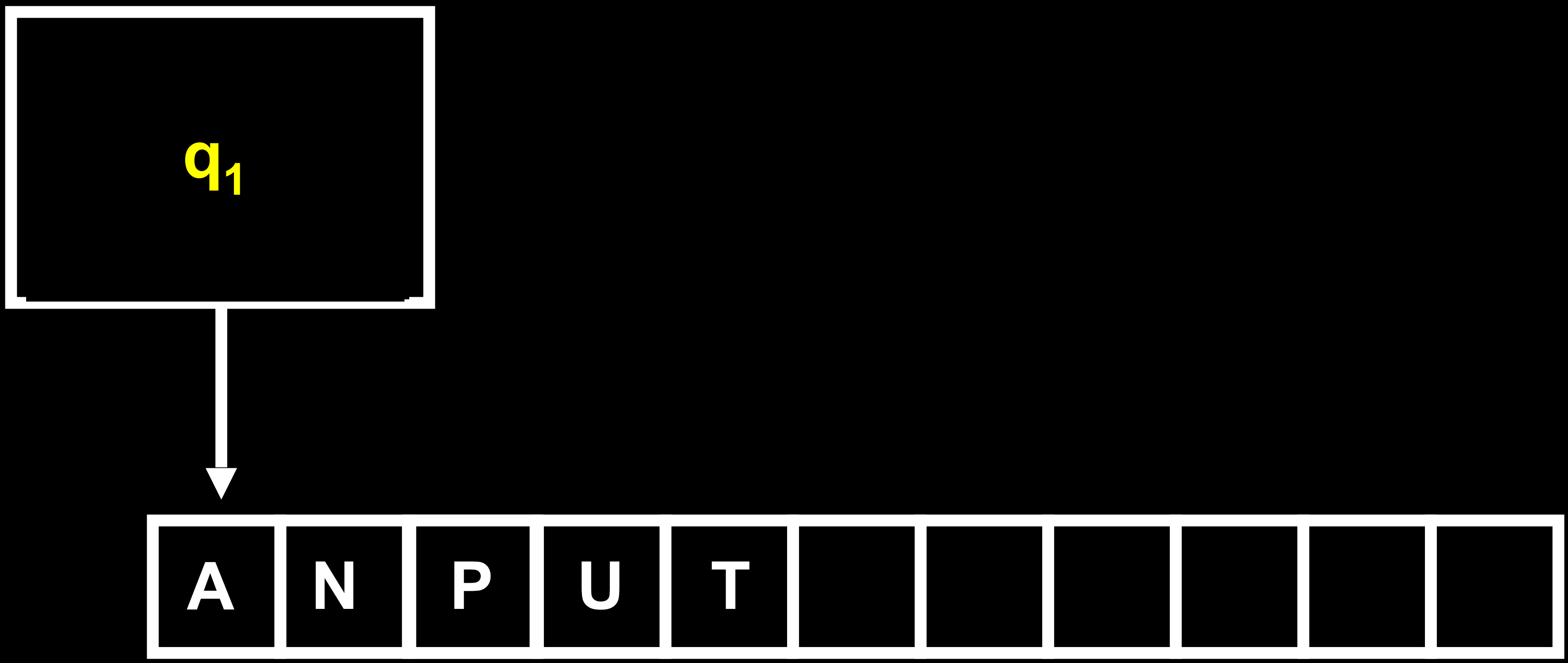
$$S_0 \rightarrow A$$
$$A \rightarrow BAB \mid B \mid \varepsilon$$
$$B \rightarrow 00 \mid \varepsilon$$

$$S_0 \rightarrow A \mid \varepsilon$$
$$A \rightarrow BAB \mid B \mid BB \mid AB \mid BA$$
$$B \rightarrow 00$$

$$S_0 \rightarrow BAB \mid 00 \mid BB \mid AB \mid BA \mid \varepsilon$$
$$A \rightarrow BAB \mid 00 \mid BB \mid AB \mid BA$$
$$B \rightarrow 00$$

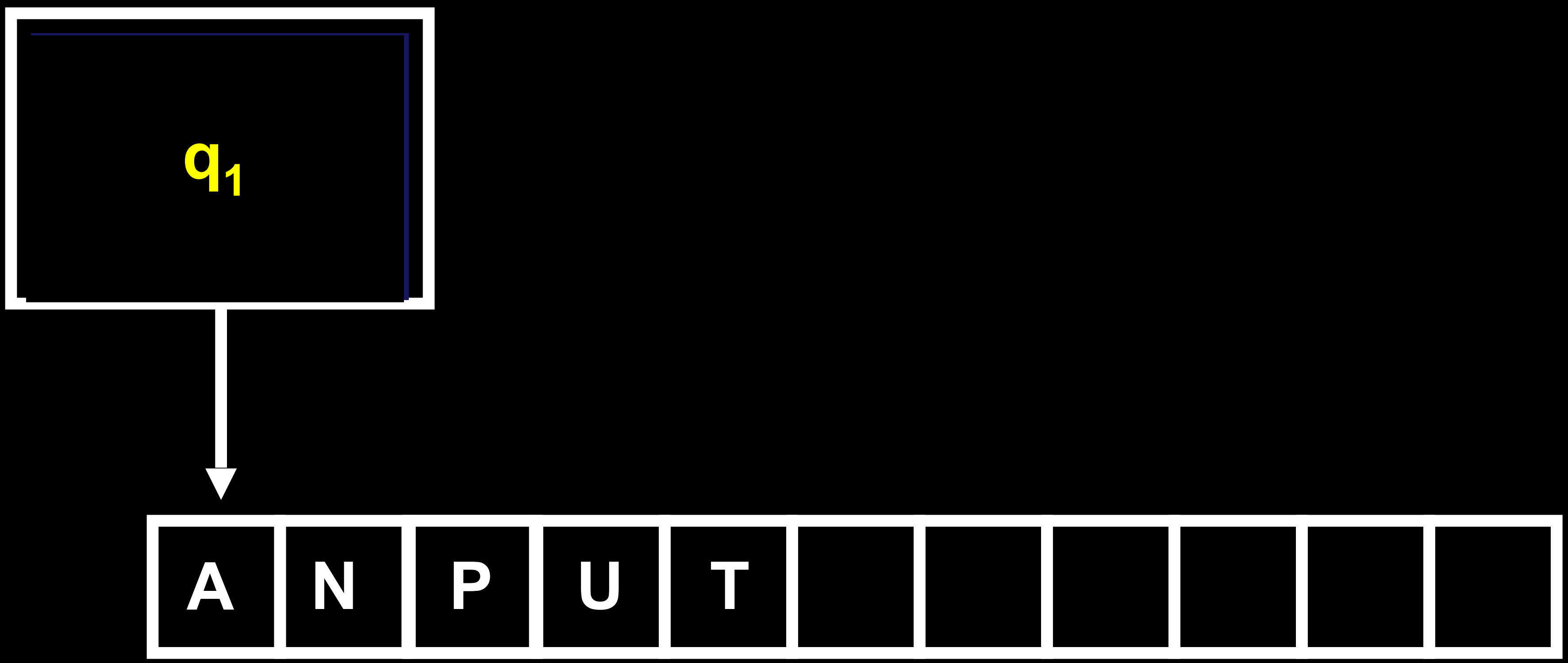
$$S_0 \rightarrow BC \mid DD \mid BB \mid AB \mid BA \mid \varepsilon, \quad C \rightarrow AB,$$
$$A \rightarrow BC \mid DD \mid BB \mid AB \mid BA, \quad B \rightarrow DD, \quad D \rightarrow 0$$

TURING MACHINE



INFINITE TAPE

TURING MACHINE



INFINITE TAPE

Theoretical Computer Science follows TURING,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

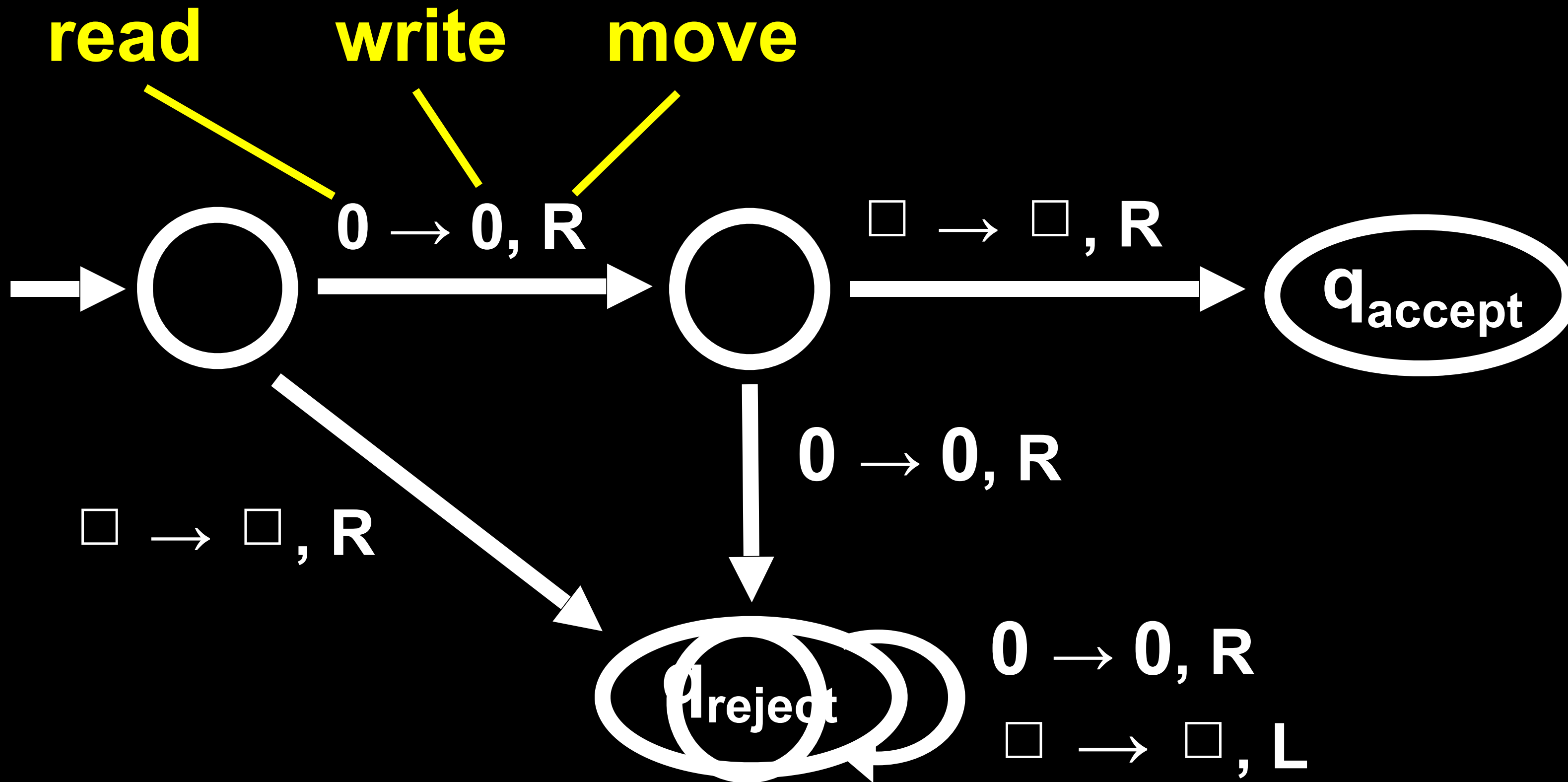
By A. M. TURING.

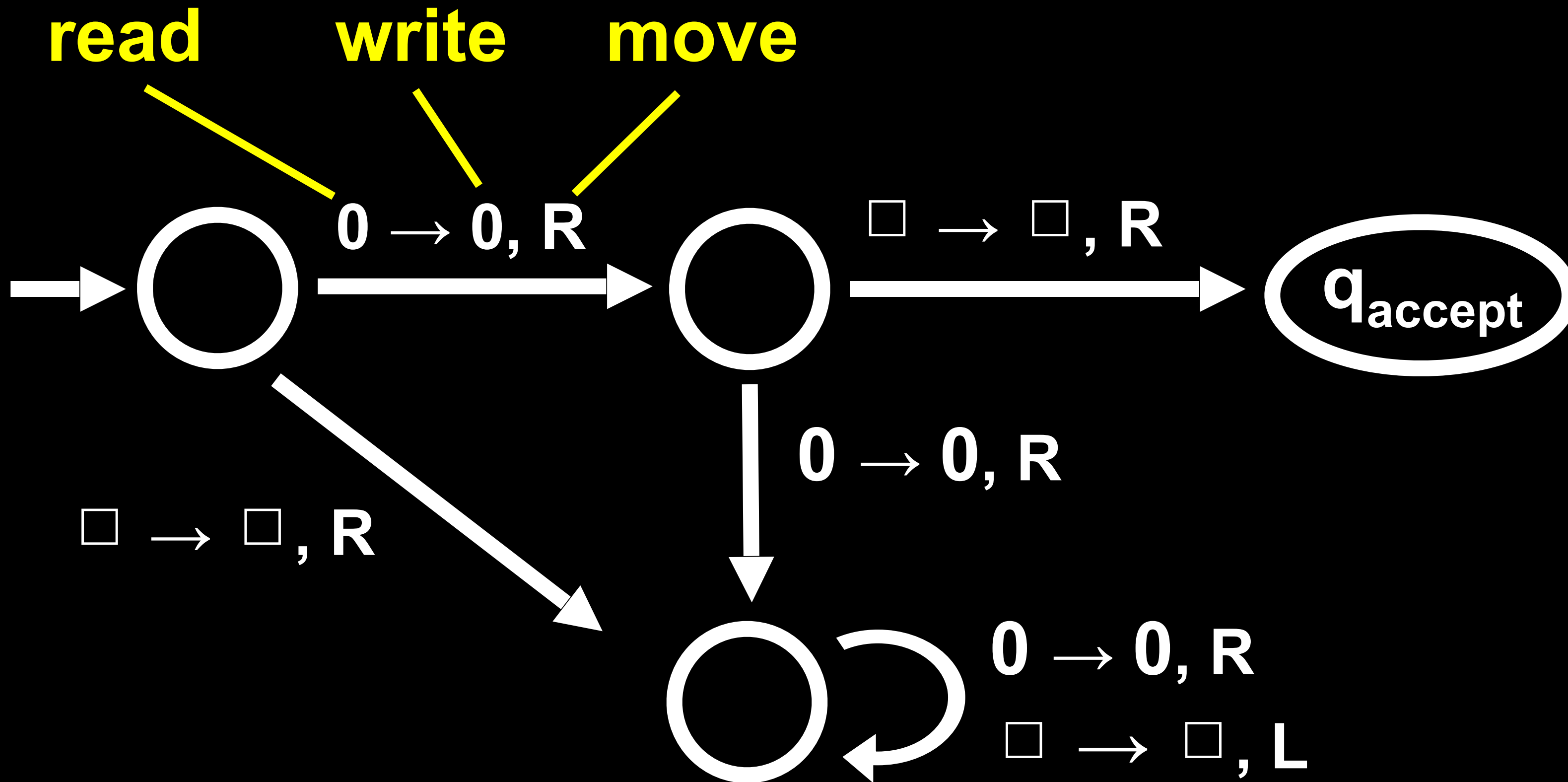
[Received 28 May, 1936.—Read 12 November, 1936.]

The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

'On computable numbers, with an application to the {Entscheidungsproblem'

Proceedings of the London Mathematical Society, 2 **vol. 42, 1937,** pp. 230-265.


read
write
move
0 → 0, R
□ → □, R
q_accept
0 → 0, R
□ → □, R
q_reject
0 → 0, R
□ → □, L



read
write
move
0 → 0, R
□ → □, R
q_accept
□ → □, R
0 → 0, R
0 → 0, R
□ → □, L

TMs VERSUS FINITE AUTOMATA

TM can both *write* to and read from the tape

The head can move *left and right*

The input doesn't have to be read entirely,

and the computation can continue after all the input has been read

Accept and Reject take immediate effect

Definition: A Turing Machine is a 7-tuple
$T = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

Q is a finite set of states

Σ is the input alphabet, where $\square \notin \Sigma$

Γ is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$

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

$q_0 \in Q$ is the start state

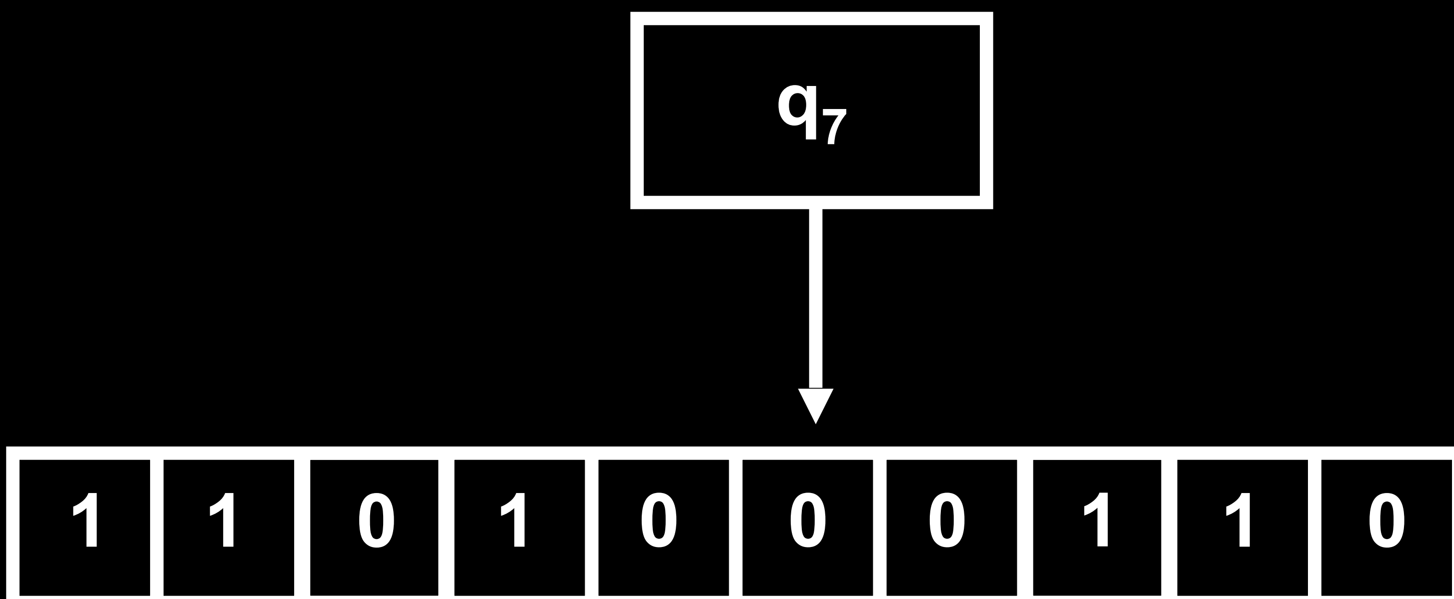
$q_{accept} \in Q$ is the accept state

$q_{reject} \in Q$ is the reject state, and $q_{reject} \neq q_{accept}$

CONFIGURATIONS

$$11010q_7 00110$$

corresponds to:



A Turing Machine M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that

1. C_1 is a *start* configuration of M on input w, ie

   C_1 is q_0w

2. each C_i *yields* C_{i+1}, ie M can legally go from C_i to C_{i+1} in a single step

ua q_i bv *yields* u q_j acv if $\delta(q_i, b) = (q_j, c, L)$

ua qi bv *yields* uac q_j v if $\delta(q_i, b) = (q_j, c, R)$

A Turing Machine M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that

1. **C_1 is a *start* configuration of M on input w, ie C_1 is q_0w**
2. **each C_i *yields* C_{i+1}, ie M can legally go from C_i to C_{i+1} in a single step**
3. **C_k is an *accepting* configuration, ie the state of the configuration is q_{accept}**

A TM *recognizes* a language iff it accepts all and only those strings in the language

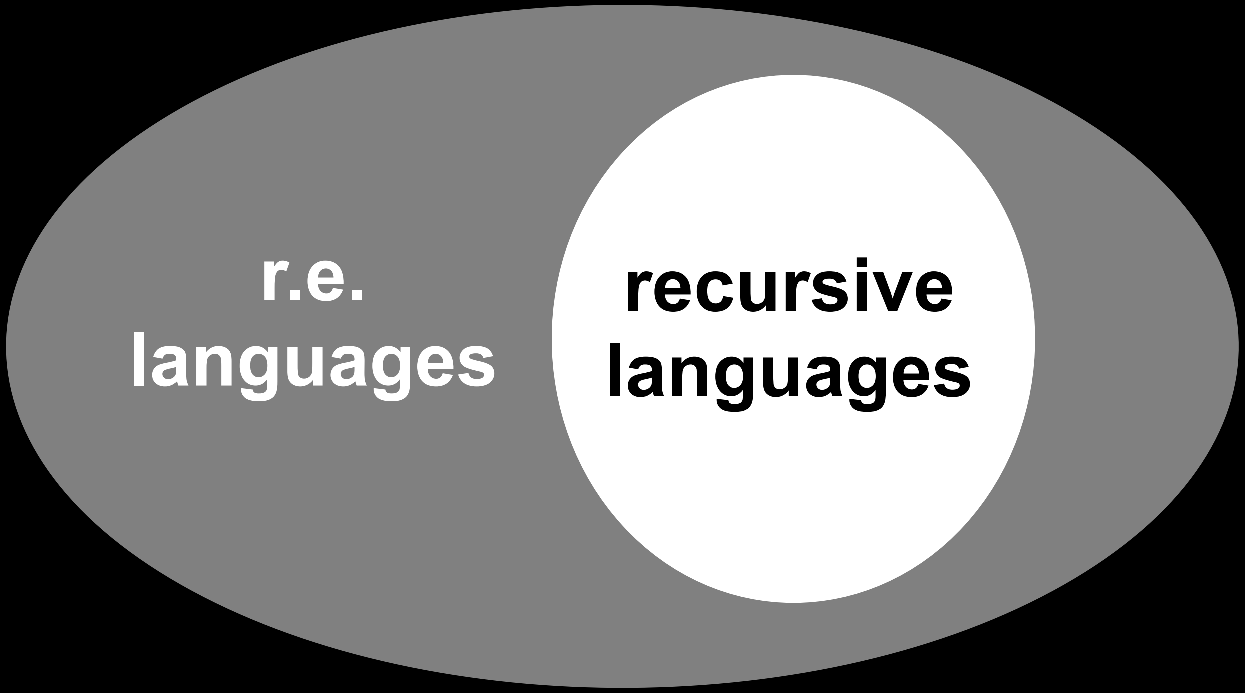
A language L is called Turing-recognizable or recursively enumerable or semi-decidable iff some TM recognizes L

A TM *decides* a language L iff it accepts all strings in L and rejects all strings not in L

A language L is called decidable or recursive iff some TM decides L

A language is called **Turing-recognizable** or **recursively enumerable (r.e.)** or **semi-decidable** if some TM **recognizes** it

A language is called **decidable** or **recursive** if some TM **decides** it



Theorem: If A and ¬A are r.e. then A is recursive

Theorem: If A and ¬A are r.e. then A is recursive

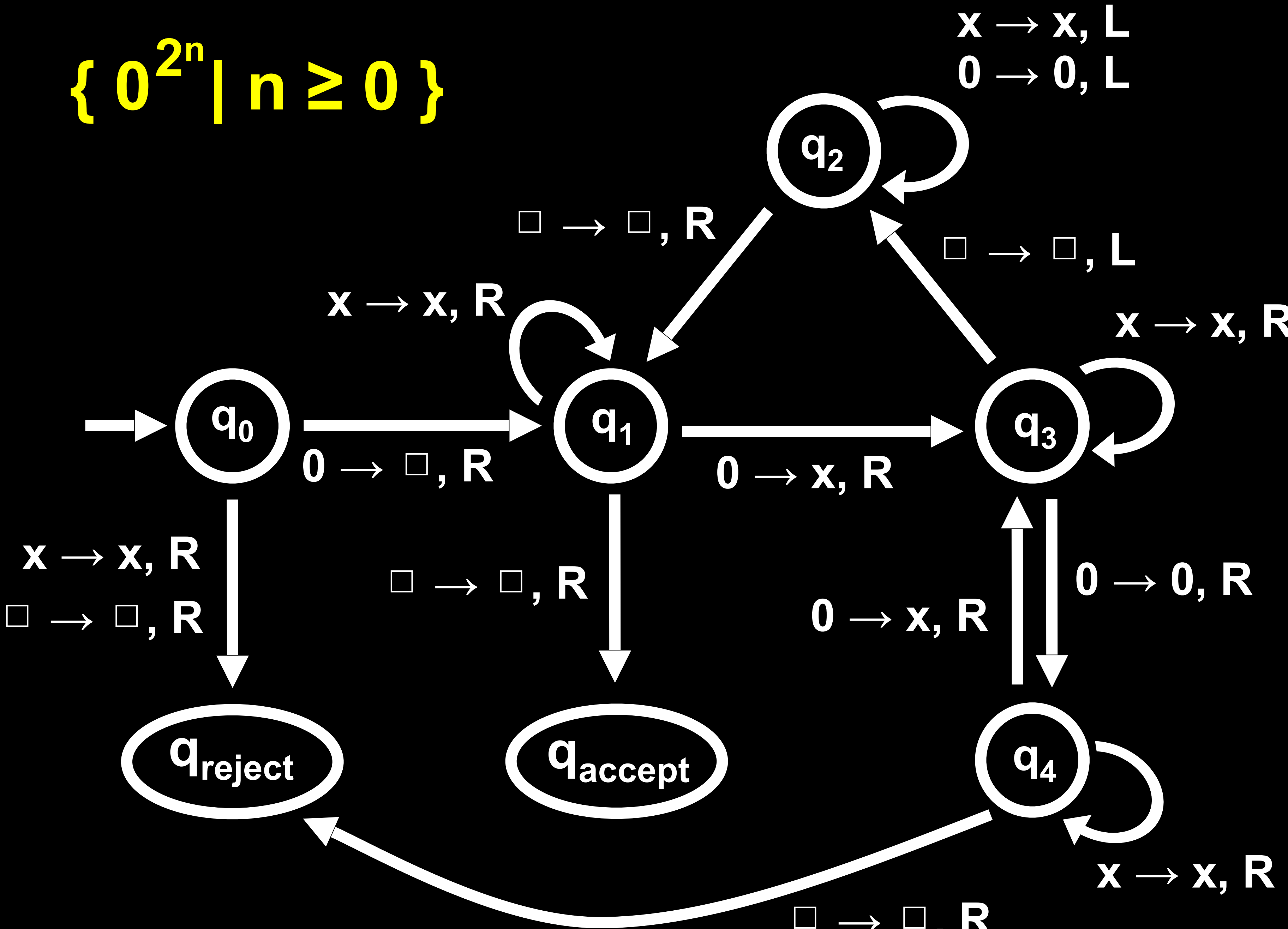
Given:
a TM that recognizes A and
a TM that recognizes ¬A,
we can build a new machine that *decides* A

$\{ 0^{2^n} \mid n \geq 0 \}$ Is decidable

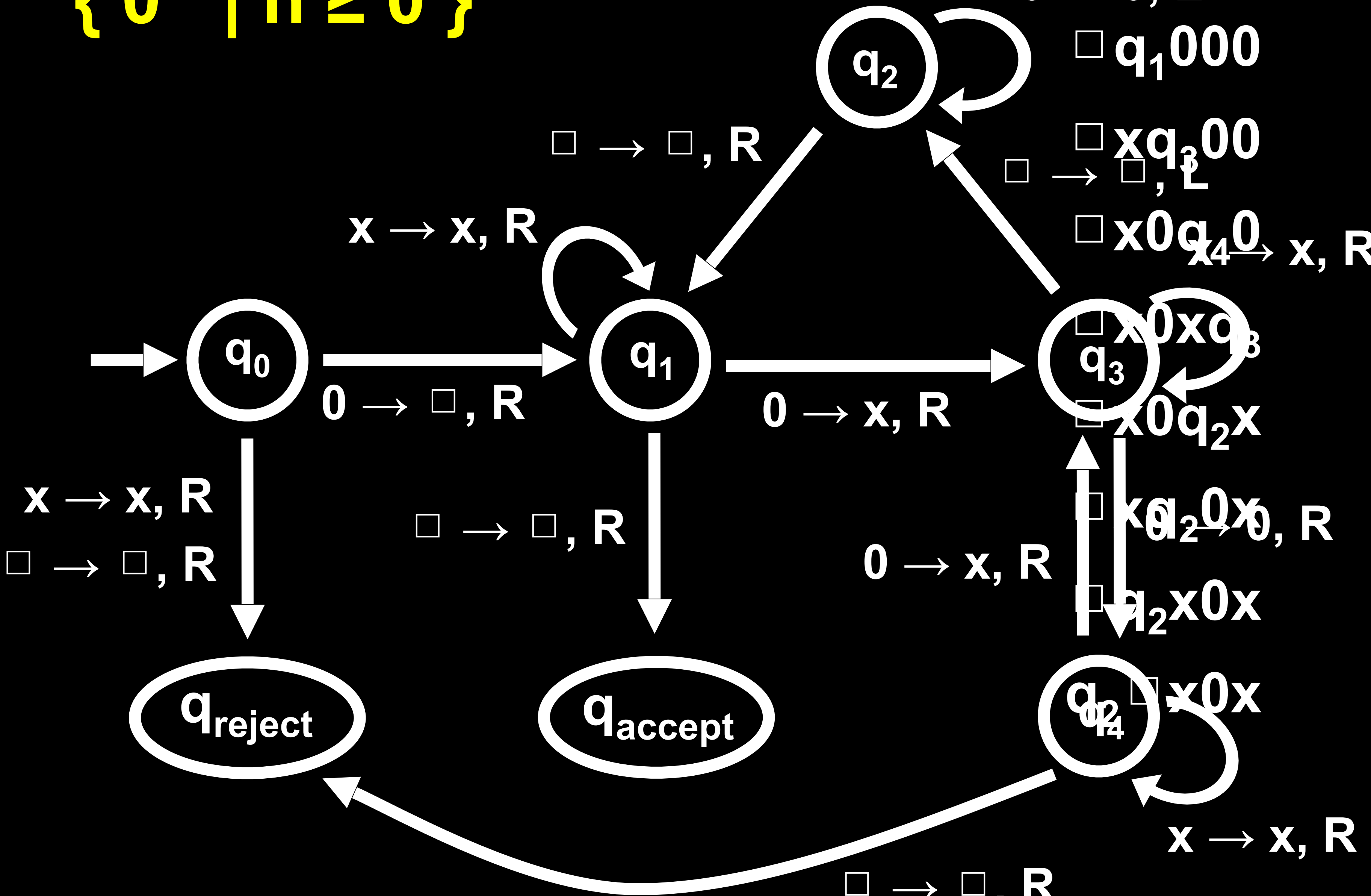
PSEUDOCODE:

1. Sweep from left to right, cross out every other **0**
2. If in stage 1, the tape had only one **0**, *accept*
3. If in stage 1, the tape had an odd number of **0**’s, *reject*
4. Move the head back to the first input symbol.
5. Go to stage 1.

$\{ 0^{2^n} \mid n \geq 0 \}$



$\{ 0^{2^n} \mid n \geq 0 \}$



$C = \{a^i b^j c^k \mid k = ij, \text{ and } i, j, k \geq 1\}$

PSEUDOCODE:

1. If the input doesn't match **a*b*c***, *reject.*
2. Move the head back to the leftmost symbol.
3. Cross off an **a**, scan to the right until **b**.
   Sweep between **b**'s and **c**'s, crossing off one of each until all **b**'s are crossed off.
4. Uncross all the **b**'s.
   If there's another **a** left, then repeat stage 3.
   If all **a**'s are crossed out,
   Check if all **c**'s are crossed off.
   If yes, then *accept*, else *reject.*

$C = \{a^i b^j c^k \mid k = ij, \text{ and } i, j, k \geq 1\}$

aabbbcccccc

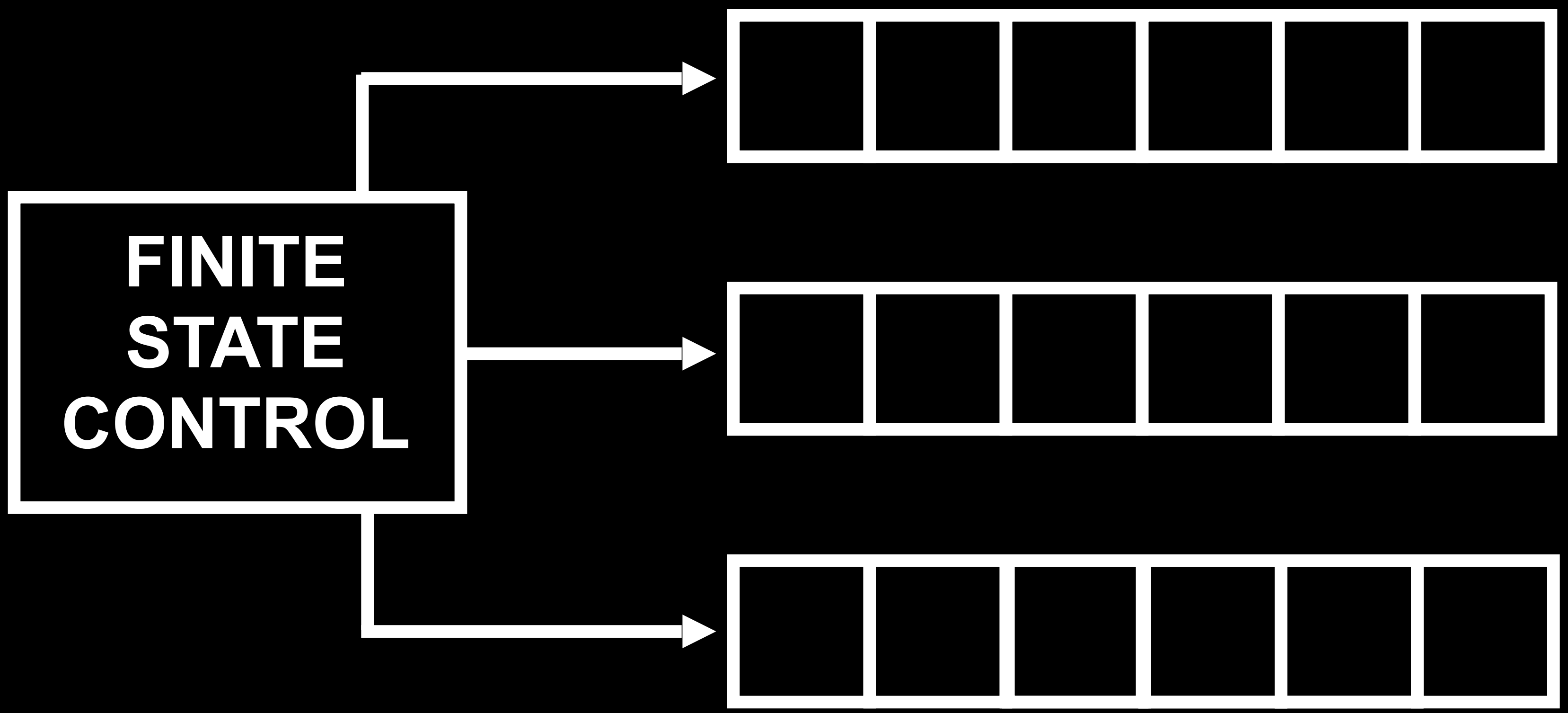
xabbbcccccc

xayyyzzzccc

xabbbzzzccc

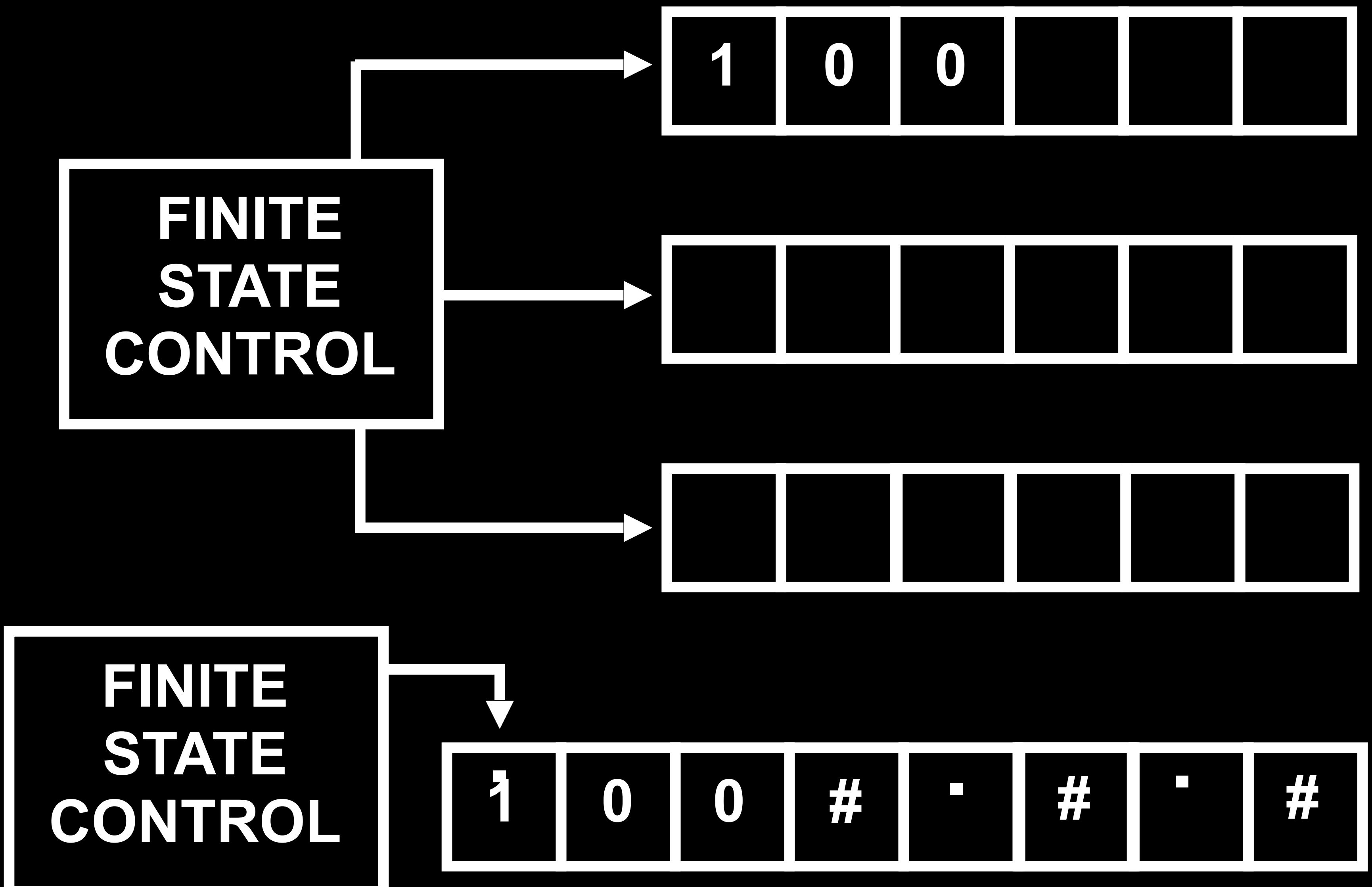
xxyyyzzzzzz

MULTITAPE TURING MACHINES

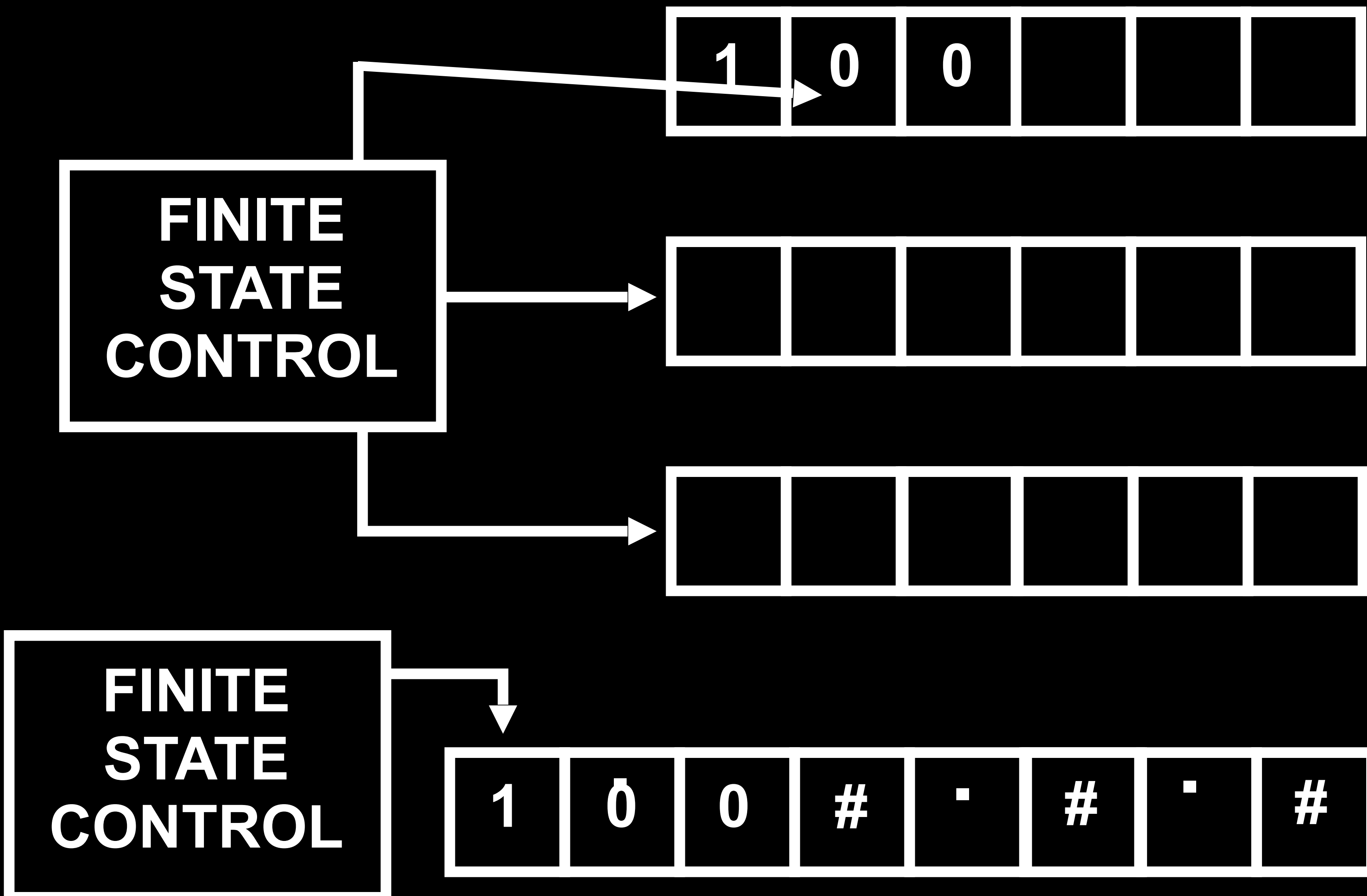


$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L,R\}^k$$

Theorem: Every Multitape Turing Machine can be transformed into a single tape Turing Machine



Theorem: **Every Multitape Turing Machine can be transformed into a single tape Turing Machine**



THE CHURCH-TURING THESIS

Intuitive Notion of Algorithms

EQUALS

Turing Machines

We can encode a TM as a string of 0s and 1s

start state

reject state

n states

$$0^n10^m10^k10^s10^t10^r10^u1\ldots$$

m tape symbols (first k are input symbols)

accept state

blank symbol

$$(\,(p, a), (q, b, L)\,) = 0^p10^a10^q10^b10$$

$$(\,(p, a), (q, b, R)\,) = 0^p10^a10^q10^b11$$

Similarly, we can encode DFAs, NFAs, CFGs, *etc.* into strings of 0s and 1s

So we can define the following languages:

A_{DFA} = { (B, w) | B is a DFA that accepts string w }

Theorem: A_{DFA} is decidable

Proof Idea: Simulate B on w

A_{NFA} = { (B, w) | B is an NFA that accepts string w }

Theorem: A_{NFA} is decidable

A_{CFG} = { (G, w) | G is a CFG that generates string w }

Theorem: A_{CFG} is decidable

Proof Idea: Transform G into Chomsky Normal Form. Try all derivations of length up to 2|w|-1

Read Chapter 3 of the book for next time