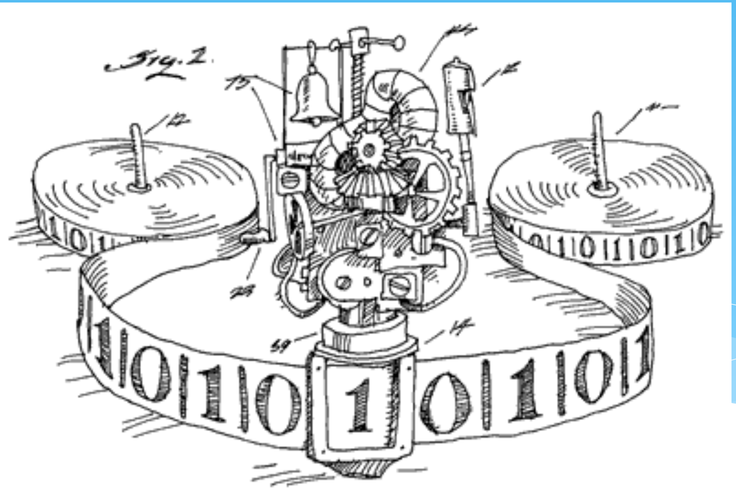
EECS 376: Foundations of Computer Science

Chris Peikert

30 January 2023



Today's Agenda

* Recap: Strings, Languages, DFAs & their abilities
* **Turing Machines** (TMs) and **Church-Turing thesis**
* Pseudocode vs TMs
* Deciders and decidability

Alphabets, Strings, Languages

- An ***alphabet*** is a finite set of characters, usually denoted Σ
  - Typically implicit, e.g., ASCII characters or binary $\{0,1\}$
- A (Σ-)***string*** is a finite sequence of characters from Σ
  - The ***length*** of a string x (# chars) is denoted $|x|$
  - The ***empty string*** is denoted ε; it has length 0
- A (Σ-)***language*** is (possibly infinite) set of (Σ-)strings: $L \subseteq \Sigma^*$
  - The language of all strings is denoted Σ^*
- **Example:** $\Sigma = \{0,1\}$, $\Sigma^* = \{\varepsilon, 0, 1, 00, \ldots\}$, $|010| = 3$, $0^3 1^2 = 00011$

What is a “problem”?

- We consider ***decision problems***, where the goal is to ***decide*** if a given object has a certain property

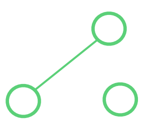
Input
376
Output
No

Is the given integer prime?

Input
abba
Output
Yes

Is the given string a palindrome?

Input



Output
No

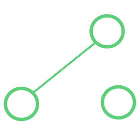
Is the given graph connected?

…The list goes on!

Languages & their Membership Problems

- *Any* *finite* object Z can be ***encoded*** as a *finite* ***string*** $\langle Z \rangle$ (e.g., in ASCII, or binary, as in a computer).
- In this view, a property is a *set of strings*: *a* ***language***

Input



Output

No

Is the given graph connected?

Input

```
010
100
000
```

Output

No

Is the given string in the language L_{CONN}?

= set of strings that encode a connected graph

The ***membership*** **(or,** ***decision*****)** ***problem*** for a language L:

Given a string x, ***decide*** if $x \in L$ (say yes/no, accept/reject, etc.)

What is a "Computer"?

* **Goal:** formalize the notion of a "computer" that can "solve" decision problems—i.e., "decide" languages.

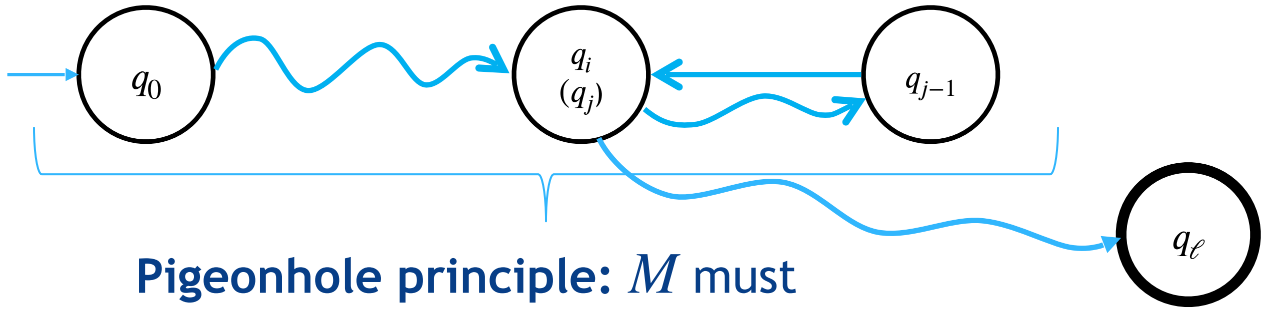
* A <u>D</u>eterministic <u>F</u>inite <u>A</u>utomaton reads the input string one character at a time, and ends in either an ***accept*** or ***reject*** (non-accept) state.
  We say that the DFA ***decides*** language L if it:
  * (i) <u>*accepts*</u> every string $x \in L$, and
  * (ii) <u>*rejects*</u> every string $x \notin L$.

* A language is ***regular*** if some DFA decides it. **Q:** Is *every* language regular?

* **Theorem:** <u>No</u> DFA decides $\{0^k 1^k \mid k \geq 0\}$.

No DFA decides $\{0^k 1^k \mid k \geq 0\}$

- Suppose that some DFA M decides $\{0^k 1^k \mid k \geq 0\}$.
- Let n = # of states of M, and let $x = 0^{n+1} 1^{n+1}$.
- **Claim:** We can write $x = uwv$ so that M is in the *same state* before and after reading substring $w \neq \varepsilon$.
- M must accept $uwwv \notin \{0^k 1^k \mid k \geq 0\}$. Contradiction!



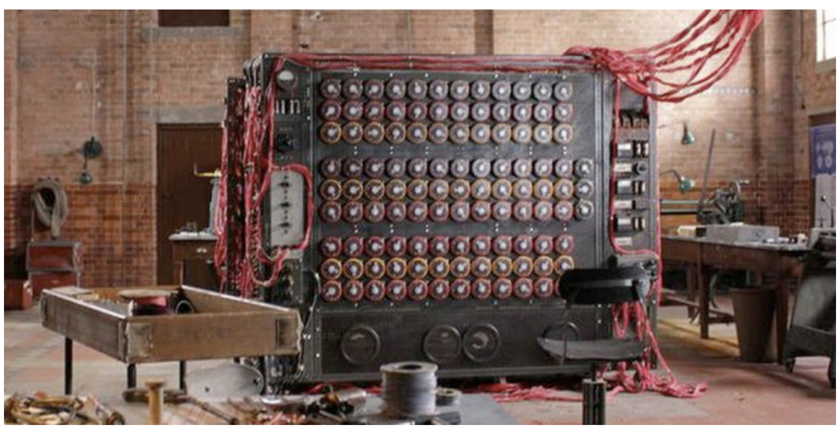
Pigeonhole principle: M must *repeat* a state while reading 0^{n+1}

Various Models of "Computers"

* DFAs
* Pushdown Automata
* Context-free Grammars
* Lambda Calculus
* Turing Machines
* RAM (random access memory) computer
* Quantum Computers
* DNA computers
* …

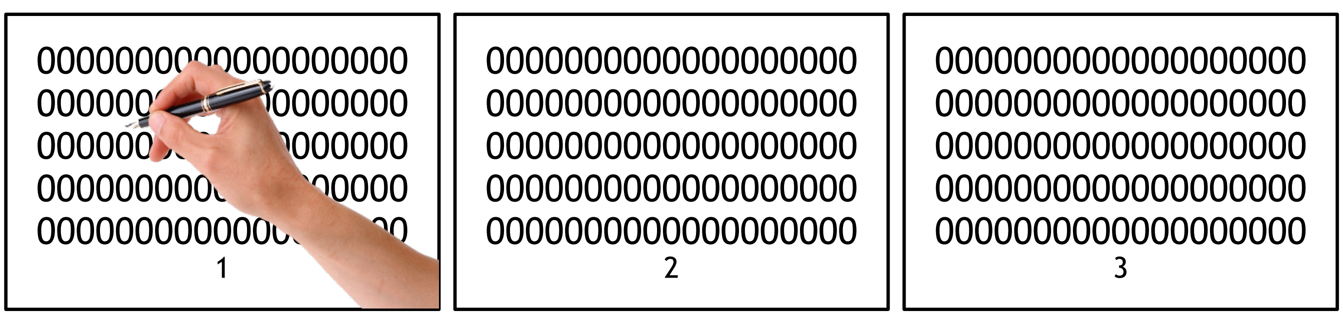
Our Model

Alan Turing (1912-1954):
British pioneering computer scientist
Inventor of the "Turing Machine."



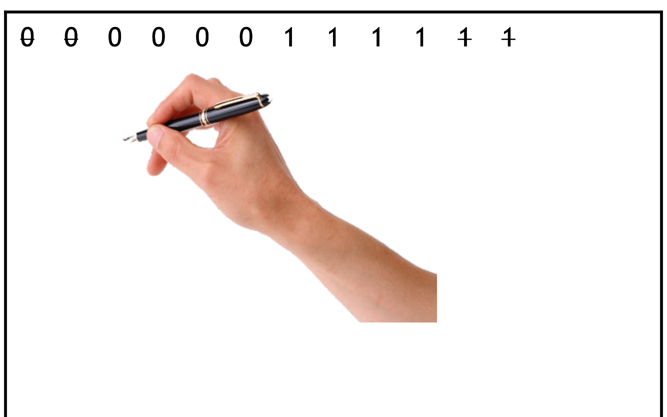
A Thought Experiment

- Imagine you are given a *huge* string x
  - $|x| \gg$ number of neurons in your brain
- The string is written *on ordered pages of paper*, and *you have a pen to write with*
- **Q:** Can you decide if $x \in \{0^k1^k \mid k \geq 0\}$?



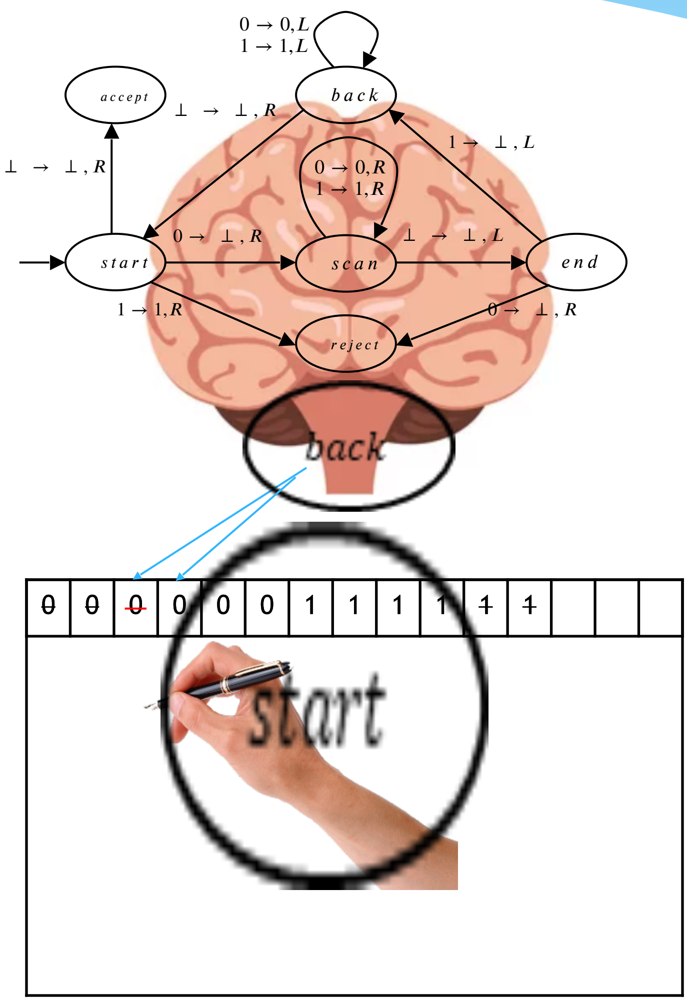
…

How do people solve problems?



- Suppose we're given a huge "input" that's written down
- What's the *bare minimum* that we need to solve the problem?
  - *Brain* to *direct* our efforts
  - *Eyes* (or other sense) to *read* with
  - *Pen* to *write* with
  - *Symbols* to write down

How do people solve problems?



Without loss of generality (?):

- We use only finitely many symbols
- The paper is an *unbounded* (infinite) array of squares that can each store *one* symbol
- At each moment, we look at a *single* square
- We read what's in the square, write an appropriate symbol, then move our gaze to an adjacent square
- (?) Our brain decides what to do next based on what we currently see and what we did so far, but it only has *finite memory*

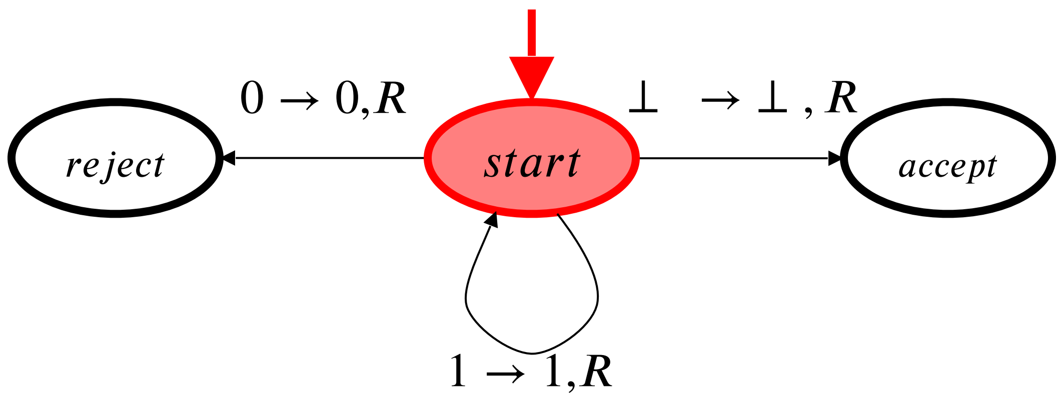
This is a ***Turing Machine*** (*TM*)

TM Example

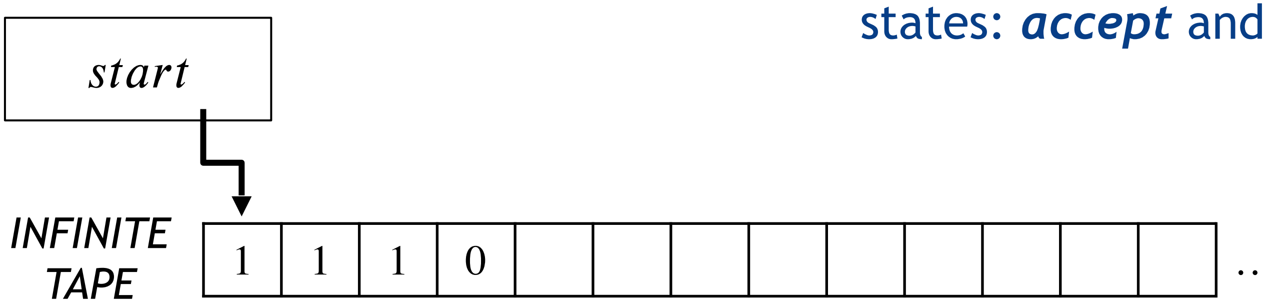
The "brain" of a TM is like a DFA, except it additionally specifies:

- what we *write* and
- whether move *left* or *right*

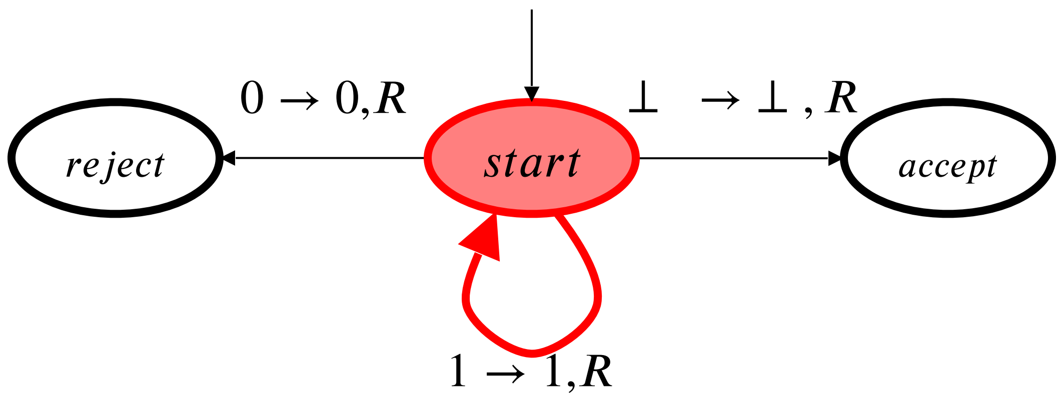
Note: "$a \to b, R$" means if the contents of the cell is a, then write b and move right.



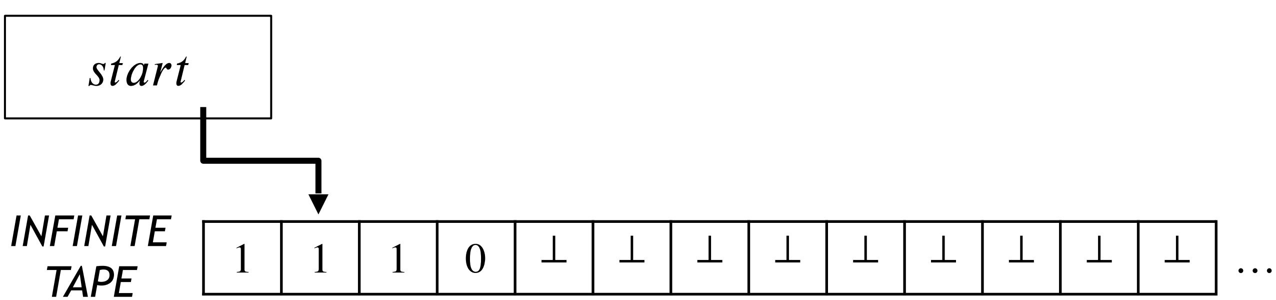
There are *two* special "termination" states: ***accept*** and ***reject***.



TM Example

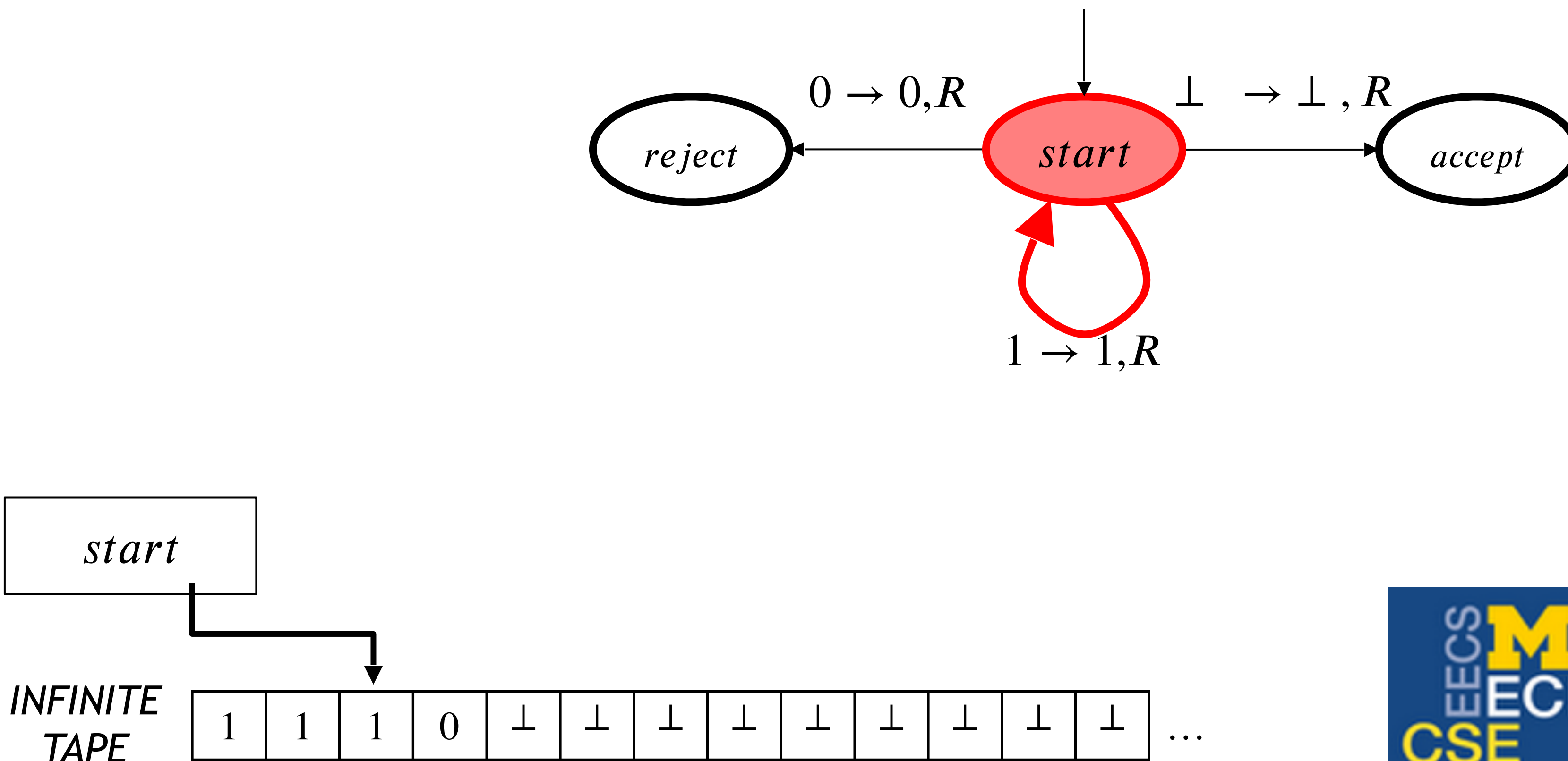


$0 \to 0, R$
$\perp \to \perp, R$
reject
start
accept
$1 \to 1, R$



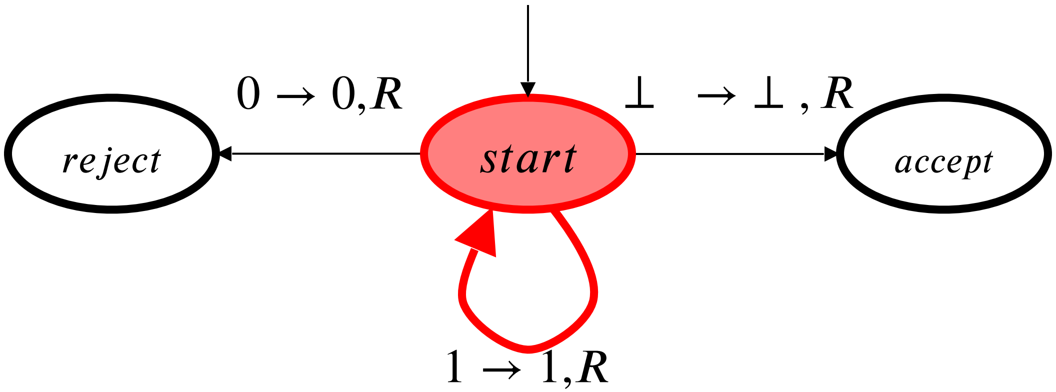
start
INFINITE TAPE
1 1 1 0 ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ …

TM Example

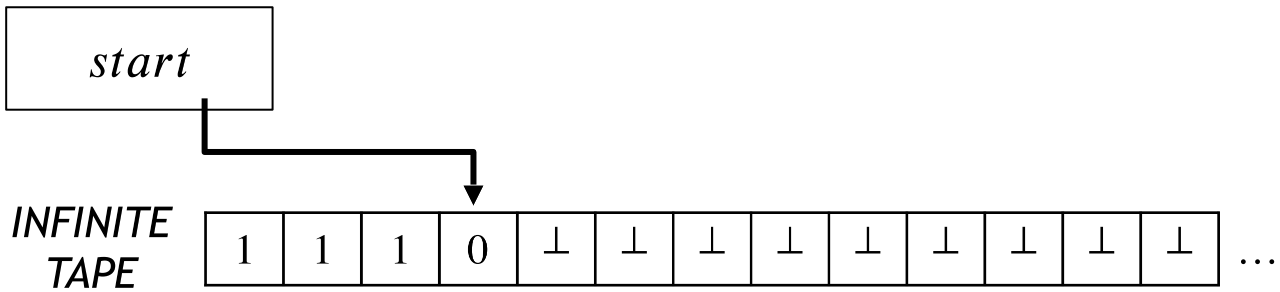


$0 \rightarrow 0, R$
$\perp \rightarrow \perp, R$
reject
start
accept
$1 \rightarrow 1, R$
start
INFINITE TAPE
1 1 1 0 ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ …

TM Example

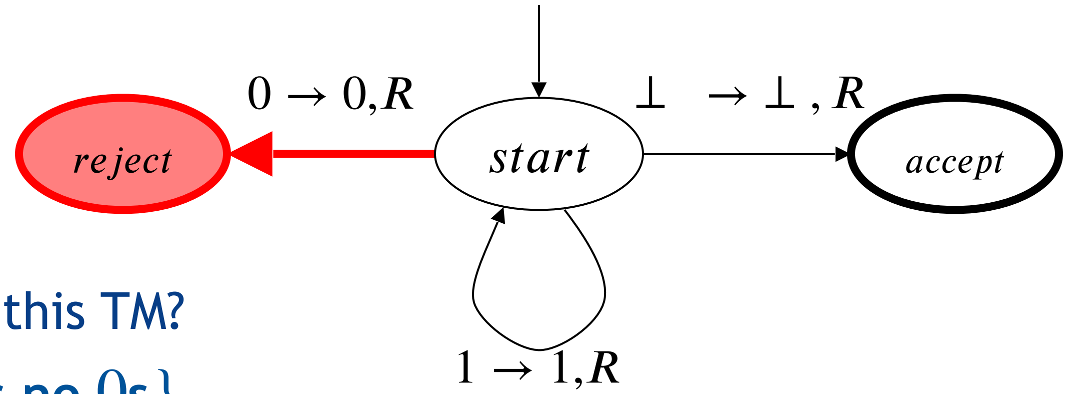


$0 \to 0, R$
$\perp \to \perp, R$
reject
start
accept
$1 \to 1, R$



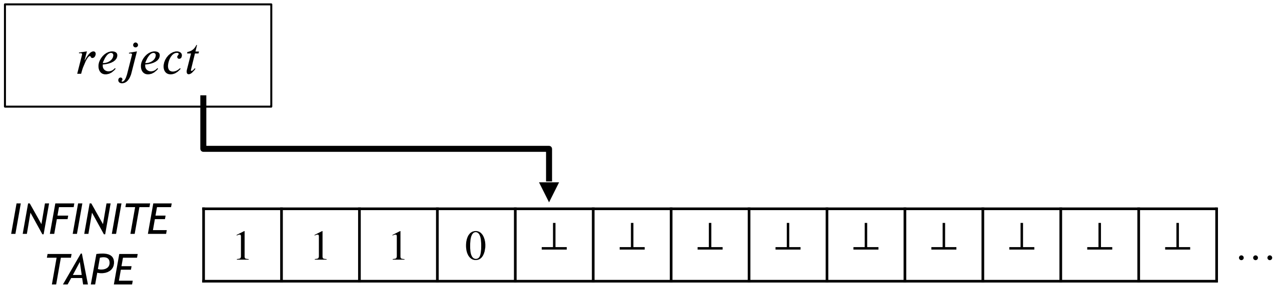
start
INFINITE TAPE
1 1 1 0 ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ …

TM Example

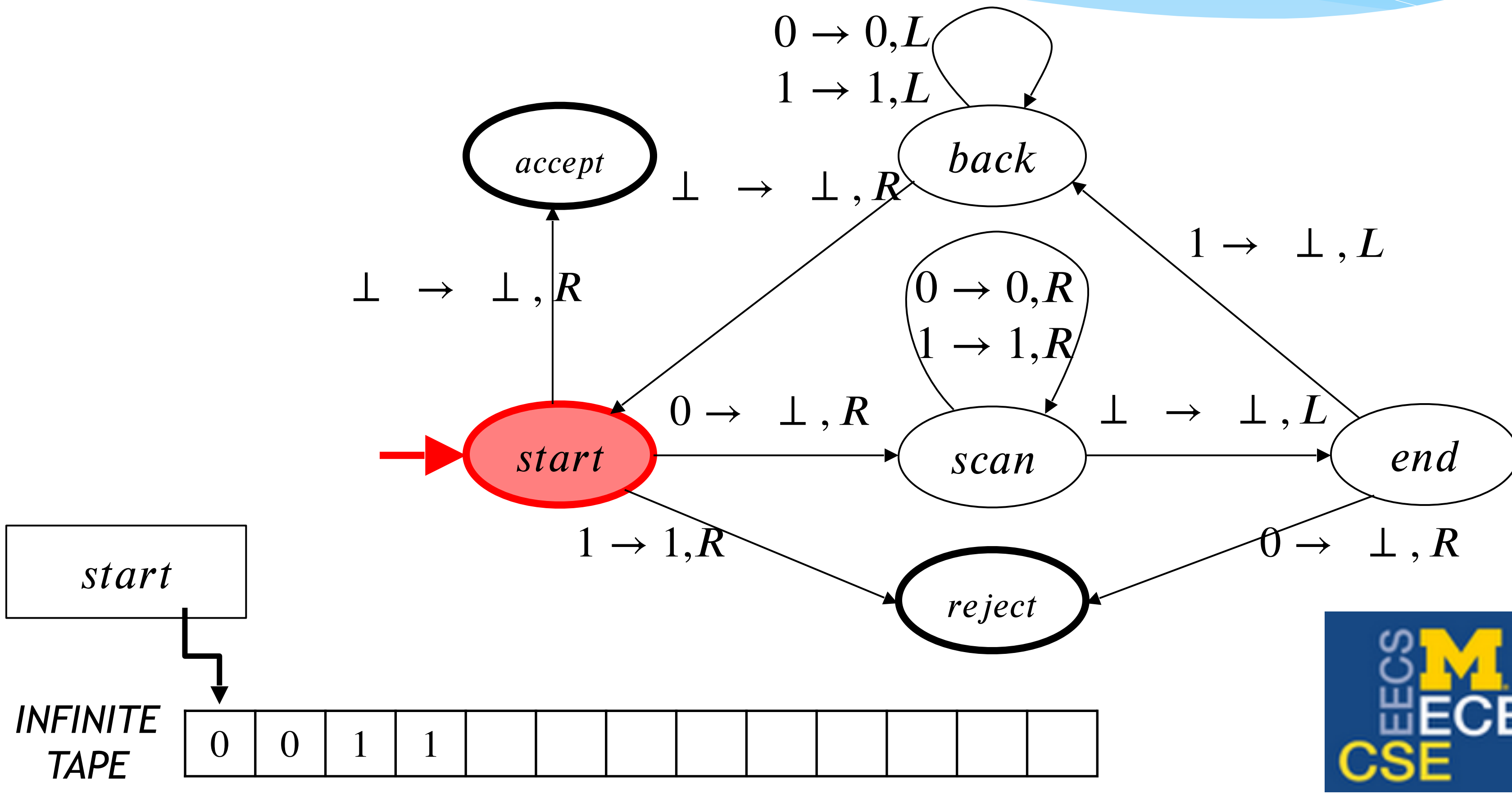


Q: Set of inputs accepted by this TM?

A: $\{x \in \{0,1\}^* \mid x$ **contains no** 0**s**$\}$

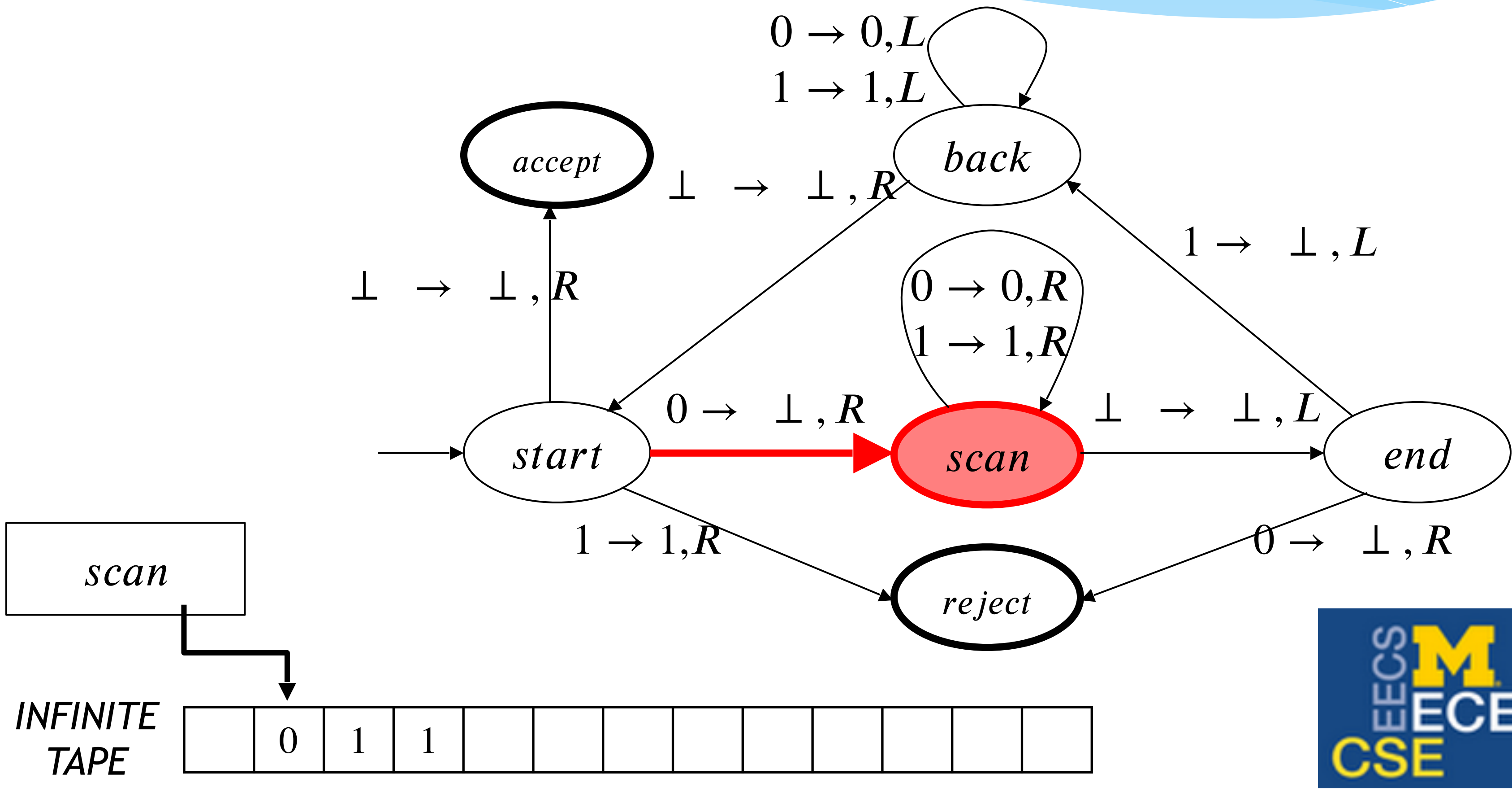


TM Example



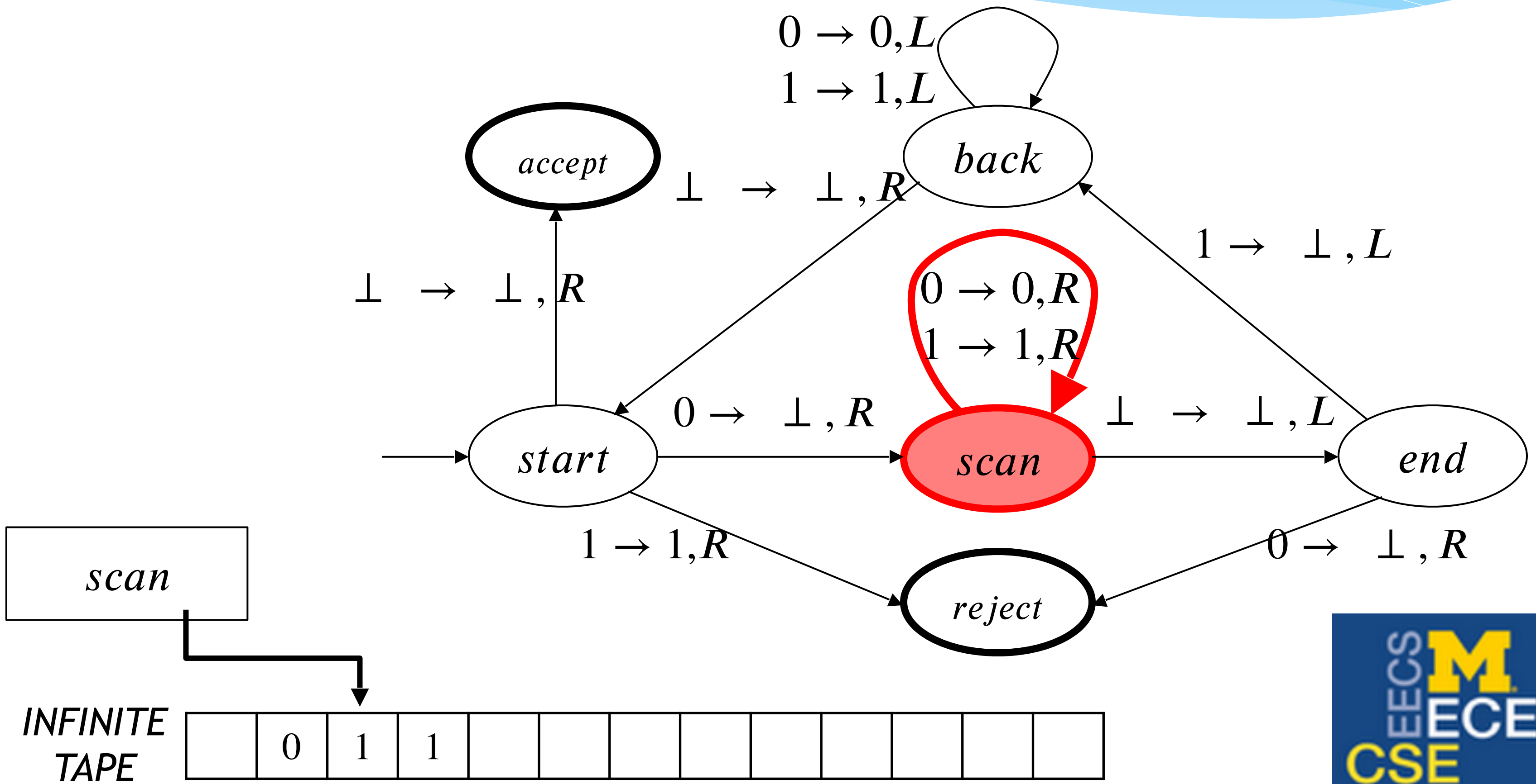
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
⊥ → ⊥ , R
1 → ⊥ , L
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
start
INFINITE TAPE
0
0
1
1

TM Example



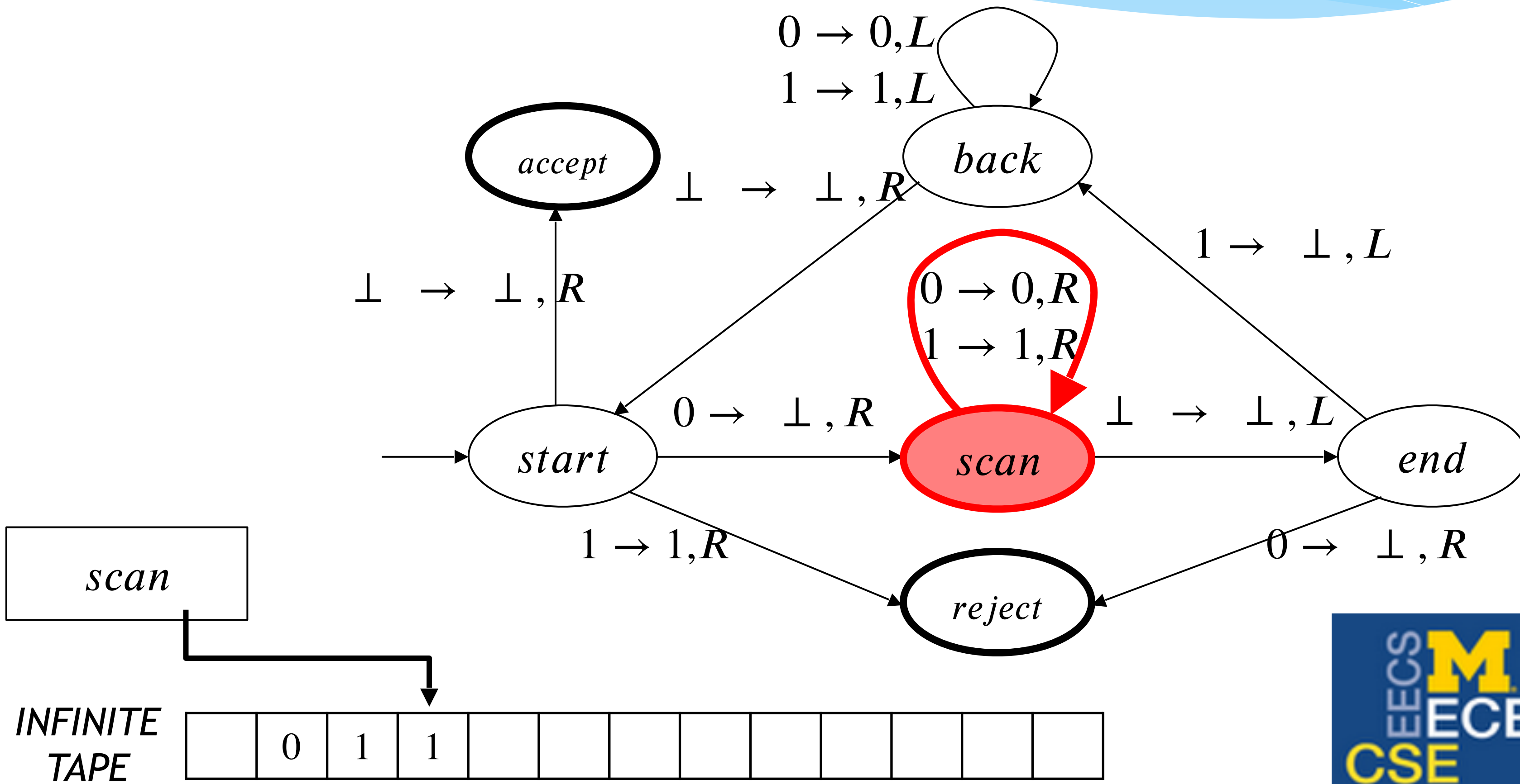
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
scan
INFINITE TAPE
0
1
1

TM Example

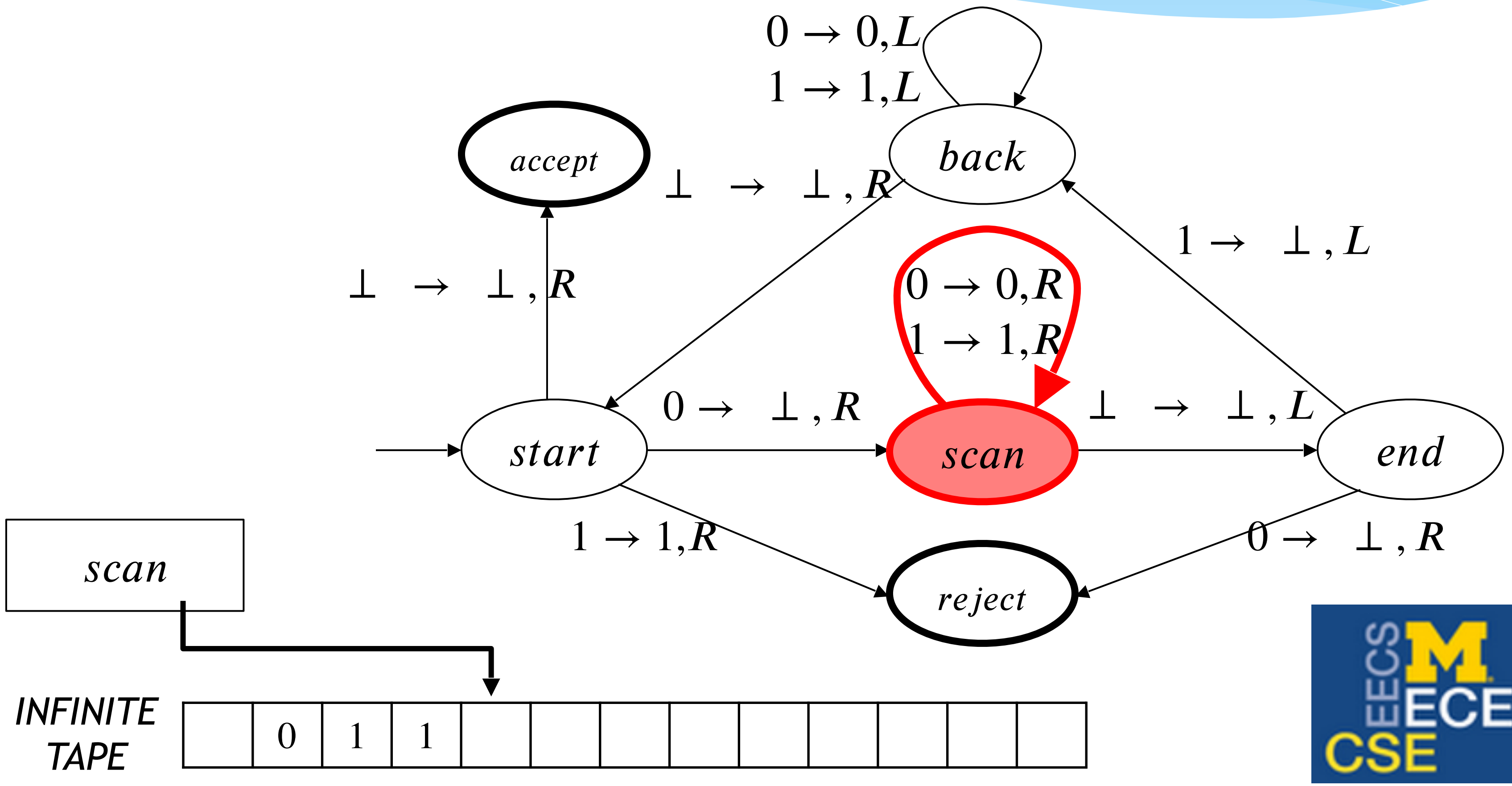


0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
scan
INFINITE TAPE
0
1
1

TM Example

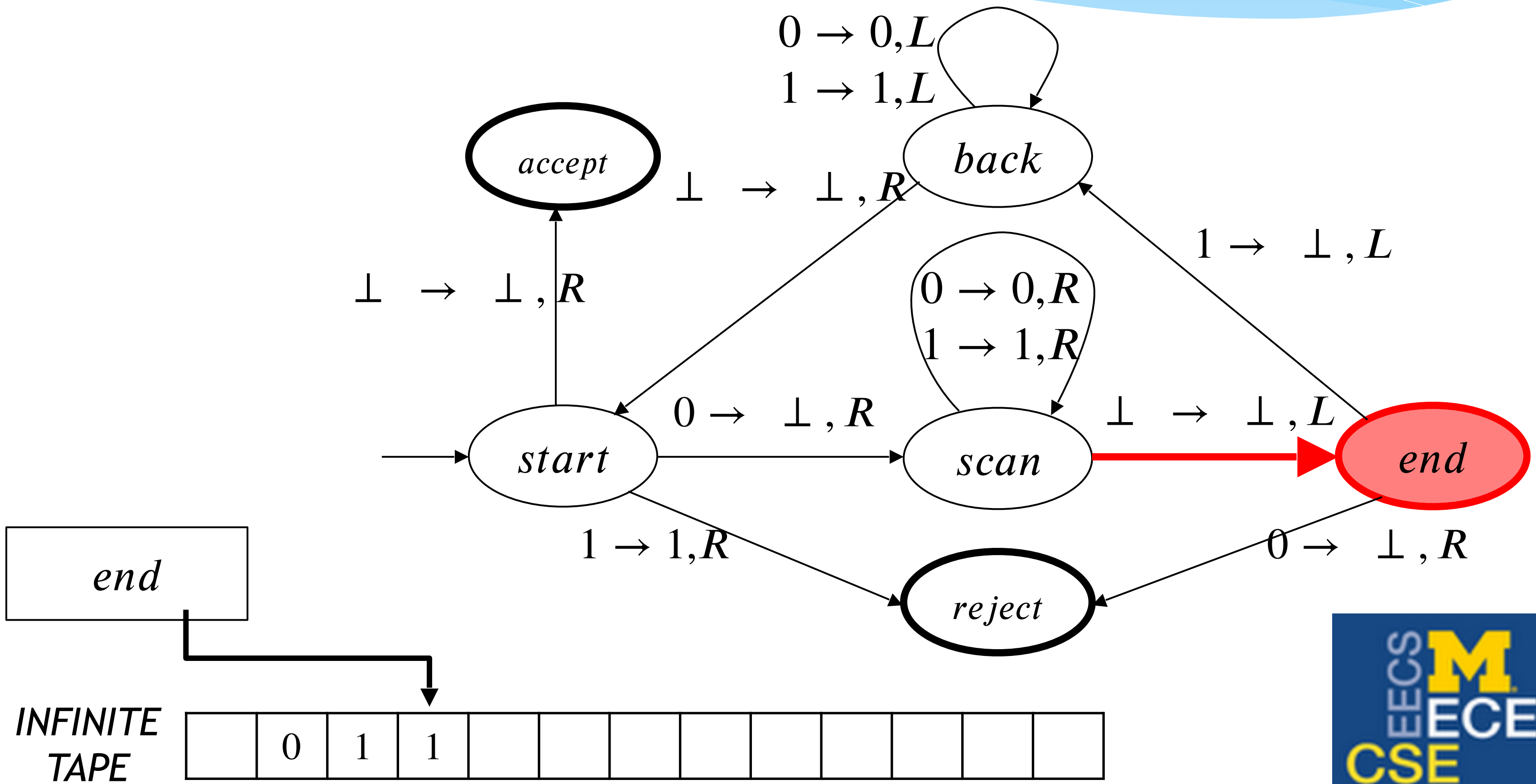

$0 \to 0, L$
$1 \to 1, L$
accept
back
$\perp \to \perp, R$
$1 \to \perp, L$
$\perp \to \perp, R$
$0 \to 0, R$
$1 \to 1, R$
$0 \to \perp, R$
$\perp \to \perp, L$
start
scan
end
$1 \to 1, R$
$0 \to \perp, R$
reject
scan
INFINITE TAPE
0
1
1

TM Example



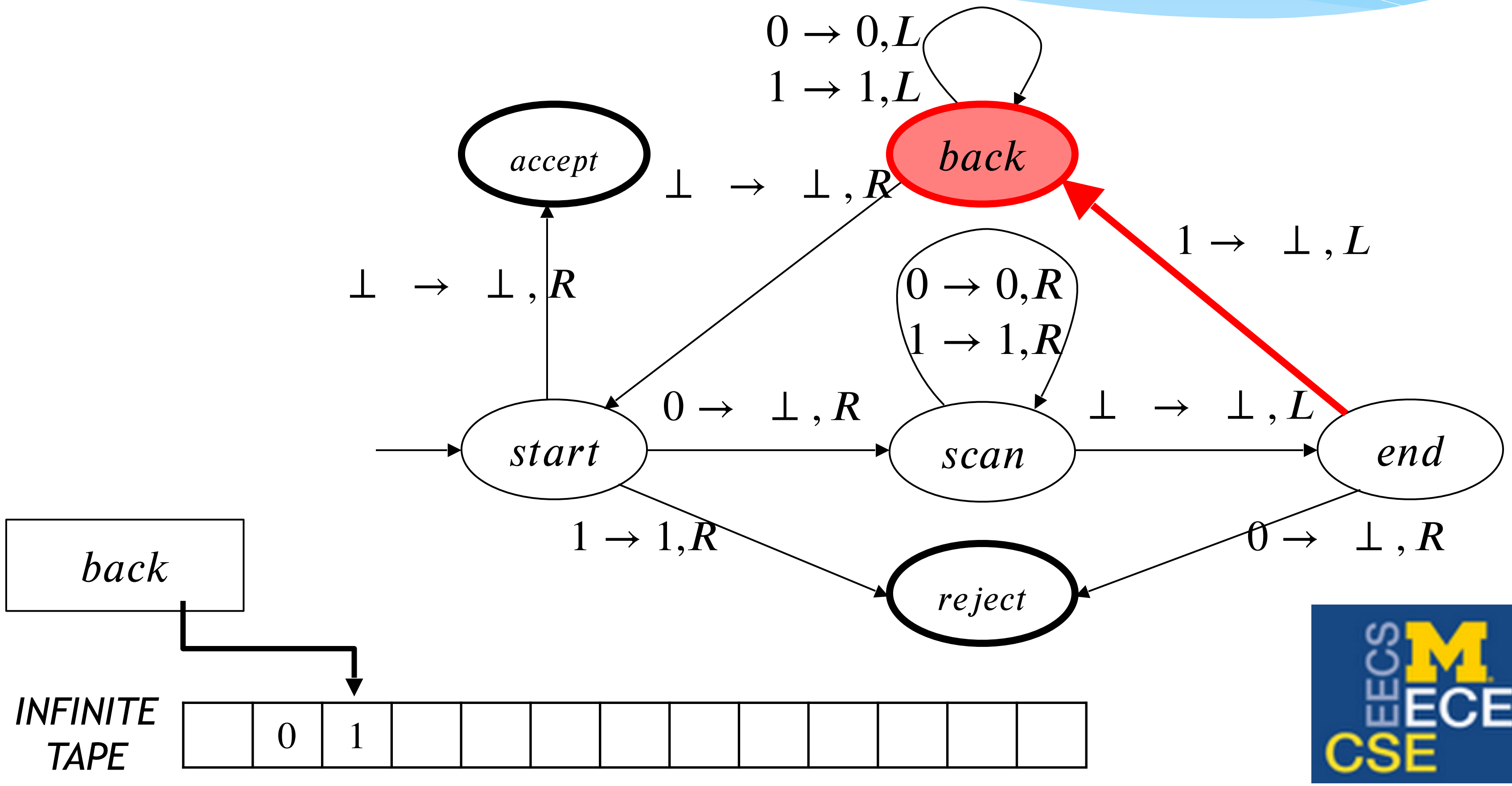
$0 \to 0,L$
$1 \to 1,L$
accept
back
$\perp \to \perp,R$
$1 \to \perp,L$
$\perp \to \perp,R$
$0 \to 0,R$
$1 \to 1,R$
$0 \to \perp,R$
$\perp \to \perp,L$
start
scan
end
$1 \to 1,R$
$0 \to \perp,R$
reject
scan
INFINITE TAPE
0
1
1

TM Example



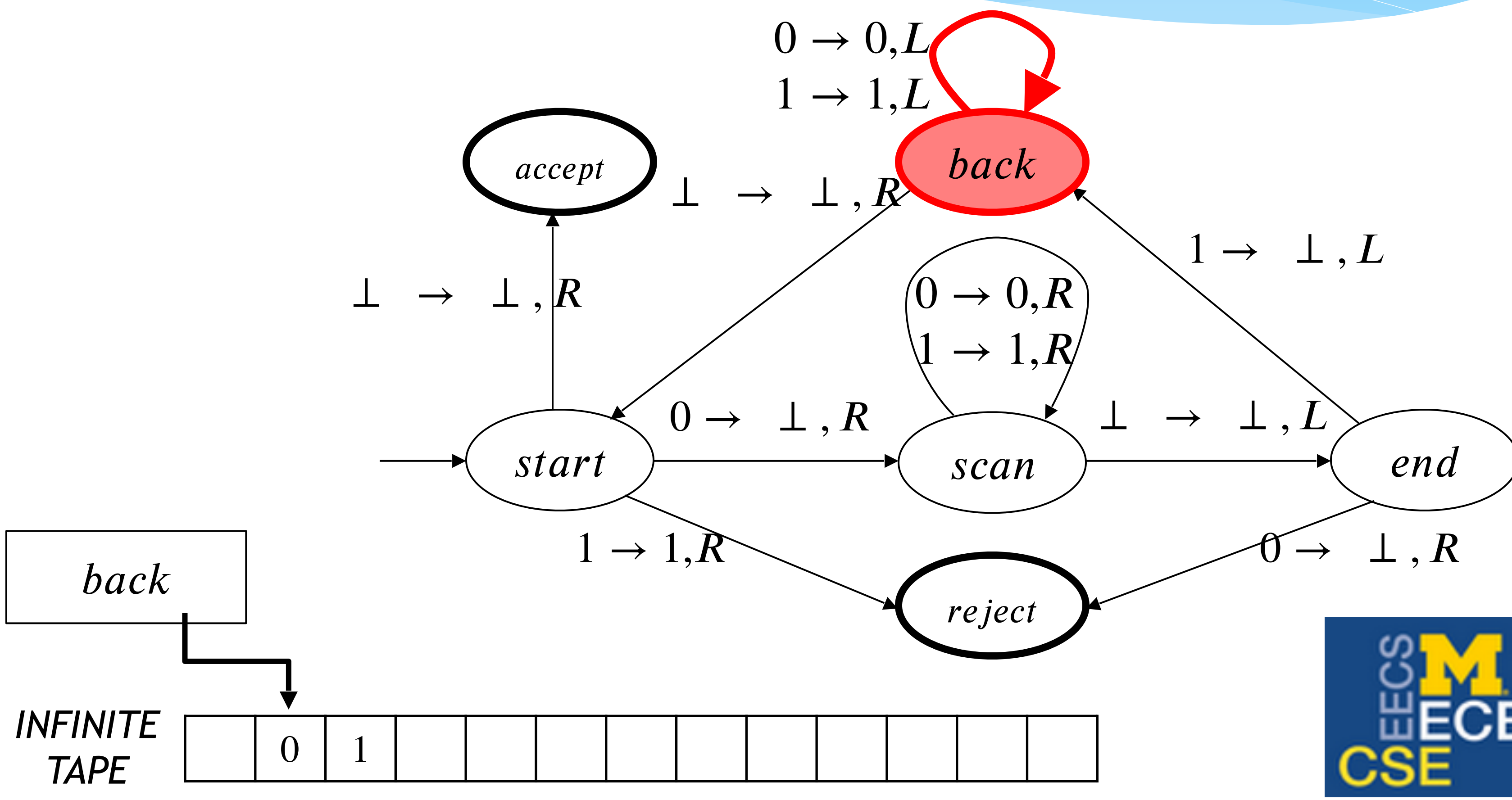
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
end
INFINITE TAPE
0
1
1

TM Example



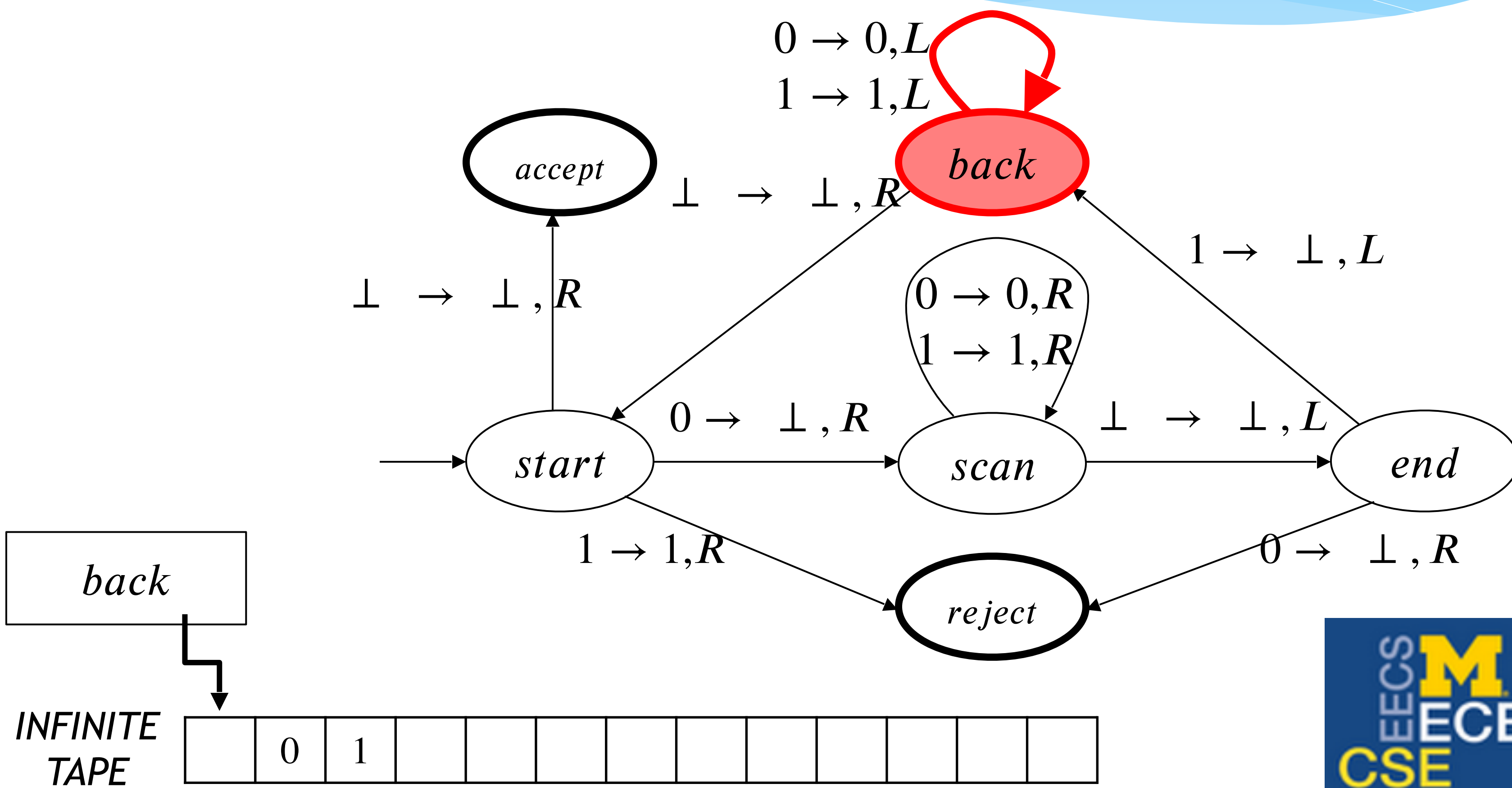
$0 \rightarrow 0, L$
$1 \rightarrow 1, L$
accept
back
$\perp \rightarrow \perp, R$
$1 \rightarrow \perp, L$
$\perp \rightarrow \perp, R$
$0 \rightarrow 0, R$
$1 \rightarrow 1, R$
$0 \rightarrow \perp, R$
$\perp \rightarrow \perp, L$
start
scan
end
$1 \rightarrow 1, R$
$0 \rightarrow \perp, R$
reject
back
INFINITE TAPE
0
1

TM Example



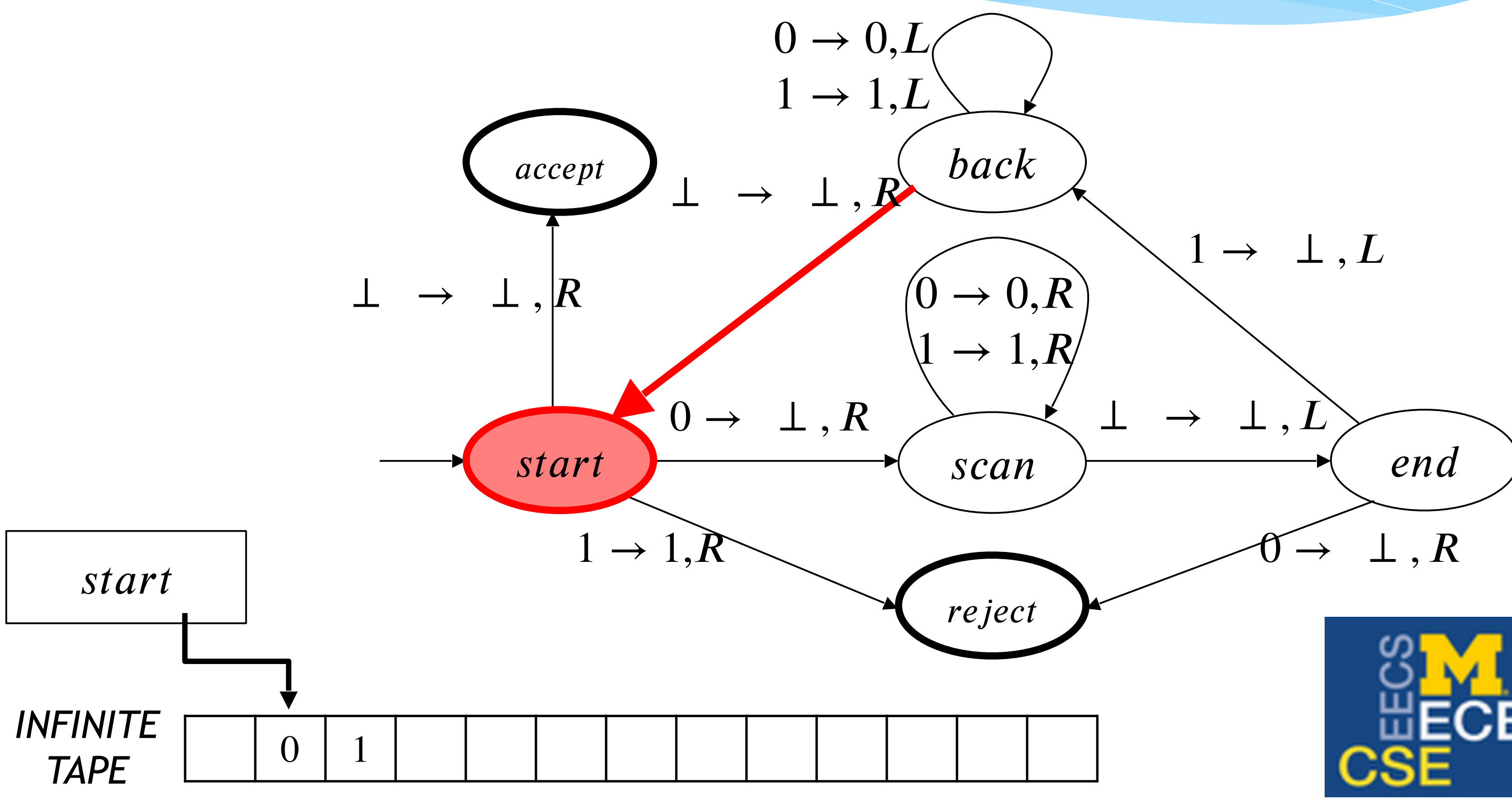
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
back
INFINITE TAPE
0
1

TM Example



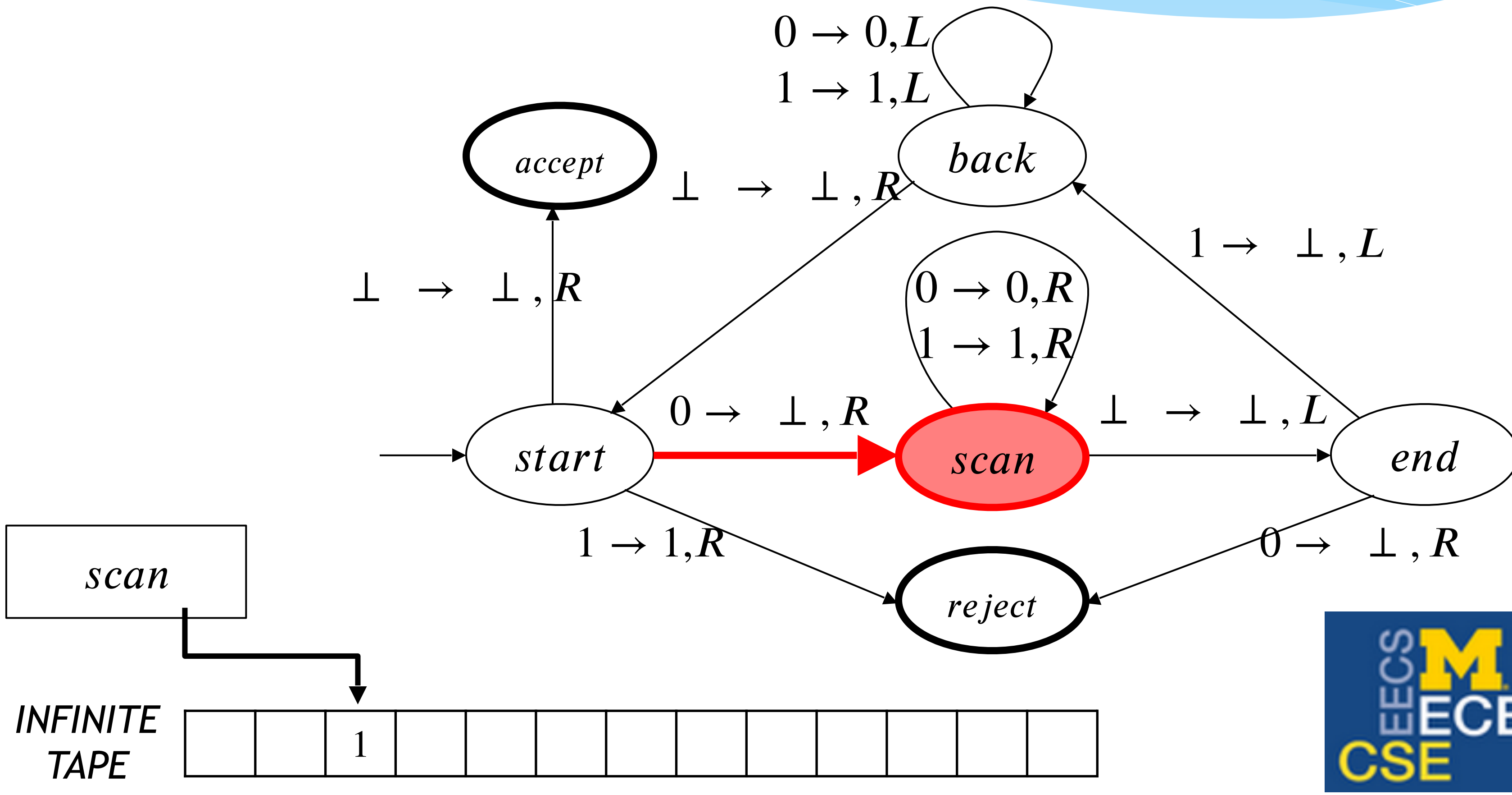
$0 \to 0, L$
$1 \to 1, L$
accept
back
$\perp \to \perp, R$
$1 \to \perp, L$
$\perp \to \perp, R$
$0 \to 0, R$
$1 \to 1, R$
$0 \to \perp, R$
$\perp \to \perp, L$
start
scan
end
$1 \to 1, R$
$0 \to \perp, R$
reject
back
INFINITE TAPE
0
1

TM Example



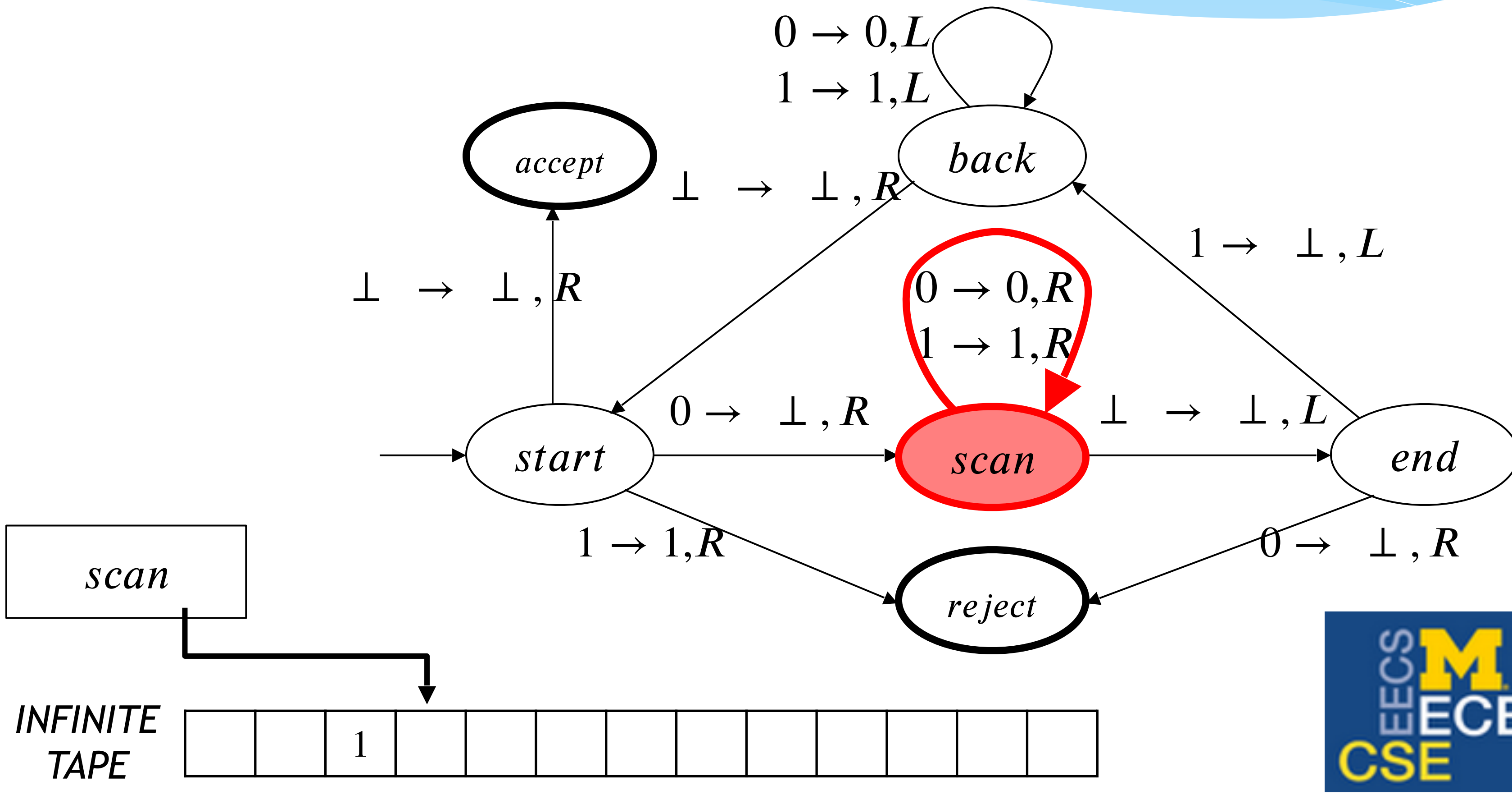
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
start
INFINITE TAPE
0
1

TM Example

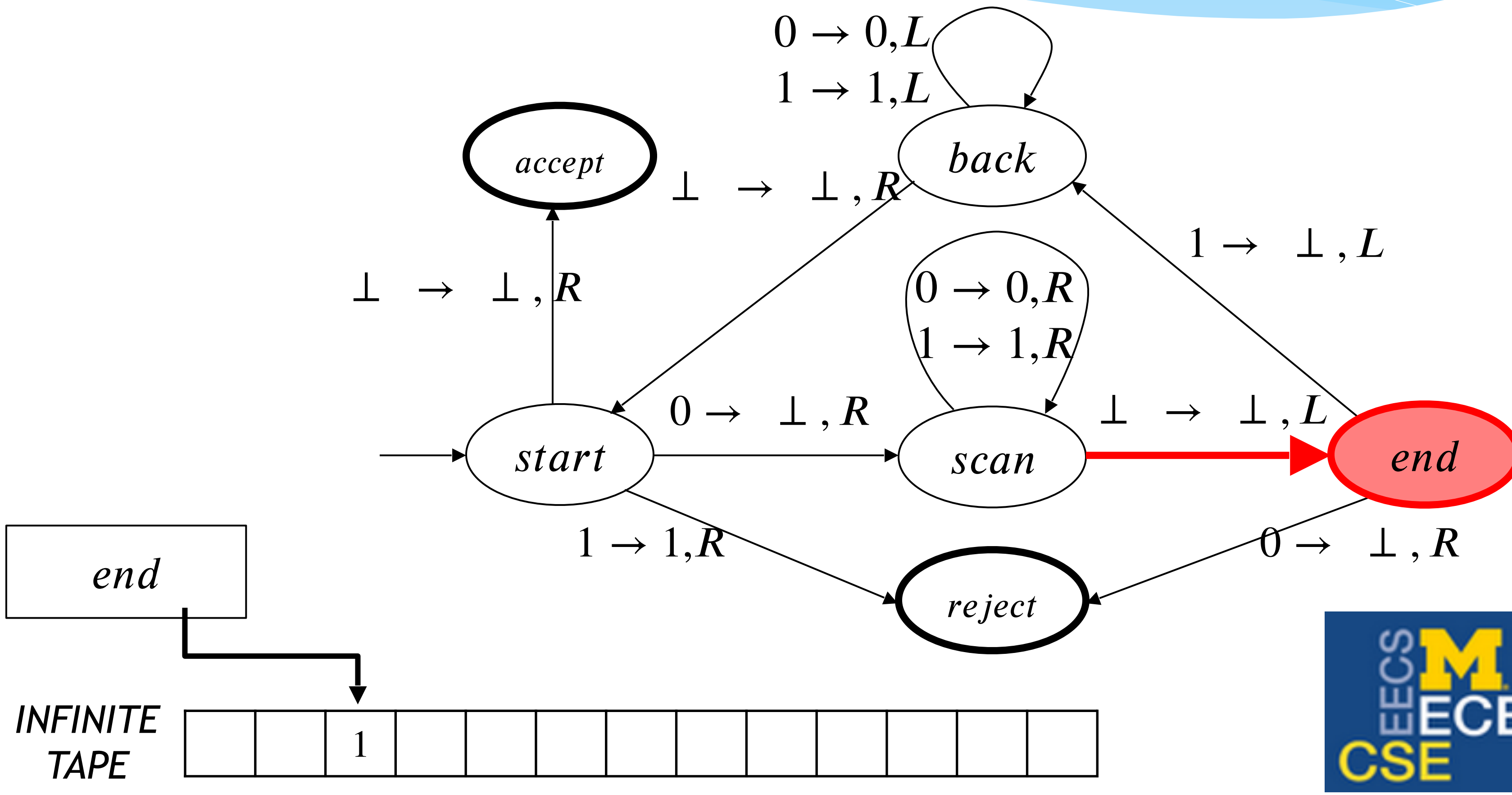


$0 \rightarrow 0, L$
$1 \rightarrow 1, L$
back
accept
$\perp \rightarrow \perp, R$
$1 \rightarrow \perp, L$
$\perp \rightarrow \perp, R$
$0 \rightarrow 0, R$
$1 \rightarrow 1, R$
$0 \rightarrow \perp, R$
$\perp \rightarrow \perp, L$
start
scan
end
$1 \rightarrow 1, R$
$0 \rightarrow \perp, R$
reject
scan
INFINITE TAPE
1

TM Example

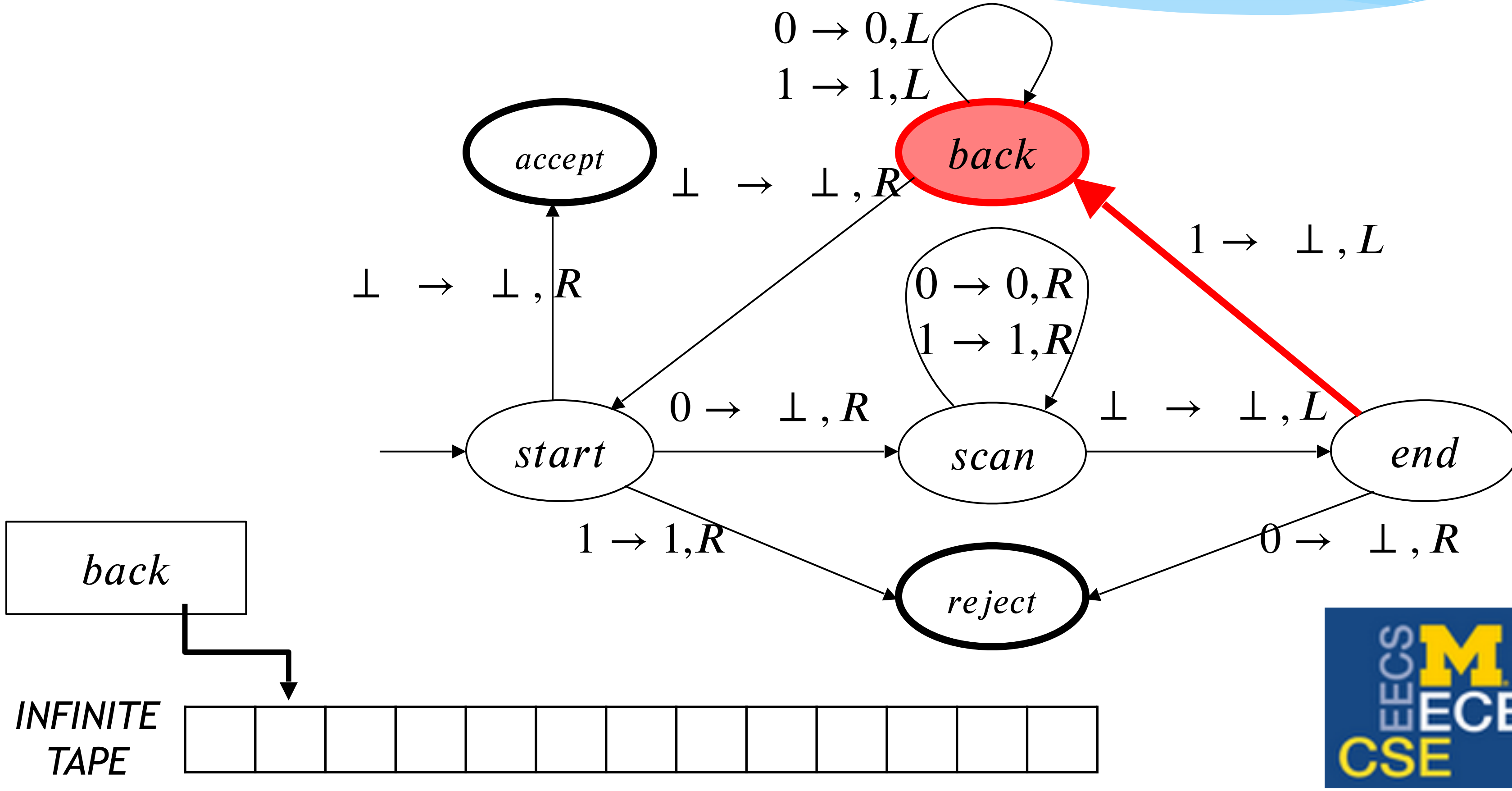

0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
scan
INFINITE TAPE
1

TM Example



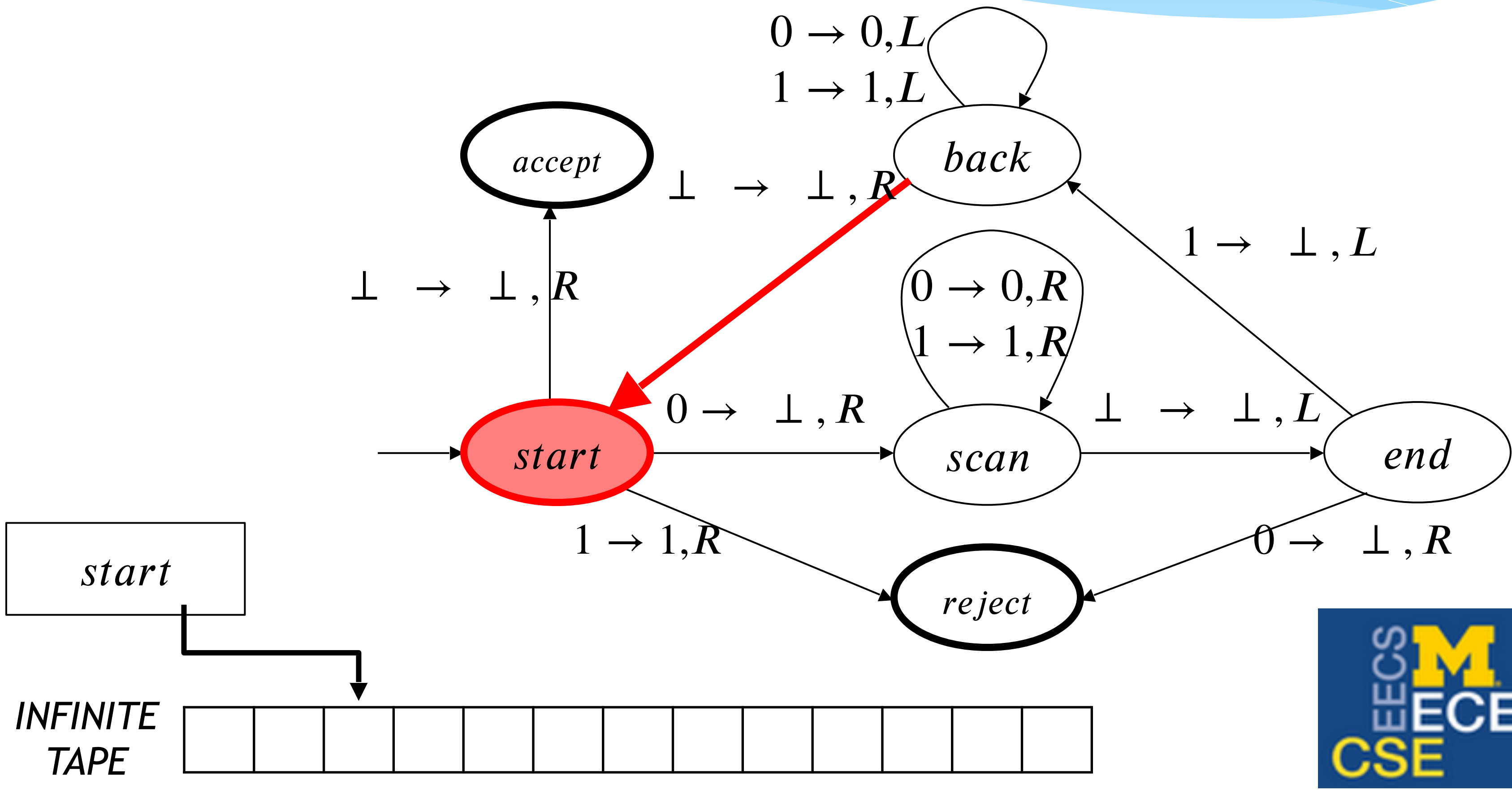
0 → 0,L
1 → 1,L
back
accept
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
end
INFINITE TAPE
1

TM Example



0 → 0,L
1 → 1,L
accept
back
⊥ → ⊥ , R
1 → ⊥ , L
⊥ → ⊥ , R
0 → 0,R
1 → 1,R
0 → ⊥ , R
⊥ → ⊥ , L
start
scan
end
1 → 1,R
0 → ⊥ , R
reject
back
INFINITE
TAPE

TM Example

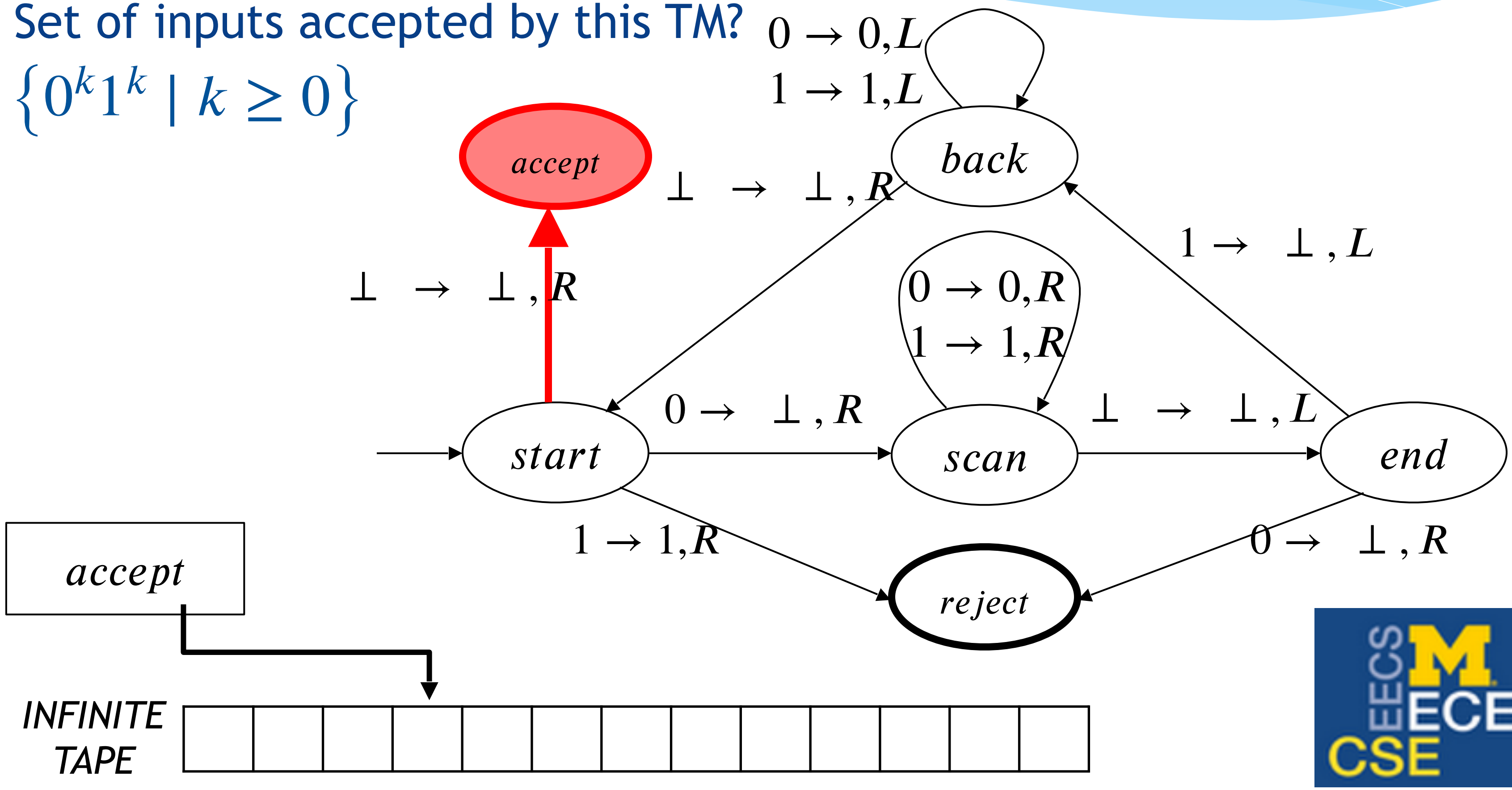


$0 \to 0, L$
$1 \to 1, L$
back
accept
$\perp \to \perp, R$
$1 \to \perp, L$
$\perp \to \perp, R$
$0 \to 0, R$
$1 \to 1, R$
$0 \to \perp, R$
$\perp \to \perp, L$
start
scan
end
$1 \to 1, R$
$0 \to \perp, R$
reject
start
INFINITE TAPE

TM Example

Q: Set of inputs accepted by this TM?

A: $\{0^k1^k \mid k \geq 0\}$



Turing Machine

- A ***Turing Machine*** is a 7-tuple $(Q, \Gamma, \Sigma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$:
  - Q is a finite set of ***states***
  - $q_0 \in Q$ is the ***initial state***
  - $F = \{q_{\textit{accept}}, q_{\textit{reject}}\} \subseteq Q$ are the ***final*** (***accept/reject***) states
  - Σ is the ***input alphabet***
  - $\Gamma \supseteq \Sigma \cup \{ \perp \}$ is the ***tape alphabet*** ($\perp \notin \Sigma$ is the ***blank symbol***)
  - $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the ***transition function***
- **Takeaway:** TMs are a well-defined type of "computer".

Simulations

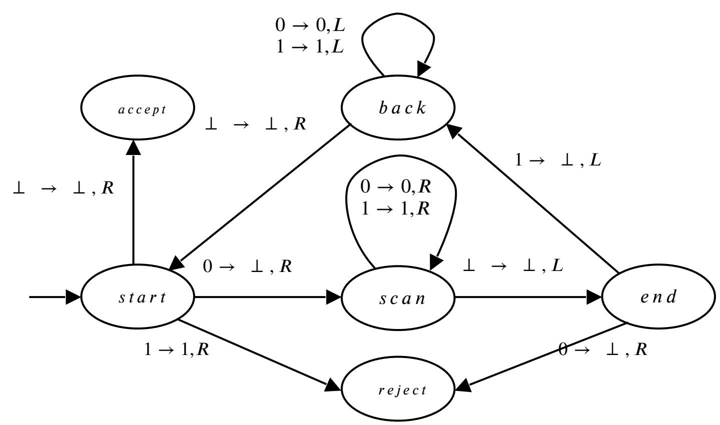
- *Intuitively*, if a "computer" M_1 can ***simulate*** another "computer" M_2, then M_1 is *at least as powerful* as M_2. They are ***equivalent*** if M_2 can also simulate M_1.

- All known computational models are either:
  - *Weaker* than TMs (e.g., DFAs, PDAs) or
  - *Equivalent* to TMs in what they can compute (e.g., random-access machines, lambda calculus, quantum computers, etc.)

- ***Church-Turing thesis:*** Any "computer" (e.g. any alien technology) can be simulated by some Turing Machine. (*This is a **conjecture**!*)

Pseudocode vs TMs

- **Claim:** Given enough memory, *<u>any</u>* TM can be simulated by a "Boolean" function on strings written in pseudocode (e.g., C++).



- **Q:** Can any "Boolean" function on strings written in pseudocode (e.g., C++) be simulated by a TM?

simulateM(string x):

// *simulates* TM **M** on string x

// - hard-coded transition function

// - maintain state & tape cells

return accept/reject according to M

Key Idea: TM ≡ "bool **M**(string x)"

Decision Programs

* **Q:** Suppose we run a function "bool **M**(string x)" (i.e., a TM) on string x. What are the possible outcomes?
  * **M** either (i) accepts, (ii) rejects, *or* (iii) it ***"loops" (forever)***

* A TM **M** ***decides*** a language L if it:
  1. <u>*accepts*</u> every string $x \in L$, and
  2. <u>*rejects*</u> every string $x \notin L$.

In this case, we say that **M** is a ***decider*** (for L), and L is ***decidable***.

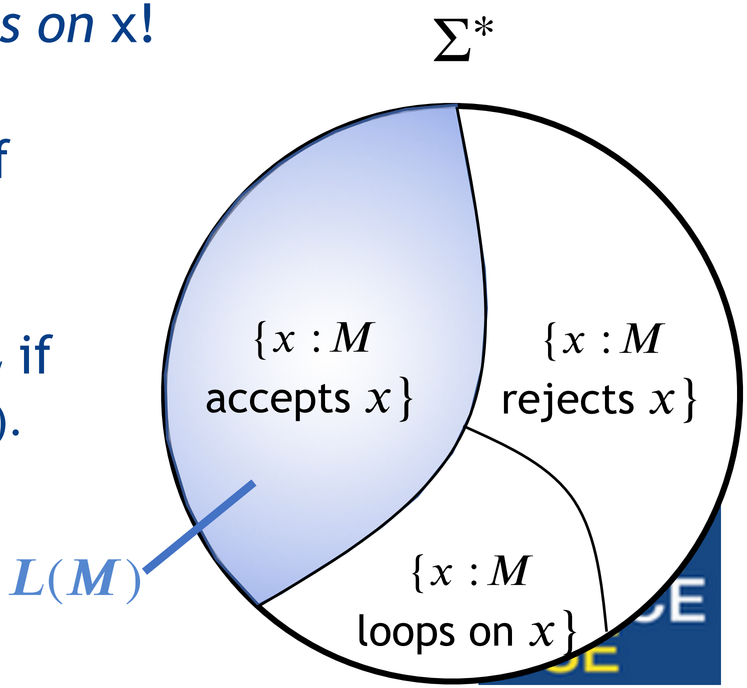
* **Note:** By definition, **M** <u>*does not loop*</u> on any input!

More Generally: Language of a TM

- **Definition:** The language of a TM M is $L(M) := \{x : M \text{ accepts } x\}$.
- **Question:** What if $x \notin L(M)$? (M(x) does not accept.)
- **Answer:** Then M either *rejects* x, or *loops on* x!

- **Conclusion:** TM M decides language L iff $L(M) = L$ and M halts on *every* input.

- **Definition**: TM M **recognizes** language L if $L(M) = L$ (regardless of whether M ever loops).
- More on this later…



Summary

* We have formalized the notions of a "problem" and "computer", as follows:
  * "Decision problem" ≡ "Is string $x \in L$ (associated language)?"
  * "Computer" ≡ TM ≡ "bool **M**(string x)"
* We also have a precise definition of what it means for a computer to solve a problem:
  * "A decision problem can be solved on a computer"

    ≡ "some TM *decides* the associated language"

Next time: Can *every* decision problem be solved on a computer?