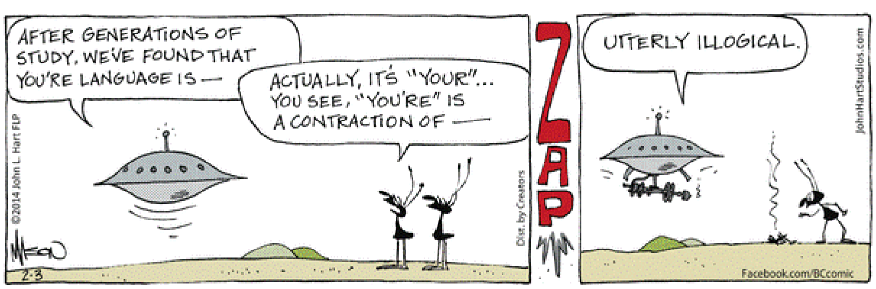
MA/CSSE 474
Theory of Computation

TM Design
Macro Language
D and SD

Your Questions?

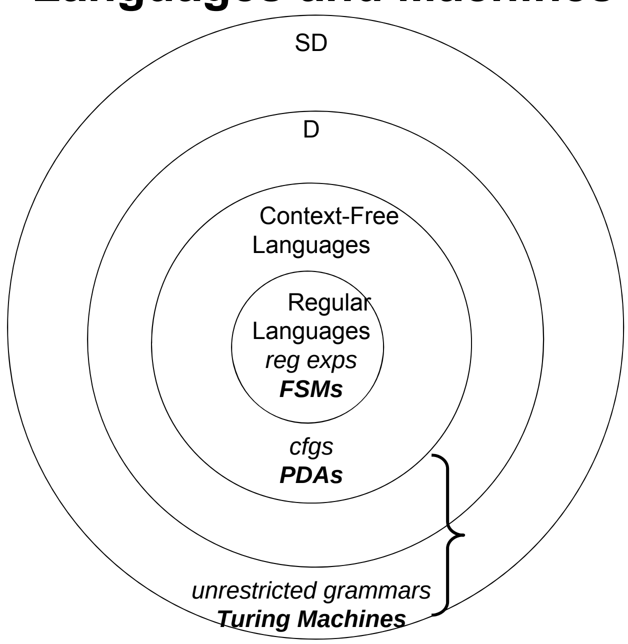
- Previous class days' material
- Reading Assignments
- HW 13 problems
- Anything else

I have included some slides online that we will not have time to do in class, but may be helpful to you anyway.

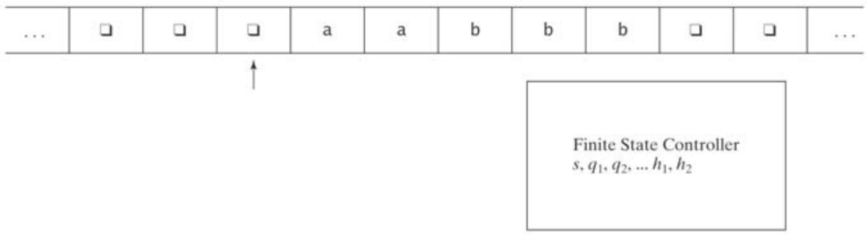


TURING MACHINES

Languages and Machines



Turing Machines



At each step, the machine must:

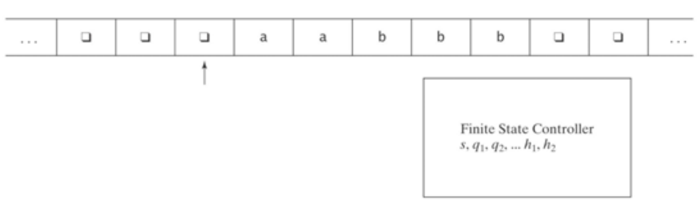
- choose its next state,
- write on the current square, and
- move left or right.

TM Formal Definition

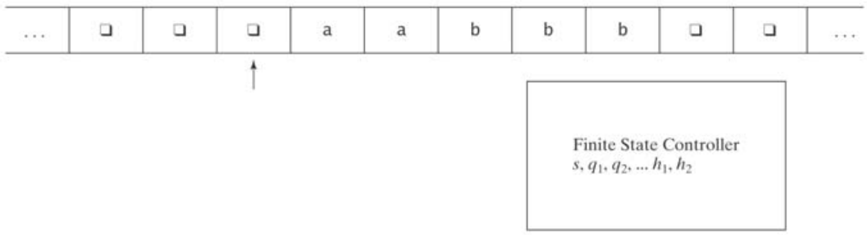
A (deterministic) Turing machine M is (K, Σ, Γ, δ, s, H), where

- K is a finite set of states;
- Σ is the input alphabet, which does not contain Æ ;
- Γ is the tape alphabet,
  which must contain Æ and have Σ as a subset.
- $s \in K$ is the initial state;
- $H \subseteq K$ is the set of halting states;
- δ is the transition function:

(K - H)	$\times$ Γ	to	K	$\times$ Γ		$\times$	$\{\rightarrow, \leftarrow\}$
non-halting state	$\times$ tape char	$\rightarrow$	state	$\times$ tape char		$\times$	direction to move (R or L)



Turing Machines



At each step, the machine must:

- choose its next state,
- write on the current square, and
- move left or right.

Notes on the Definition

1. The input tape is infinite in both directions.
2. δ is a function, not a relation. So this is a definition for deterministic Turing machines.
3. δ must be defined for all (state, tape symbol) pairs unless the state is a halting state.
4. Turing machines do not necessarily halt (unlike FSM's and most PDAs). Why? To halt, they must enter a halting state. Otherwise they loop.
5. Turing machines generate output, so they can compute functions.

An Example

M takes as input a string in the language:

$\{a^i b^j, 0 \leq j \leq i\}$,

and adds b's as required to make the number of b's equal the number of a's.

The input to *M* will look like this:

...	❑	a	a	a	b	❑	❑	❑	...

The output should be:

...	❑	a	a	a	b	b	b	❑	...

Formal Definition of M

M = ({1, 2, 3, 4, 5, 6}, {a, b}, {a, b, ❑, \$, #}, δ, 1, {6}), where δ =

```
(   ((1, ❑), (2, ❑, →)),
    ((1, a), (2, q, →))
    ((1, b), (2, q, →)),
    ((1, $), (2, ❑, →)),
    ((1, #), (2, ❑, →)),
    ((2, ❑), (6, $, →)),
    ((2, a), (3, $, →)),
    ((2, b), (3, $, →)),
    ((2, $), (3, $, →)),
    ((2, #), (3, $, →)),
    ((3, ❑), (4, #, ←)),
    ((3, a), (3, a, →)),
    ((3, b), (4, #, ←)),
    ((3, $), (3, $, →)),
    ((3, #), (3, #, →)),
    ((4, ❑), (5, ❑, →)),
    ((4, a), (3, $, →)),
    ((4, $), (4, $, ←)),
    ((4, #), (4, #, ←)),
    ((5, ❑), (6, ❑, ←)),
    ((5, $), (5, a, →)),
    ((5, #), (5, b, →))  )
```

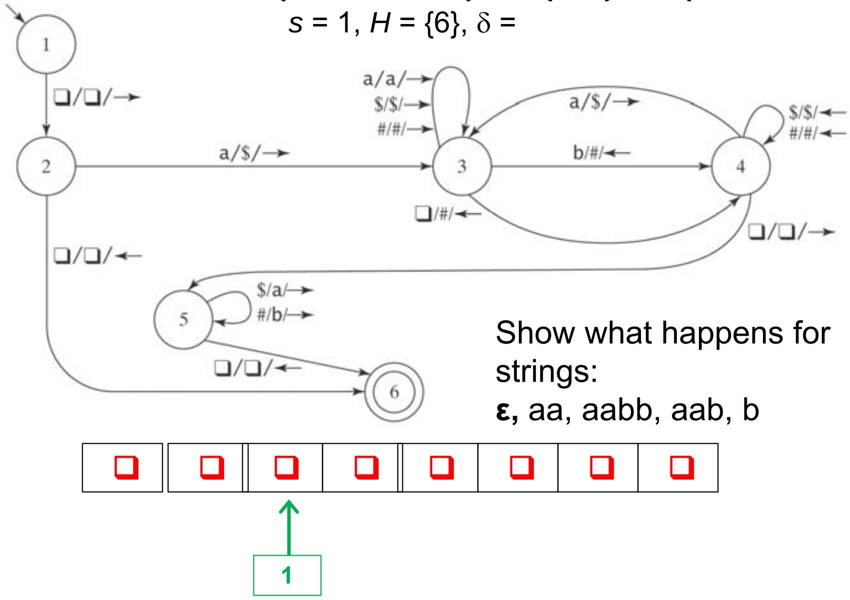
These four transitions are required because *M* must be defined for every state/ input pair, but since it isn't possible to see anything except ❑ in state 1, it doesn't matter what they do.

Three more unusable elements of δ. We'll omit the rest here for clarity.

State 6 is a halting state and so has no transitions out of it

The Details (ε)

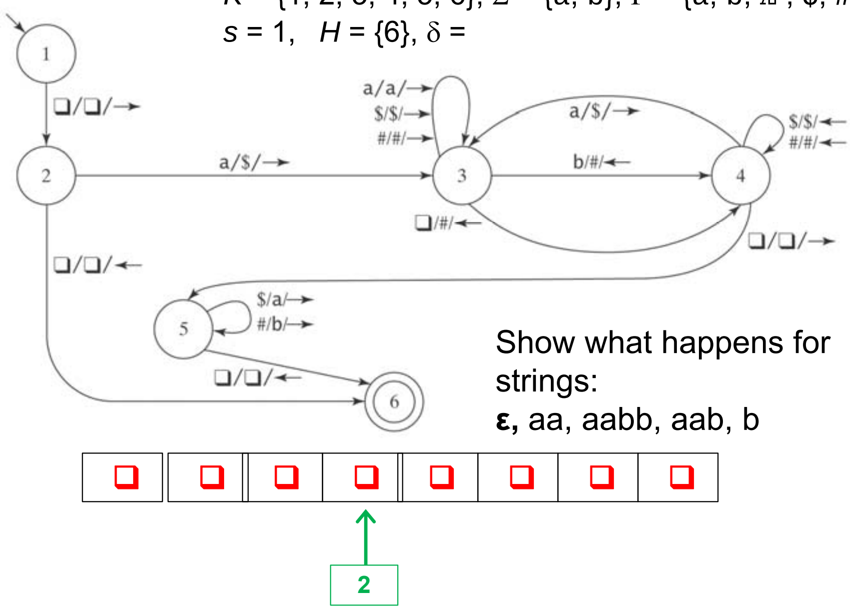
K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

The Details (ε)

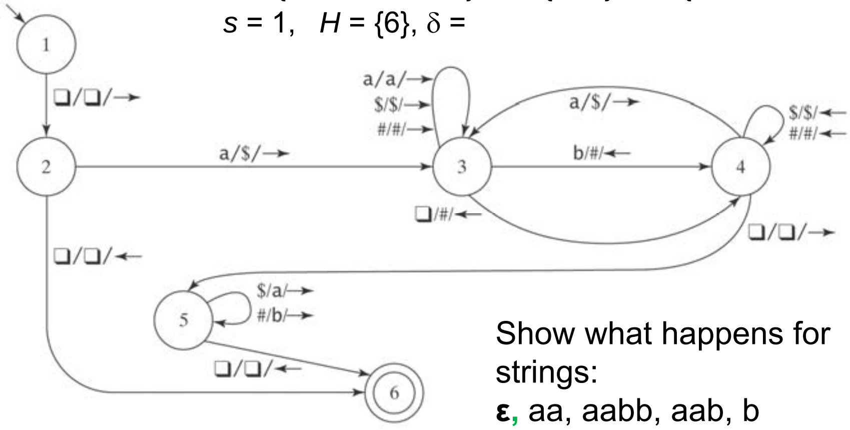
K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

The Details (ε)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



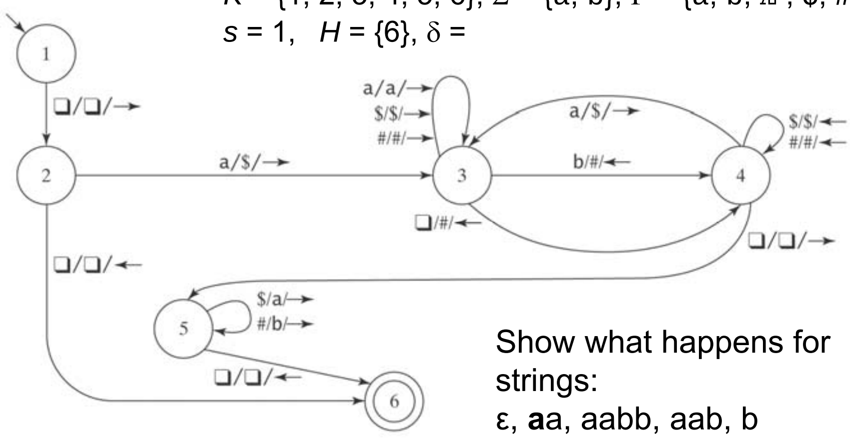
Show what happens for strings:
ε, aa, aabb, aab, b

❑	❑	❑	❑	❑	❑	❑	❑

6

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



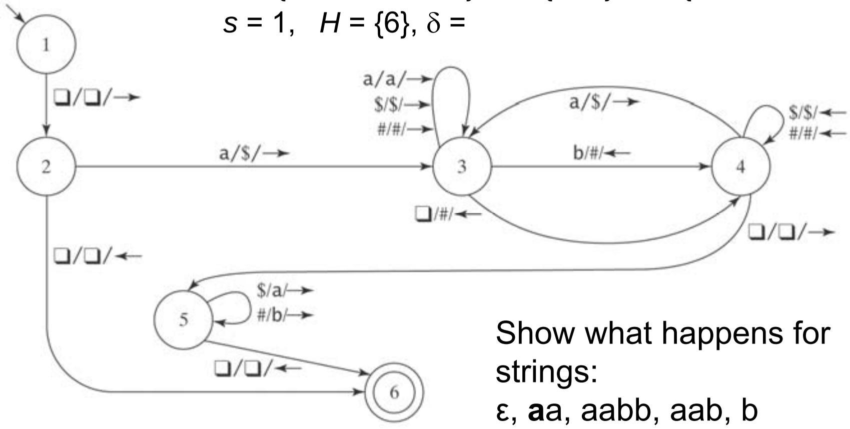
Show what happens for strings:
ε, **a**a, aabb, aab, b

❑	❑	❑	a	a	❑	❑	❑

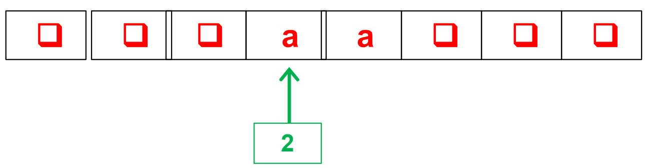
1

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =

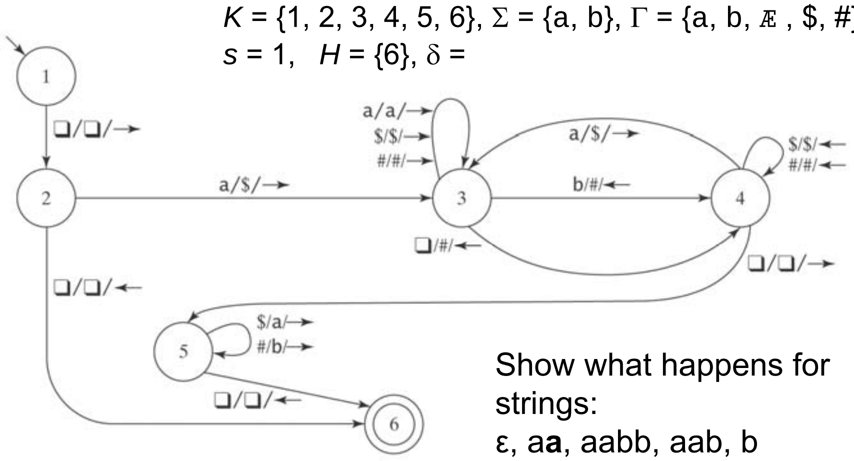


Show what happens for strings:
ε, **a**a, aabb, aab, b

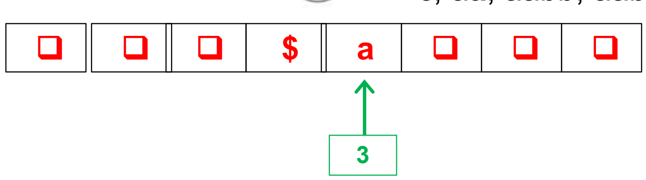


The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =

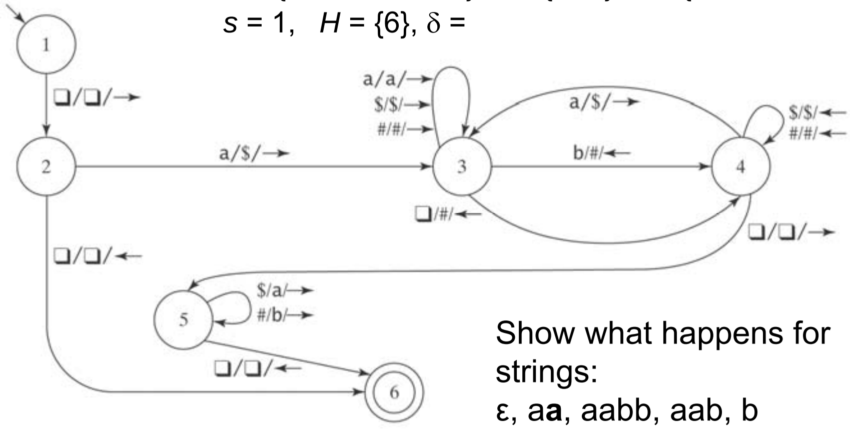


Show what happens for strings:
ε, a**a**, aabb, aab, b



The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, *H* = {6}, δ =



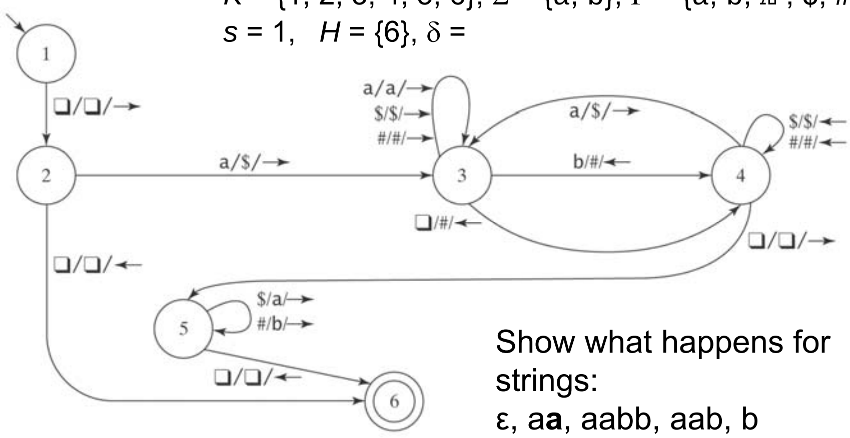
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	a	❑	❑	❑

↑ 3

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, *H* = {6}, δ =



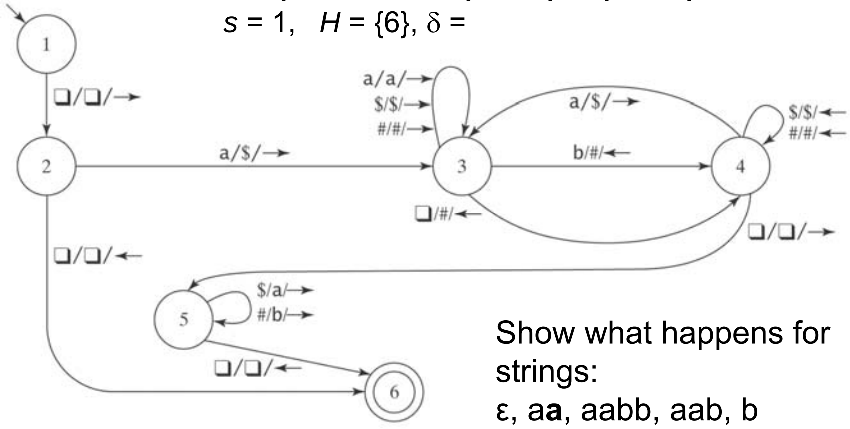
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	a	#	❑	❑

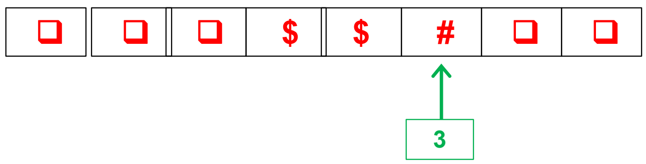
↑ 4

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑ , \$, #},
s = 1, H = {6}, δ =

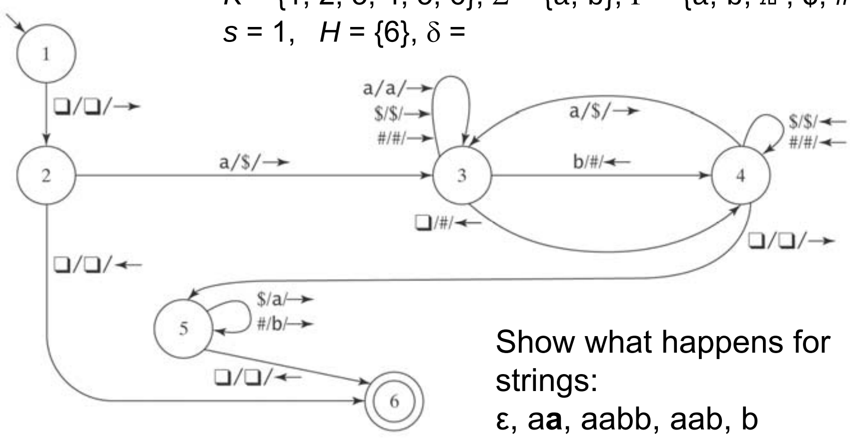


Show what happens for strings:
ε, a**a**, aabb, aab, b

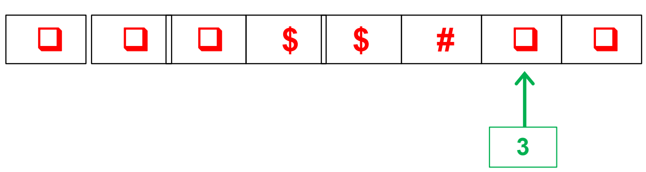


The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑ , \$, #},
s = 1, H = {6}, δ =

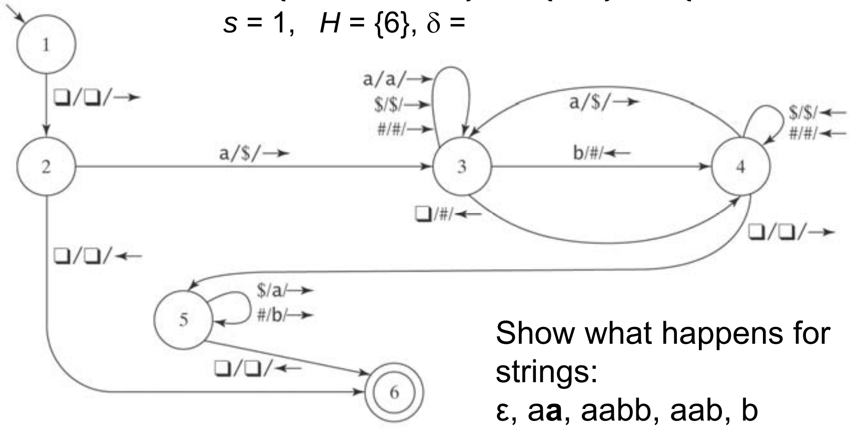


Show what happens for strings:
ε, a**a**, aabb, aab, b



The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



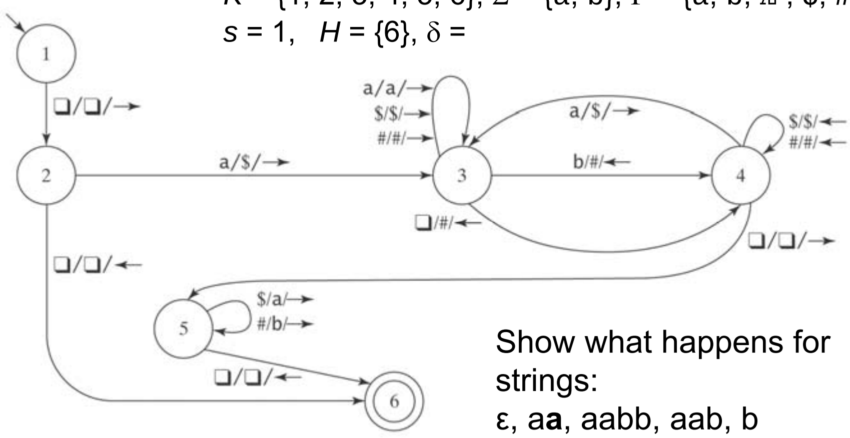
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	❑	❑

3

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



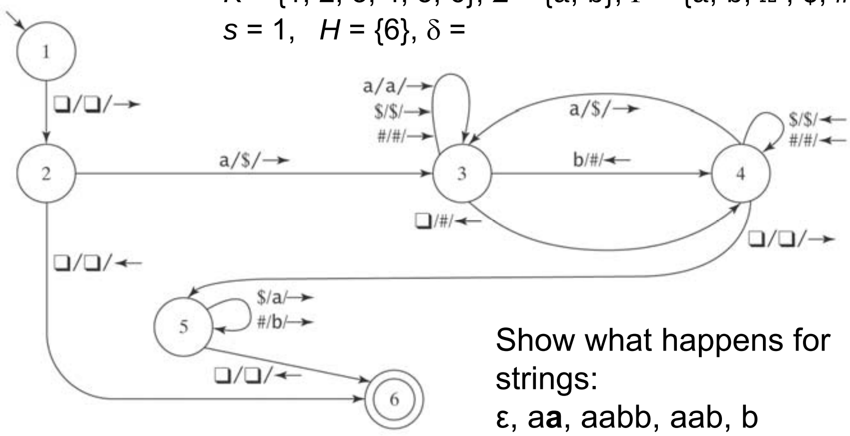
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	#	❑

4

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



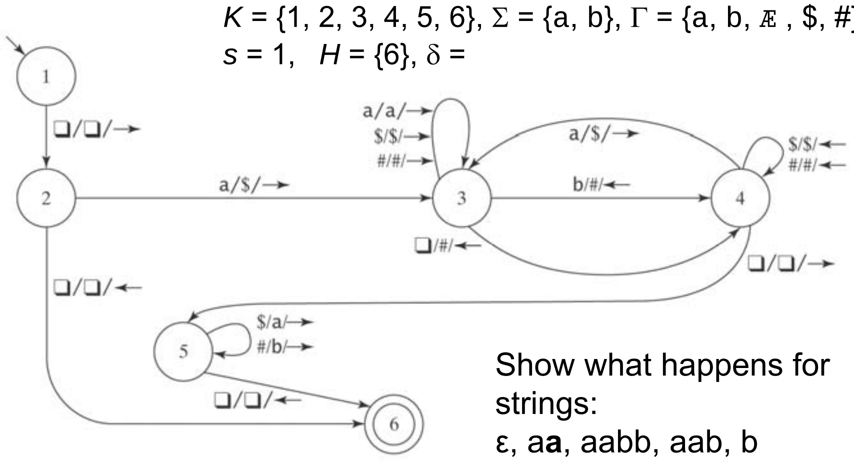
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	#	❑

4

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



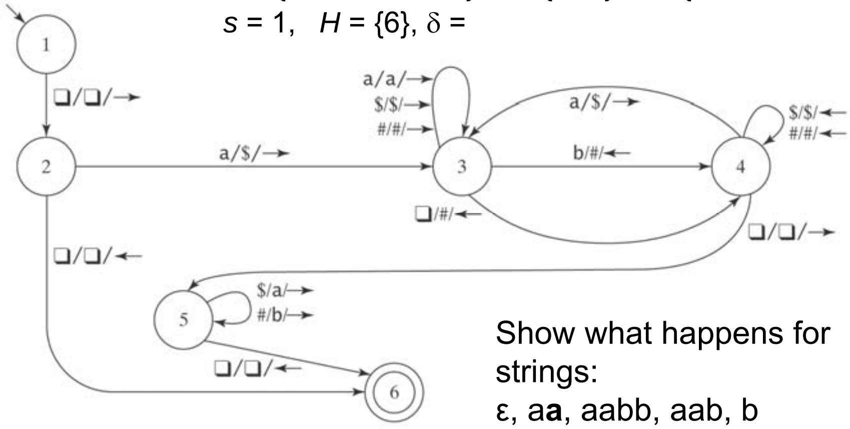
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	#	❑

4

The Details(aa)

$K = \{1, 2, 3, 4, 5, 6\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, Æ, \$, \#\}$,
$s = 1$, $H = \{6\}$, $\delta =$



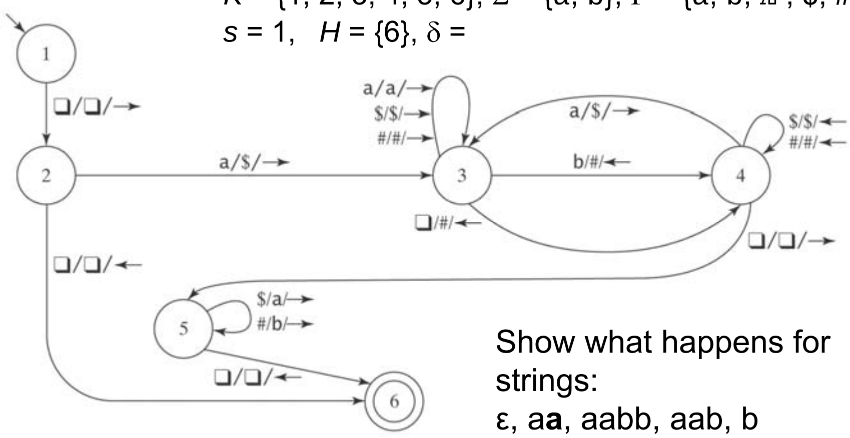
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	#	❑

4

The Details(aa)

$K = \{1, 2, 3, 4, 5, 6\}$, $\Sigma = \{a, b\}$, $\Gamma = \{a, b, Æ, \$, \#\}$,
$s = 1$, $H = \{6\}$, $\delta =$



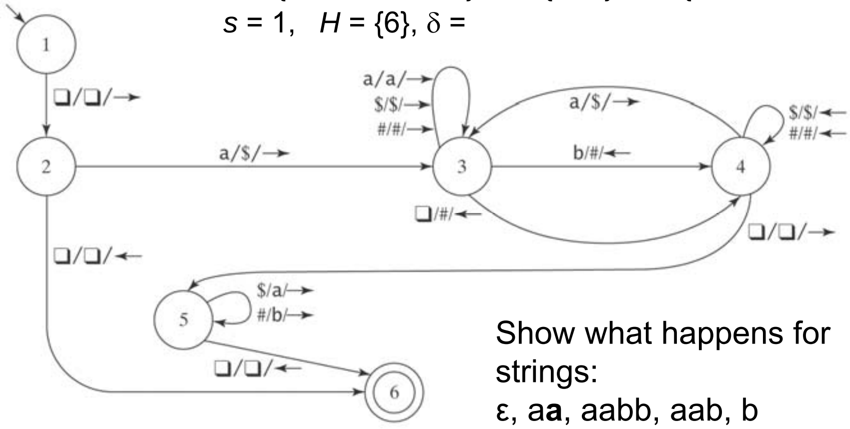
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	$	$	#	#	❑

5

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =

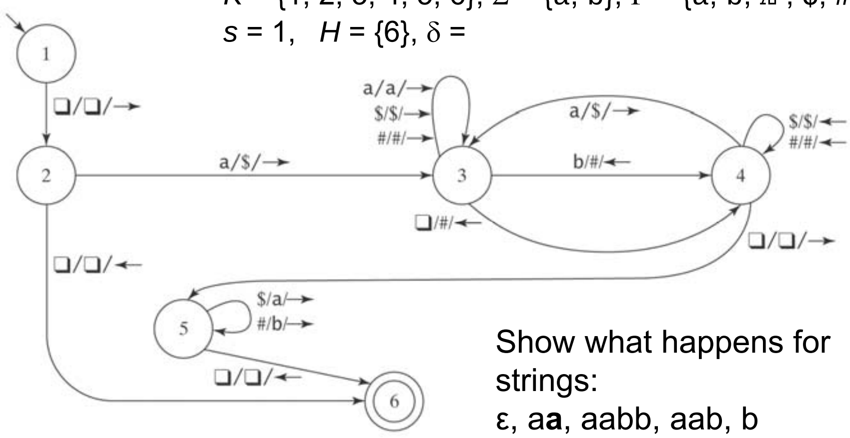


Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	\$	#	#	❑
				↑ 5			

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =

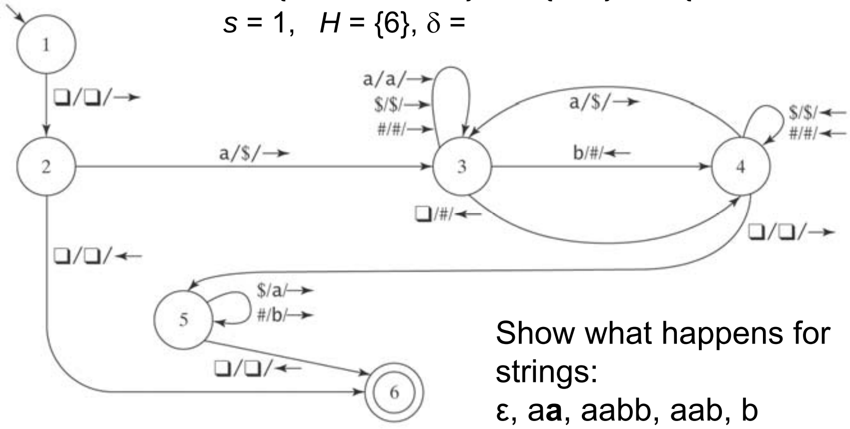


Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	a	#	#	❑
					↑ 5		

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑, \$, #},
s = 1, H = {6}, δ =



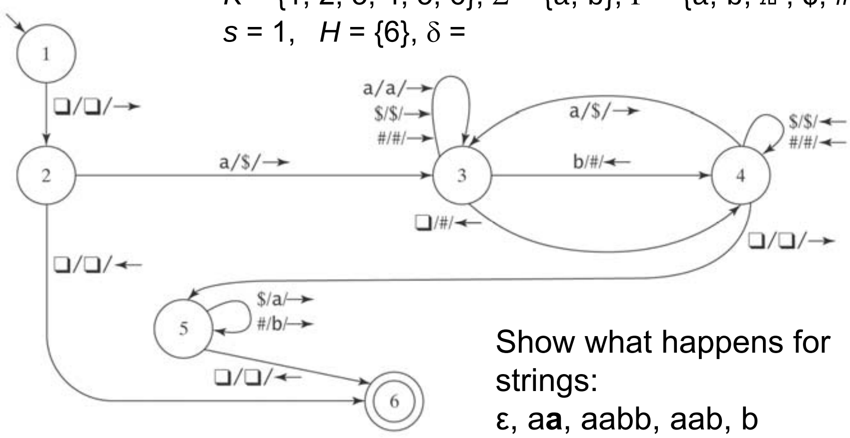
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	a	b	#	❑

5

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑, \$, #},
s = 1, H = {6}, δ =



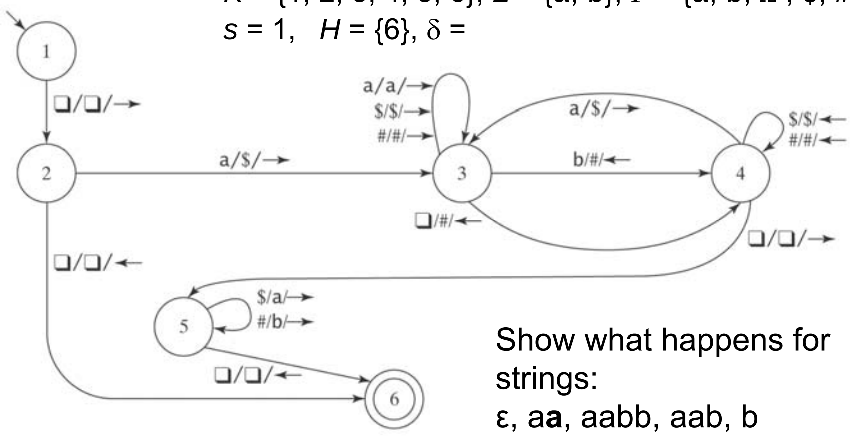
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	a	b	b	❑

5

The Details(aa)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



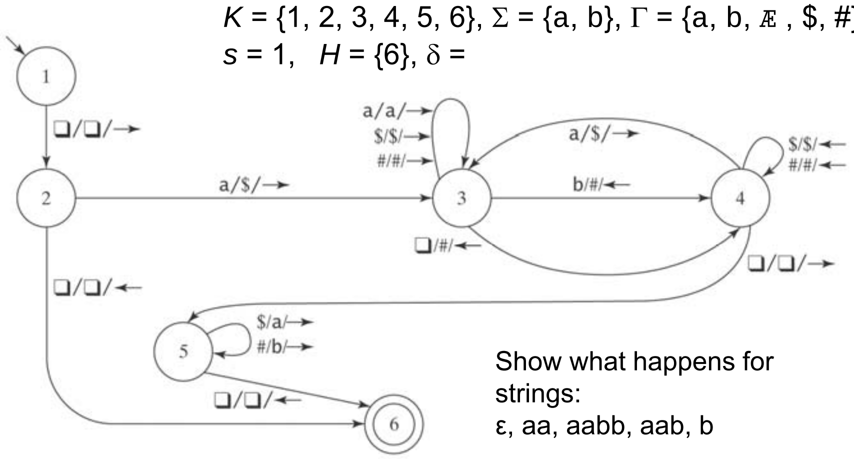
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	a	b	b	❑

6

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

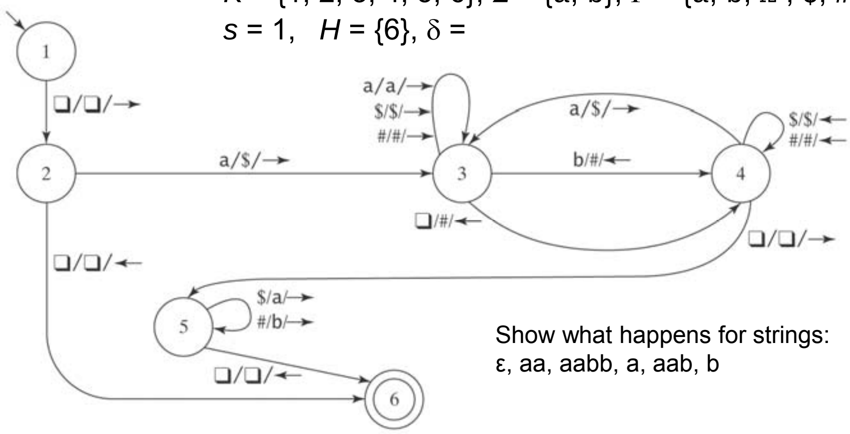
❑	❑	❑	a	a	b	b	❑

1

The steps are the same as previous example until we read the b; skip to there

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =



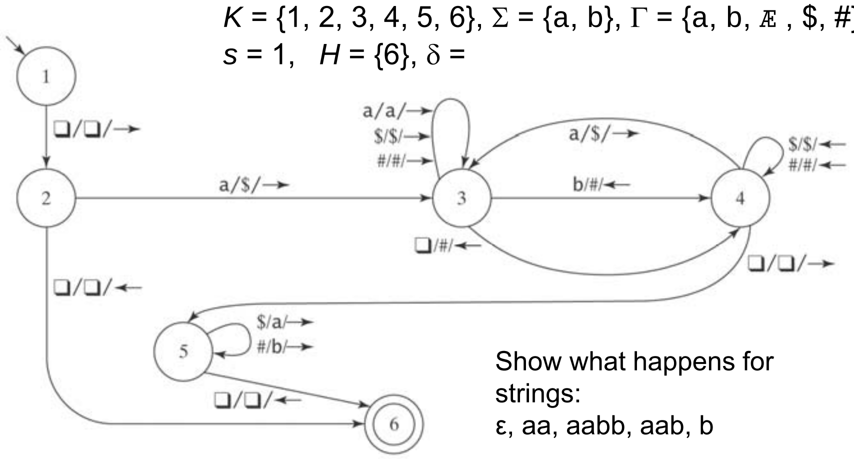
Show what happens for strings:
ε, aa, aabb, a, aab, b

❑	❑	❑	\$	a	b	b	❑

3

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , \$, #},
s = 1, H = {6}, δ =



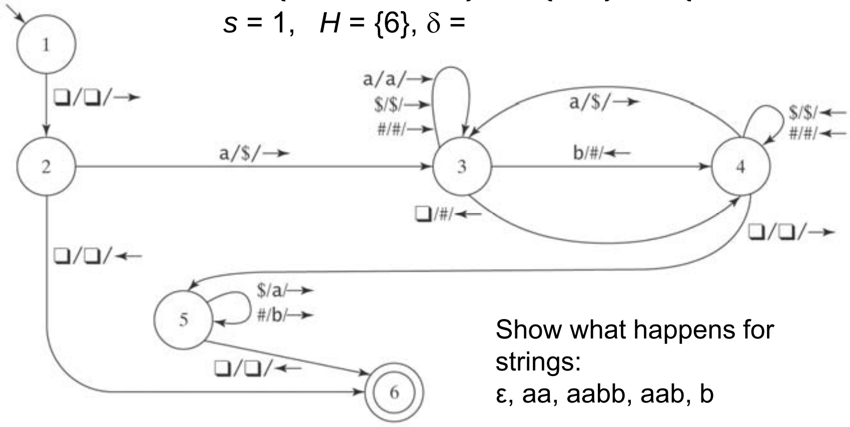
Show what happens for strings:
ε, aa, aabb, aab, b

❑	❑	❑	\$	a	#	b	❑

4

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



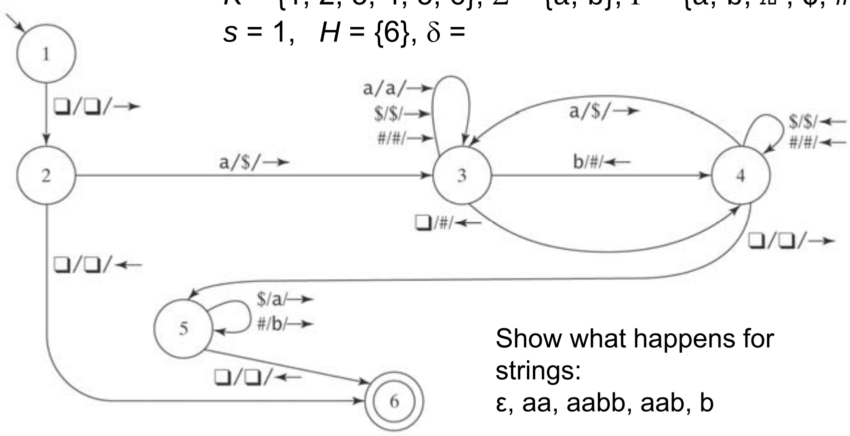
Show what happens for strings:
ε, aa, aabb, aab, b

❑	❑	❑	$	$	#	b	❑

3

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



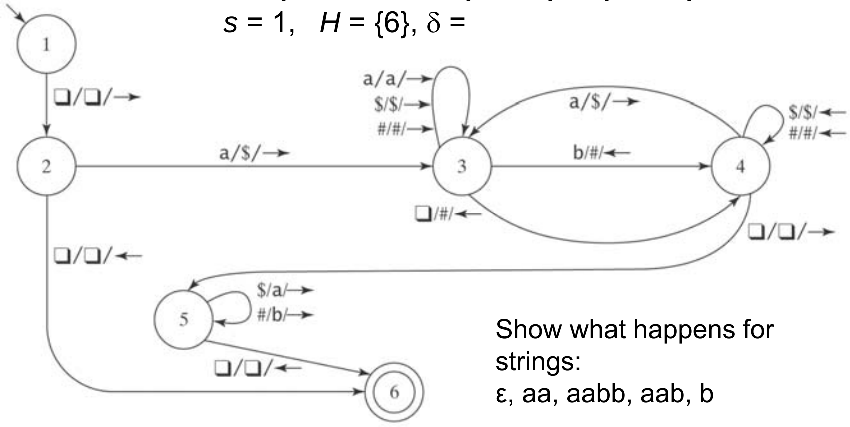
Show what happens for strings:
ε, aa, aabb, aab, b

❑	❑	❑	$	$	#	b	❑

3

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

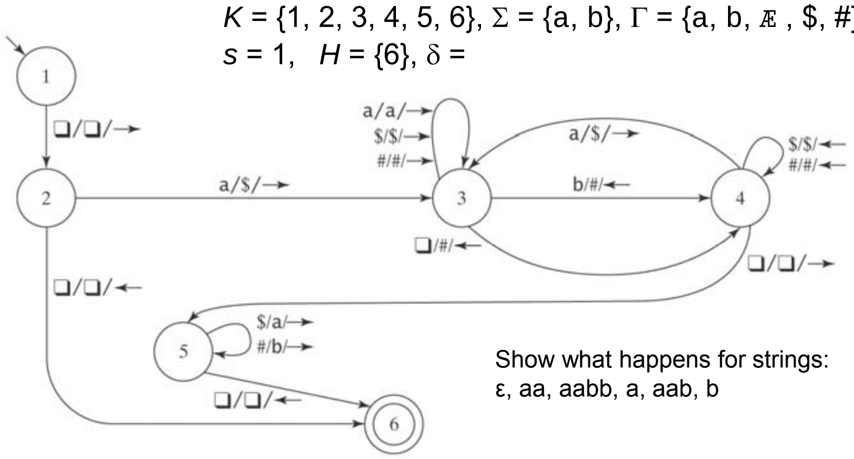
❑	❑	❑	$	$	#	#	❑

4

In state 4, go left until we hit a blank

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



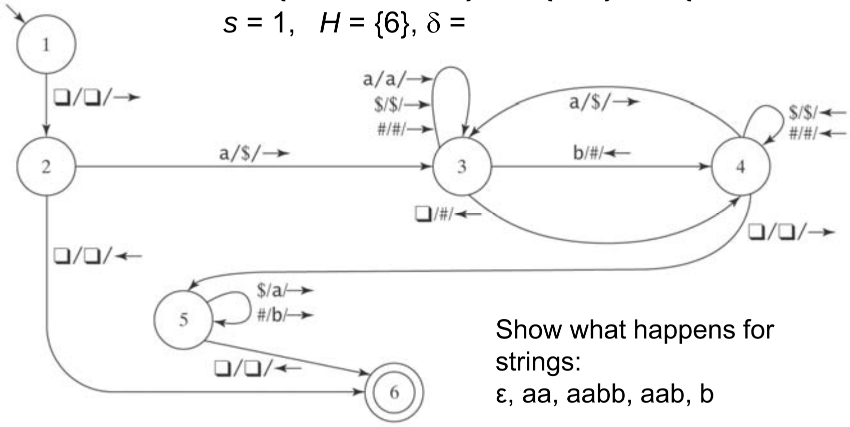
Show what happens for strings:
ε, aa, aabb, a, aab, b

❑	❑	❑	$	$	#	#	❑

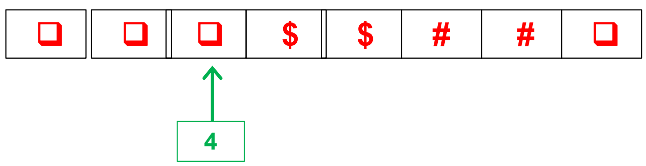
4

The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =

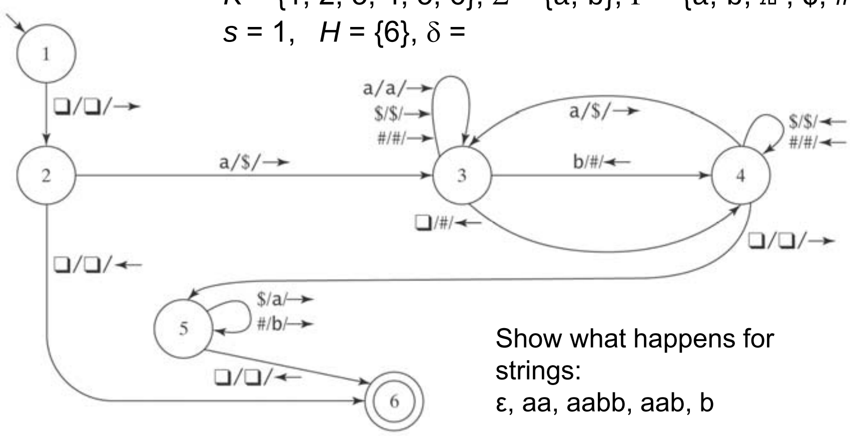


Show what happens for strings:
ε, aa, aabb, aab, b

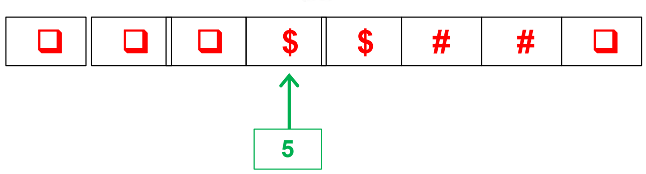


The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =

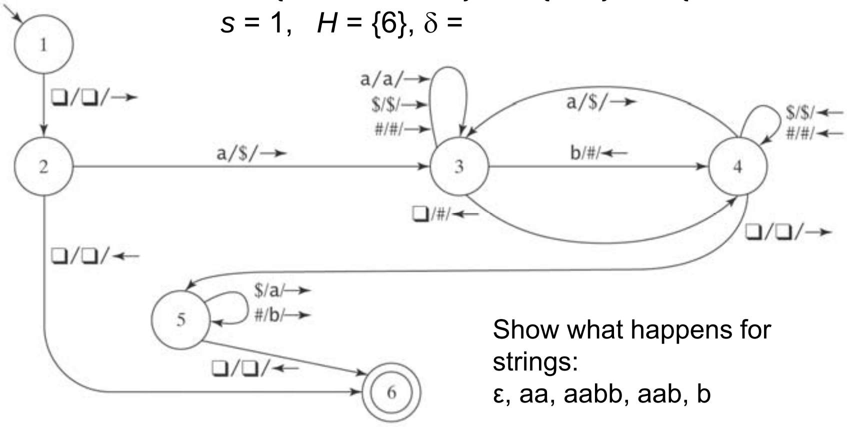


Show what happens for strings:
ε, aa, aabb, aab, b



The Details (aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

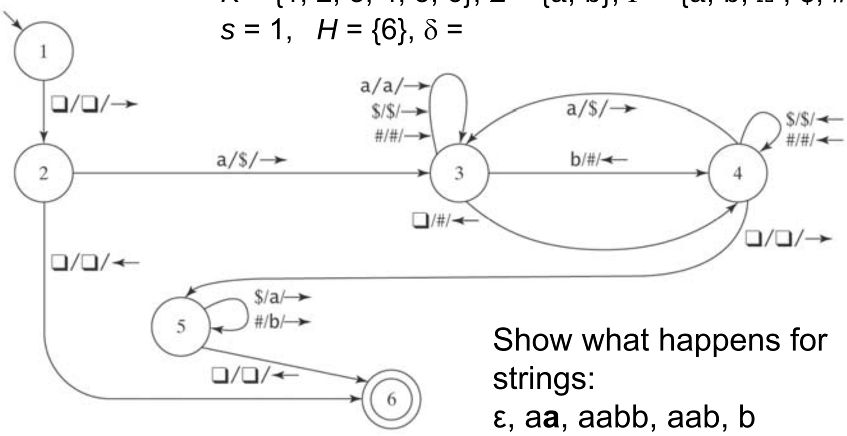
❑	❑	❑	$	$	#	#	❑

↑ 5

Go right, replacing $ with a and # with b, then move left, as in the last part of the previous example.

The Details(aabb)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, Æ , $, #},
s = 1, H = {6}, δ =



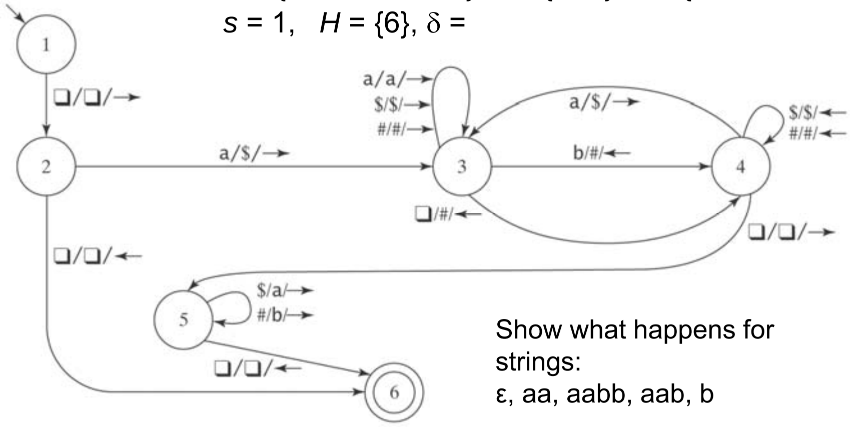
Show what happens for strings:
ε, a**a**, aabb, aab, b

❑	❑	❑	a	a	b	b	❑

↑ 6

The Details (aab)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑, $, #},
s = 1, *H* = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

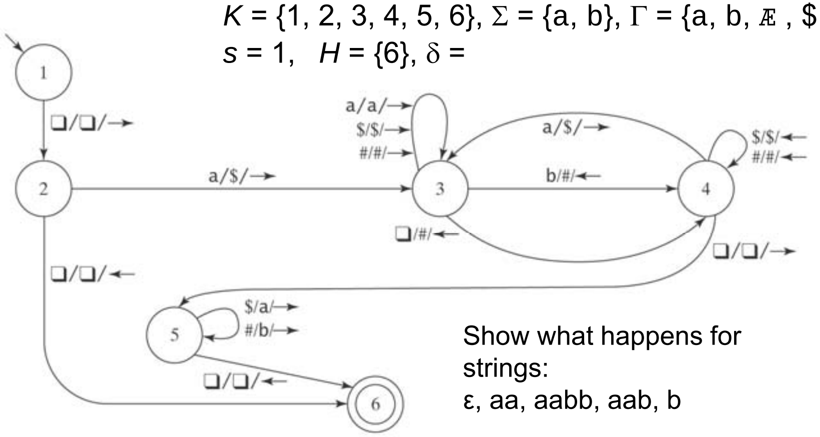
❑	❑	❑	a	a	b	❑	❑

1

You should try this one.

The Details (b)

K = {1, 2, 3, 4, 5, 6}, Σ = {a, b}, Γ = {a, b, ❑, $, #},
s = 1, *H* = {6}, δ =



Show what happens for strings:
ε, aa, aabb, aab, b

❑	❑	❑	b	❑	❑	❑	❑

1

In the first step, move right. Then there is no transition in the diagram. But there is an implied transition to a dead state, i.e. a new halting state that does not accept.

Notes on Programming

Turing machinesa strong procedural feel, with one phase coming after another.

There are common idioms, such as
"scan left until you find a blank"

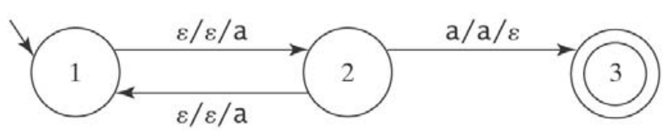
There are some common ways to scan back and forth marking things off.

Often there is a final phase to fix up the output.

Even a very simple machine can be a nuisance to write.

Halting

- A **DFSM** M, on input w, is guaranteed to halt in $|w|$ steps.
- A **PDA** M, on input w, is not guaranteed to halt. To see why, consider again M =



But there exists an algorithm to construct an equivalent PDA M' that is guaranteed to halt.

- A **TM** M, on input w, is not guaranteed to halt. And there is no algorithm to construct an equivalent TM that is guaranteed to halt.

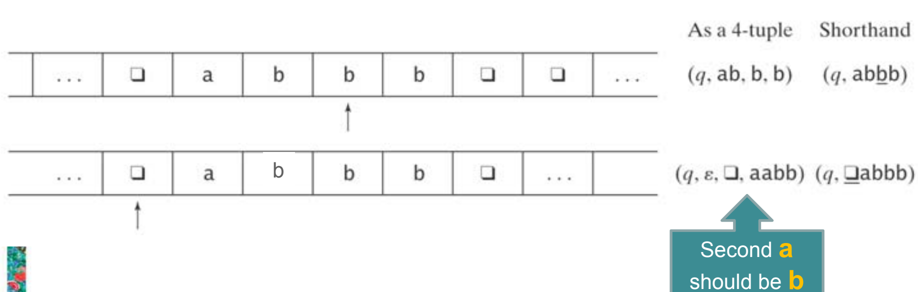
Formalize TM computations

A *configuration* of TM $M = (K, \Sigma, \Gamma, s, H)$ is an element of:

$K \times ((\Gamma - \{Æ\})\, \Gamma^*) \cup \{\varepsilon\} \times \Gamma \times (\Gamma^*\, (\Gamma - \{Æ\})) \cup \{\varepsilon\}$

state	before current tape square	current tape square	after current tape square

Example Configurations



(1) $(q, \text{ab}, \text{b}, \text{b})$ written more simply as $(q, \text{ab}\underline{\text{b}}\text{b})$
(2) $(q, \varepsilon, Æ, \text{abbb})$ written more simply as $(q, \underline{Æ}\text{abbb})$

Initial configuration is always $(s, \underline{Æ}w)$

Yields and Computations

$(q_1, w_1) \vdash_M (q_2, w_2)$ iff (q_2, w_2) is derivable *via* δ in one step.

For any TM M, let $\vdash_M{}^*$ be the reflexive, transitive closure of $|\text{-}_M$.

Configuration C_1 ***yields*** configuration C_2 if: $C_1 \vdash_M{}^* C_2$.

A ***path*** through M is a sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that C_0 is the initial configuration and:

$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

A **computation** by M is a path that halts.

If a computation is of *length n* (has n steps), we can also write:

$C_0 \vdash_M{}^n C_n$

Exercise

If n and m are non-negative integers, ***monus(n, m)*** *is defined to be n – m* if $n > m$, and 0 if $n \leq m$.

Draw the diagram for a TM M whose input is $1^n;1^m$ and whose output is $1^{monus(n, m)}$. When M halts, the read/write head should be positioned on the blank before the first 1.

(not used in class 2018)Exercise

A TM to recognize { ww^R : $w \in \{a, b\}^*$ }.

If the input string is in the language, the machine should halt with y as its current tape symbol

If not, it should halt with n as its current tape symbol.

The final symbols on the rest of the tape may be anything.

TMs are complicated

- … and *very* low-level!
- We need higher-level "abbreviations".
  - Macros
  - Subroutines

A Macro language for Turing Machines

(1) Define some basic machines

- Symbol writing machines

  For each $x \in \Gamma$, define M_x, written as just x, to be a machine that writes x. Read-write head ends up in original position.

- Head-moving machines

  R: for each $x \in \Gamma$, $\delta(s, x) = (h, x, \rightarrow)$
  L: for each $x \in \Gamma$, $\delta(s, x) = (h, x, \leftarrow)$

- Machines that simply halt:

  h, which simply halts (don't care whether it accepts).
  n, which halts and rejects.
  y, which halts and accepts.

You need to learn (but not memorize) this simple language. I will use it and I expect you to use it on HW and tests.

Checking Inputs and Combining Machines

Next we need to describe how to:

- Check the tape and branch based on what character we see, and
- Combine the basic machines to form larger ones.

To do this, we need two forms:

- $M_1\ M_2$

- $M_1 \xrightarrow{\textit{<condition>}} M_2$

Turing Machines Macros Cont'd

Example:

$$\begin{array}{lcl} >M_1 & \xrightarrow{\text{a}} & M_2 \\ \text{b}\downarrow & & \\ M_3 & & \end{array}$$

- Start in the start state of M_1.
- Compute until M_1 reaches a halt state.
- Examine the tape and take the appropriate transition.
- Start in the start state of the next machine, etc.
- Halt if any component reaches a halt state and has no place to go.
- If any component fails to halt, then the entire machine may fail to halt.

More macros

$M_1 \overset{\xrightarrow{\text{a}}}{\underset{\text{b}}{\longrightarrow}} M_2$ becomes $M_1 \xrightarrow{\text{a, b}} M_2$

$M_1 \xrightarrow{\text{all elems of } \Gamma} M_2$ becomes $M_1 \longrightarrow M_2$ or M_1M_2

Variables

$M_1 \xrightarrow[\text{except a}]{\text{all elems of } \Gamma} M_2$ becomes $M_1 \xrightarrow{x \leftarrow \neg \text{a}} M_2$

and x takes on the value of the current square

$M_1 \xrightarrow{\text{a, b}} M_2$ becomes $M_1 \xrightarrow{x \leftarrow \text{a, b}} M_2$

and x takes on the value of the current square

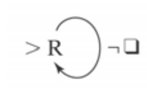
$M_1 \xrightarrow{x = y} M_2$

if $x = y$ then take the transition

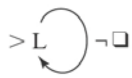
e.g., $> \xrightarrow{x \leftarrow \neg Æ} Rx$

if the current square is not blank, go right and copy it.

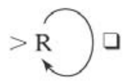
Blank/Non-blank Search Machines



Find the first blank square to the right of the current square. $R_{Æ}$



Find the first blank square to the left of the current square. $L_{Æ}$



Find the first nonblank square to the right of the current square. $R_{\neg Æ}$

Find the first nonblank square to the left of the current square $L_{\neg Æ}$

More Search Machines

L_a	Find the first occurrence of a to the left of the current square.
$R_{a,b}$	Find the first occurrence of a or b to the right of the current square.
$\begin{array}{lcl} L_{a,b} & \xrightarrow{a} & M_1 \\ \downarrow b & & \\ M_2 & & \end{array}$	Find the first occurrence of a or b to the left of the current square, then go to M_1 if the detected character is a; go to M_2 if the detected character is b.
$L_{x\leftarrow a,b}$	Find the first occurrence of a or b to the left of the current square and set *x* to the value found.
$L_{x\leftarrow a,b}Rx$	Find the first occurrence of a or b to the left of the current square, set *x* to the value found, move one square to the right, and write *x* (a or b).