

CS21

Decidability and Tractability

Lecture 8

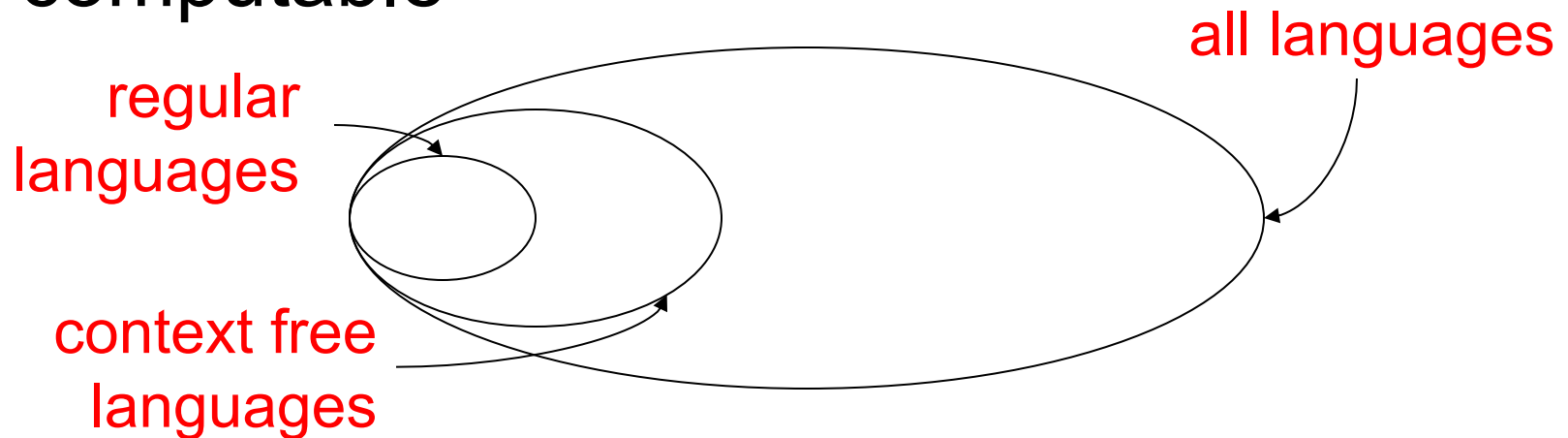
January 24, 2018

Outline

- Turing Machines and variants
 - multitape TMs
 - nondeterministic TMs
- Church-Turing Thesis

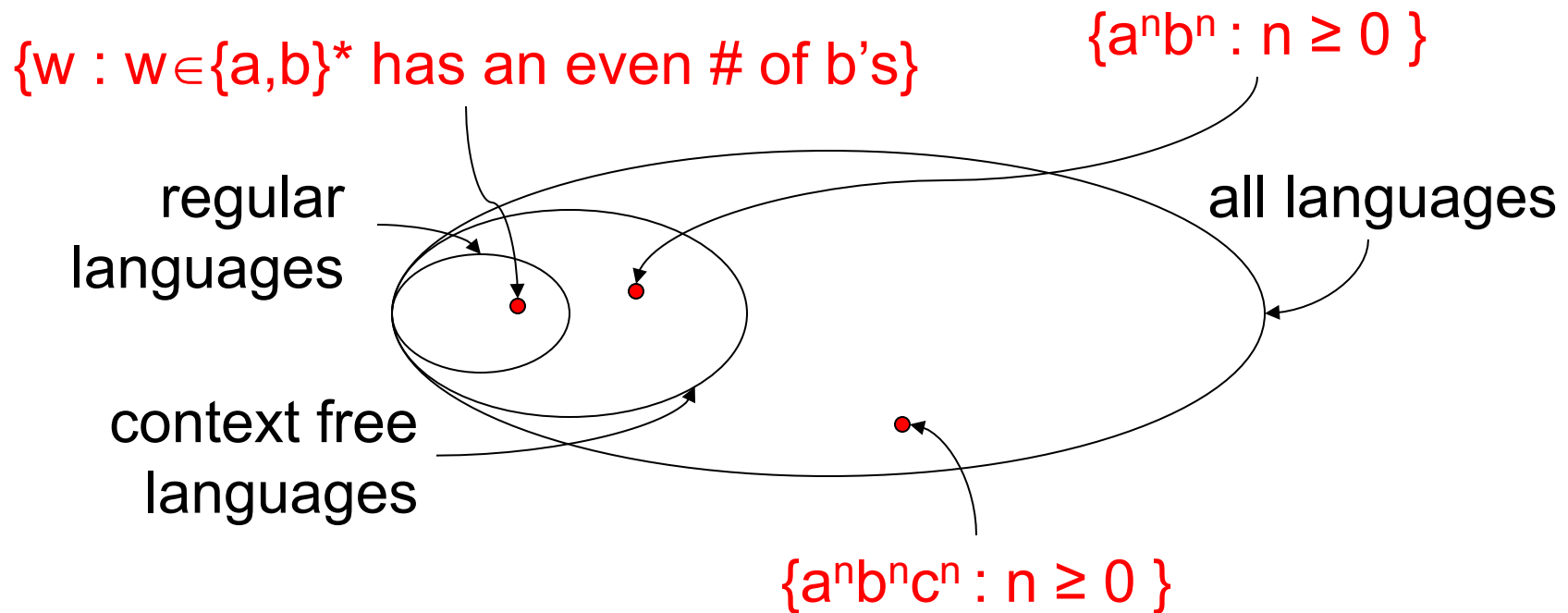
So far...

- several models of computation
 - finite automata
 - pushdown automata
- fail to capture our intuitive notion of what is computable



So far...

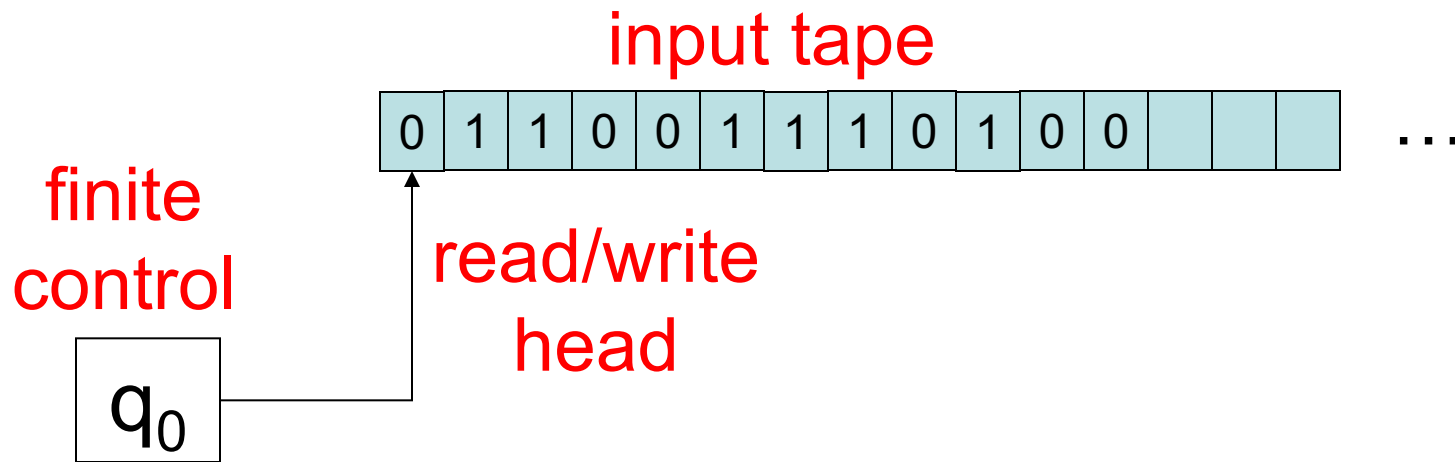
- We proved (using constructions of FA and NPDAs and the two pumping lemmas):



A more powerful machine

- limitation of NPDA related to fact that their memory is stack-based (last in, first out)
- What is the **simplest** alteration that adds general-purpose “memory” to our machine?
- Should be able to recognize, e.g., $\{a^n b^n c^n : n \geq 0\}$

Turing Machines



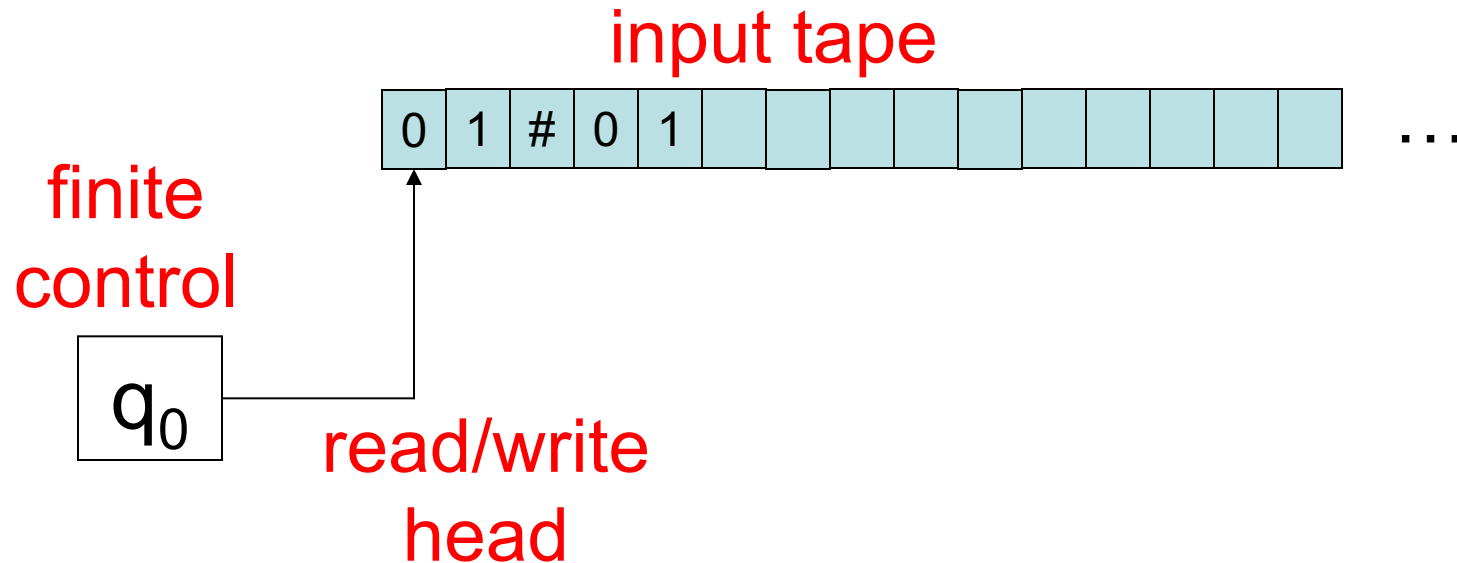
- New capabilities:
 - infinite tape
 - can read OR write to tape
 - read/write head can move left and right

Turing Machine

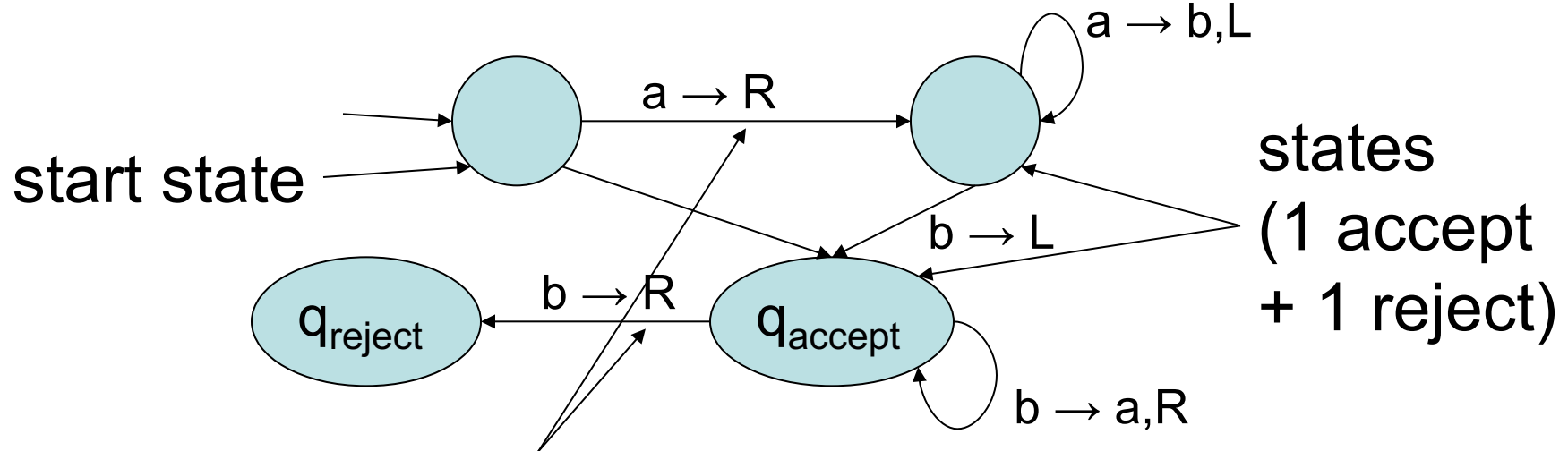
- Informal description:
 - input written on left-most squares of tape
 - rest of squares are blank
 - at each point, take a step determined by
 - current symbol being read
 - current state of finite control
 - a step consists of
 - writing new symbol
 - moving read/write head left or right
 - changing state

Example Turing Machine

language $L = \{w\#w : w \in \{0,1\}^*\}$



Turing Machine diagrams

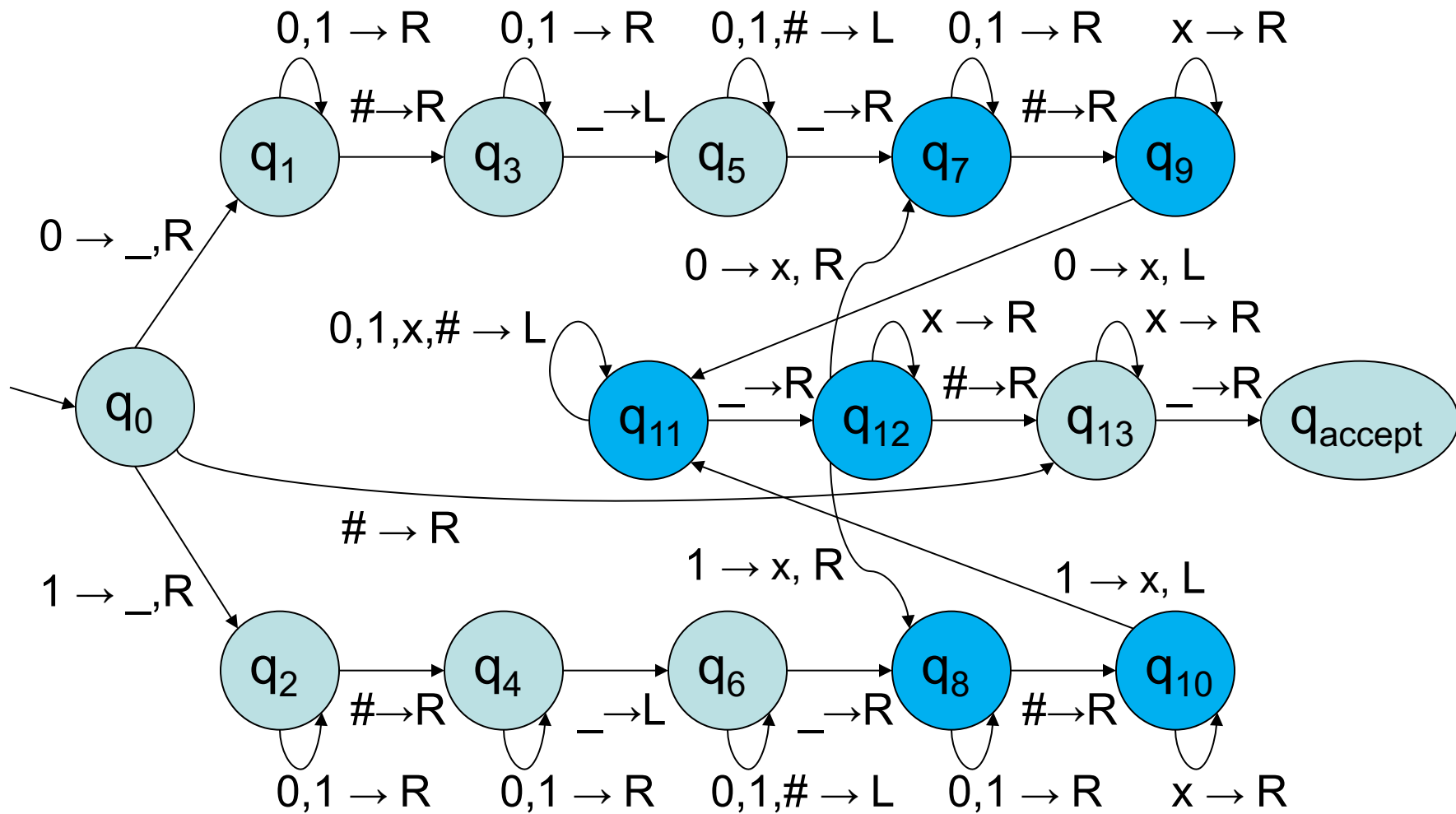


transition label: (tape symbol read \rightarrow
tape symbol written, direction moved)

- $a \rightarrow R$ means “read a, move right”
- $a \rightarrow L$ means “read a, move left”
- $a \rightarrow b, R$ means “read a, write b, move right”

“_” means
blank tape
square

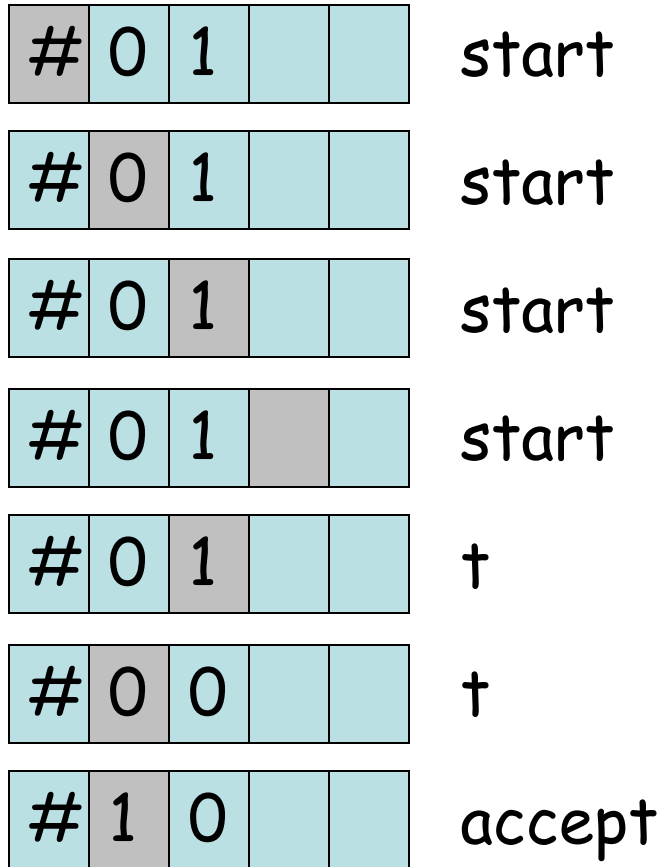
Example TM diagram



TM formal definition

- A TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:
 - Q is a finite set called the **states**
 - Σ is a finite set called the **input alphabet**
 - Γ is a finite set called the **tape alphabet**
 - $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is a function called the **transition function**
 - q_0 is an element of Q called the **start state**
 - $q_{\text{accept}}, q_{\text{reject}}$ are the **accept** and **reject states**

Example TM operation



program
for "binary
successor"

q	σ	$\delta(q, \sigma)$
start	0	(start, 0, R)
start	1	(start, 1, R)
start	_	(t, _, L)
start	#	(start, #, R)
t	0	(accept, 1, -)
t	1	(t, 0, L)
t	#	(accept, #, R)

TM configurations

- At every step in a computation, a **configuration** determined by
 - the contents of the tape
 - the state
 - the location of the read/write head
- next step completely determined by current configuration
- shorthand: string **uqv** with $u, v \in \Gamma^*$, $q \in Q$

meaning:

- tape contents: uv followed by blanks
- in state q
- reading first symbol of v

TM configurations

- configuration C_1 **yields** configuration C_2 if TM can legally* move from C_1 to C_2 in 1 step
 - notation: $C_1 \Rightarrow C_2$
 - also: “yields in 1 step” notation: $C_1 \Rightarrow^1 C_2$
 - “yields in k steps” notation: $C_1 \Rightarrow^k C_2$
- if there exists configurations D_1, D_2, \dots, D_{k-1} for which $C_1 \Rightarrow D_1 \Rightarrow D_2 \Rightarrow \dots \Rightarrow D_{k-1} \Rightarrow C_2$
 - also: “yields in some # of steps” ($C_1 \Rightarrow^* C_2$)

*Convention: TM halts upon entering q_{accept} , q_{reject}

TM configurations

- Formal definition of “yields”:

$$uaq_i bv \Rightarrow uq_j acv$$

if $\delta(q_i, b) = (q_j, c, L)$, and

$$uaq_i bv \Rightarrow uacq_j v$$

if $\delta(q_i, b) = (q_j, c, R)$

$$(q_i \neq q_{\text{accept}}, q_{\text{reject}})$$

- two special cases:

– left end: $q_i bv \Rightarrow q_j cv$ if $\delta(q_i, b) = (q_j, c, L)$

– right end: uaq_i same as $uaq_i _$

$$\begin{aligned} u, v &\in \Gamma^* \\ a, b, c &\in \Gamma \\ q_i, q_j &\in Q \end{aligned}$$

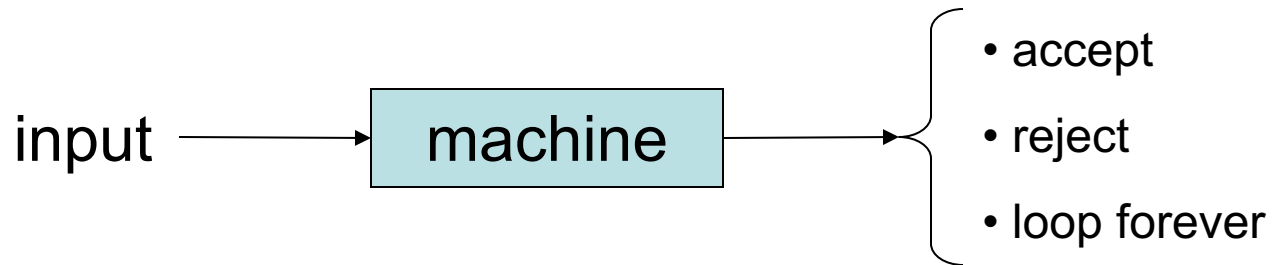
TM acceptance

- start configuration: q_0w (w is input)
- accepting config.: any config. with state q_{accept}
- rejecting config.: any config. with state q_{reject}

TM M accepts input w if there exist configurations C_1, C_2, \dots, C_k

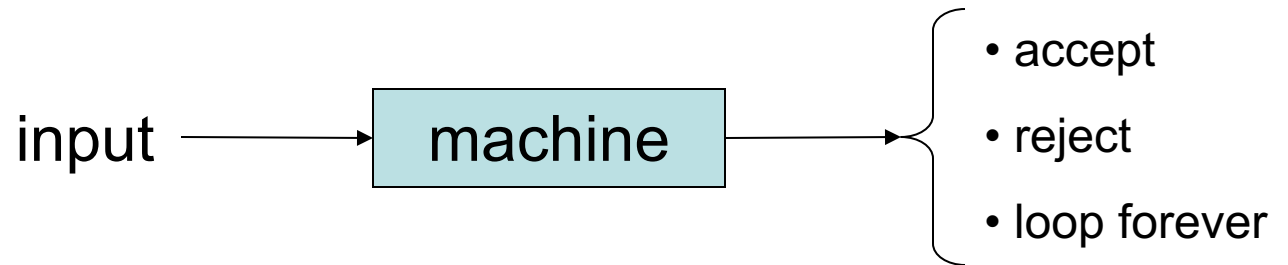
- C_1 is start configuration of M on input w
- $C_i \Rightarrow C_{i+1}$ for $i = 1, 2, 3, \dots, k-1$
- C_k is an accepting configuration

Deciding and Recognizing



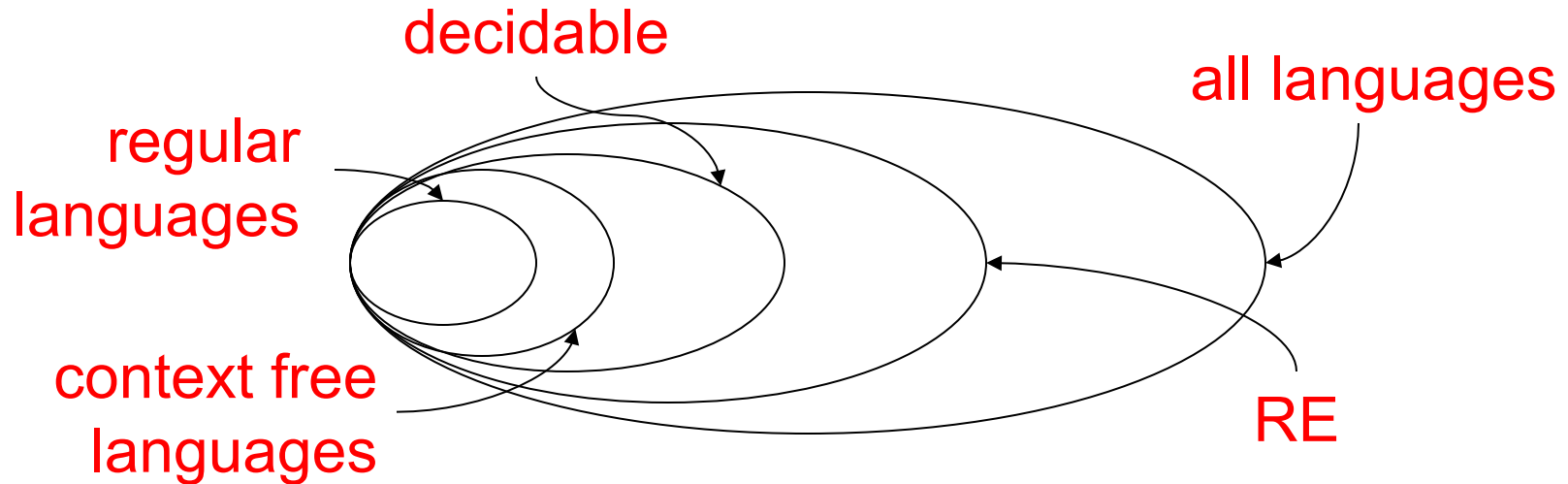
- TM M :
 - $L(M)$ is the language it **recognizes**
 - if M rejects every $x \notin L(M)$ it **decides** L
 - set of languages recognized by some TM is called **Turing-recognizable** or **recursively enumerable (RE)**
 - set of languages decided by some TM is called **Turing-decidable** or **decidable** or **recursive**

Deciding and Recognizing



- TM M :
 - $L(M)$ is the language it **recognizes**
 - if M rejects every $x \notin L(M)$ it **decides** L
 - set of languages recognized by some TM is called **Turing-recognizable** or **recursively enumerable (RE)**
 - set of languages decided by some TM is called **Turing-decidable** or **decidable** or **recursive**

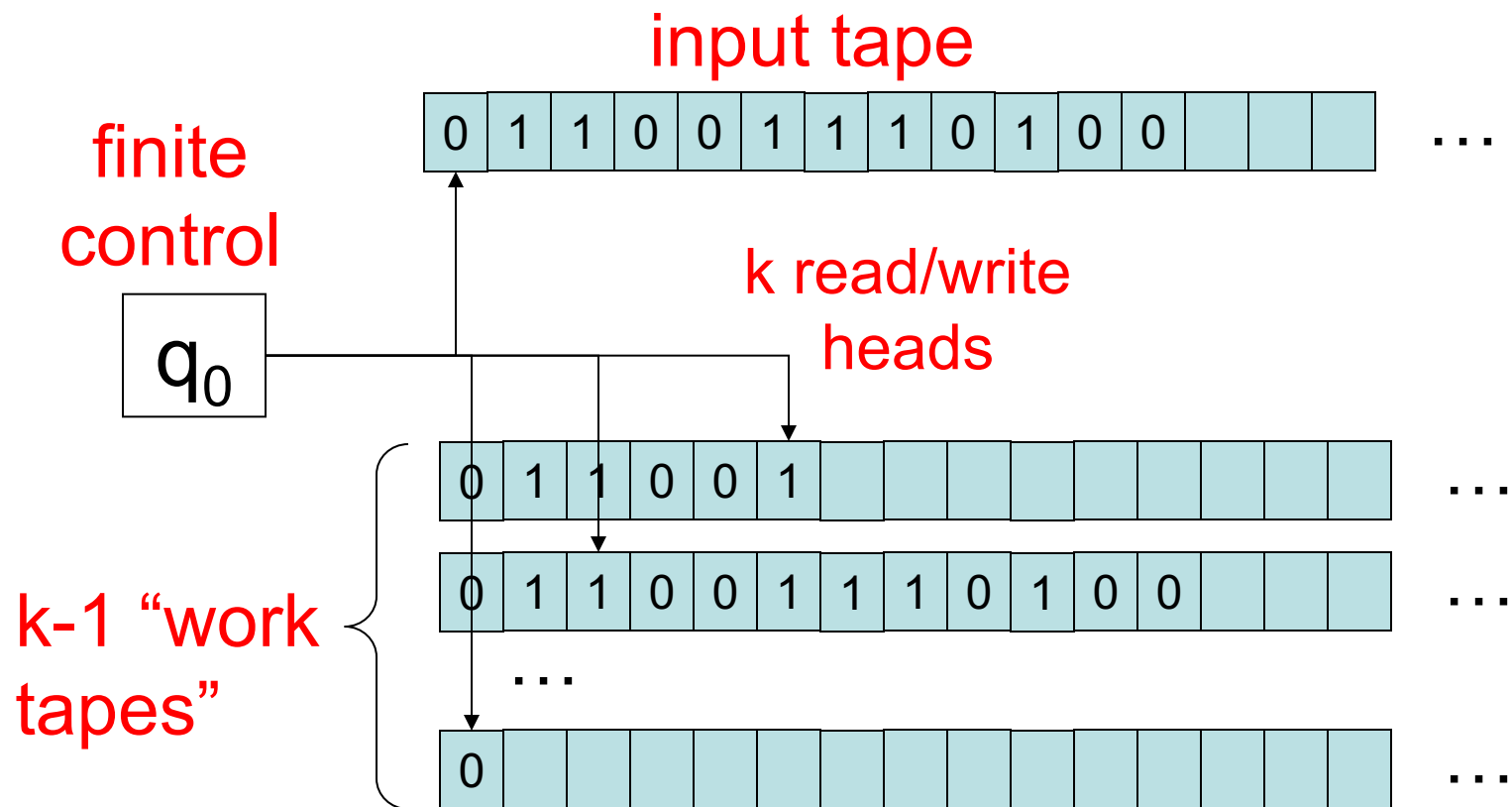
Classes of languages



- We know: regular \subset CFL (proper containment)
- CFL \subset decidable
 - proof?
 - decidable \subset RE \subset all languages
 - proof?

Multitape TMs

- A useful variant: k-tape TM



Multitape TMs

- Informal description of **k-tape** TM:
 - input written on left-most squares of tape **#1**
 - rest of squares are blank **on all tapes**
 - at each point, take a step determined by
 - current **k** symbols being read **on k tapes**
 - current state of finite control
 - a step consists of
 - writing **k** new symbols **on k tapes**
 - moving each of **k** read/write heads left or right
 - changing state

Multitape TM formal definition

- A TM is a 7-tuple

$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ where:

- everything is the same as a TM except the transition function:

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

$$\delta(q_i, a_1, a_2, \dots, a_k) = (q_j, b_1, b_2, \dots, b_k, L, R, \dots, L) =$$

“in state q_i , reading a_1, a_2, \dots, a_k on k tapes, move to state q_j , write b_1, b_2, \dots, b_k on k tapes, move L, R on k tapes as specified.”

Multitape TMs

Theorem: every k -tape TM has an equivalent single-tape TM.

Proof:

- Idea: simulate k -tape TM on a 1-tape TM.