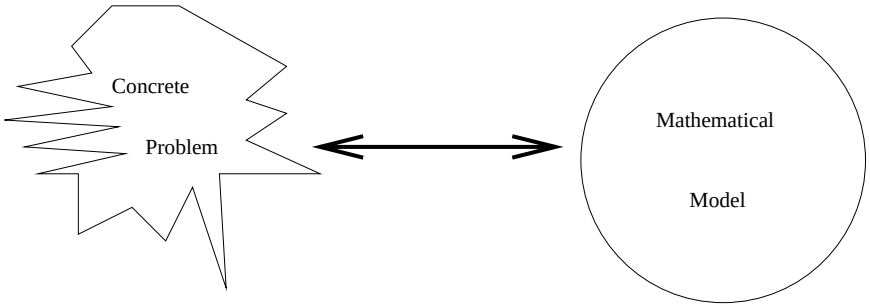
In-depth introduction to main models, concepts of theory of computation:

- **Automata Theory**: warm-up and review
- **Computability**: what can be computed in principle
- **Logic**: how can we express our requirements
- **Complexity**: what can be computed in practice



Formal Models of Computation:

- FA $\equiv$ Regular Language
- PDA $\equiv$ CFL
- TM = all powerful computer
- logical formula

deep understanding of formal models of computation

proofs are important

$M = (Q, \Sigma, \delta, s)$

Q: finite set of states; $s \in Q$

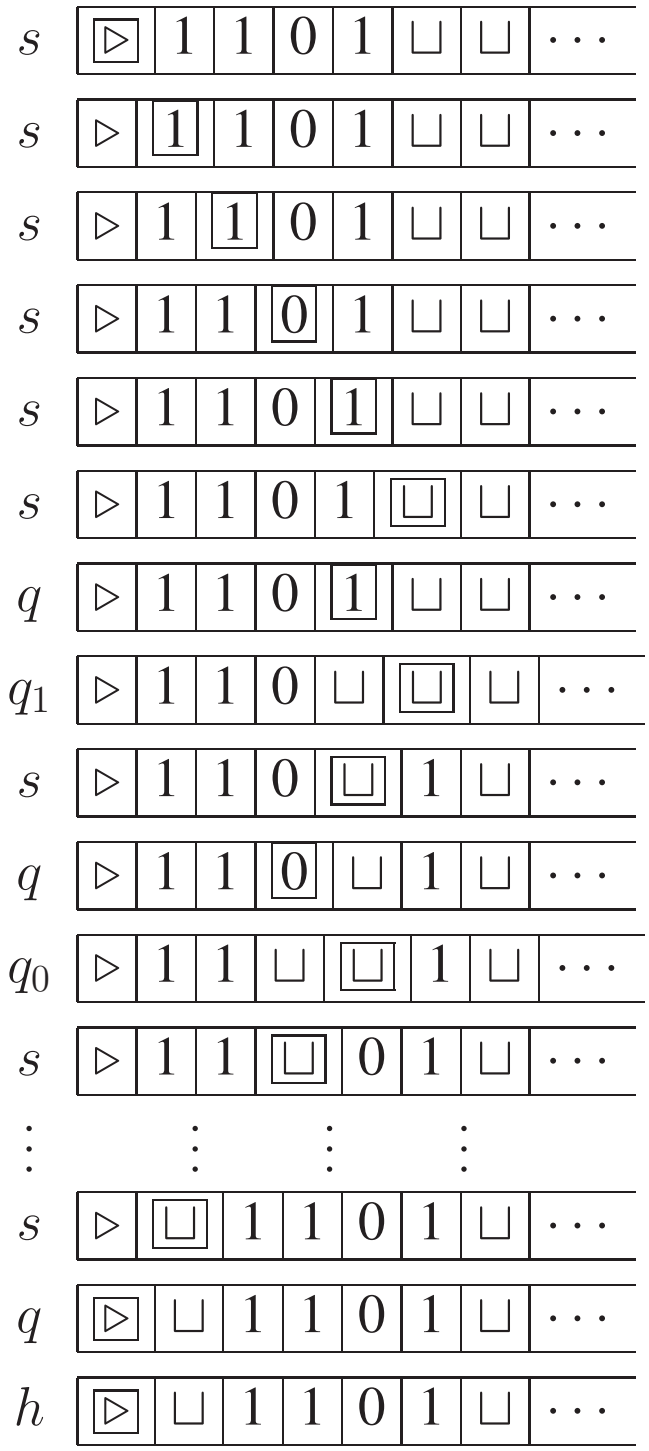
Σ: finite set of symbols; $\triangleright, \sqcup \in \Sigma$

δ: $Q \times \Sigma \rightarrow (Q \cup \{h\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$

s	$\triangleright$	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

mvRt.tm	s	q	q_0	q_1
0	$s, 0, \rightarrow$	$q_0, \sqcup, \rightarrow$		
1	$s, 1, \rightarrow$	$q_1, \sqcup, \rightarrow$		
$\sqcup$	$q, \sqcup, \leftarrow$		$s, 0, \leftarrow$	$s, 1, \leftarrow$
$\triangleright$	$s, \triangleright, \rightarrow$	$h, \triangleright, -$		
comment	find $\sqcup$	memorize & erase	change $\sqcup$ to 0	change $\sqcup$ to 1

	s	q	q_0	q_1
0	$s, 0, \rightarrow$	$q_0, \sqcup, \rightarrow$		
1	$s, 1, \rightarrow$	$q_1, \sqcup, \rightarrow$		
$\sqcup$	$q, \sqcup, \leftarrow$		$s, 0, \leftarrow$	$s, 1, \leftarrow$
$\triangleright$	$s, \triangleright, \rightarrow$	$h, \triangleright, -$		



Hilbert's Program [1900]: Give a complete axiomization of all of mathematics!

Such a complete axiomization would have provided a mechanical procedure to churn out exactly all true statements in mathematics.

This led to active interest in 1930's in the question: **"What is a mechanical procedure?"**

Church: Lambda calculus
Gödel: Recursive function
Kleene: Formal system
Markov: Markov algorithm
Post: Post machine
Turing: Turing machine

Fact: The above models are all exactly equivalent.

(And they are also equivalent to what is computable by any appropriate formal model of a real computer that has added to it a potentially unbounded amount of secondary storage.)

Church's Thesis: The intuitive idea of "effectively computable" is captured by the precise definition of computable by any of the above models.

Why is the Turing machine as powerful as any other computational model?

Intuitive answer: Imagine any computational device. It has:

- Finitely many states
- Ability to scan limited amount per step: one page at a time
- Ability to print limited amount per step: one page at a time
- Next state determined by current state and page currently being read

Note: Without the potentially infinite supply of tape cells, paper, extra disks, extra tapes, etc. we have just a (potentially huge) **finite state machine**.

The PC on your desk, with 20 GB of hard disk is a finite state machine with over $2^{160,000,000,000}$ states!

This is better modeled as a TM with a bounded number of states, and a potentially infinite tape.

Definition 1.1 A string $w \in \Sigma^\star$ is a *palindrome* iff it is the same as its reversal, i.e., $w = w^R$. □

Examples of palindromes:

- 101
- 1101001011
- ABLE WAS I ERE I SAW ELBA
- A MAN A PLAN A CANAL PANAMA

Proposition 1.2 *The set of PALINDROMES (over a fixed alphabet, Σ) is a recursive set.*

Proof: [Verbal sketch:]

□

Fact 1.3 *Time $O(n^2)$ is necessary and sufficient for a one-tape Turing machine to accept the set, PALINDROMES.*

Proof: Sufficiency is obvious. To prove necessity do problems 2.8.4, 2.8.5 from [Papadimitriou, *Computational Complexity*]. □

Definition 1.4 A k-tape Turing machine, $M = (Q, \Sigma, \delta, s)$

Q: finite set of states; $s \in Q$

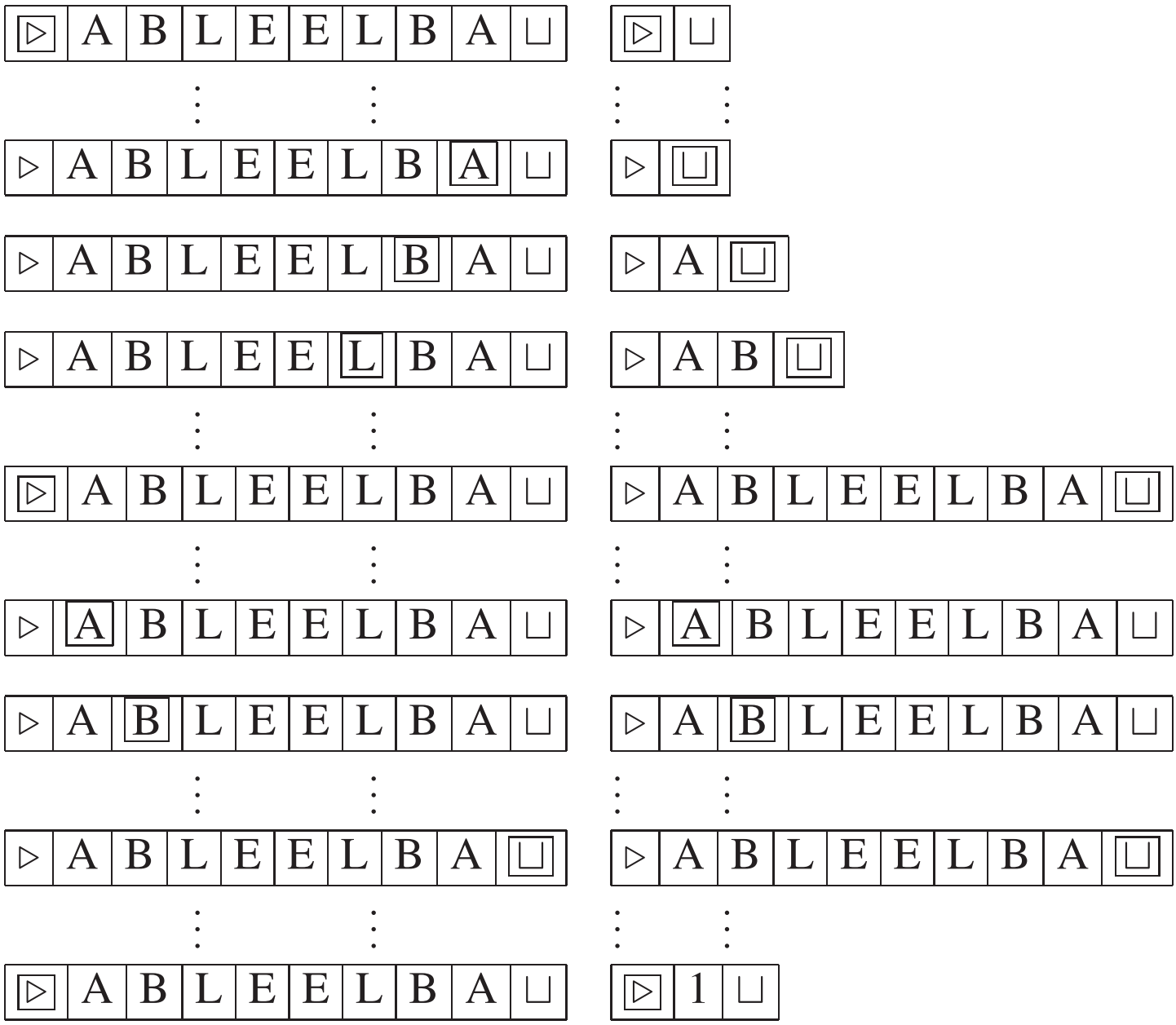
Σ: finite tape alphabet;

δ: $Q \times \Sigma^k \rightarrow (Q \cup \{h\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$

□

Proposition 1.5 *PALINDROMES can be accepted in* DTIME$[n]$ *on a 2-tape TM.*

Proof: (that PALINDROMES $\in$ DTIME$[n]$)



□