CS311 Computational Structures

Turing Machines

Lecture 11

Andrew P. Black
Andrew Tolmach

Portland State University

Alan Mathison Turing

Portland State UNIVERSITY

Alan Mathison Turing

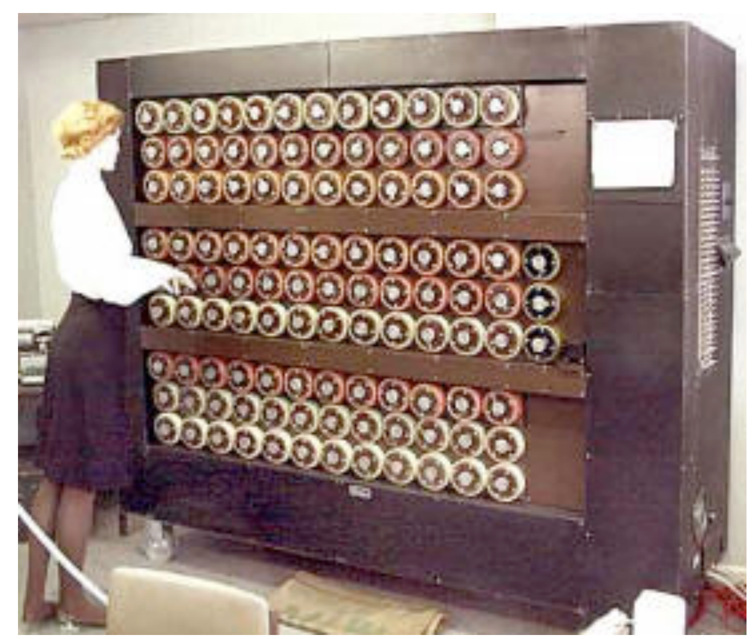
- Founder of modern computer science
- Born 1912, London
- Died 1954, Cheshire (suicide)
- Computability (Turing Machines)
- Artificial Intelligence (Turing test)
- Cryptography (breaking Enigma)
- Physical computer design/ applications
- Mathematics, logic, biology, physics,...

Turing personal chronology

- 1912 (23 June): Birth, Paddington, London
- 1926-31: Sherborne School
- 1930: Death of friend Christopher Morcom
- 1931-34: Undergraduate at King's College, Cambridge University
- 1932-35: Quantum mechanics, probability, logic
- 1935: Elected fellow of King's College, Cambridge
- 1936: The Turing machine, computability, universal machine
- 1936-38: Princeton University. Ph.D. Logic, algebra, number theory
- 1938-39: Return to Cambridge. Introduced to German Enigma cipher machine
- 1939-40: The Bombe, machine for Enigma decryption
- 1939-42: Breaking of U-boat Enigma, saving battle of the Atlantic
- 1943-45: Chief Anglo-American crypto consultant. Electronic work.
- 1945: National Physical Laboratory, London
- 1946: Computer and software design leading the world.
- 1947-48: Programming, neural nets, and artificial intelligence
- 1948: Manchester University
- 1949: First serious mathematical use of a computer
- 1950: The Turing Test for machine intelligence
- 1951: Elected FRS. Non-linear theory of biological growth
- 1952: Arrested as a homosexual, loss of security clearance
- 1953-54: Unfinished work in biology and physics
- 1954 (7 June): Death (suicide) by cyanide poisoning, Wilmslow, Cheshire.

Source: Turing web site maintained by Hodges

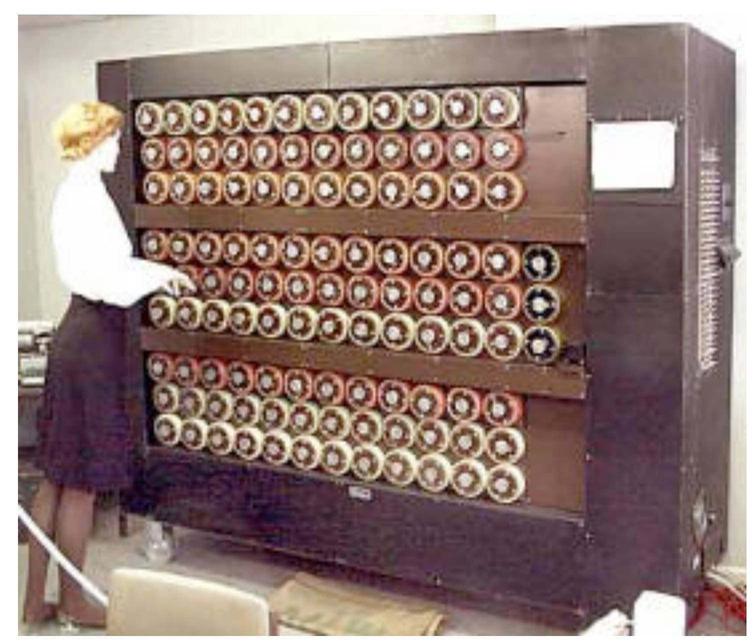
Turing  Quiz



Portland State UNIVERSITY

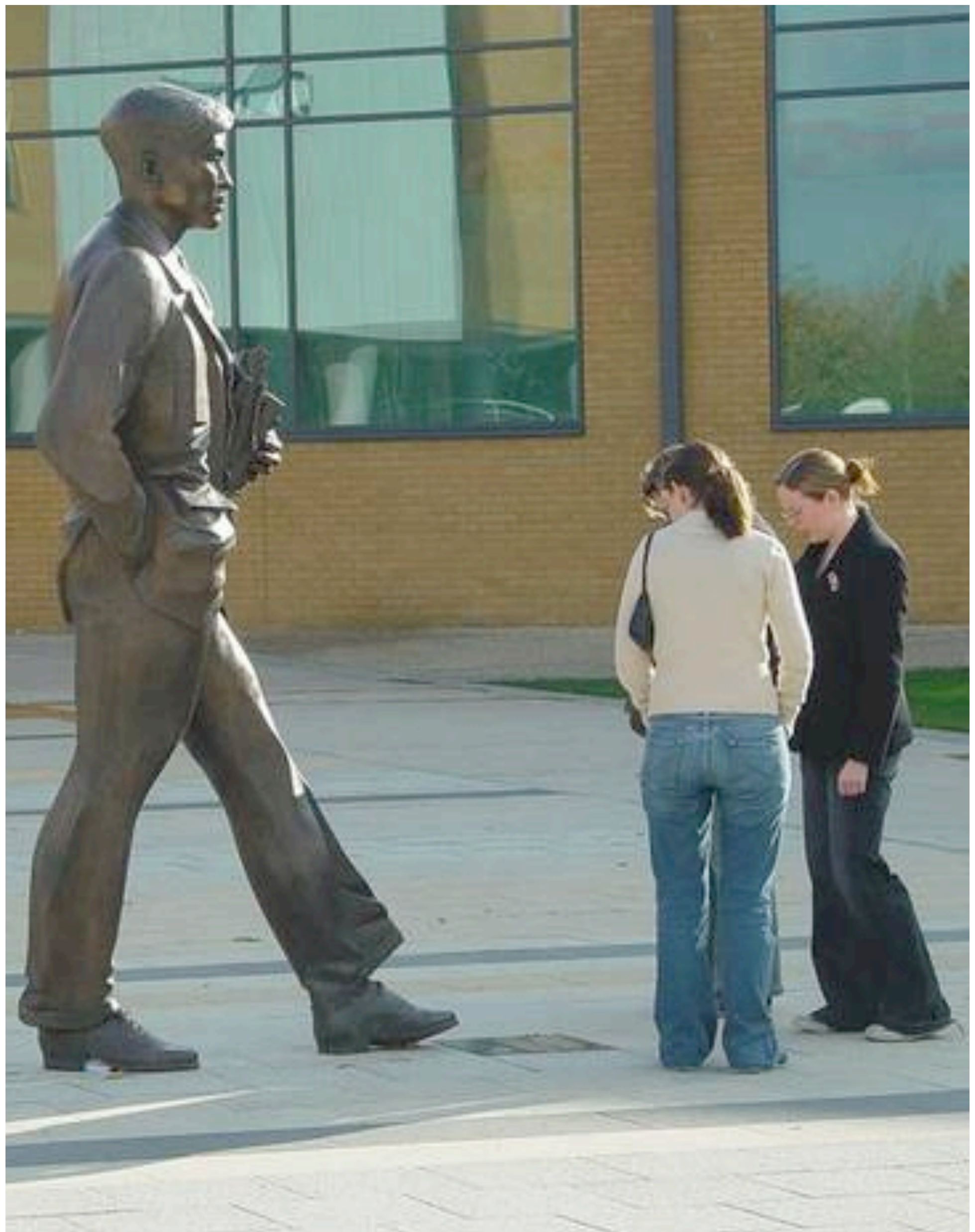
Turing  Quiz

Hut 6



Sackville Park, Manchester

University of Surrey



Sackville Park,
Manchester

Some History

1928: Hilbert proposed the Entscheidungsproblem

- Is there an algorithm for deciding the truth or falsity of any theorem in a mathematical system?

1931: Gödel's Incompleteness Theorems

- Gödel numbering

1936: Church defines "effective calculability" based on λ-calculus

1936: Turing defines the Turing machine

1936: Church and Turing (independently) answer the Entscheidungsproblem: *No*

Turing sources

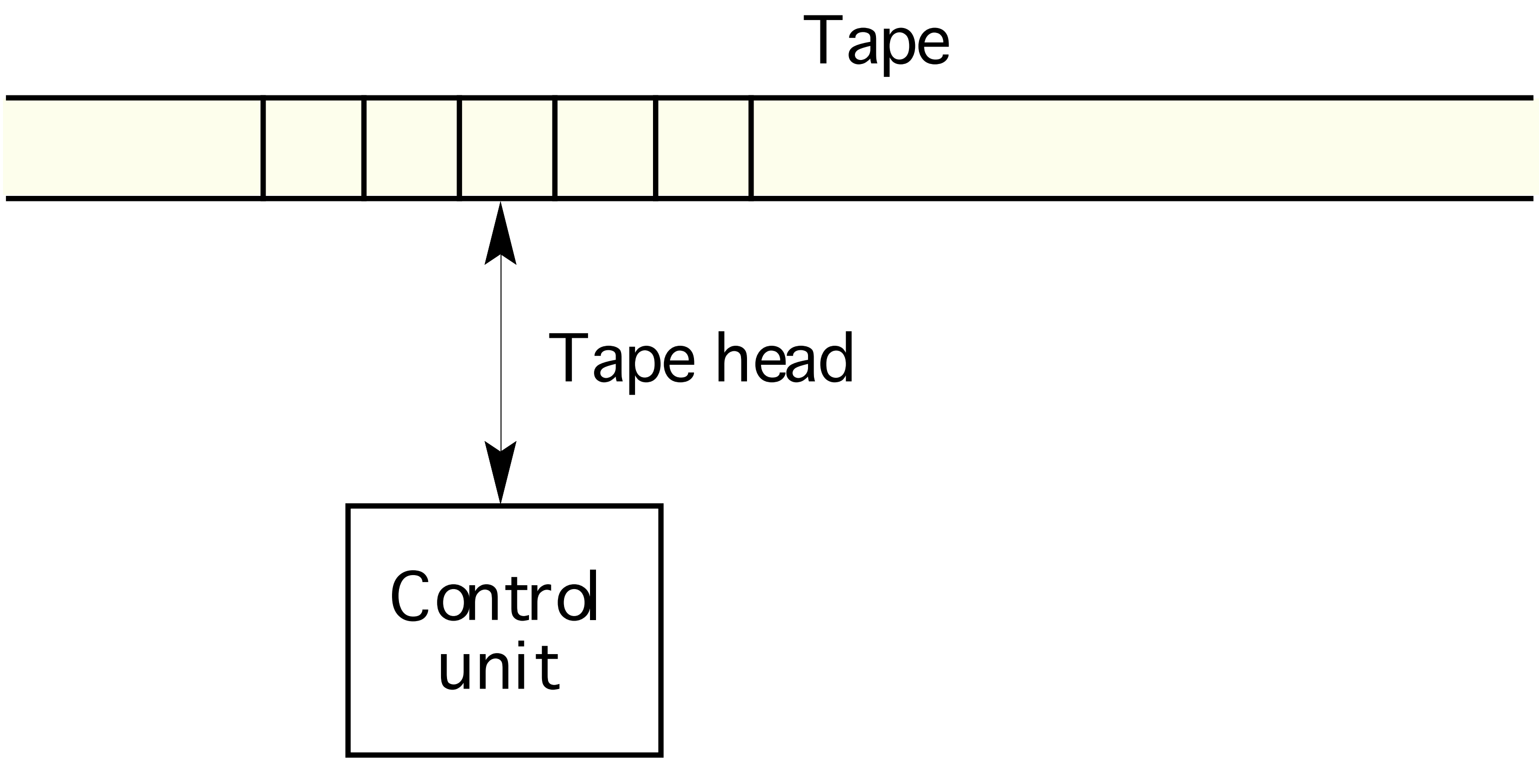
- Biography: Andrew Hodges, *Alan Turing: the Enigma*, Walker and Company, New York.
- Web site maintained by biographer:
  - http://www.turing.org.uk/turing/

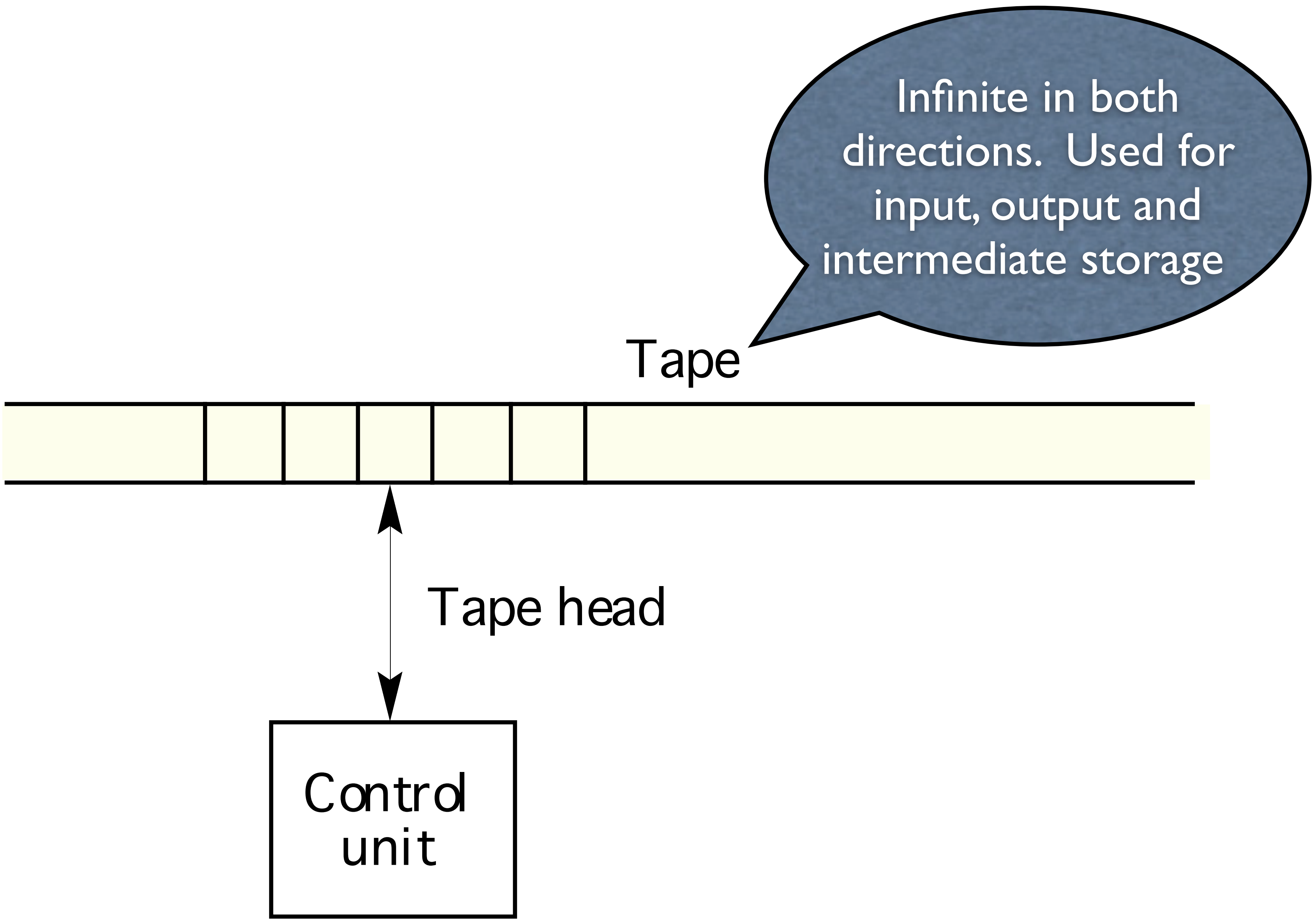
What is a Turing Machine?

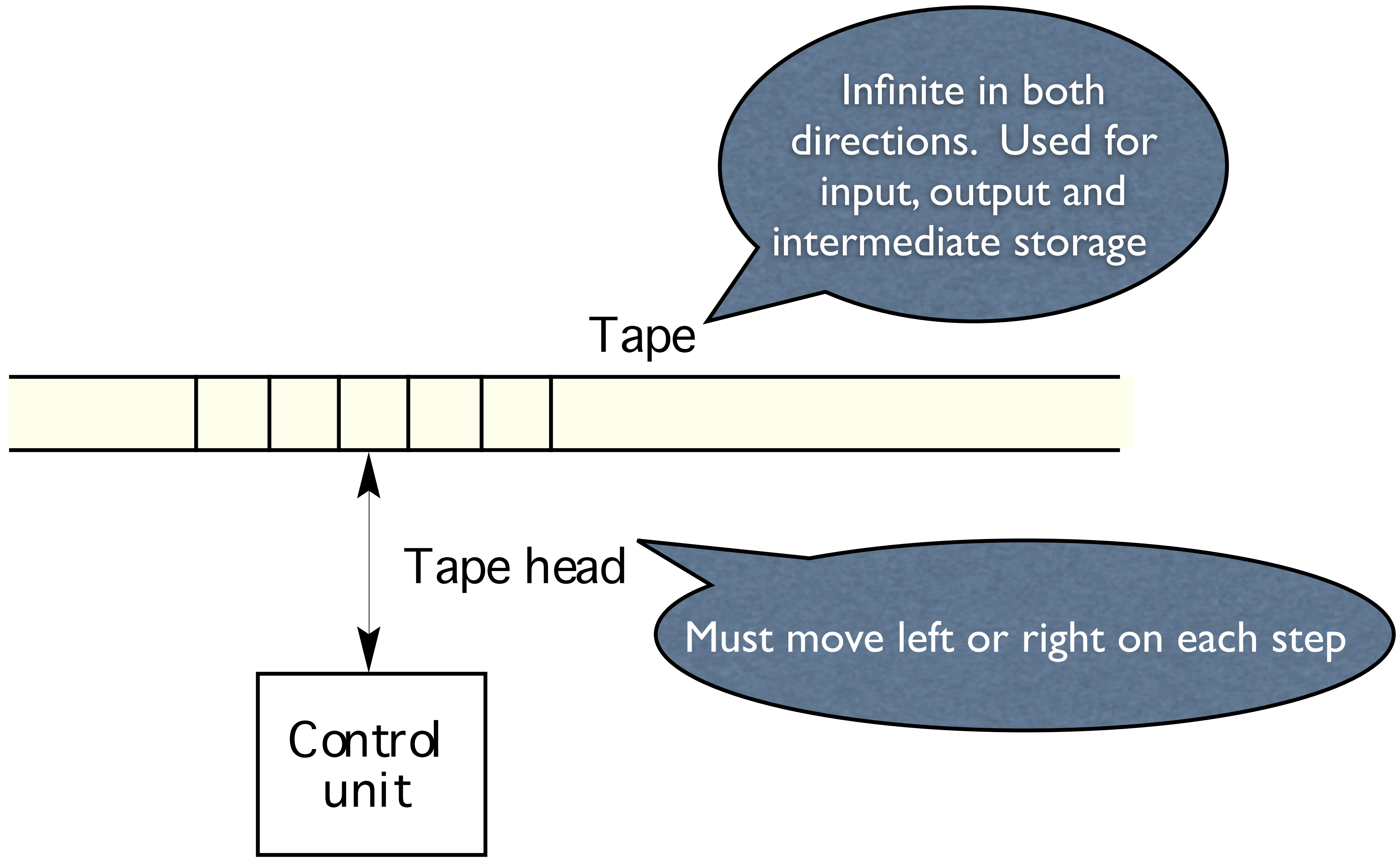
- It's a theoretical machine that does the work of a computer
  - When Turing wrote about a "computer" he meant a person!
- Should be able to do anything a computer could possibly do.
  - So if TM can't do a computation, that computation just isn't possible

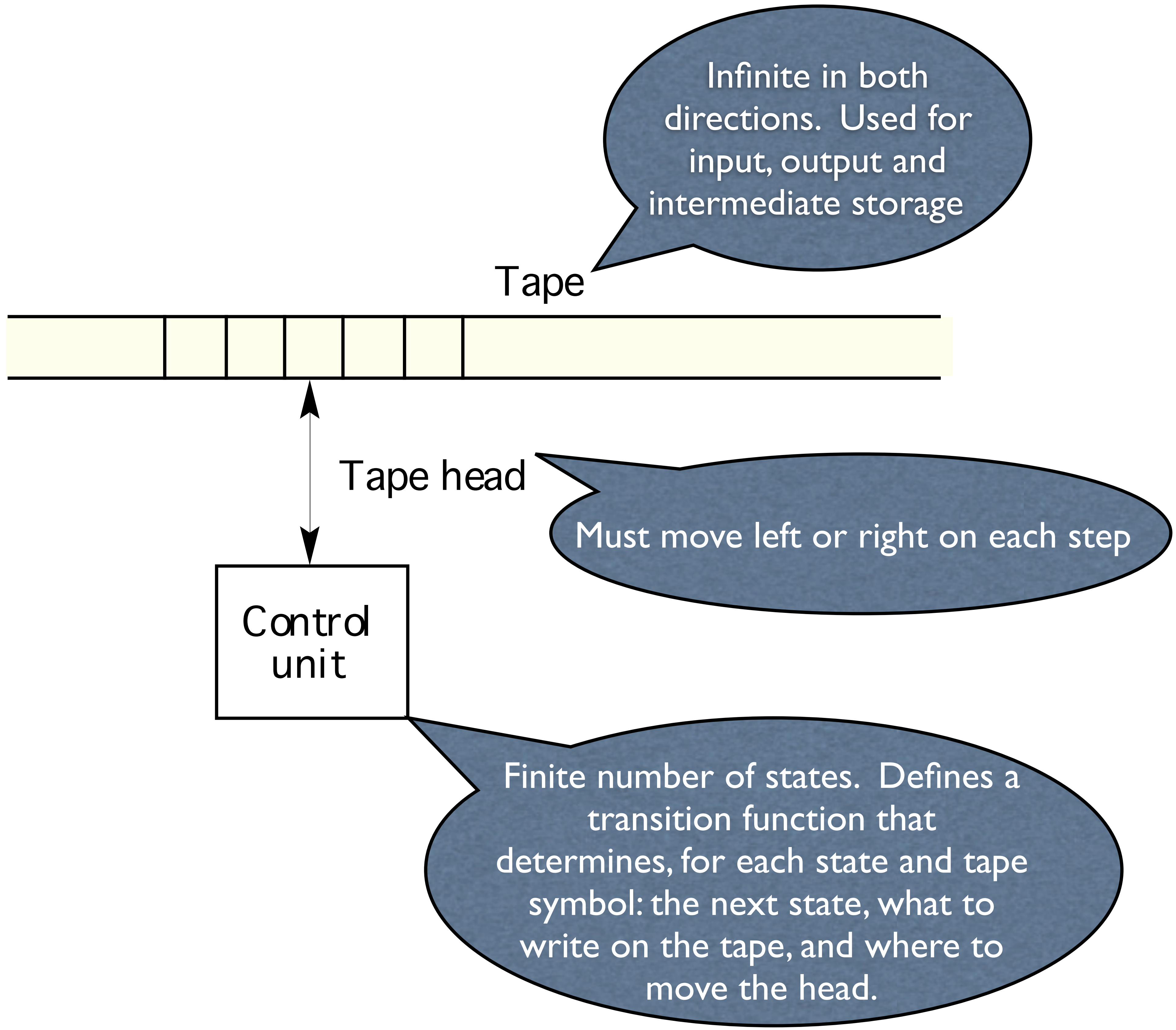
What constrains a "Computer"?

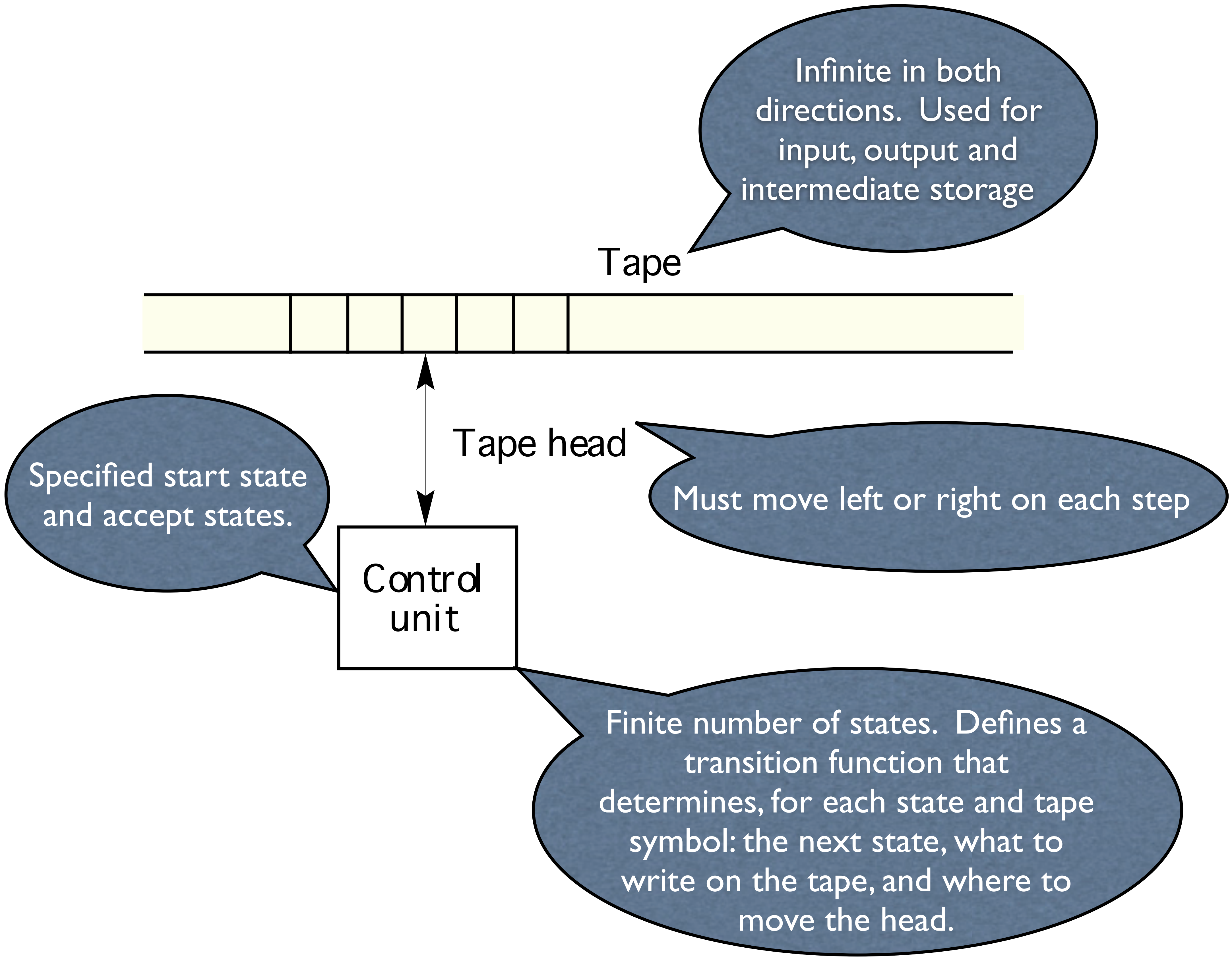
- Limited "mental state"
  - Computer can only act in a limited number of different modes
- Unlimited scratch space...
  - Computer can record a configuration and return to the computation after taking a break
  - Can always extend the written configuration
- ...but limited field of awareness
  - Can only look at a limited part of the configuration at one time

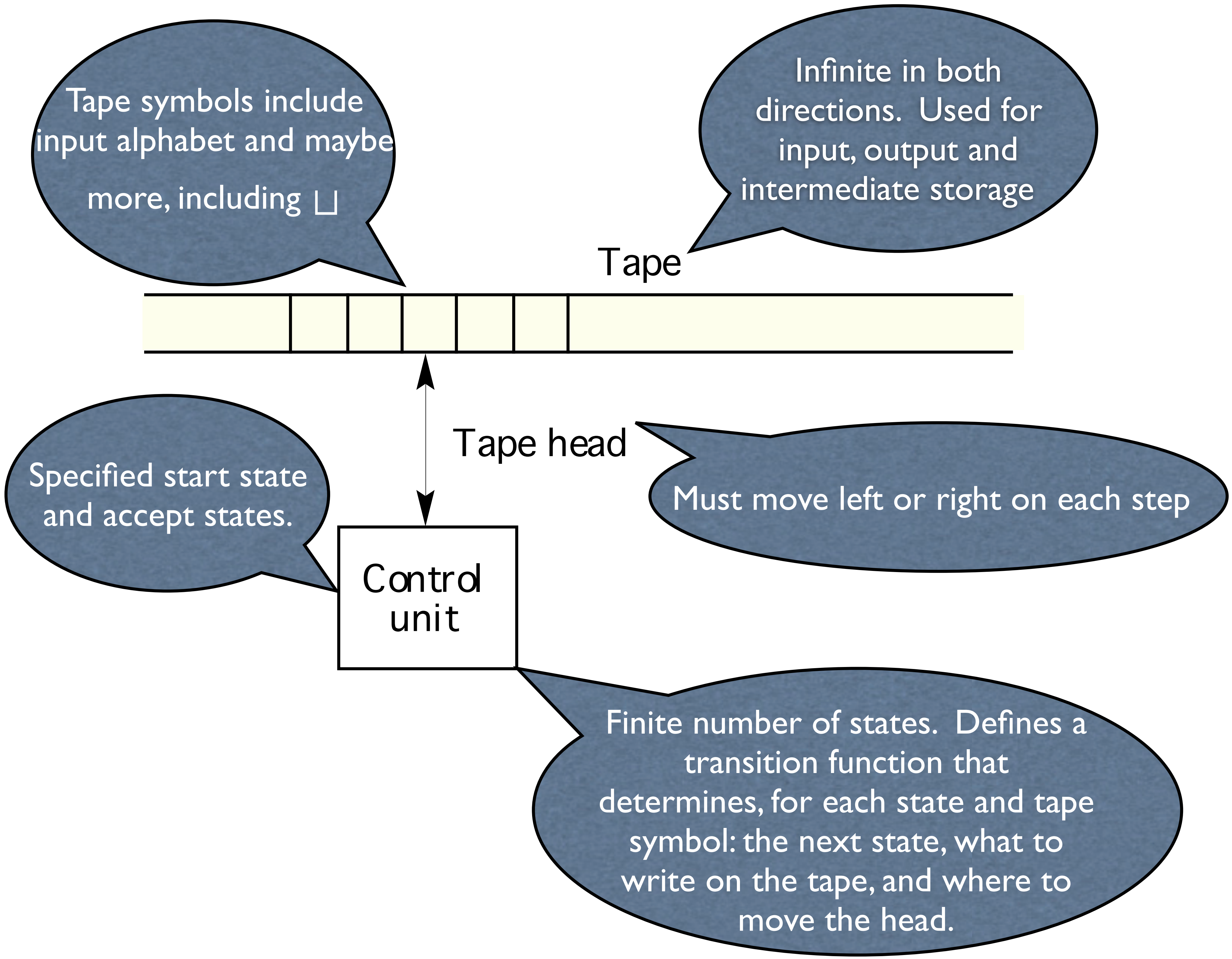

Tape
Tape head
Control
unit


Infinite in both directions. Used for input, output and intermediate storage
Tape
Tape head
Control unit


Infinite in both directions. Used for input, output and intermediate storage
Tape
Tape head
Must move left or right on each step
Control unit

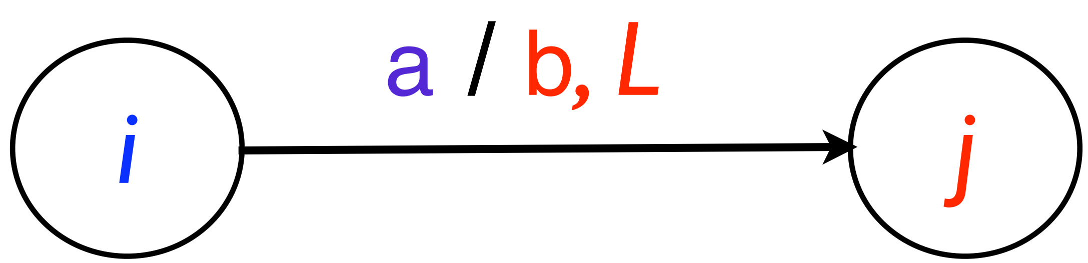

Infinite in both directions. Used for input, output and intermediate storage
Tape
Tape head
Must move left or right on each step
Control unit
Finite number of states. Defines a transition function that determines, for each state and tape symbol: the next state, what to write on the tape, and where to move the head.


Infinite in both directions. Used for input, output and intermediate storage
Tape
Tape head
Must move left or right on each step
Specified start state and accept states.
Control unit
Finite number of states. Defines a transition function that determines, for each state and tape symbol: the next state, what to write on the tape, and where to move the head.


Tape symbols include input alphabet and maybe more, including ⊔
Infinite in both directions. Used for input, output and intermediate storage
Tape
Tape head
Specified start state and accept states.
Must move left or right on each step
Control unit
Finite number of states. Defines a transition function that determines, for each state and tape symbol: the next state, what to write on the tape, and where to move the head.

Turing machine transitions

Turing machine state diagrams have transitions like this:



IALC uses
a/b →
or
a/b ←

meaning:

if machine is in state i, and the symbol under the head is a, then:

the machine can write b, move the head one cell Left, and enter state j.

For simplicity we write a / *L* for a / a, *L*

Portland State University

- Assumptions/Conventions:
  - The input is on the tape before the machine starts
  - All other cells on the tape contain the special blank symbol ⊔
  - The head starts at the left-hand end of the input
  - There is a designated start state
  - There is a set of designated accept states
  - The machine *halts* when no transition is possible
  - We never allow any transitions from the accept states
  - Anything else?

A Turing machine can *recognize* a language

- A Turing machine accepts a string if
  - after starting with the string on the tape,
  - the machine eventually enters an accept state
- How can a TM fail to accept?
  - It can halt in some non-accepting state; OR
  - it can run forever!

Recall: a machine *recognizes* a language if it accepts all and only the strings in that language

n-ary addition as a Language

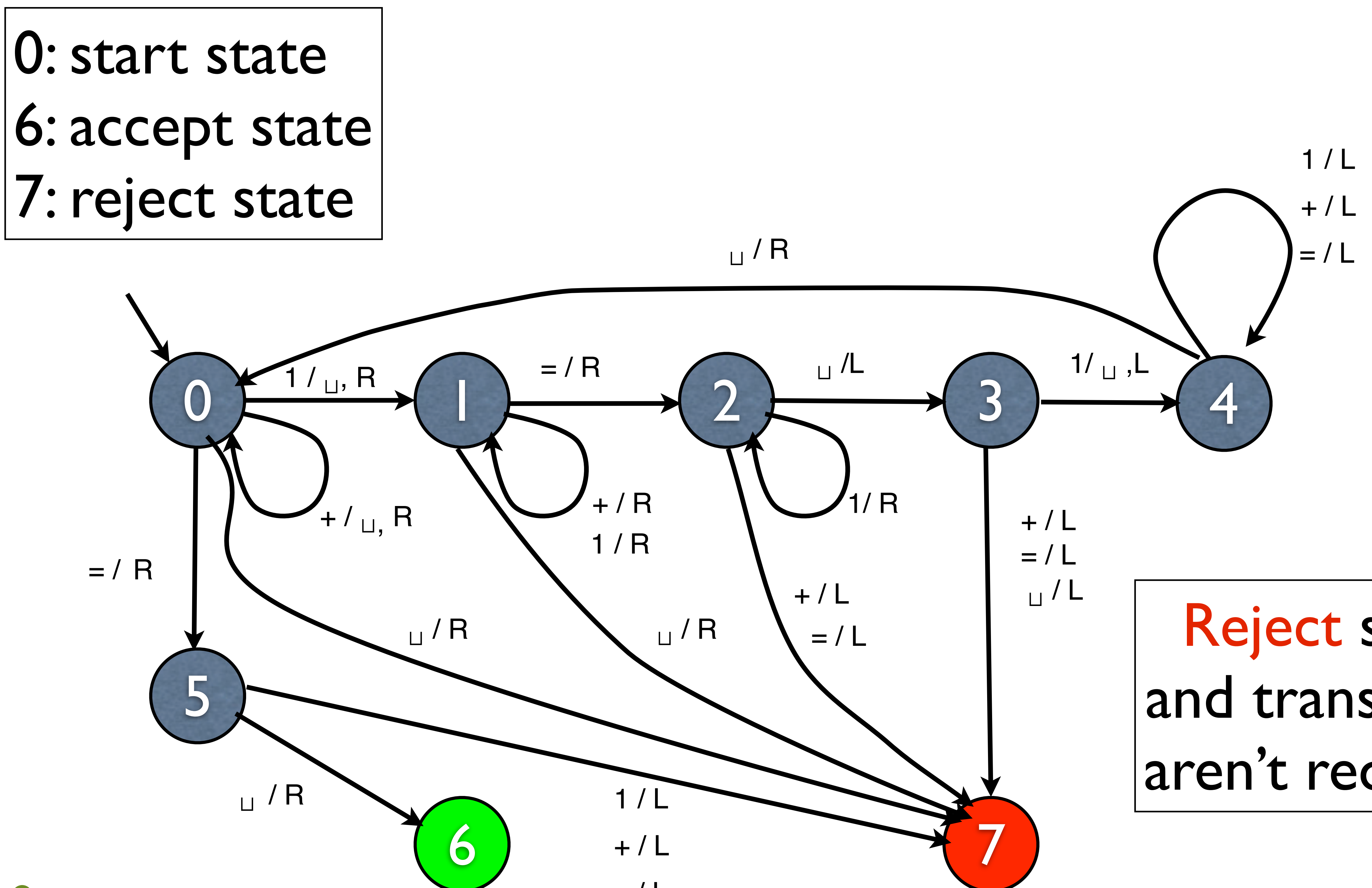
- Contains strings like:
  - 1+11=111
  - 1+1+11+111+1=11111111
- Does not contain strings like:
  - 111
  - 1+1=1

Recognizing *n*-ary addition

- Let L = $\{1^{k1}+1^{k2}+...+1^{kn}=1^{k1+k2+...+kn}\}$ over alphabet {1,+,=}
- Machine starts at left end of tape and loops:
  - Look at current symbol
  - If it's a 1, change to blank, move to right end
    - if that is a 1, change it to blank, return to left end of input, repeat loop
    - otherwise, reject
  - If it's a +, change to blank, move right one; repeat loop
  - If it's a =, move right: if next symbol is blank, accept; otherwise reject

Portland State UNIVERSITY

Machine for *n*-ary addition

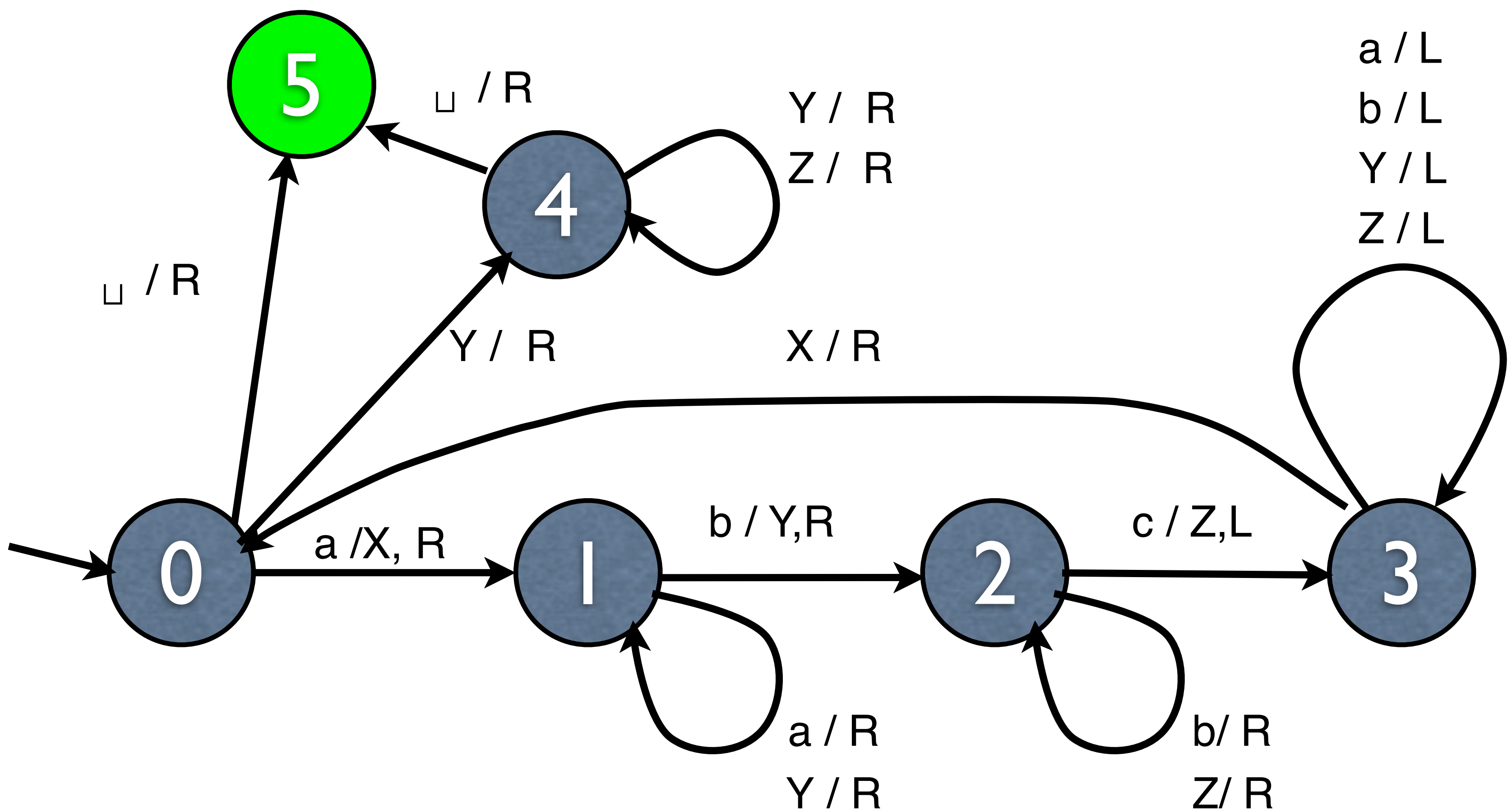


Reject state and transitions aren't required

Idea for recognizing $\{a^n b^n c^n\}$

- Starting at left-hand of input, if current cell is empty, accept.
- Otherwise, if cell contains an a, change this to an X and move right looking for a corresponding b.
- If b is found, change it to a Y and move right looking for a corresponding c.
- If c is found, change it to a Z and return to left to start over again.
- If there are no more a's, scan to the right making sure that there are no more b's or c's either.

Machine: Recognizing $\{a^n b^n c^n\}$



state 0: looking for an a
state 1: looking for matching b
state 2: looking for matching c
state 3: returning to rightmost X
state 4: checking there's no b or c
state 5: accept

Compare to Pushdown Automata

1. A Turing machine reads its input and does "scratch work" on the same tape
2. The read-write head can move anywhere on the tape at any time

Formal Definition

- A Turing machine M is defined as a 7-tuple: $M = (Q,\Sigma,\Gamma,\delta,q_0,\sqcup,F)$ where:
  - Q is a set of states
  - Σ is the input alphabet not containing $\sqcup$
  - Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
  - $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ is the transition function
    - This is a **partial** function, not necessarily defined on all inputs
  - $q_0 \in Q$ is the start state
  - $\sqcup$ is the blank symbol
  - $F \subseteq Q$ is the set of accept states

TM Configurations

- The behavior of a TM at each step is governed by its **configuration**
  - the current state
  - the current tape contents
  - the current tape head location
- We write *u q v* for the configuration where
  - *q* is the current state
  - *uv* is the current tape contents (with blanks to the left and right)
  - the first symbol of *v* is under the tape head

Portland State UNIVERSITY

TM Acceptance, Formally

- Configuration C_i **yields** configuration C_j if the TM can legally go from C_i to C_j in a single step using δ
- M **accepts** *w* if there is a sequence of configurations $C_1, C_2, \ldots, C_n$ where
  - C_1 is the **start** configuration q_0 *w*
  - C_i yields C_{i+1} for $i = 1, \ldots, n-1$.
  - C_n is any configuration with state $\in F$

Recognizable vs. Decidable

- A language L is Turing recognizable if some Turing machine recognizes it.
  - Some strings not in L may cause the TM to loop
  - Turing recognizable = recursively enumerable.
- A language L is Turing decidable if some Turing machine decides it
  - To decide is to return a definitive answer; the TM must halt on all inputs
  - Turing decidable = decidable = recursive.
- We'll see some recognizable but undecidable languages later on.

Decidable Language

$$A_1 = \{0^{2^n} | n \geq 0\}$$

- Decidable by M_1:
  1. Sweep right, crossing off every other 0
  2. if in (1) there was a single 0, *accept*
  3. otherwise, if in (1) there was an odd number of 0s, halt
  4. return the head to the left, and repeat from (1)

State diagram for M_1

In State	Reading	Write	Move	New State
0	#	#	R	0
0	0	0	R	7
1	#	#	L	4
1	0	0	R	2
1	x	x	R	1
2	#	#	R	9
2	0	x	R	1
2	x	x	R	2
4	#	#	R	0
4	0	0	L	4
4	x	x	L	4
7	#	#	L	h
7	0	x	R	1
7	x	x	R	7

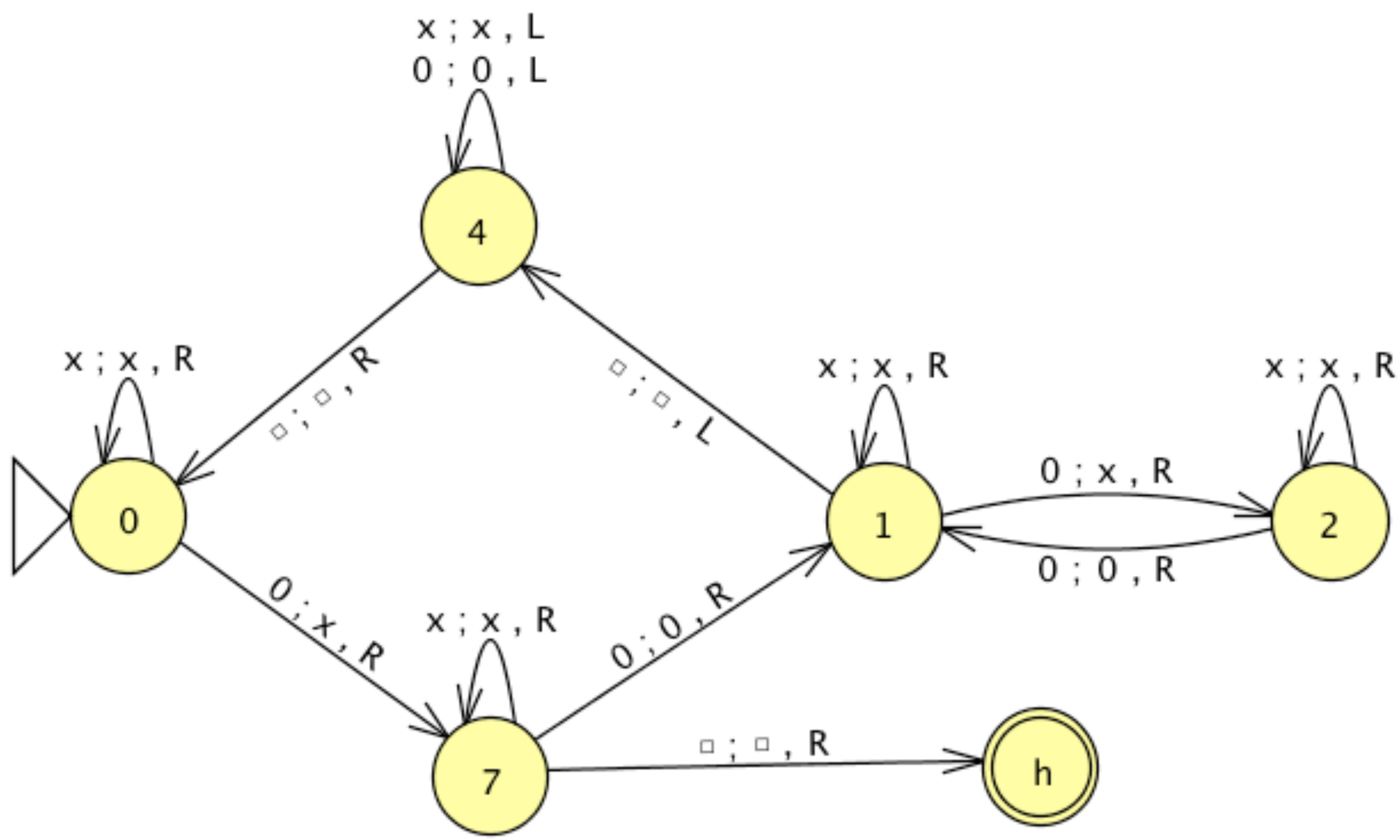
State 0: I'm looking at first input symbol

State 7: I've seen exactly one 0 on this pass

State 1: I've seen an even number of 0s on this pass

State 2: I've seen an odd number of 0s on this pass

State 4: I'm skipping back to the start



JFLAP version: TM for 0^s^n

TM as Transducer

- Because the TM leaves its tape behind when it halts, it can also be viewed as a *transducer* that turns input into output
- Example: TM that does unary multiplication, turning 111x11= into 111x11=111111

Unary Multiplication TM

- Does multiplication by repeated addition:
  - copies the 1s from the second number to the end of the tape
  - repeats this for each 1 in the first number

q0: looking at first unused symbol in input

q1: found a 1 in the first factor; skipping to the x symbol

q2: found x

q4: found a 1 in the second factor; changed it to Y; skipping to the =

q3: used all the 1s in the second factor; convert the Ys back to 1s

q5: skipping to end of tape

q6: skipping back to =

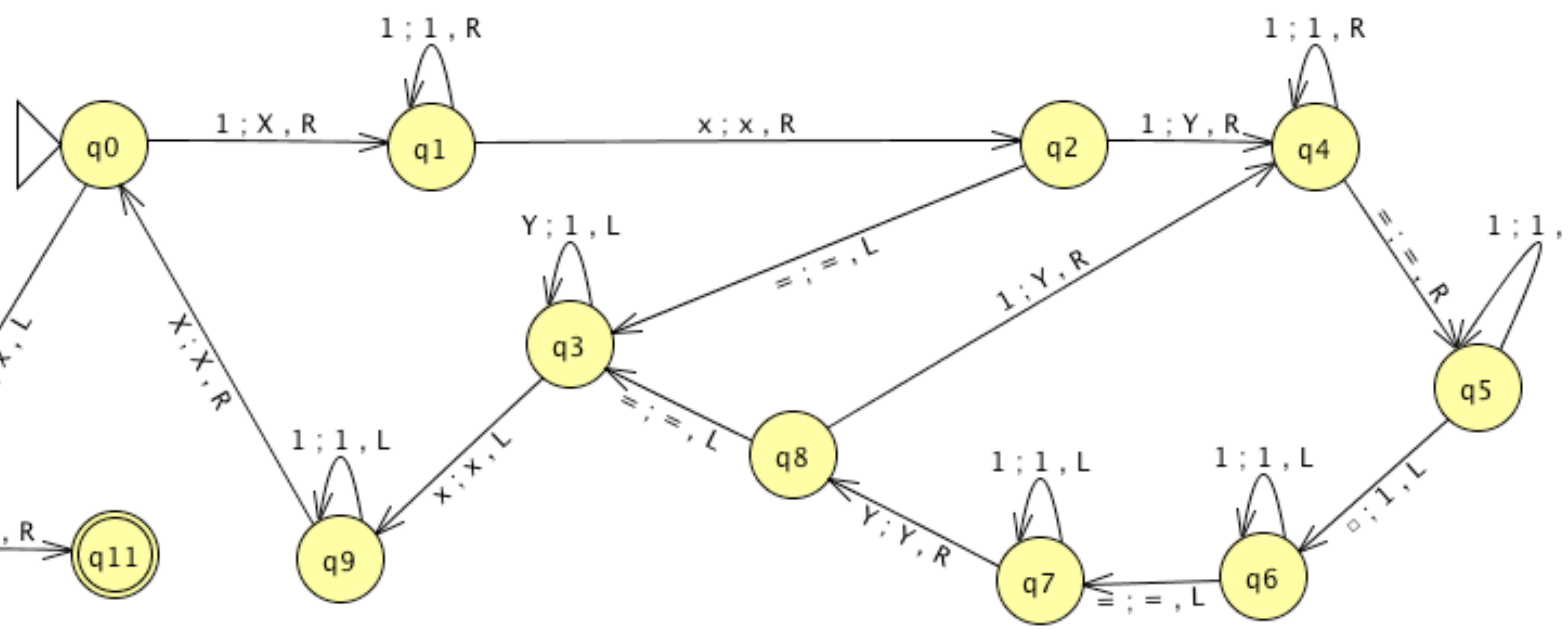
q7: skipping back over second factor

q8: found Y marking previously used 1 from second factor

q9: skip back over 1's in first factor

q12: consumed all of the first factor; convert xs back to 1s

q11: found ␣



JFLAP file: TM for x times y