بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه چهارم: زبان‌های غیر مستقل از زمینه و ماشین تورینگ

Theory of computation - 002 - S04 - Non-CFG Languanges and Turing Machine

Review

Last time:

- Context free grammars (CFGs)
- Context free languages (CFLs)
- Pushdown automata (PDA)
- Converting CFGs to PDAs

Review

Last time:

- Context free grammars (CFGs)
- Context free languages (CFLs)
- Pushdown automata (PDA)
- Converting CFGs to PDAs

Today: (Sipser §2.3, §3.1)

- Proving languages not Context Free
- Turing machines
- T-recognizable and T-decidable languages

Review

Last time:

- Context free grammars (CFGs)
- Context free languages (CFLs)
- Pushdown automata (PDA)
- Converting CFGs to PDAs

Today: (Sipser §2.3, §3.1)

- Proving languages not Context Free
- Turing machines
- T-recognizable and T-decidable languages

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\longrightarrow$ Done.

$\longleftarrow$ Need to know the fact, not the proof

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\longrightarrow$ Done.

$\longleftarrow$ Need to know the fact, not the proof

Corollaries:

1) Every regular language is a CFL.
2) If A is a CFL and B is regular then $A \cap B$ is a CFL.

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\rightarrow$ Done.

$\leftarrow$ Need to know the fact, not the proof

Corollaries:

1) Every regular language is a CFL.
2) If A is a CFL and B is regular then $A \cap B$ is a CFL.

Proof sketch of (2):

While reading the input, the finite control of the PDA for A simulates the DFA for B.

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\rightarrow$ Done.

$\leftarrow$ Need to know the fact, not the proof

Corollaries:

1) Every regular language is a CFL.
2) If A is a CFL and B is regular then $A \cap B$ is a CFL.

Proof sketch of (2):

While reading the input, the finite control of the PDA for A simulates the DFA for B.

Note 1: If A and B are CFLs then $A \cap B$ may not be a CFL (will show today).

Therefore the class of CFLs is not closed under $\cap$.

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\longrightarrow$ Done.

$\longleftarrow$ Need to know the fact, not the proof

Corollaries:

1) Every regular language is a CFL.
2) If A is a CFL and B is regular then $A \cap B$ is a CFL.

Proof sketch of (2):

While reading the input, the finite control of the PDA for A simulates the DFA for B.

Note 1: If A and B are CFLs then $A \cap B$ may not be a CFL (will show today).
Therefore the class of CFLs is not closed under $\cap$.

Note 2: The class of CFLs is closed under $\cup$, $\circ$, $*$ (see Pset 2).

Equivalence of CFGs and PDAs

Recall Theorem: A is a CFL iff some PDA recognizes A

$\rightarrow$ Done.

$\leftarrow$ Need to know the fact, not the proof

Corollaries:

1) Every regular language is a CFL.
2) If A is a CFL and B is regular then $A \cap B$ is a CFL.

Proof sketch of (2):

While reading the input, the finite control of the PDA for A simulates the DFA for B.

Note 1: If A and B are CFLs then $A \cap B$ may not be a CFL (will show today).
Therefore the class of CFLs is not closed under $\cap$.

Note 2: The class of CFLs is closed under $\cup$, $\circ$, $*$ (see Pset 2).

Proving languages not Context Free

Let $B = \left\{ 0^k 1^k 2^k \,\middle|\, k \geq 0 \right\}$. We will show that B isn't a CFL.

Proving languages not Context Free

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Informally: All long strings in A are pumpable and stay in A.

Proving languages not Context Free

Let $B = \{0^k1^k2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

$$s = \overbrace{\rule{6em}{0.4pt}}^{\geq p} \in A$$

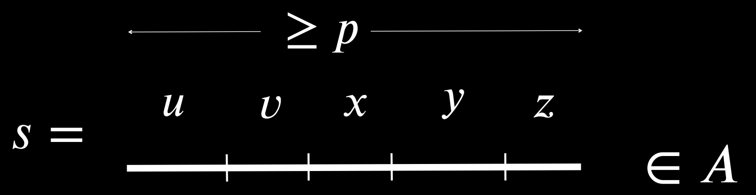
Informally: All long strings in A are pumpable and stay in A.

Proving languages not Context Free

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$



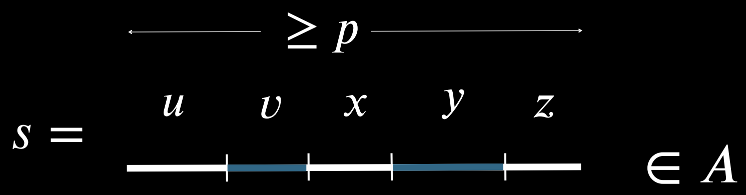
Informally: All long strings in A are pumpable and stay in A.

Proving languages not Context Free

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$



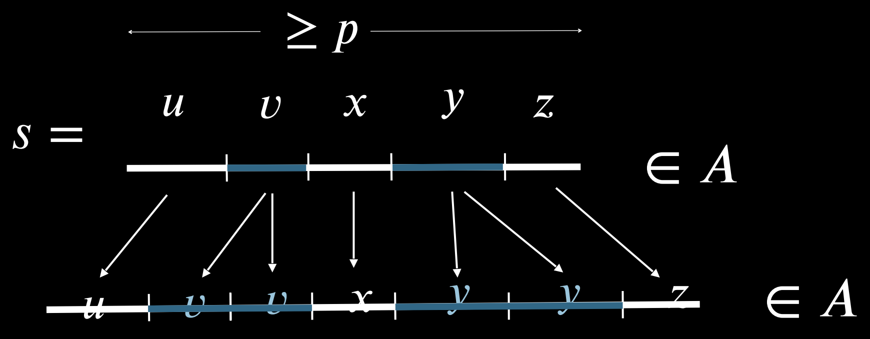
Informally: All long strings in A are pumpable and stay in A.

Proving languages not Context Free

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$



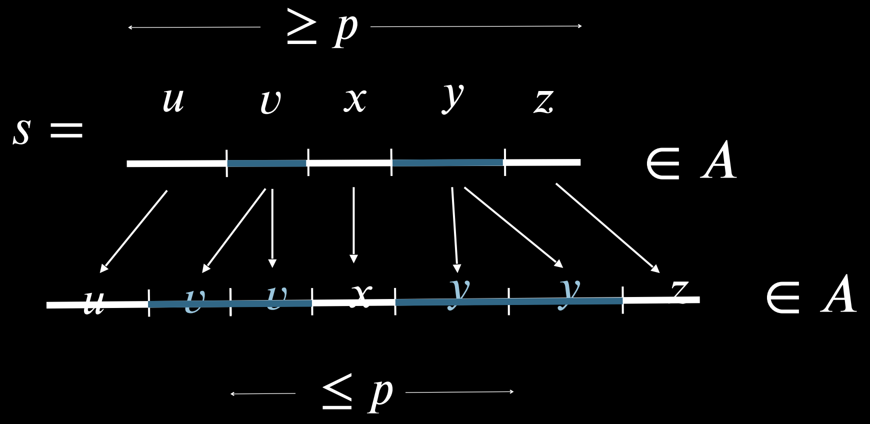
Informally: All long strings in A are pumpable and stay in A.

Proving languages not Context Free

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$. We will show that B isn't a CFL.

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$



Informally: All long strings in A are pumpable and stay in A.

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$

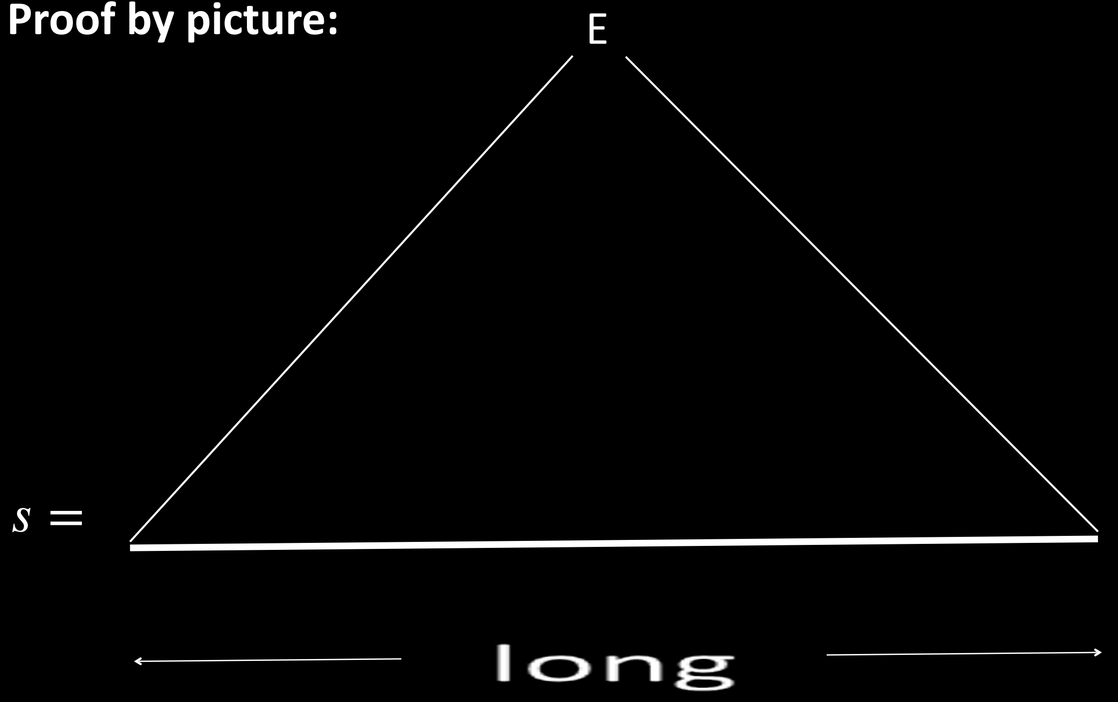
long

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:



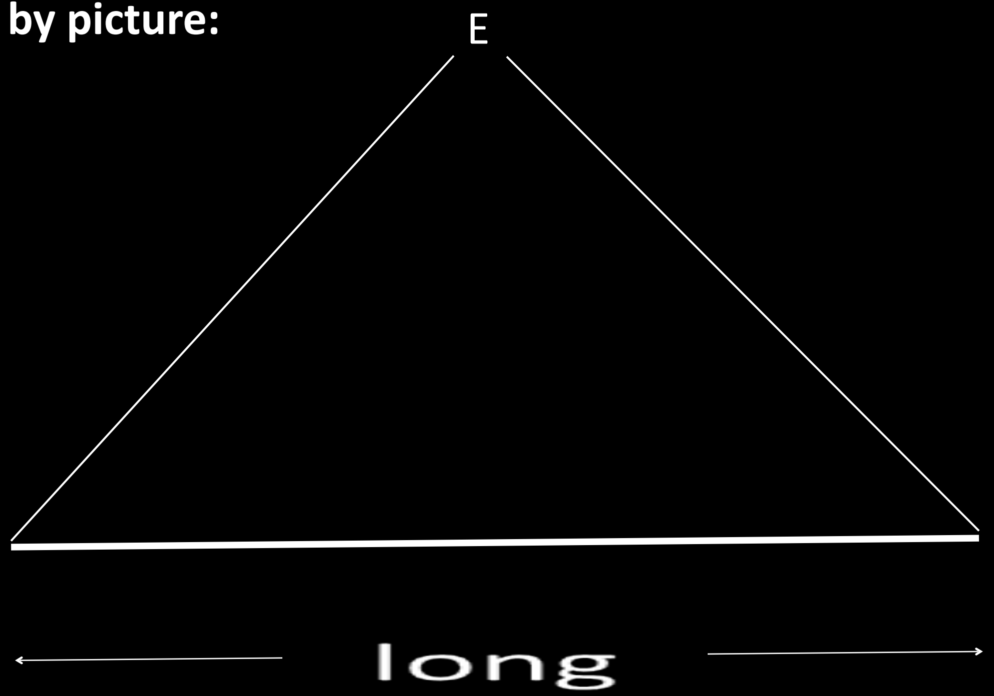
Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$



Long s →
tall parse tree

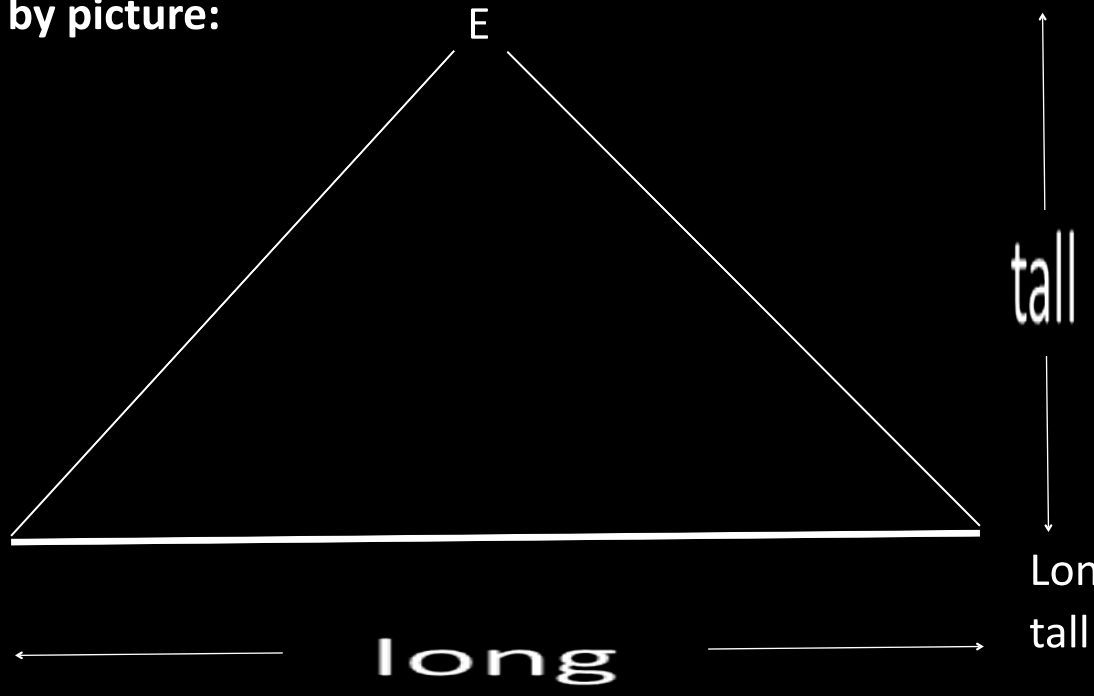
Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$



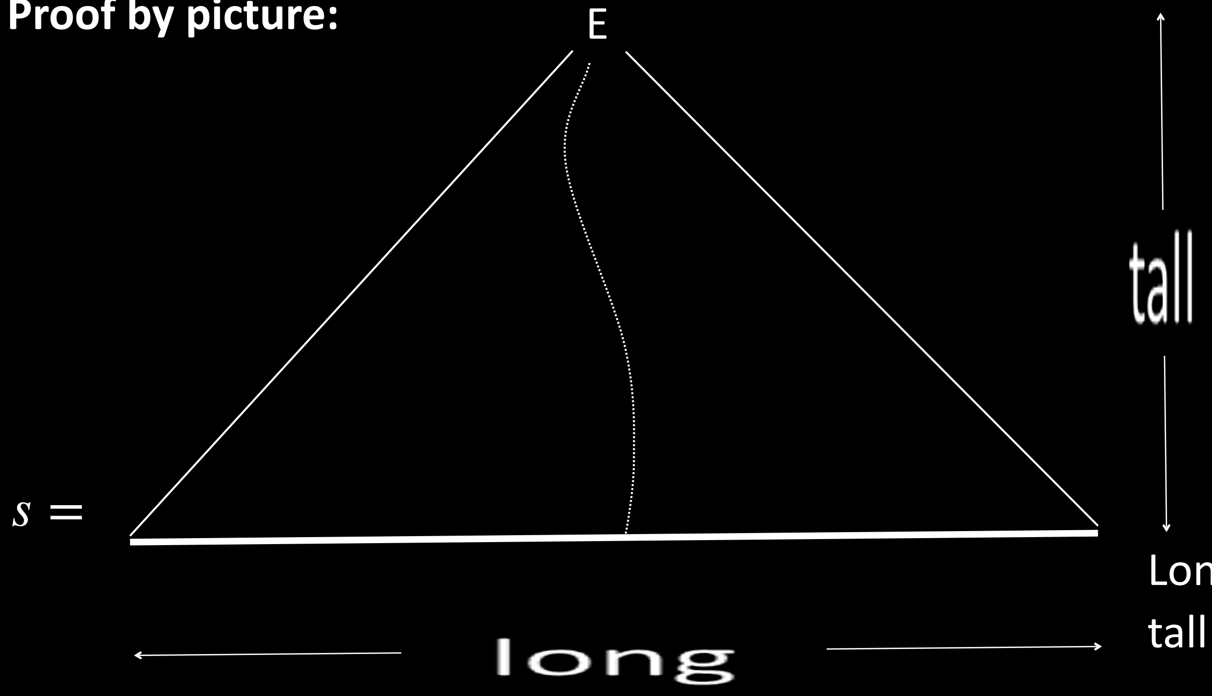
Long s →
tall parse tree

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:



Long $s \rightarrow$ tall parse tree

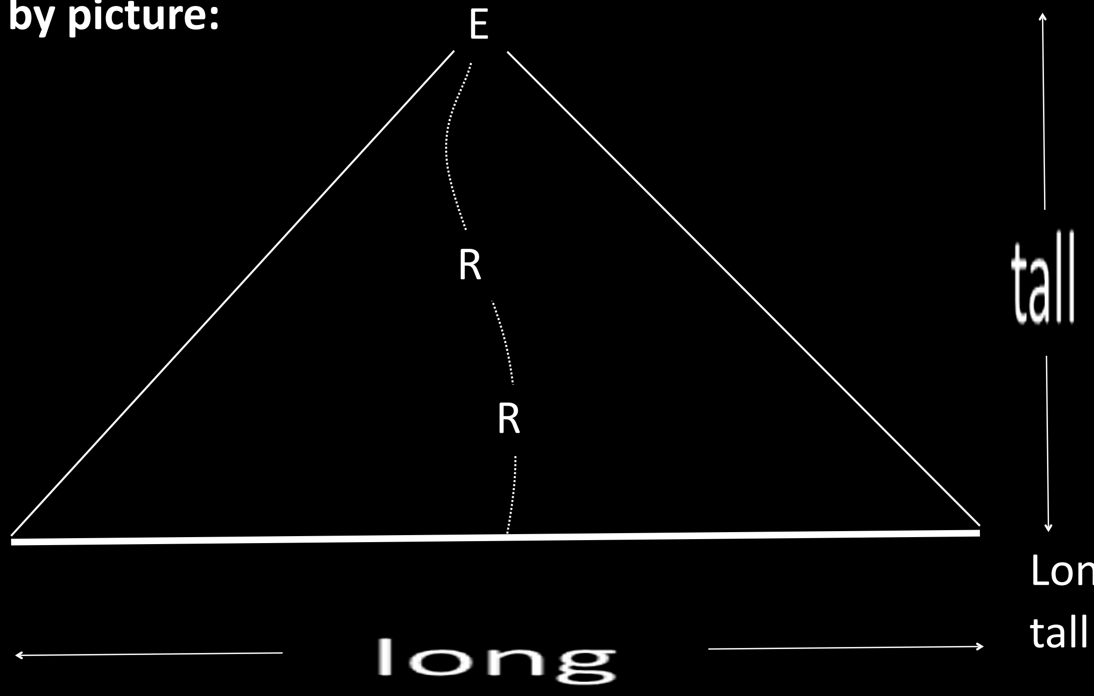
Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$



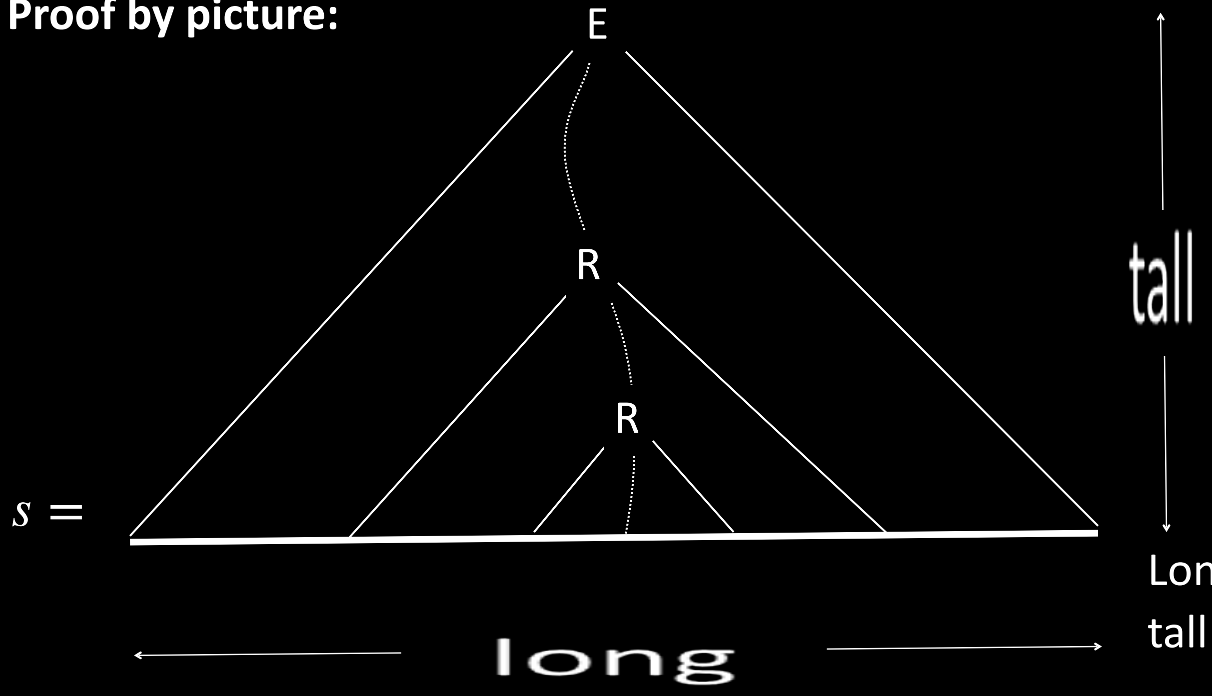
Long $s \rightarrow$
tall parse tree

Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:



Long $s \rightarrow$
tall parse tree

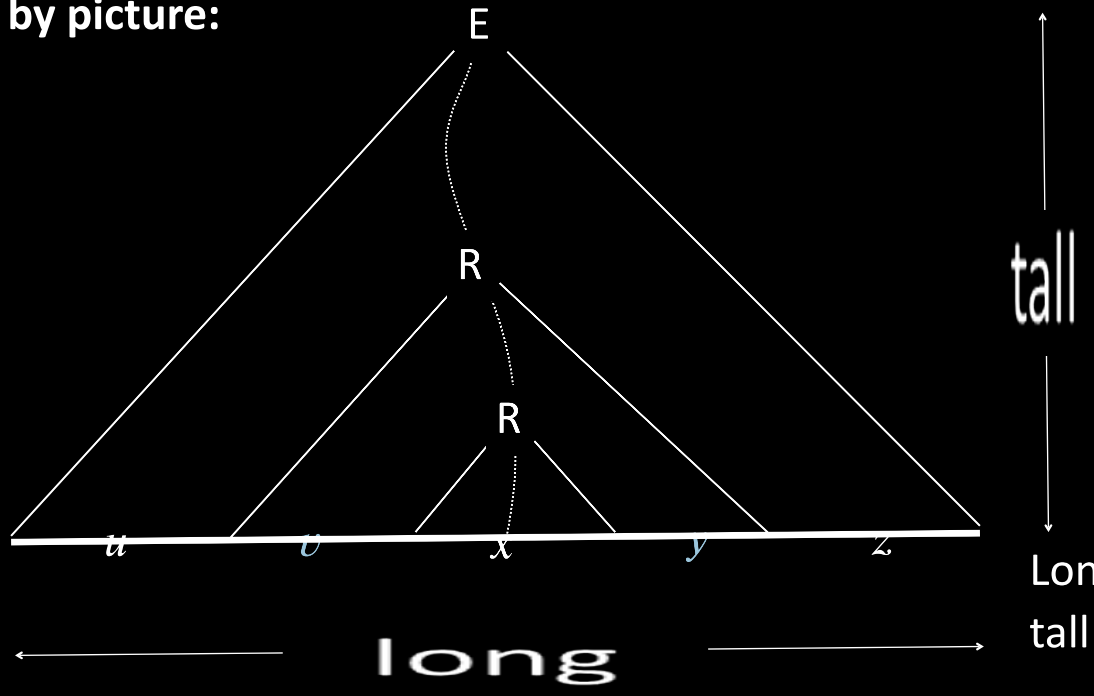
Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$



Long $s \rightarrow$
tall parse tree

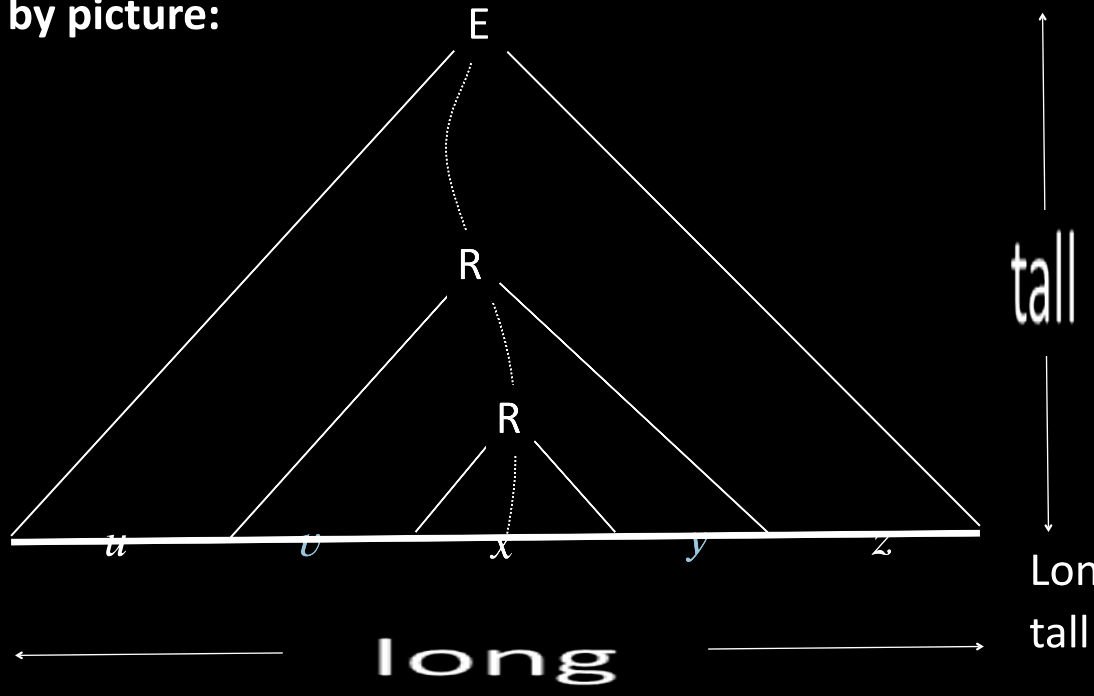
Pumping Lemma – Proof

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

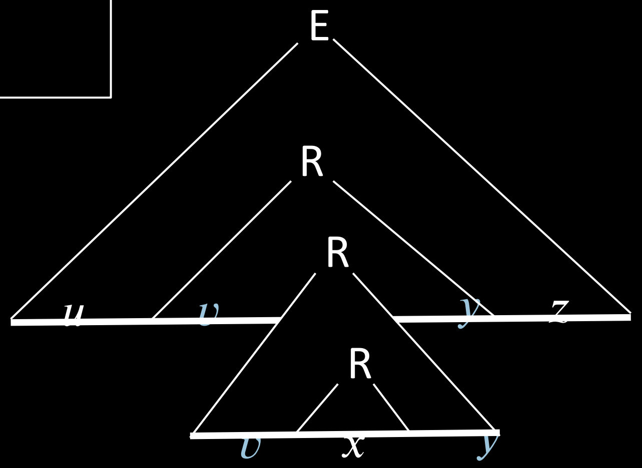
1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

$s =$



Long $s \rightarrow$
tall parse tree



Generates $uvvxyyz$

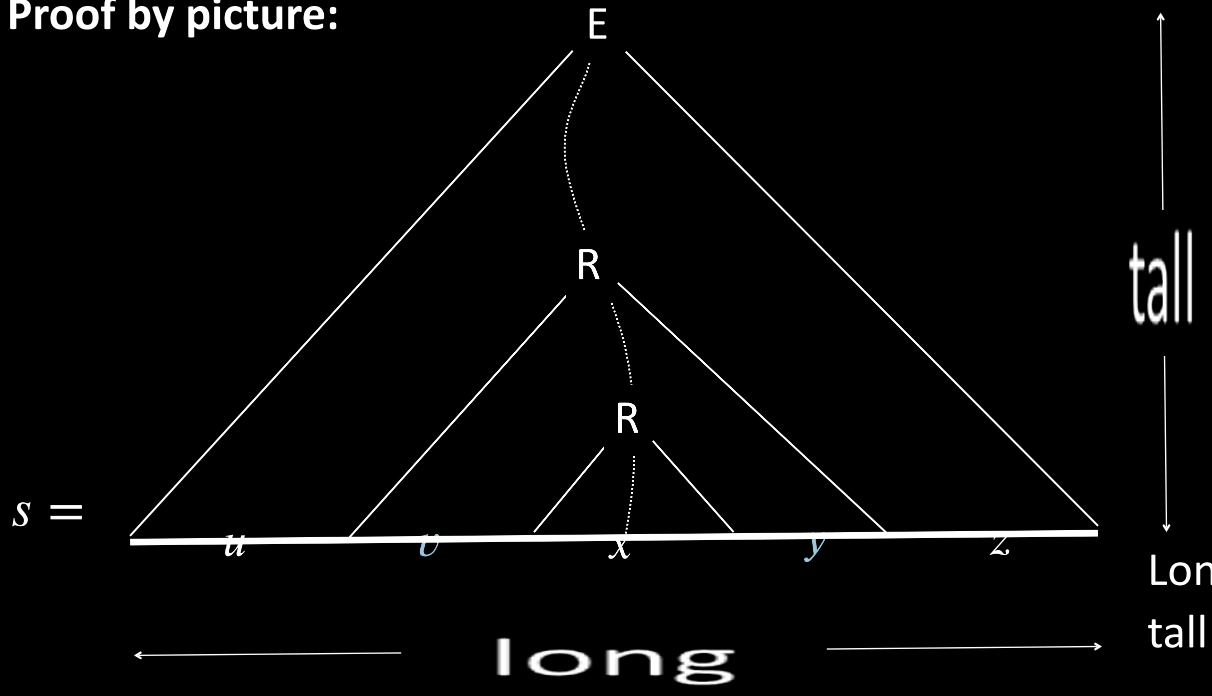
$= uv^2xy^2z$

Pumping Lemma – Proof

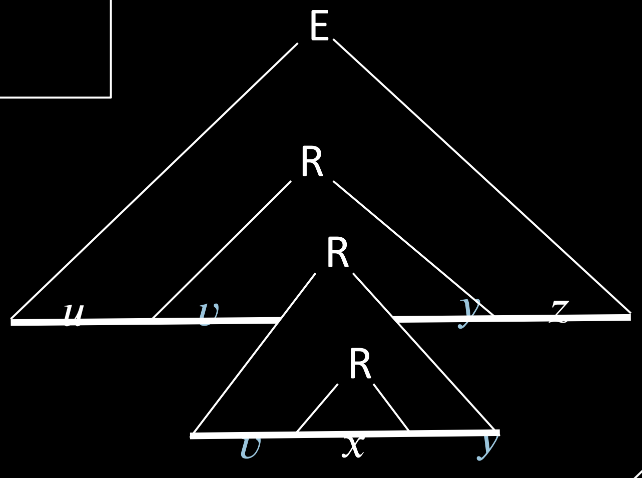
Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

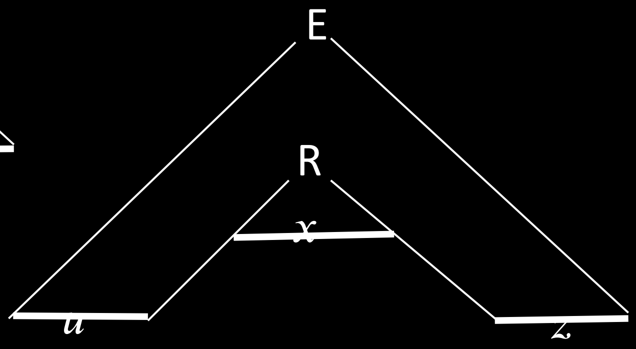


Long $s \rightarrow$ tall parse tree



Generates $uvvxyyz$

$= uv^2xy^2z$



Generates uxz

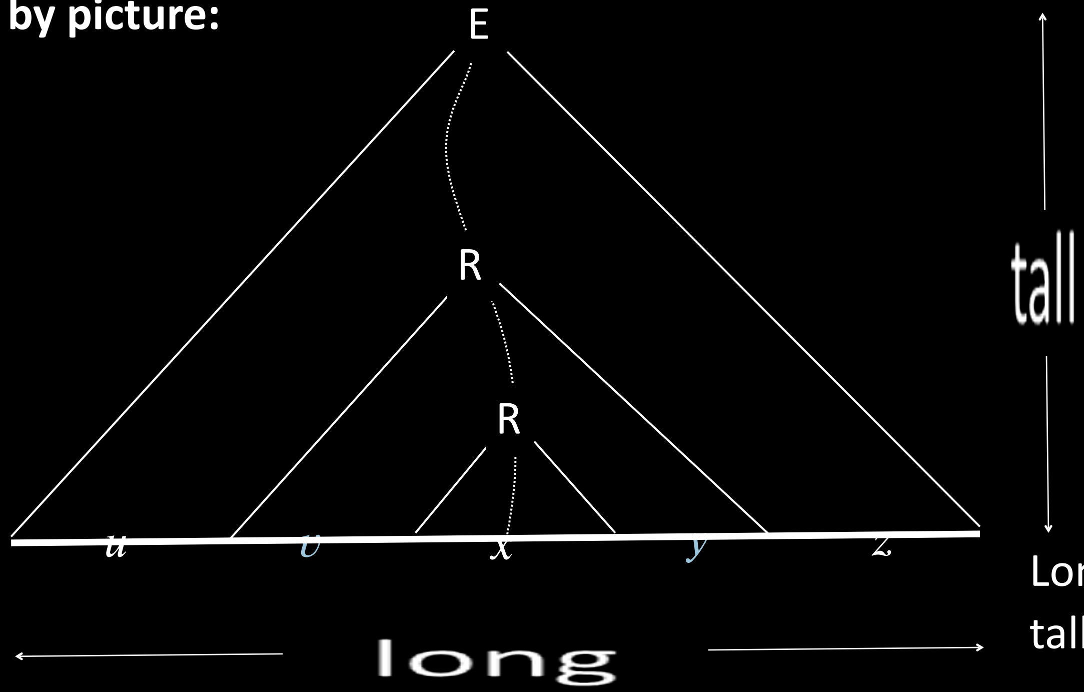
$= uv^0xy^0z$

Pumping Lemma – Proof

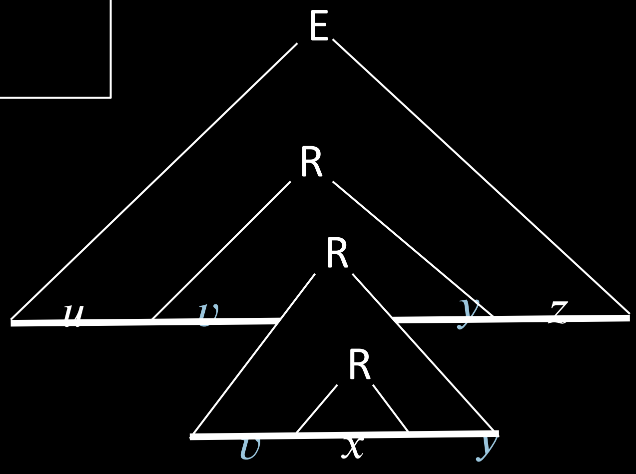
Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Proof by picture:

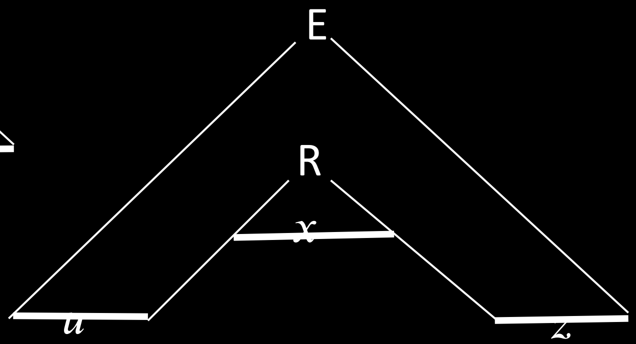
$s =$ 

Long s → tall parse tree



Generates $uvvxyyz$

$= uv^2xy^2z$



Generates uxz

$= uv^0xy^0z$

"cutting and pasting" argument

Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

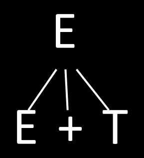
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E ⟶ E+T)

$=$ the max branching of the parse tree



Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree

E

E + T

Let $h =$ the height of the parse tree for s.

Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E ⟶ E+T)

$=$ the max branching of the parse tree

E

E + T

Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

So $|s| \leq b^h$.

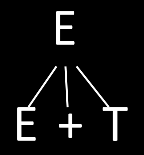
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq$ ε
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

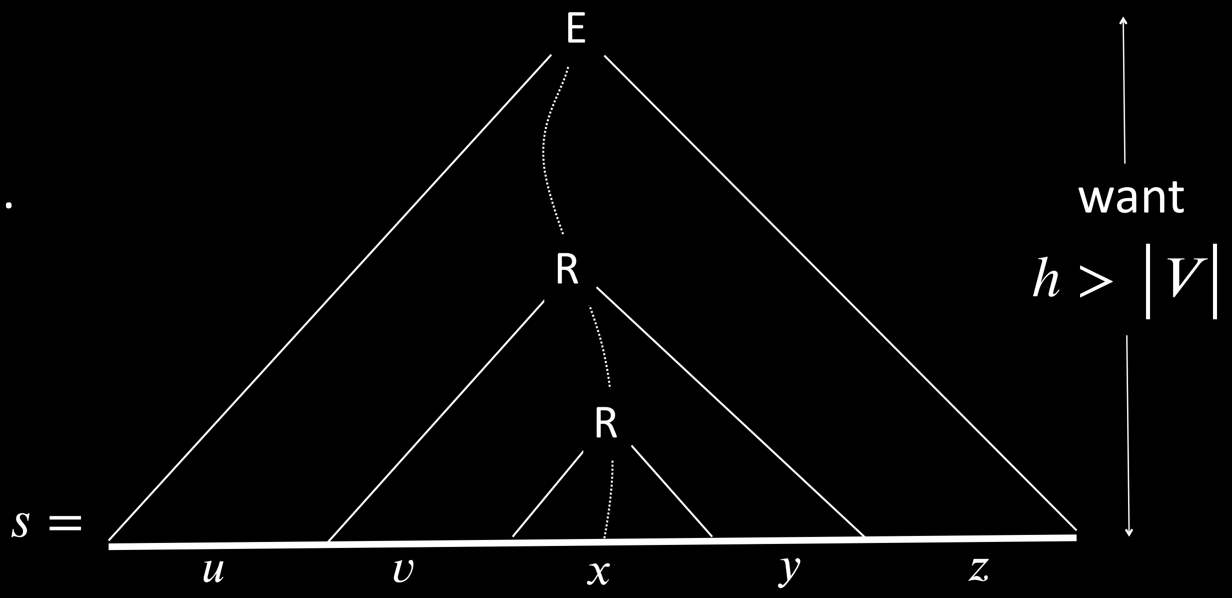
$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

So $|s| \leq b^h$.



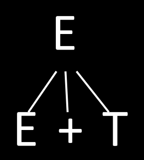
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

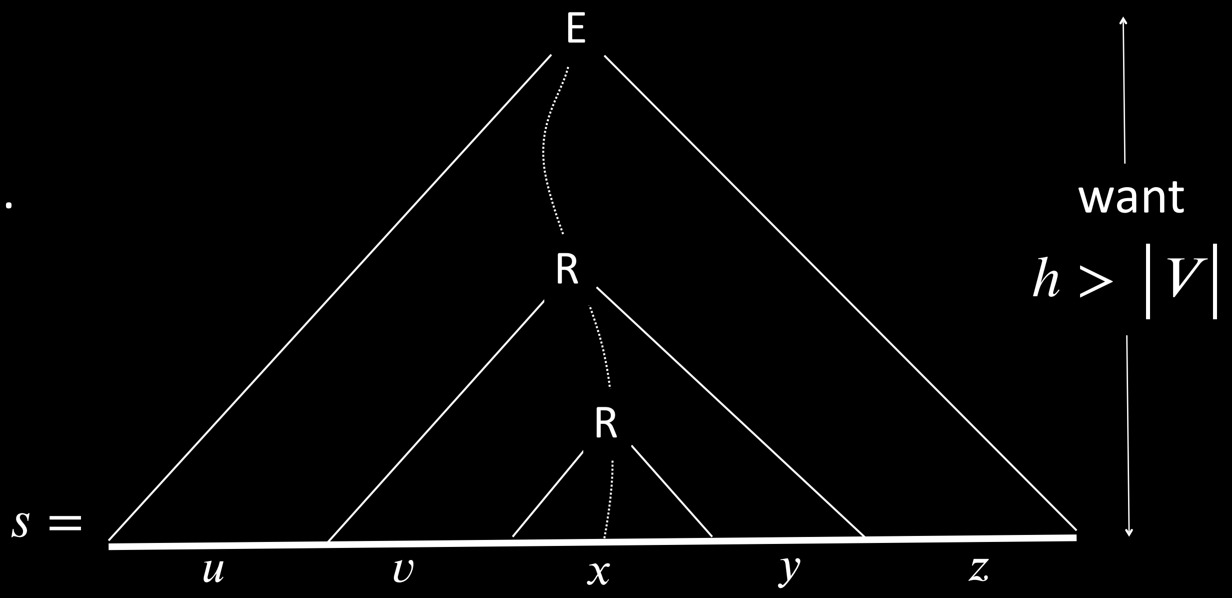
$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

So $|s| \leq b^h$.



use $|s| > b^{|V|}$

set $p = b^{|V|} + 1$

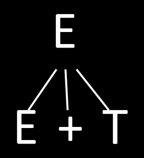
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree

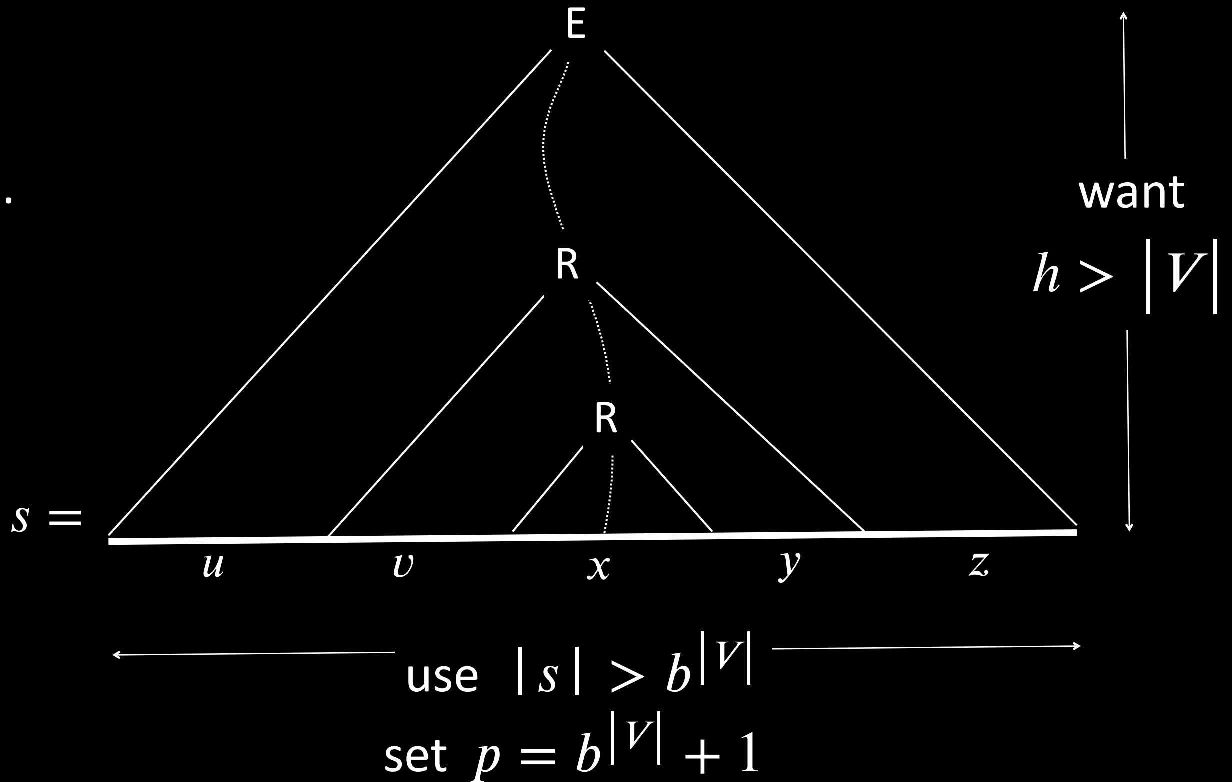


Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.



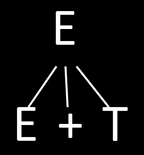
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree



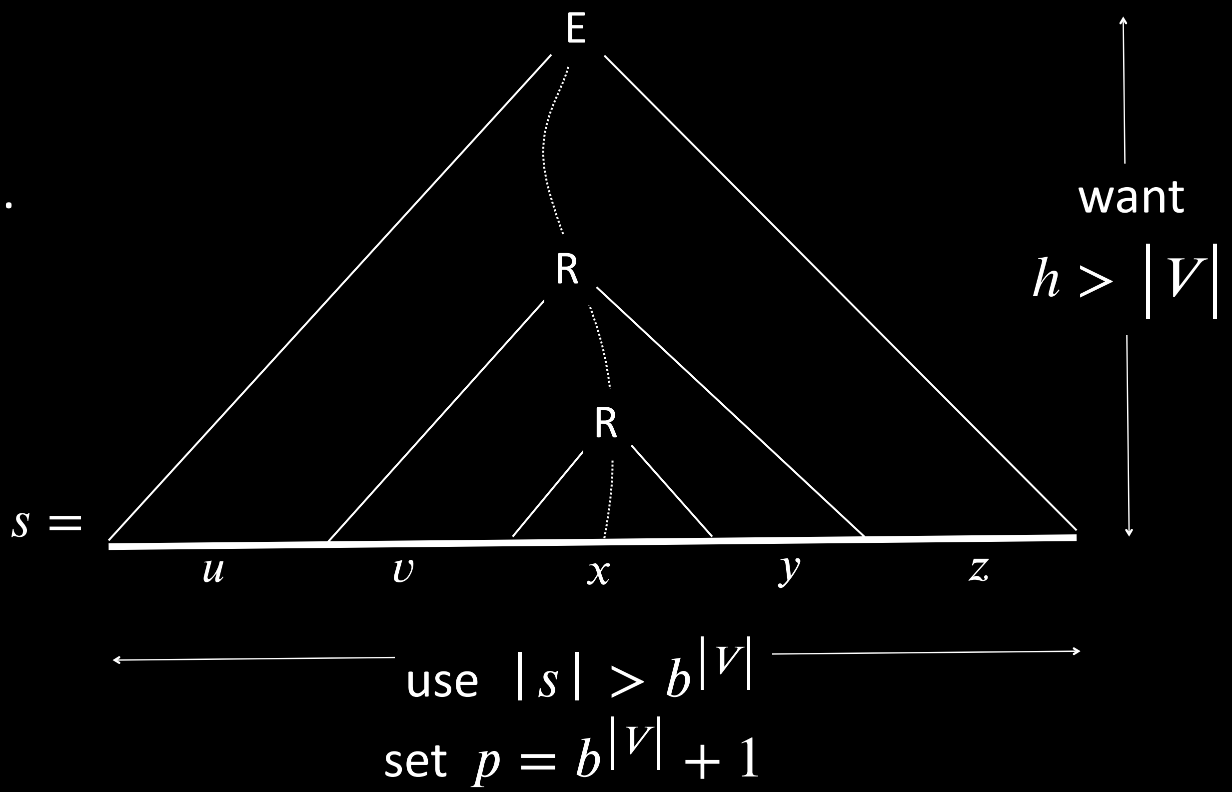
Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.



Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree

E

E + T

Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

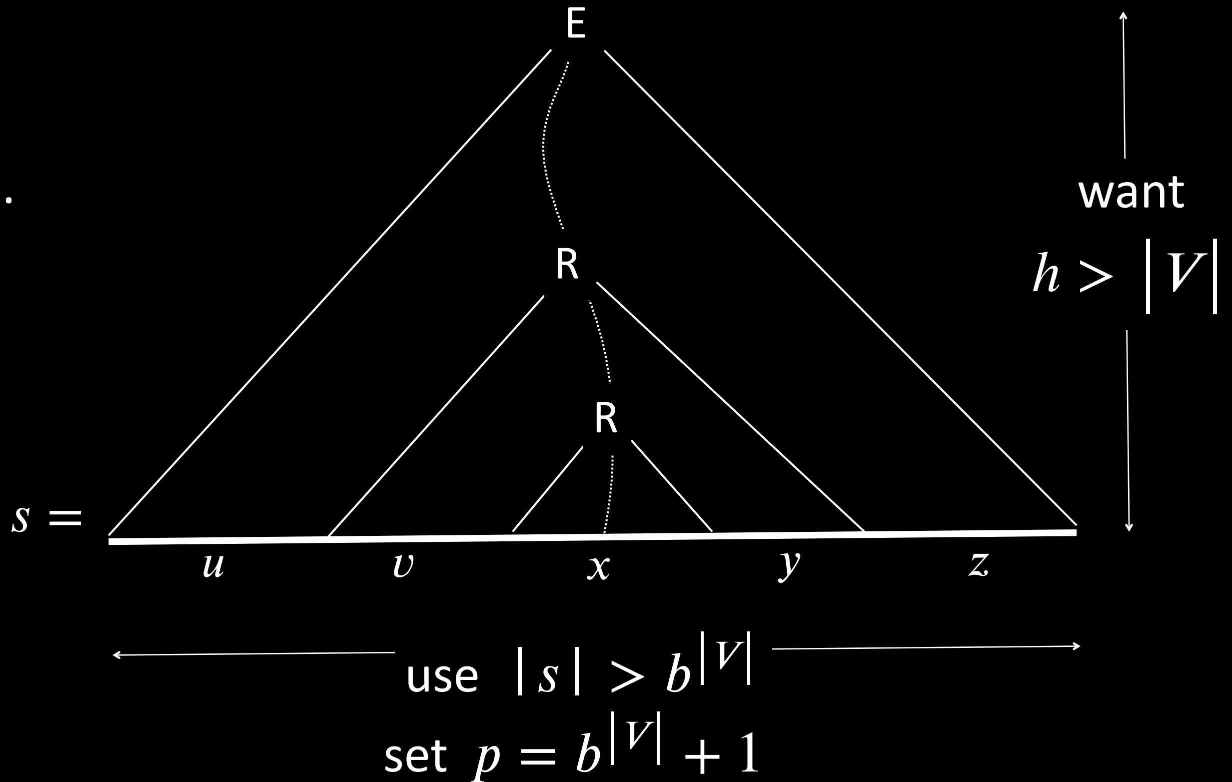
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.

So some variable R must repeat on a path.



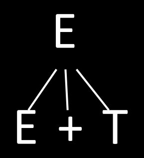
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$ ...cutting and pasting
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

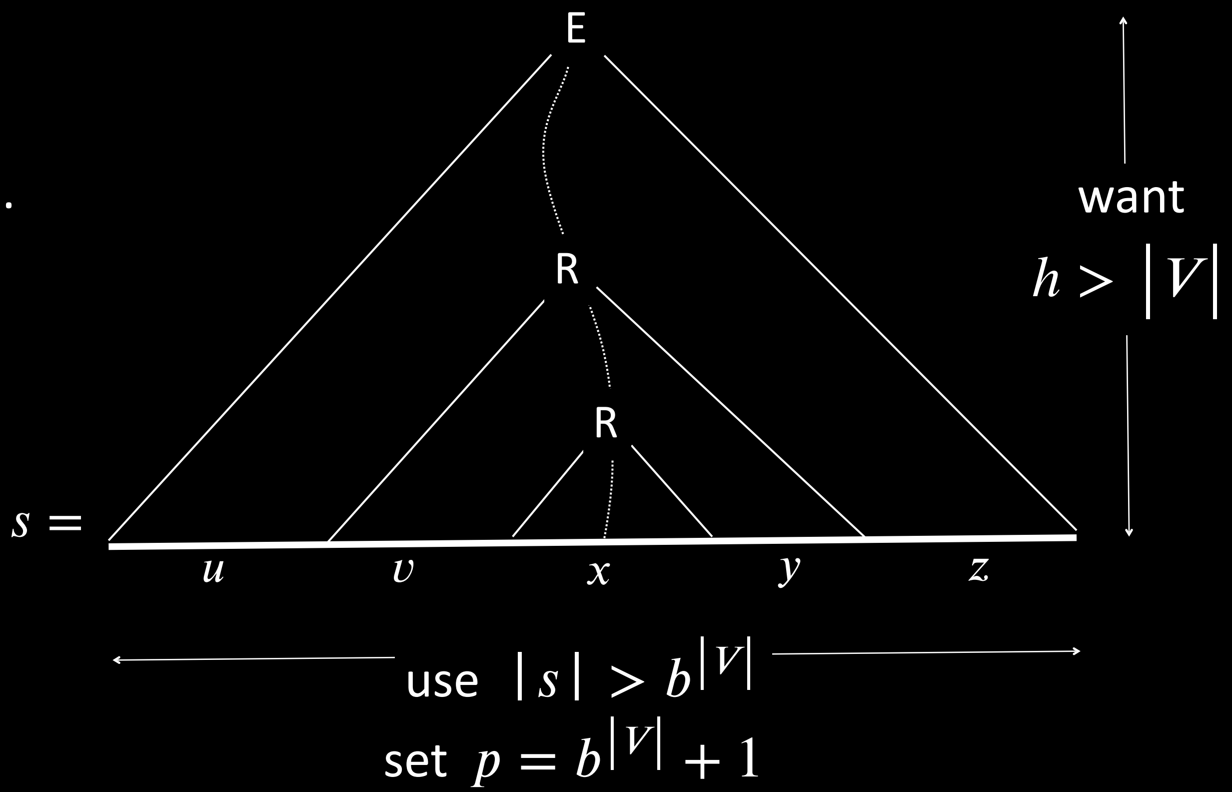
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.

So some variable R must repeat on a path.



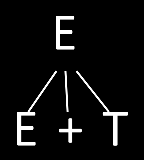
Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$ ...cutting and pasting
2) $vy \neq \varepsilon$ ...start with the smallest parse tree for s
3) $|vxy| \leq p$

Let $b =$ the length of the longest right hand side of a rule (E $\rightarrow$ E+T)

$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

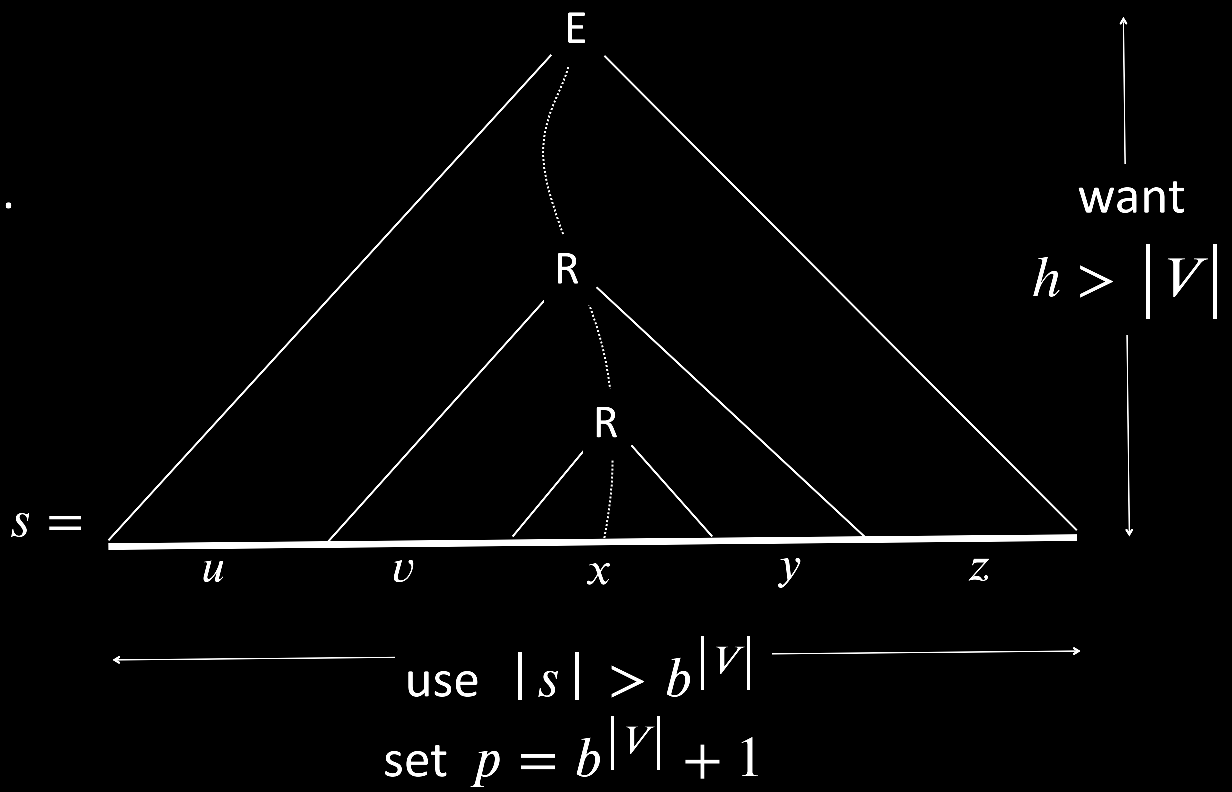
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.

So some variable R must repeat on a path.



Pumping Lemma – Proof details

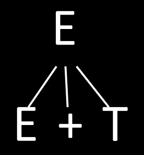
For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$ ...cutting and pasting
2) $vy \neq \varepsilon$ ...start with the smallest parse tree for s
3) $|vxy| \leq p$

|vxy|>0 otherwise higher R would be enough
|vy|>0 otherwise lower R would be enough

Let $b =$ the length of the longest right hand side of a rule (E $\longrightarrow$ E+T)

$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

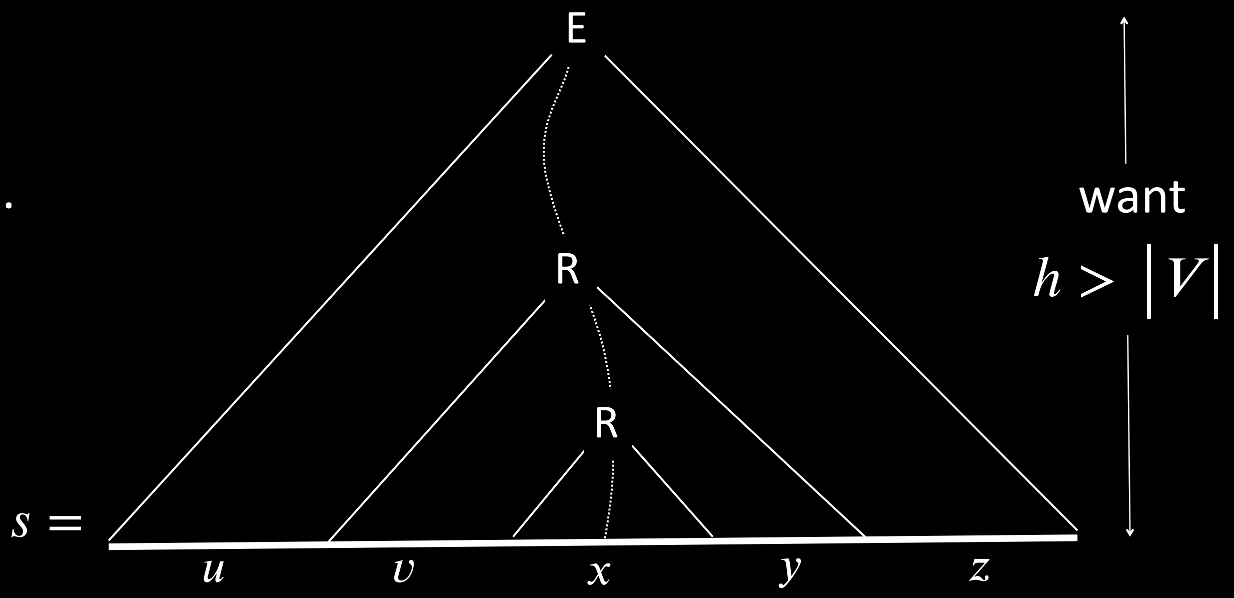
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.

So some variable R must repeat on a path.



use $|s| > b^{|V|}$

set $p = b^{|V|} + 1$

Pumping Lemma – Proof details

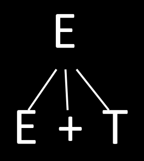
For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

1) $uv^ixy^iz \in A$ for all $i \geq 0$ ...cutting and pasting
2) $vy \neq \varepsilon$ ...start with the smallest parse tree for s
3) $|vxy| \leq p$ ...pick the lowest repetition of a variable

|vxy|>0 otherwise higher R would be enough
|vy|>0 otherwise lower R would be enough

Let $b =$ the length of the longest right hand side of a rule (E $\rightarrow$ E+T)

$=$ the max branching of the parse tree



Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.

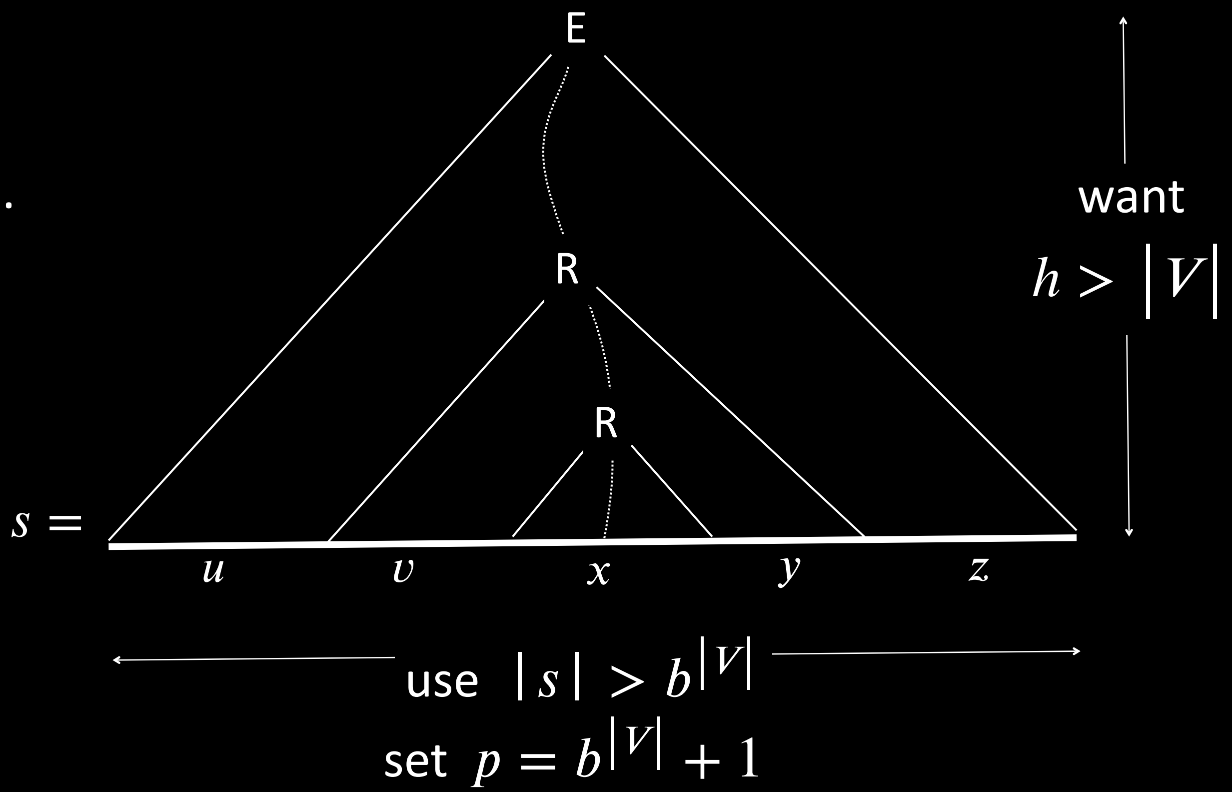
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.

So some variable R must repeat on a path.



Pumping Lemma – Proof details

For $s \in A$ where $|s| \geq p$, we have $s = uvxyz$ where:

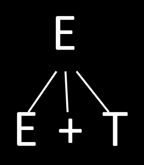
1) $uv^ixy^iz \in A$ for all $i \geq 0$ ...cutting and pasting
2) $vy \neq \varepsilon$ ...start with the smallest parse tree for s
3) $|vxy| \leq p$ ...pick the lowest repetition of a variable

|vxy|>0 otherwise higher R would be enough
|vy|>0 otherwise lower R would be enough

A tree of height at most |V|+1 generates string of length at most $p = b^{|V|+1}$

Let $b =$ the length of the longest right hand side of a rule (E

$=$ the max branching of the parse tree



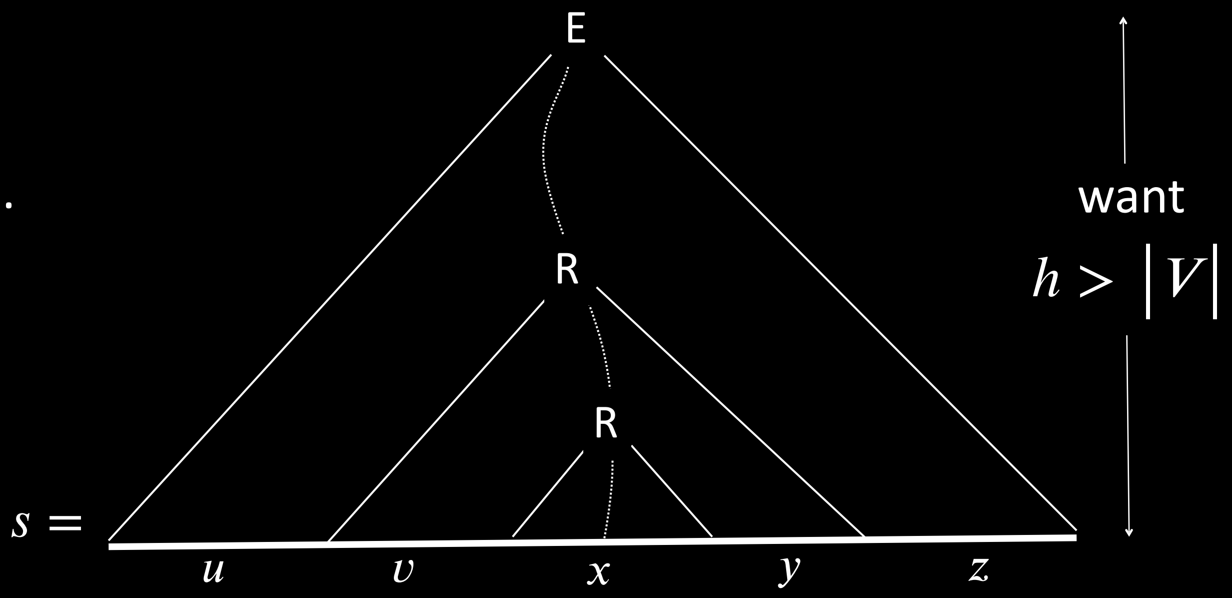
Let $h =$ the height of the parse tree for s.

A tree of height h and max branching b has at most b^h leaves.
So $|s| \leq b^h$.

Let $p = b^{|V|+1}$ where $|V| =$ # variables in the grammar.

So if $|s| \geq p > b^{|V|}$ then $|s| > b^{|V|}$ and so $h > |V|$.

Thus at least $|V| + 1$ variables occur in the longest path.
So some variable R must repeat on a path.



use $|s| > b^{|V|}$
set $p = b^{|V|} + 1$

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$

Show: B is not a CFL

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p 1^p 2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

So uv^2xy^2z has unequal numbers of 0s, 1s, and 2s.

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k 1^k 2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p 1^p 2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

So uv^2xy^2z has unequal numbers of 0s, 1s, and 2s.

Thus $uv^2xy^2z \notin B$, violating Condition 1. Contradiction!

Therefore our assumption (B is a CFL) is false. We conclude that B is not a CFL .

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

$s = 00\cdots0011\cdots1122\cdots22$

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

So uv^2xy^2z has unequal numbers of 0s, 1s, and 2s.

Thus $uv^2xy^2z \notin B$, violating Condition 1. Contradiction!

Therefore our assumption (B is a CFL) is false. We conclude that B is not a CFL .

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B <u>is</u> a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

So uv^2xy^2z has unequal numbers of 0s, 1s, and 2s.

Thus $uv^2xy^2z \notin B$, violating Condition 1. Contradiction!

Therefore our assumption (B is a CFL) is false. We conclude that B is not a CFL .

$s = 00\cdots0011\cdots1122\cdots22$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Proof by Contradiction:

Assume (to get a contradiction) that B is a CFL .

The CFL pumping lemma gives p as above. Let $s = 0^p1^p2^p \in B$.

Pumping lemma says that can divide $s = uvxyz$ satisfying the 3 conditions.

Condition 3 ($|vxy| \leq p$) implies that vxy cannot contain both 0s and 2s.

So uv^2xy^2z has unequal numbers of 0s, 1s, and 2s.

Thus $uv^2xy^2z \notin B$, violating Condition 1. Contradiction!

Therefore our assumption (B is a CFL) is false. We conclude that B is not a CFL .

$s = 00\cdots0011\cdots1122\cdots22$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Check-in 5.1

Example 1 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $B = \{0^k1^k2^k \mid k \geq 0\}$

Show: B is not a CFL

Check-in 5.1

Let $A_1 = \{0^k1^k2^l \mid k, l \geq 0\}$ (equal #s of 0s and 1s)

Let $A_2 = \{0^l1^k2^k \mid k, l \geq 0\}$ (equal #s of 1s and 2s)

Observe that PDAs can recognize A_1 and A_2. What can we now conclude?

a) The class of CFLs is not closed under intersection.

b) The Pumping Lemma shows that $A_1 \cup A_2$ is not a CFL .

c) The class of CFLs is closed under complement.

$s =$ $00\cdots0011\cdots1122\cdots22$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Check-in 5.1

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

$s_1 = 000\cdots001000\cdots001$

Try $s_1 = 0^p10^p1 \in F$.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

$s_1 = 000\cdots001000\cdots001$

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

$s_1 = 000\cdots001000\cdots001$

$u \quad v \quad x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \ge p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \ge 0$
2) $vy \ne \varepsilon$
3) $|vxy| \le p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

$s_1 =$ 000⋯001000⋯001

u | v | x | y | z

$\leftarrow \le p \rightarrow$

Try $s_2 = 0^p1^p0^p1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

$$s_1 = \underbrace{000\cdots00}_{u}\,\underbrace{1}_{v}\,\underbrace{0}_{x}\,\underbrace{00}_{y}\,\underbrace{0\cdots001}_{z}$$

$\leftarrow \leq p \rightarrow$

Try $s_2 = 0^p1^p0^p1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

$$s_2 = 0\cdots01\cdots10\cdots01\cdots1$$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

Try $s_2 = 0^p1^p0^p1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

$s_1 =$ $000\cdots001000\cdots001$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

$s_2 =$ $0\cdots01\cdots10\cdots01\cdots1$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^i xy^i z \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p 1 0^p 1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

Try $s_2 = 0^p 1^p 0^p 1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

Condition 3 implies that vxy does not overlap two runs of 0s or two runs of 1s.

$s_1 =$ 000⋯001000⋯001

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

$s_2 =$ 0⋯01⋯10⋯01⋯1

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

Try $s_2 = 0^p1^p0^p1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

Condition 3 implies that vxy does not overlap two runs of 0s or two runs of 1s.

Therefore, if uv^2xy^2z have at most 4 runs,

$s_1 = $ 000⋯001000⋯001

$u \quad v \quad x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

$s_2 = $ 0⋯01⋯10⋯01⋯1

$u \quad v \quad x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 <u>can</u> be pumped and stay inside F. Bad choice of s.

Try $s_2 = 0^p1^p0^p1^p \in F$.

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

Condition 3 implies that vxy does not overlap two runs of 0s or two runs of 1s.

Therefore, if uv^2xy^2z have at most 4 runs,

then two runs of 0s or two runs of 1s have unequal length.

$s_1 = 000\cdots001000\cdots001$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

$s_2 = 0\cdots01\cdots10\cdots01\cdots1$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Example 2 of Proving Non-CF

Pumping Lemma for CFLs: For every CFL A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = uvxyz$ where

1) $uv^ixy^iz \in A$ for all $i \geq 0$
2) $vy \neq \varepsilon$
3) $|vxy| \leq p$

Let $F = \{ww \mid w \in \Sigma^*\}$. $\Sigma = \{0,1\}$.

Show: F is not a CFL.

Assume (for contradiction) that F is a CFL.

The CFL pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s_1 = 0^p10^p1 \in F$. But s_1 can be pumped and stay inside F. Bad choice of s.

$s_1 = 000\cdots001000\cdots001$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Try $s_2 = 0^p1^p0^p1^p \in F$.

$s_2 = 0\cdots01\cdots10\cdots01\cdots1$

$u \mid v \mid x \mid y \mid z$

$\leftarrow \leq p \rightarrow$

Show s_2 cannot be pumped $s_2 = uvxyz$ satisfying the 3 conditions.

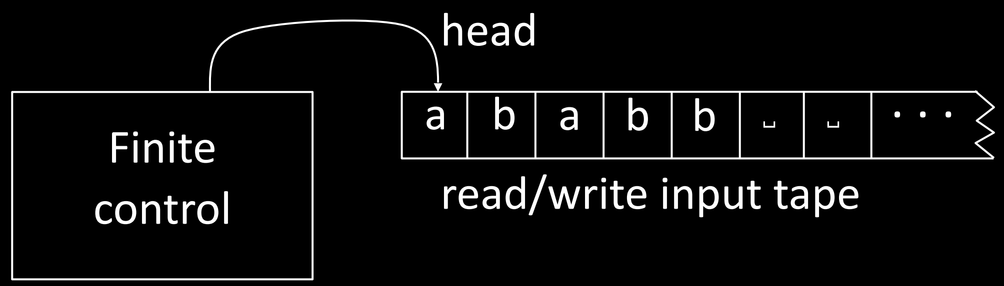
Condition 3 implies that vxy does not overlap two runs of 0s or two runs of 1s.

Therefore, if uv^2xy^2z have at most 4 runs,
then two runs of 0s or two runs of 1s have unequal length.

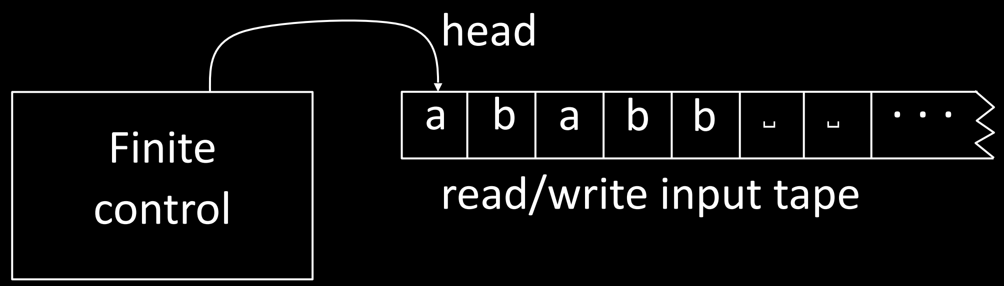
So $uv^2xy^2z \notin F$ violating Condition 1. Contradiction! Thus F is not a CFL.

Turing Machine

Turing Machines (TMs)



Turing Machines (TMs)



1) Head can read and write
2) Head is two way (can move left or right)
3) Tape is infinite (to the right)
4) Infinitely many blanks “␣“ follow input
5) Can accept or reject any time (not only at end of input)

TM – example

TM – example

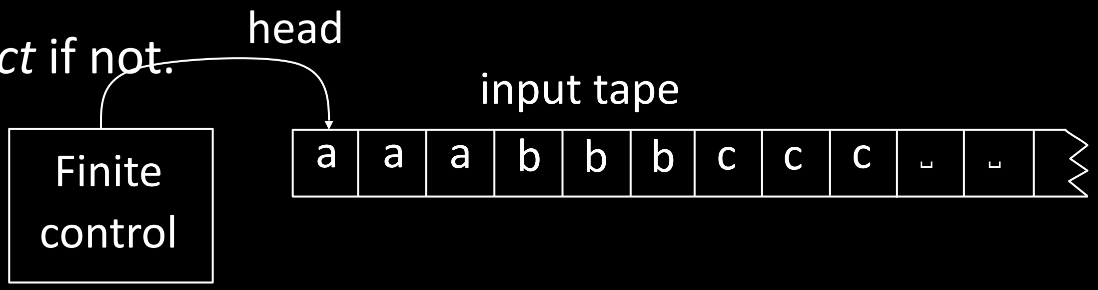
TM recognizing $B = \left\{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \,\middle|\, k \geq 0 \right\}$

1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).

TM – example

TM recognizing $B = \left\{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \,\middle|\, k \geq 0 \right\}$

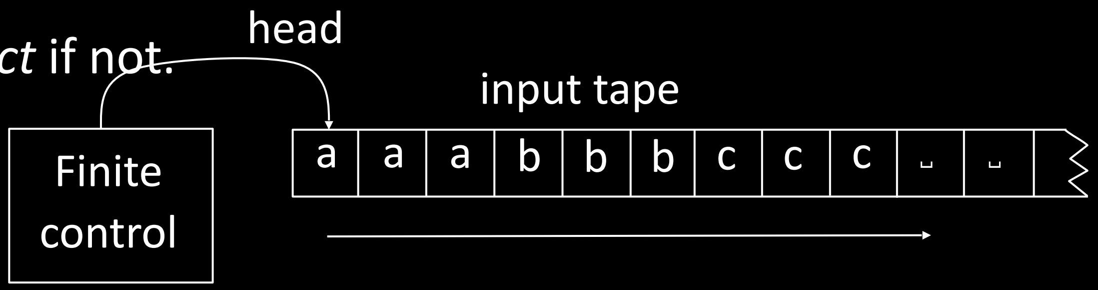
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$

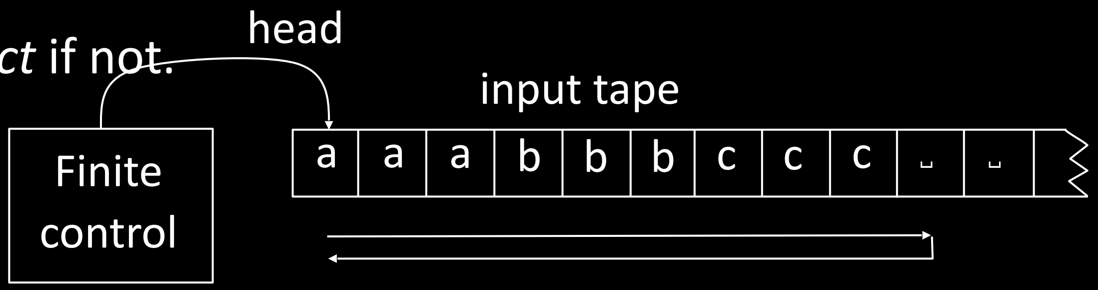
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$

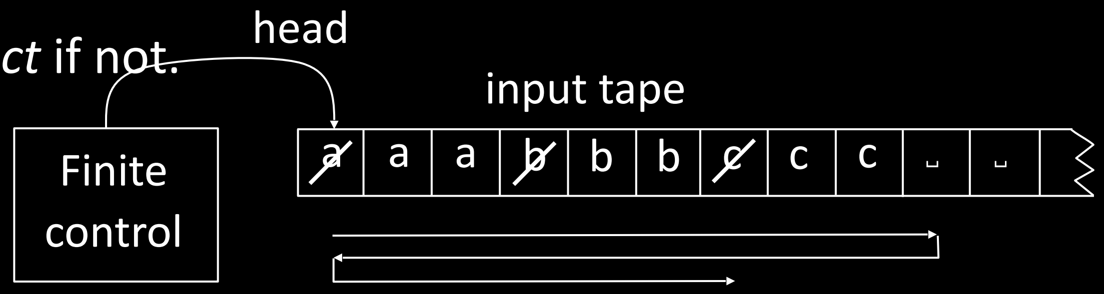
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

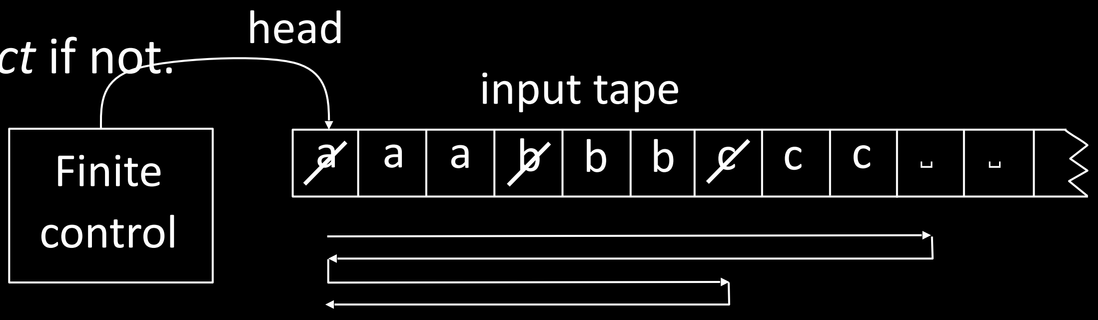
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

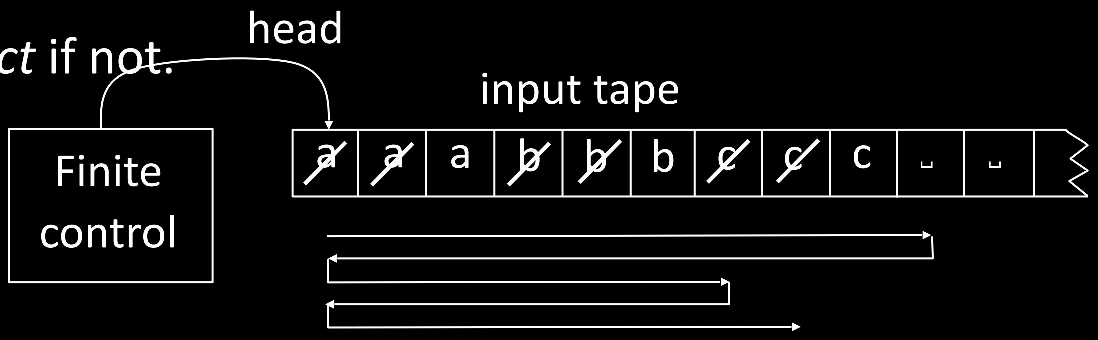
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \left\{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \,\middle|\, k \geq 0 \right\}$

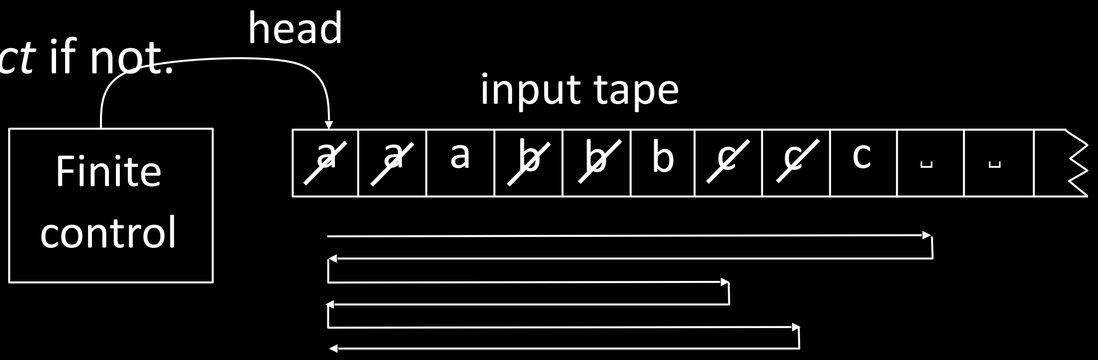
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

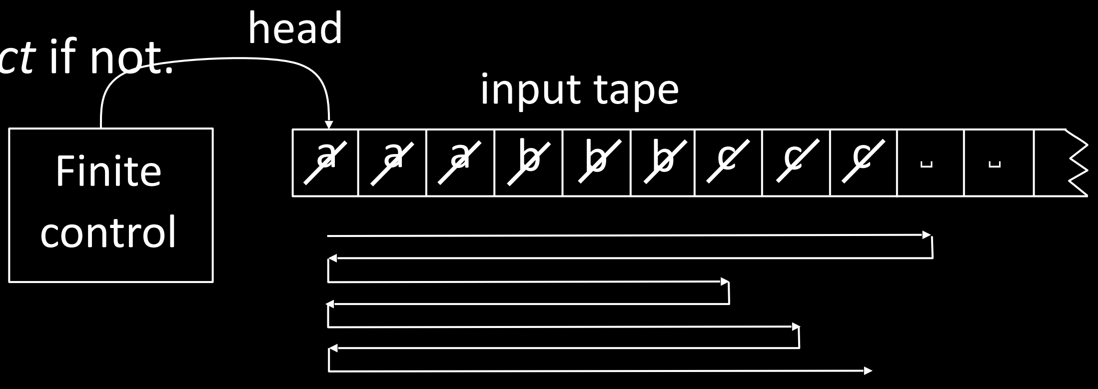
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

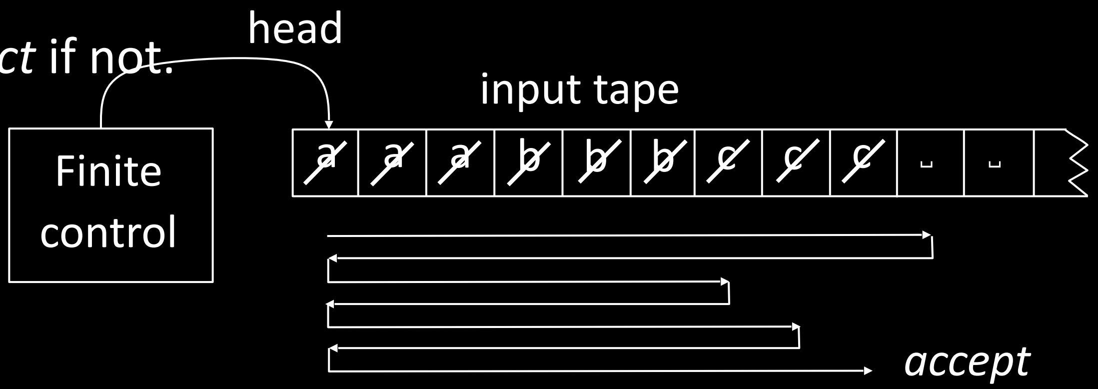
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

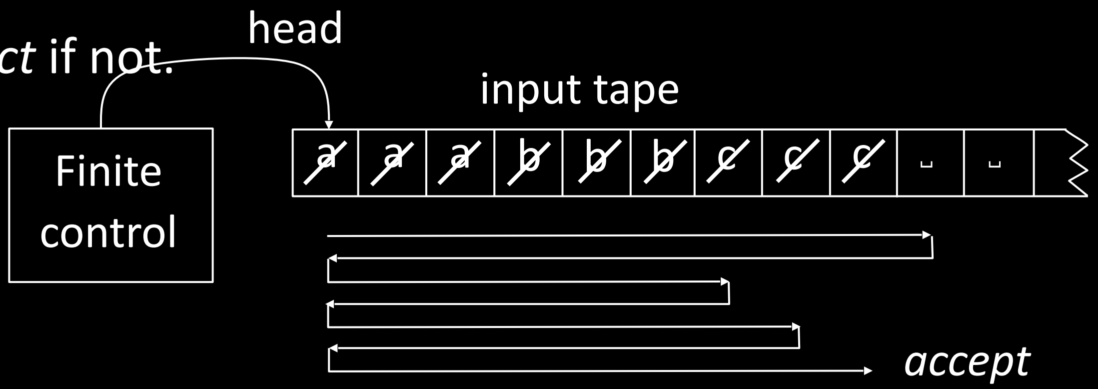
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

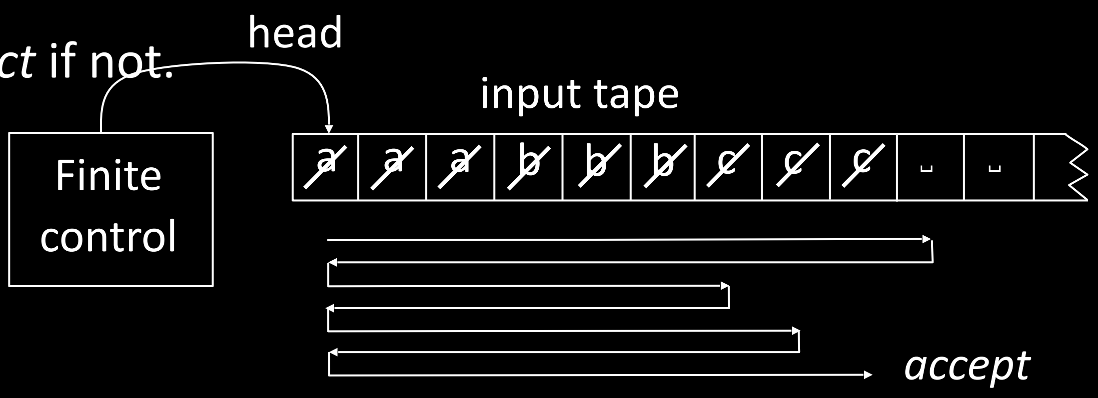
1) Scan right until ␣ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.
6) If all symbols remain, return to left end and repeat from (3).



TM – example

TM recognizing $B = \{ \mathrm{a}^k \mathrm{b}^k \mathrm{c}^k \mid k \geq 0 \}$

1) Scan right until $\sqcup$ while checking if input is in $\mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2) Return head to left end.
3) Scan right, crossing off single a, b, and c.
4) If the last one of each symbol, *accept*.
5) If the last one of some symbol but not others, *reject*.



Check-in 5.2

How do we get the effect of “crossing off” with a Turing machine?

a) We add that feature to the model.

b) We use a tape alphabet $\Gamma = \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \cancel{\mathrm{a}}, \cancel{\mathrm{b}}, \cancel{\mathrm{c}}, \sqcup\}$.

c) All Turing machines come with an eraser.

TM – Formal Definition

Defn: A Turing Machine (TM) is a 7-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta, q_0,\ q\text{acc}\ ,\ q\text{rej})$

Σ input alphabet

Γ tape alphabet ($\Sigma \subseteq \Gamma$)

δ: $\mathrm{Q} \times \Gamma \rightarrow Q \times \Gamma \times$ {L, R} (L = Left, R = Right)

$\delta(q, \mathrm{a}) = (r, \mathrm{b}, \ \mathrm{R})$

TM – Formal Definition

Defn: A Turing Machine (TM) is a 7-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta, q_0,\ q\text{acc}\ ,\ q\text{rej})$

Σ input alphabet

Γ tape alphabet ($\Sigma \subseteq \Gamma$)

δ: $\mathrm{Q} \times \Gamma \rightarrow Q \times \Gamma \times$ {L, R} (L = Left, R = Right)

$\delta(q, \mathrm{a}) = (r, \mathrm{b},\ \mathrm{R})$

On input w a TM M may halt (enter $q$acc or $q$rej)
or M may run forever ("loop").

So M has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

TM – Formal Definition

Defn: A <u>Turing Machine</u> (TM) is a 7-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta, q_0,\ q\text{acc},\ q\text{rej})$

Σ input alphabet

Γ tape alphabet ($\Sigma \subseteq \Gamma$)

δ: $\mathrm{Q} \times \Gamma \to Q \times \Gamma \times$ {L, R} (L = Left, R = Right)

$\delta(q, \mathrm{a}) = (r, \mathrm{b}, \mathrm{R})$

On input w a TM M may halt (enter $q$acc or $q$rej)
or M may run forever (“loop”).

So M has 3 possible outcomes for each input w:

1. *<u>Accept</u>* w (enter $q$acc)
2. *<u>Reject</u>* w by halting (enter $q$rej)
3. *<u>Reject</u>* w by looping (running forever)

Check-in 5.3

TM – Formal Definition

Defn: A Turing Machine (TM) is a 7-tuple
$(Q, \Sigma, \Gamma, \delta, q_0, q\text{acc}, q\text{rej})$

Σ input alphabet

Γ tape alphabet $(\Sigma \subseteq \Gamma)$

δ: $\mathrm{Q} \times \Gamma \rightarrow Q \times \Gamma \times$ {L, R} (L = Left, R = Right)

$\delta(q, \mathrm{a}) = (r, \mathrm{b}, \mathrm{R})$

On input w a TM M may halt (enter $q$acc or $q$rej)
or M may run forever ("loop").

So M has 3 possible outcomes for each input w:

1. *Accept* w (enter $q$acc)
2. *Reject* w by halting (enter $q$rej)
3. *Reject* w by looping (running forever)

Check-in 5.3

This Turing machine model is deterministic.
How would we change it to be nondeterministic?

a) Add a second transition function.

b) Change δ to be δ: $\mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\})$

c) Change the tape alphabet Γ to be infinite.

TM Recognizers and Deciders

TM Recognizers and Deciders

Let M be a TM. Then $L(M) = \{w \mid M \text{ accepts } w\}$.

Say that M recognizes A if $A = L(M)$.

Defn: A is Turing-recognizable if $A = L(M)$ for some TM M.

TM Recognizers and Deciders

Let M be a TM. Then $L(M) = \{w|\ M \text{ accepts } w\}$.

Say that M recognizes A if $A = L(M)$.

Defn: A is Turing-recognizable if $A = L(M)$ for some TM M.

Defn: TM M is a decider if M halts on all inputs.

TM Recognizers and Deciders

Let M be a TM. Then $L(M) = \{w \mid M$ accepts $w\}$.

Say that M recognizes A if $A = L(M)$.

Defn: A is Turing-recognizable if $A = L(M)$ for some TM M.

Defn: TM M is a decider if M halts on all inputs.

Say that M decides A if $A = L(M)$ and M is a decider.

Defn: A is Turing-decidable if $A = L(M)$ for some TM decider M.

TM Recognizers and Deciders

Let M be a TM. Then $L(M) = \{w \mid M \text{ accepts } w\}$.

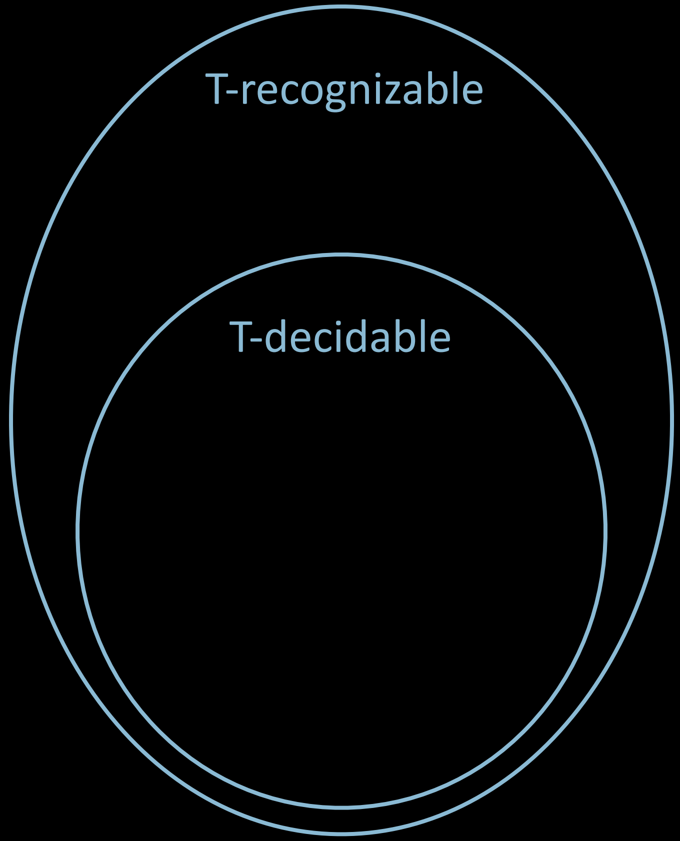
Say that M recognizes A if $A = L(M)$.

Defn: A is Turing-recognizable if $A = L(M)$ for some TM M.

Defn: TM M is a decider if M halts on all inputs.

Say that M decides A if $A = L(M)$ and M is a decider.

Defn: A is Turing-decidable if $A = L(M)$ for some TM decider M.



TM Recognizers and Deciders

Let M be a TM. Then $L(M) = \{w \mid M \text{ accepts } w\}$.

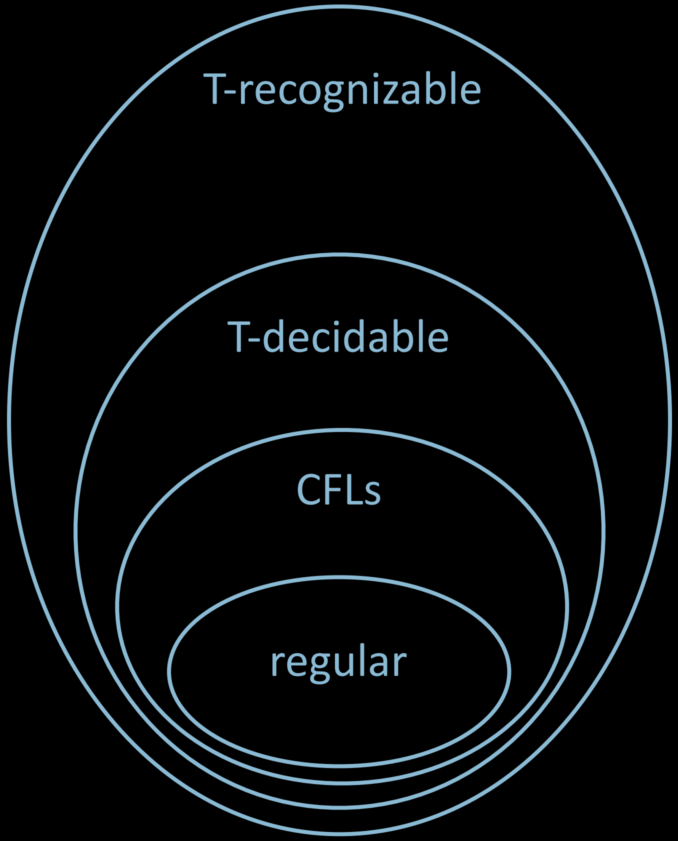
Say that M recognizes A if $A = L(M)$.

Defn: A is Turing-recognizable if $A = L(M)$ for some TM M.

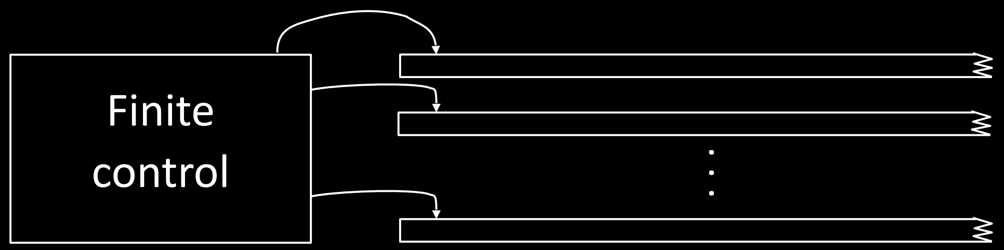
Defn: TM M is a decider if M halts on all inputs.

Say that M decides A if $A = L(M)$ and M is a decider.

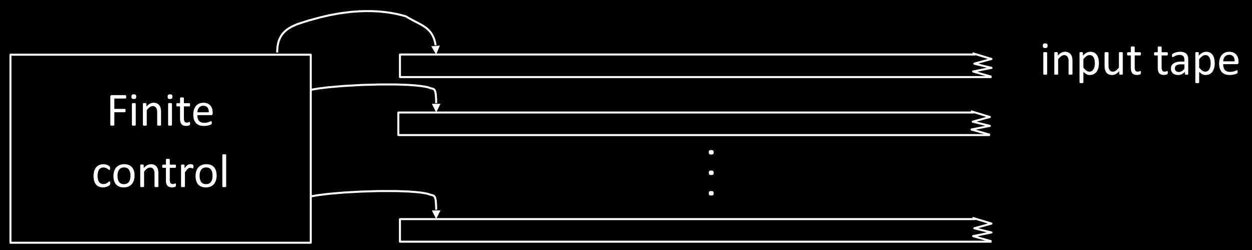
Defn: A is Turing-decidable if $A = L(M)$ for some TM decider M.



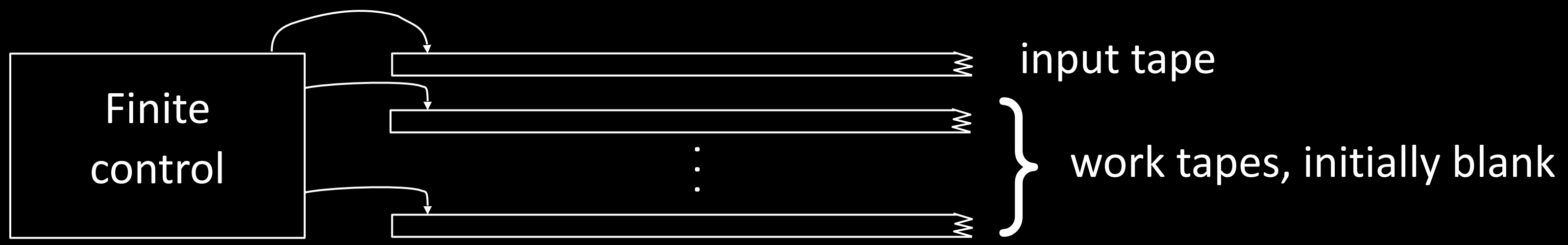
Multi-tape Turing machines



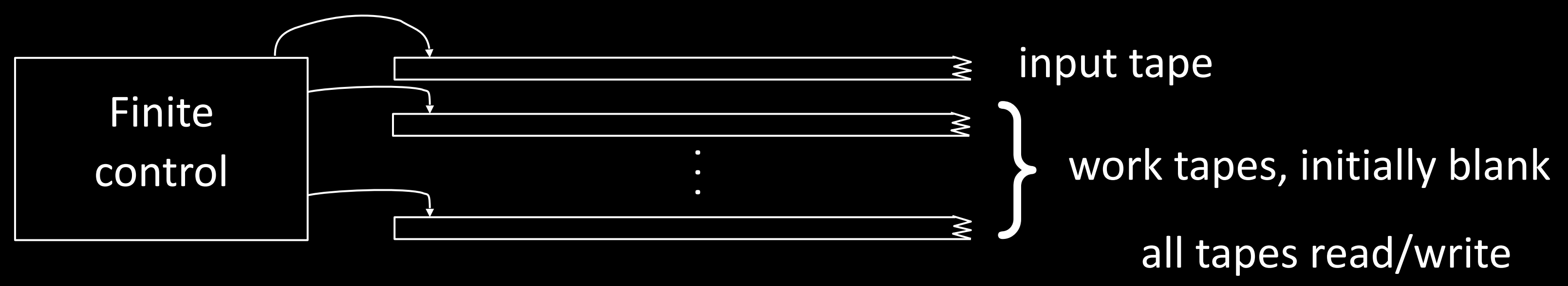
Multi-tape Turing machines



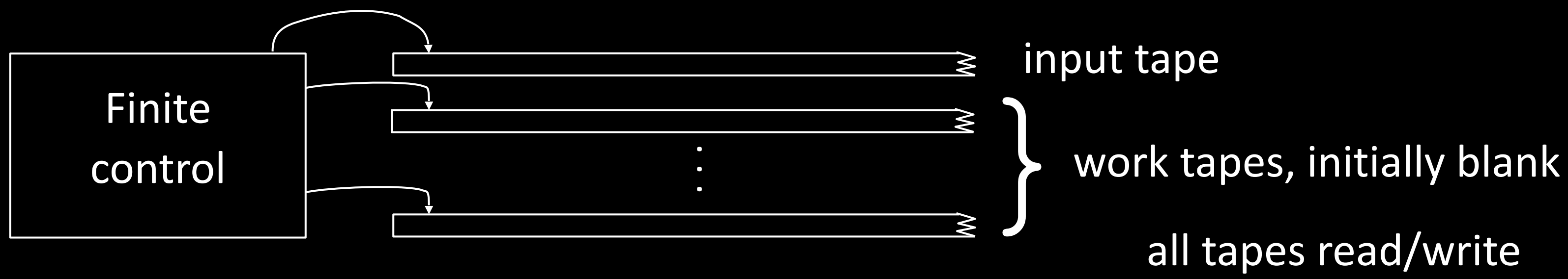
Multi-tape Turing machines



Multi-tape Turing machines

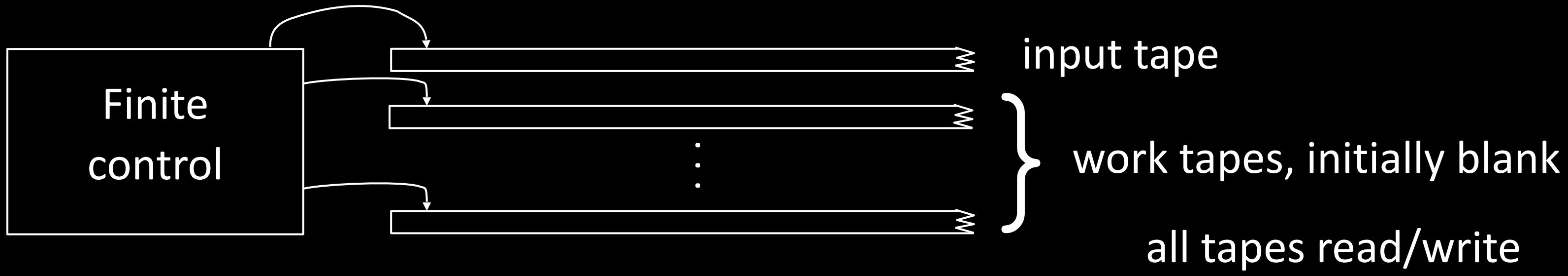


Multi-tape Turing machines



Theorem: A is T-recognizable iff some multi-tape TM recognizes A

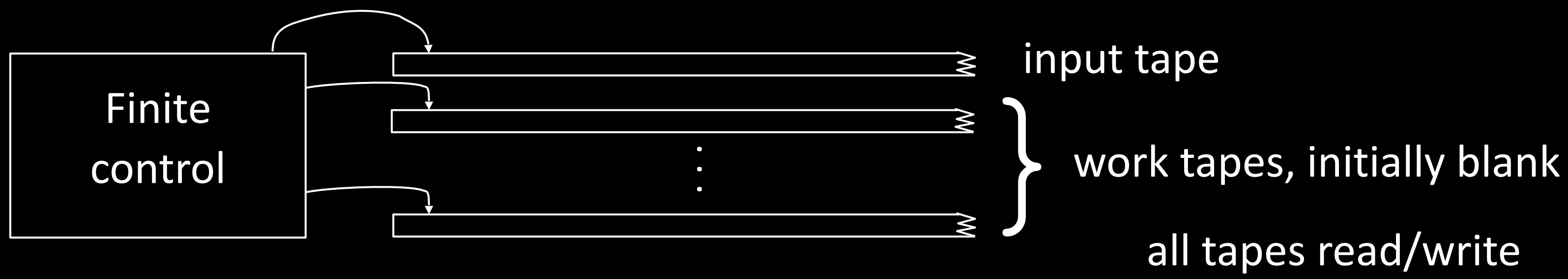
Multi-tape Turing machines



Theorem: A is T-recognizable iff some multi-tape TM recognizes A

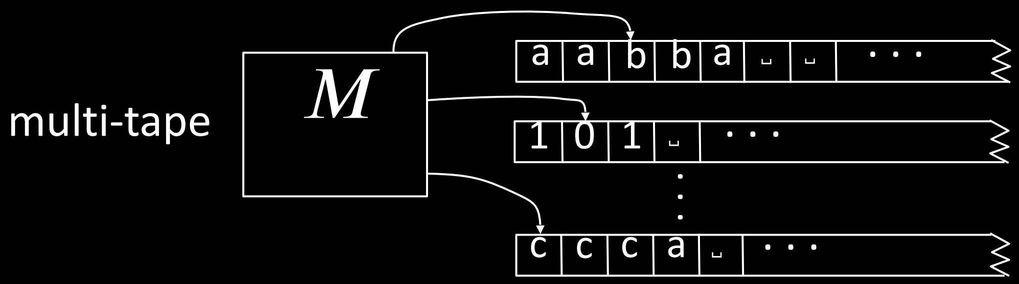
Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

Multi-tape Turing machines

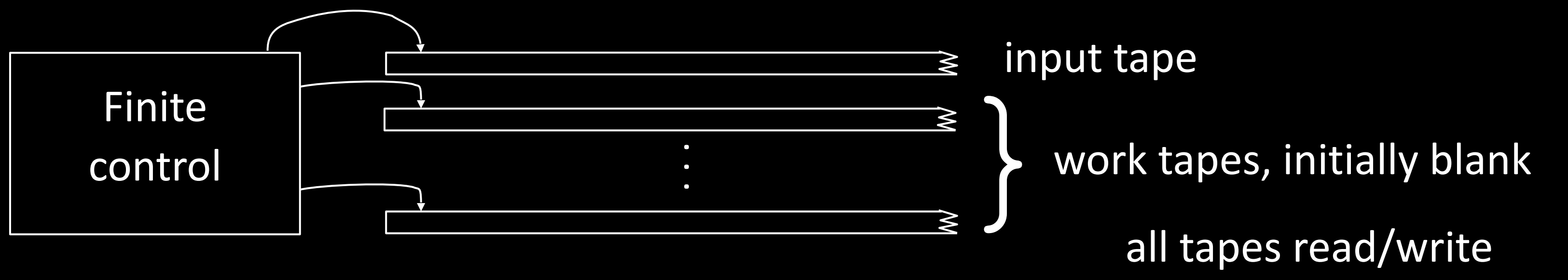


Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

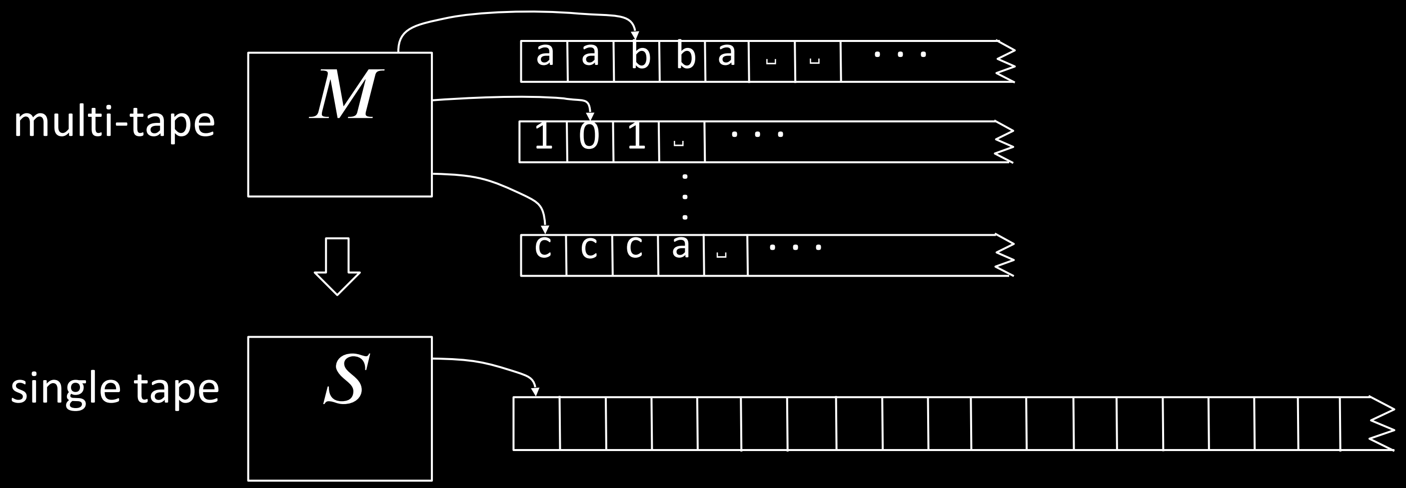


Multi-tape Turing machines

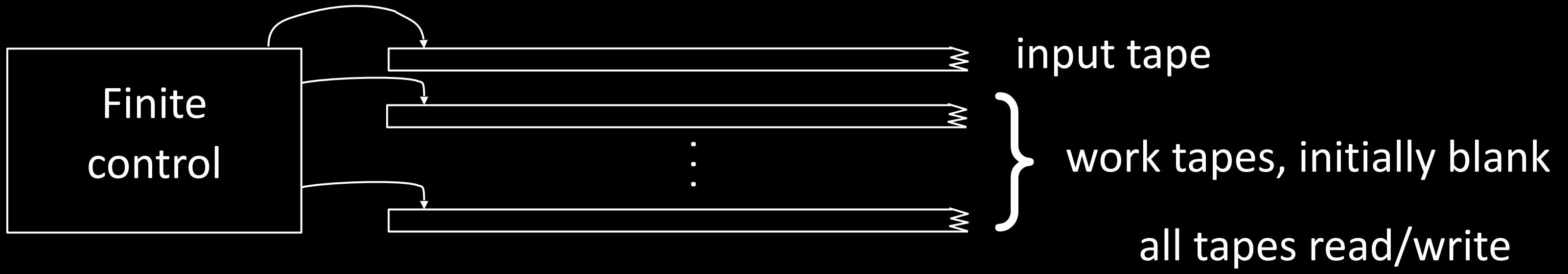


Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:



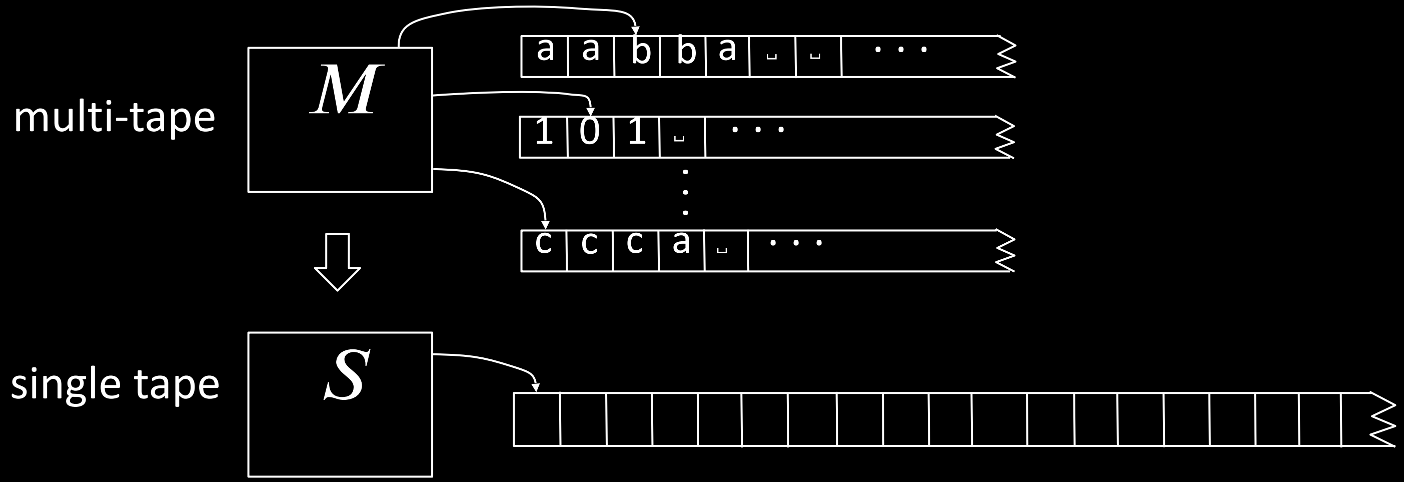
Multi-tape Turing machines



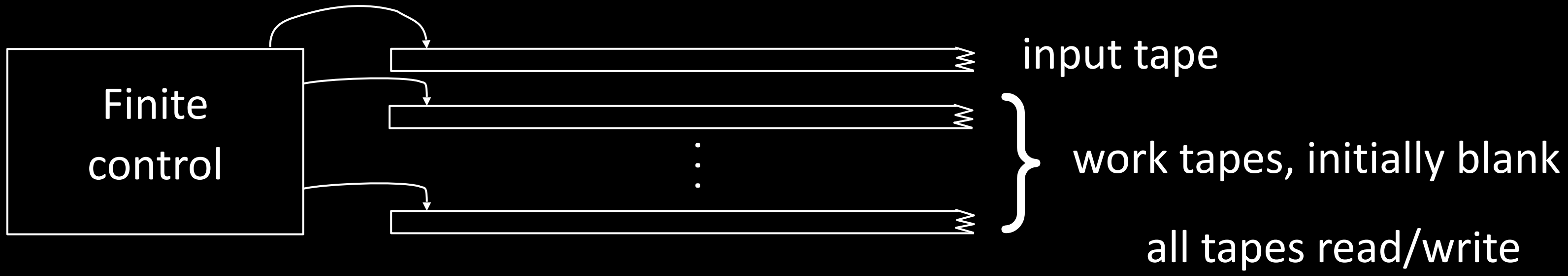
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in “blocks”.

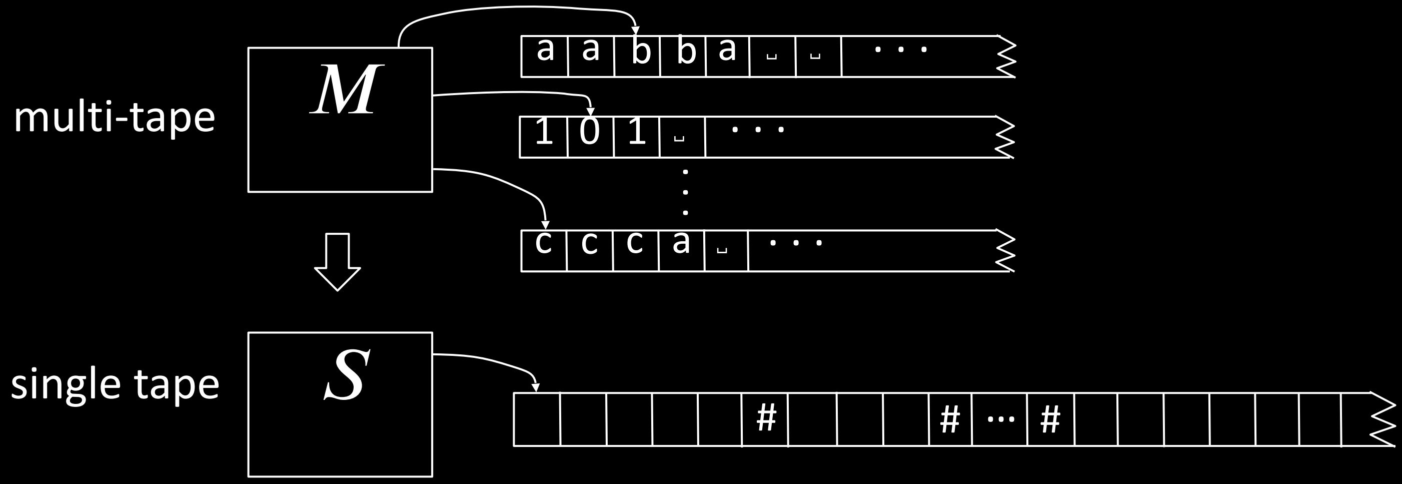


Multi-tape Turing machines



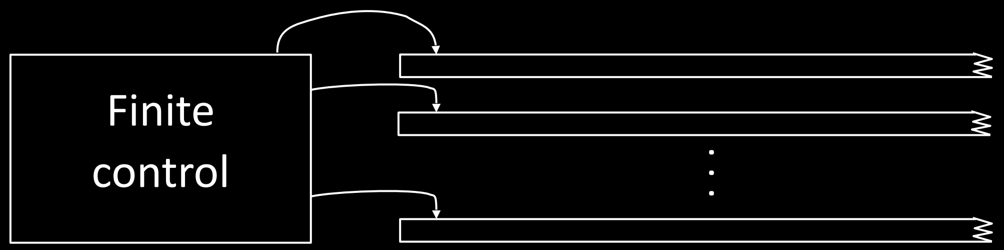
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:



S simulates M by storing the contents of multiple tapes on a single tape in "blocks".

Multi-tape Turing machines



input tape

work tapes, initially blank

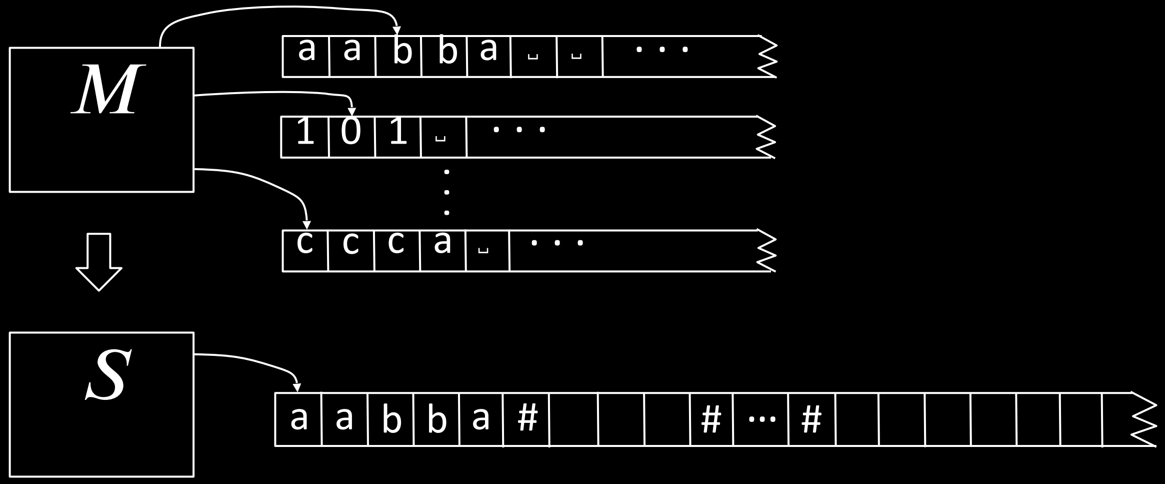
all tapes read/write

Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

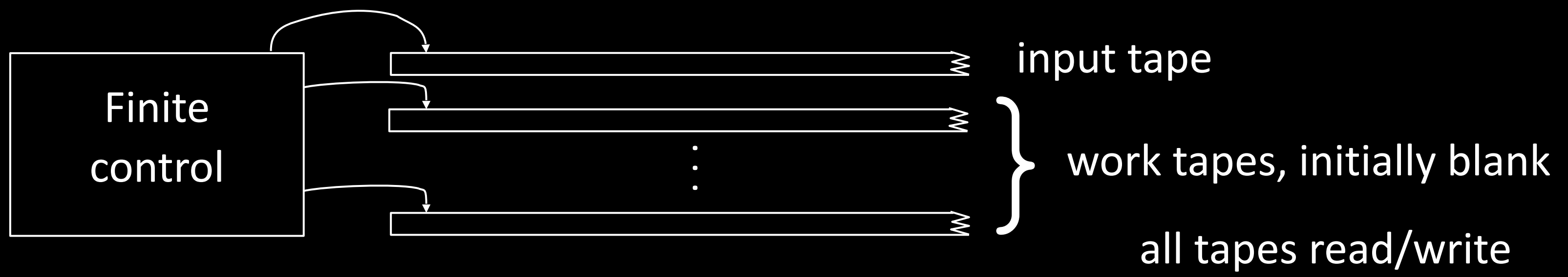
multi-tape

single tape



S simulates M by storing the contents of multiple tapes on a single tape in “blocks”.

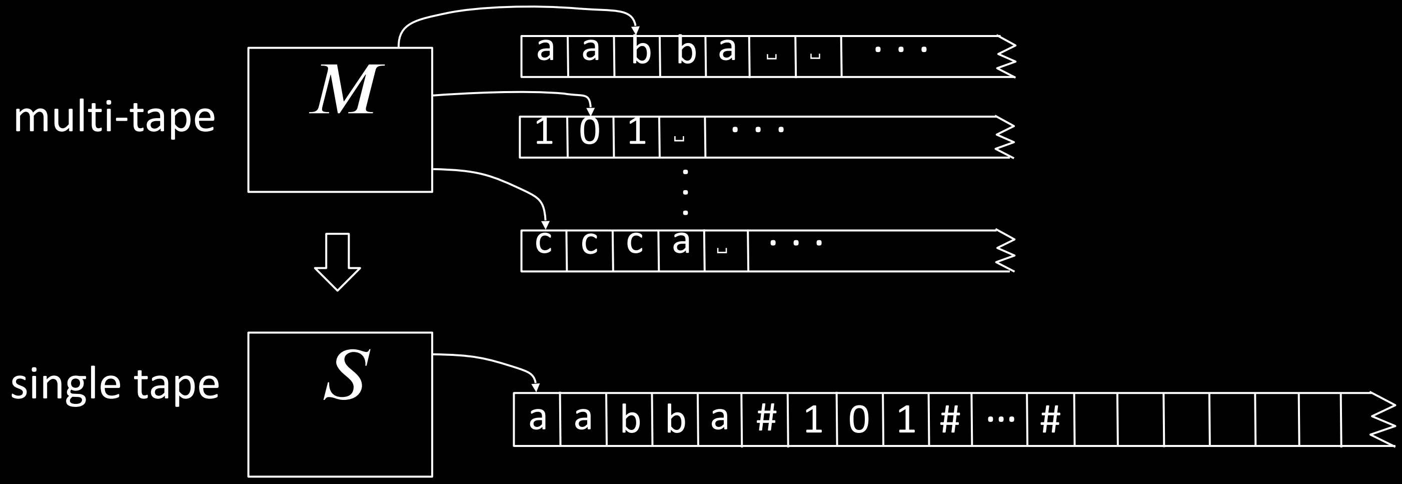
Multi-tape Turing machines



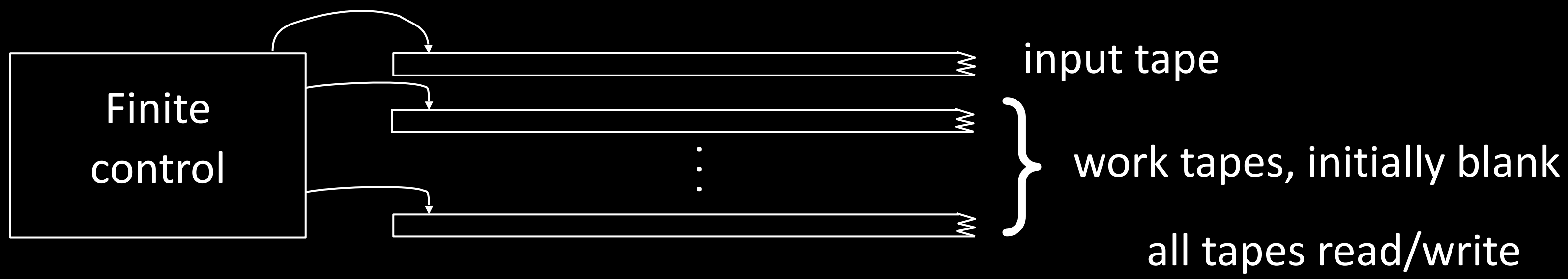
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in “blocks”.

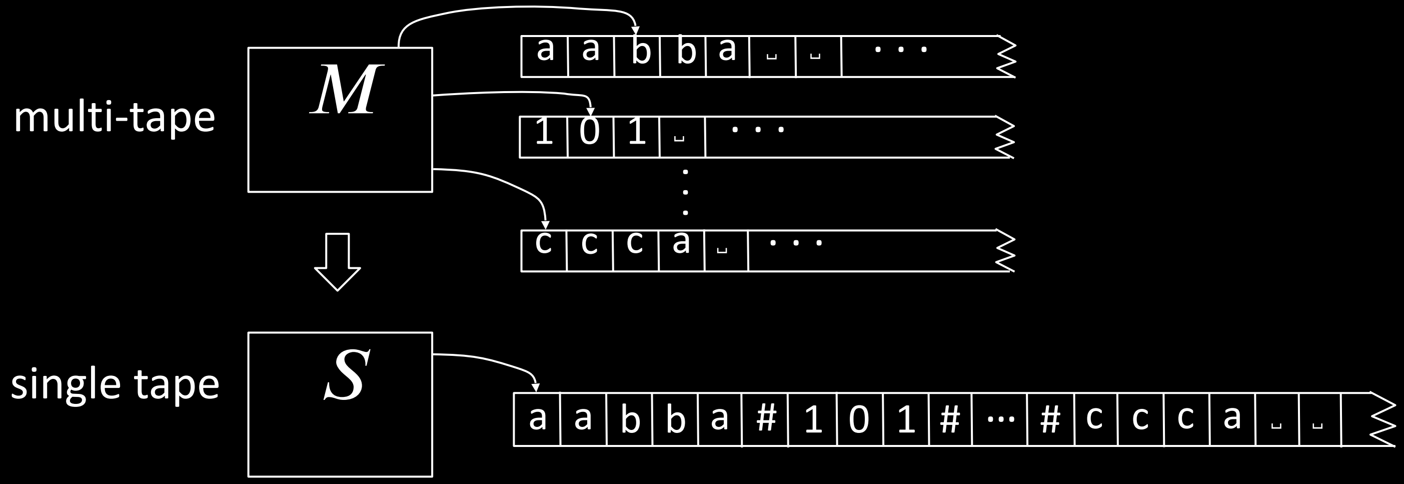


Multi-tape Turing machines



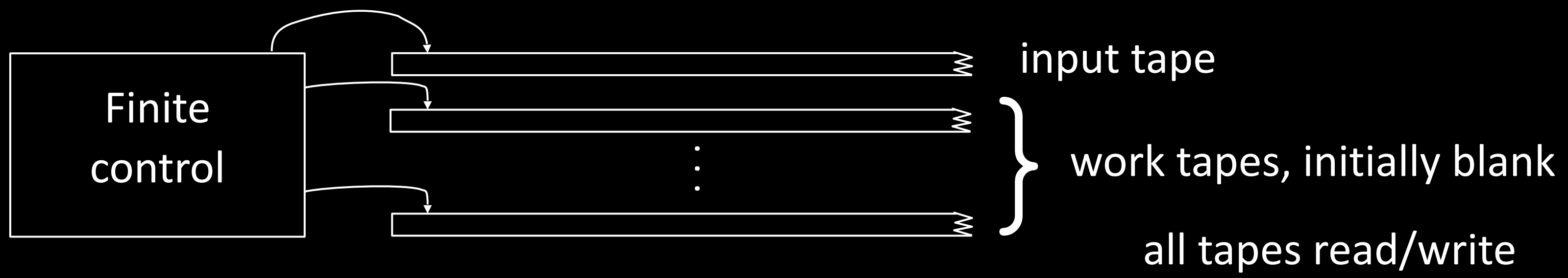
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert multi-tape to single tape:



S simulates M by storing the contents of multiple tapes on a single tape in “blocks”.

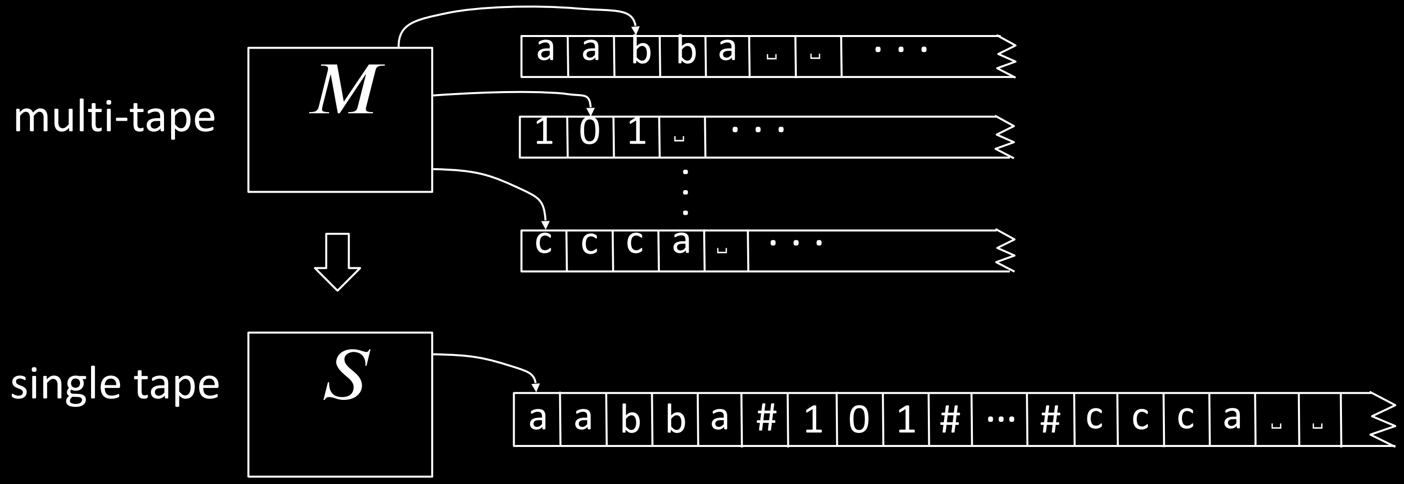
Multi-tape Turing machines



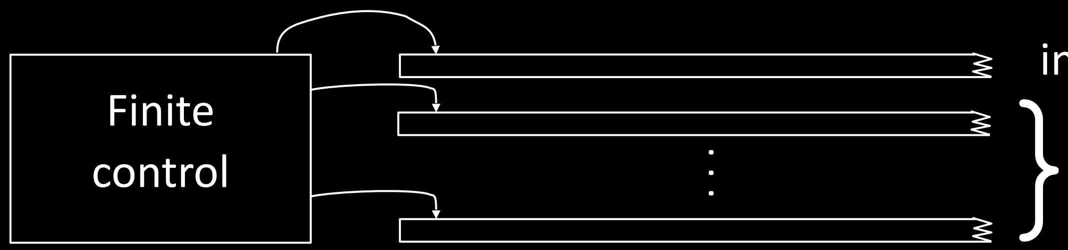
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in “blocks”. Record head positions with dotted symbols.



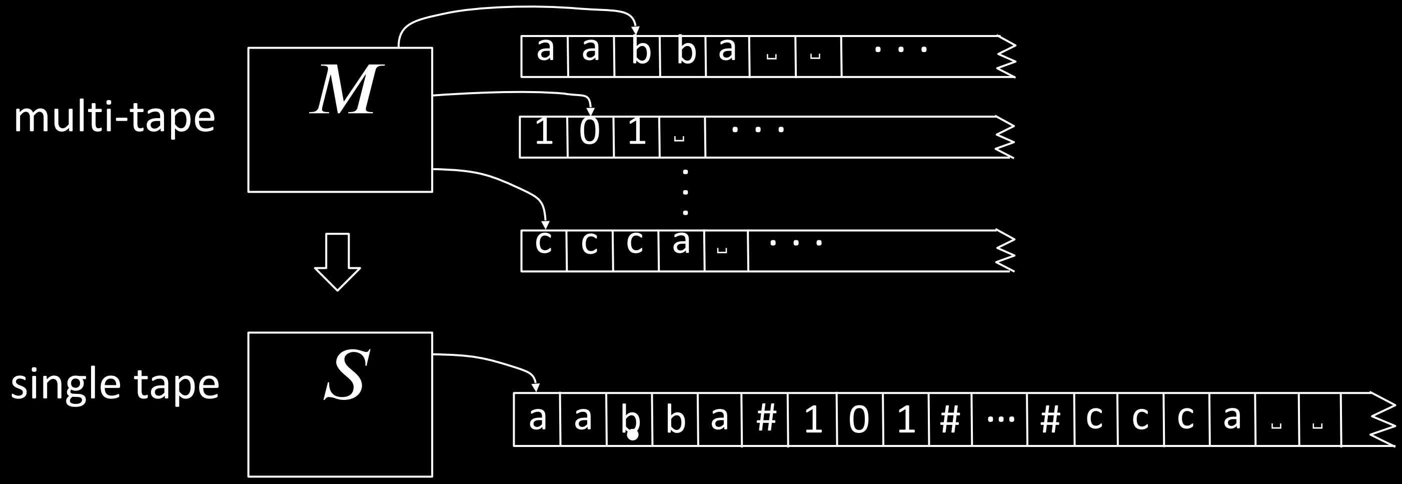
Multi-tape Turing machines



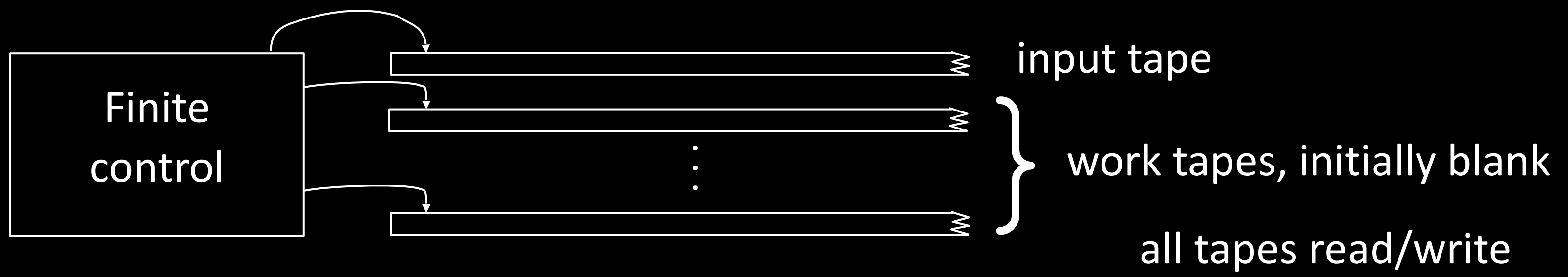
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.



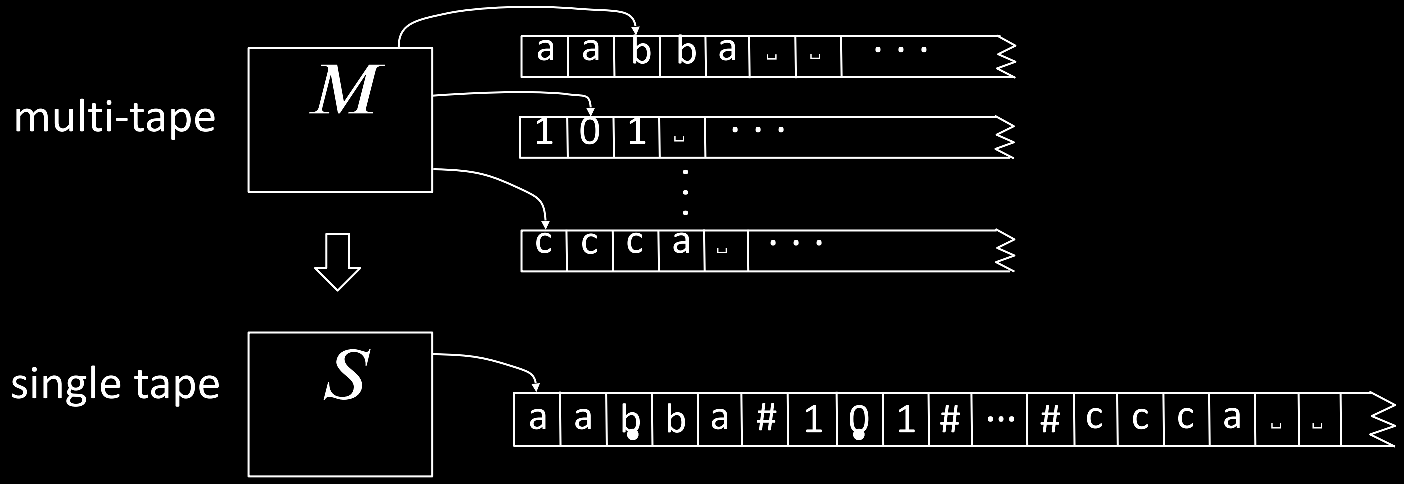
Multi-tape Turing machines



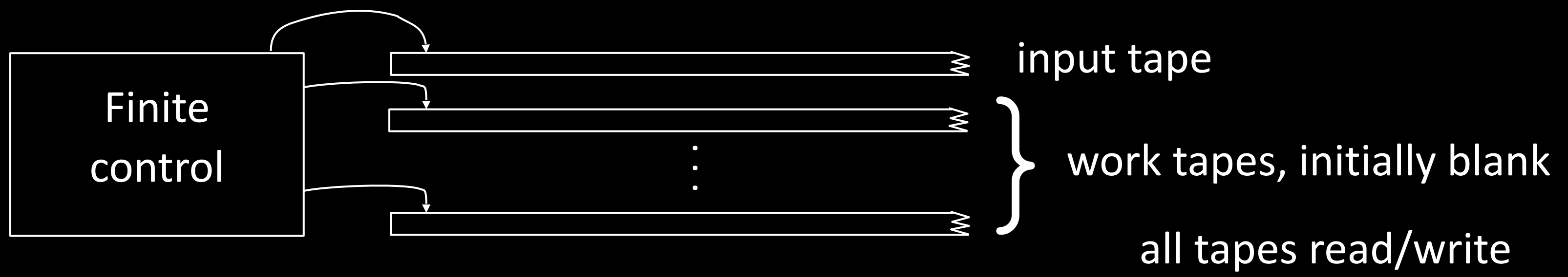
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.



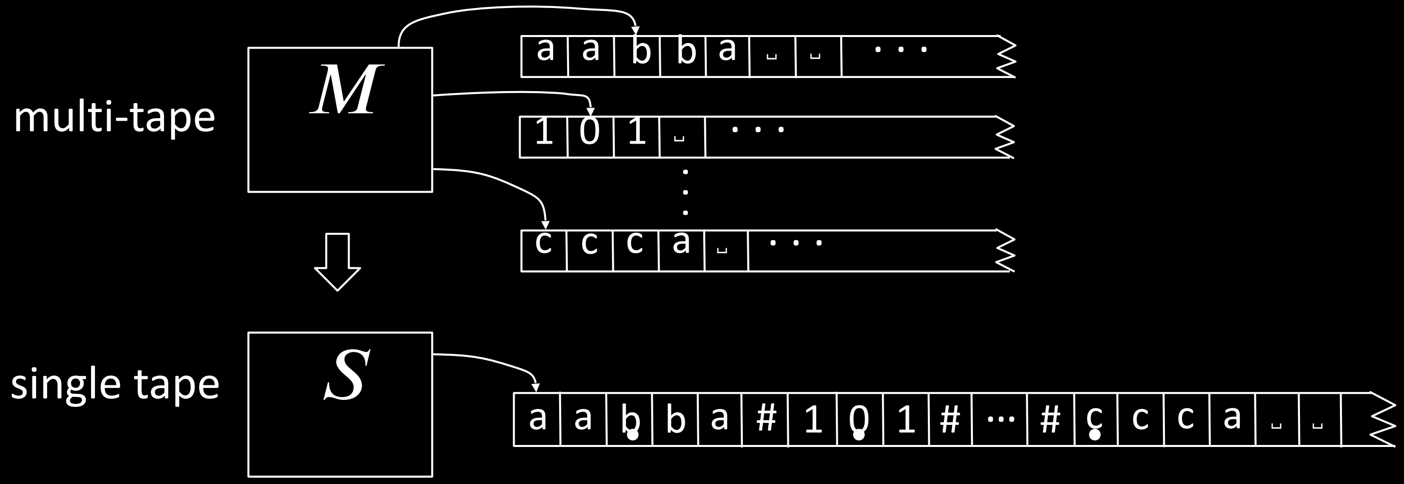
Multi-tape Turing machines



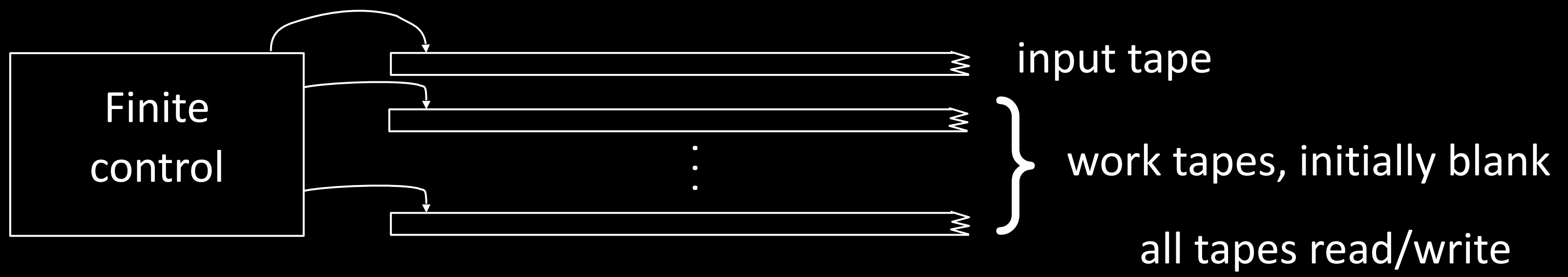
Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.

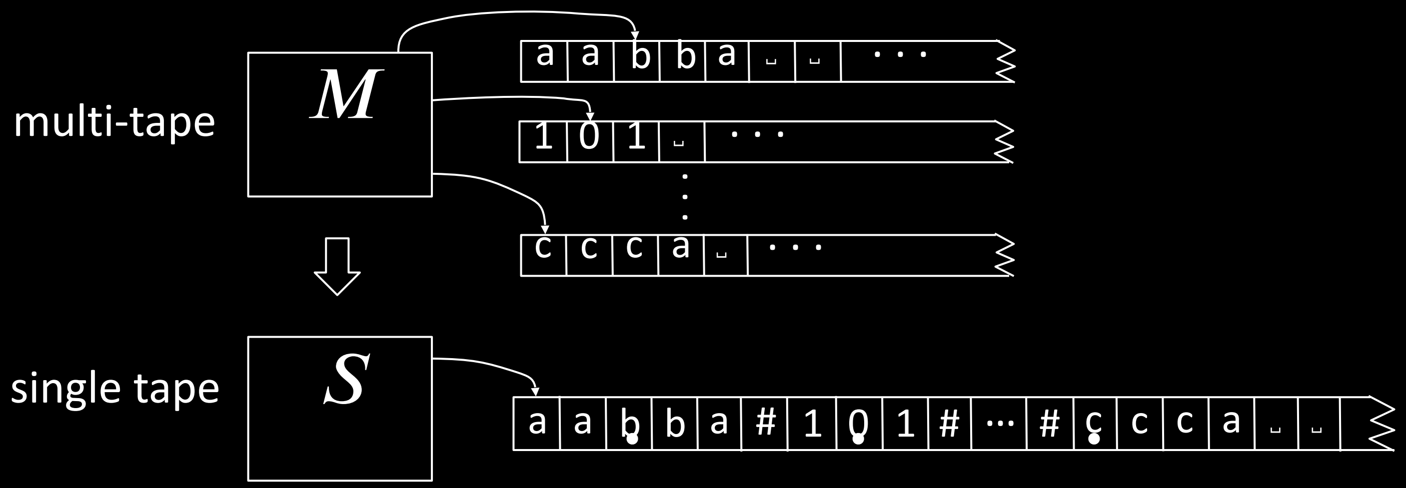


Multi-tape Turing machines



Theorem: A is T-recognizable iff some multi-tape TM recognizes A

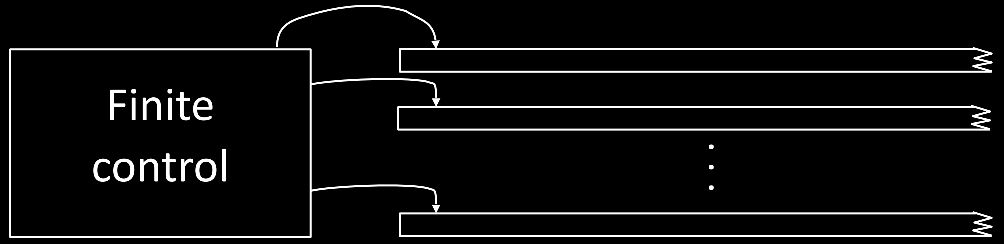
Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert multi-tape to single tape:



S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.

Some details of S:

Multi-tape Turing machines



input tape

work tapes, initially blank

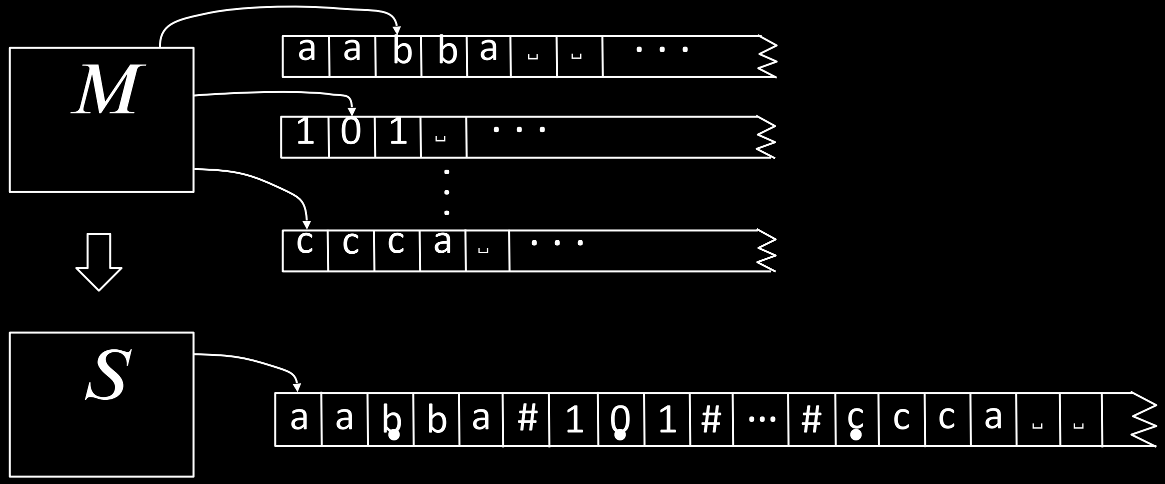
all tapes read/write

Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert multi-tape to single tape:

multi-tape

single tape

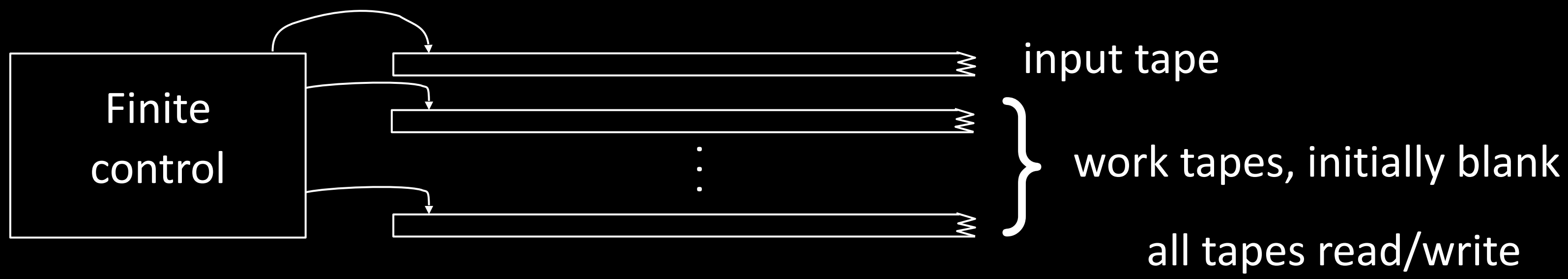


S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.

Some details of S:

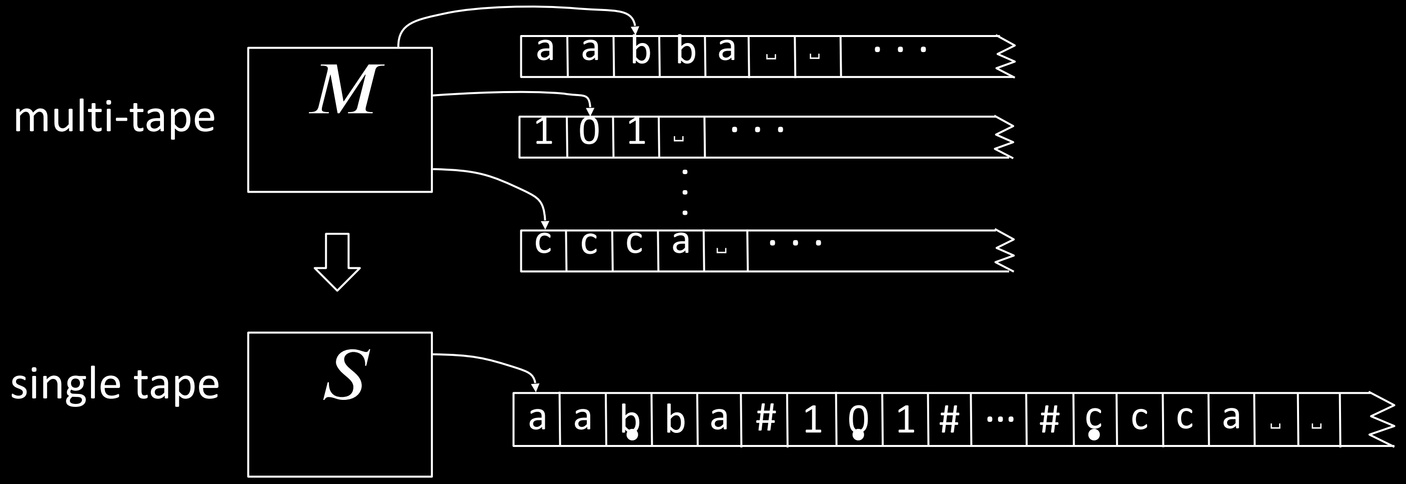
1) To simulate each of M's steps
   a. Scan entire tape to find dotted symbols.
   b. Scan again to update according to M's δ.

Multi-tape Turing machines



Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:

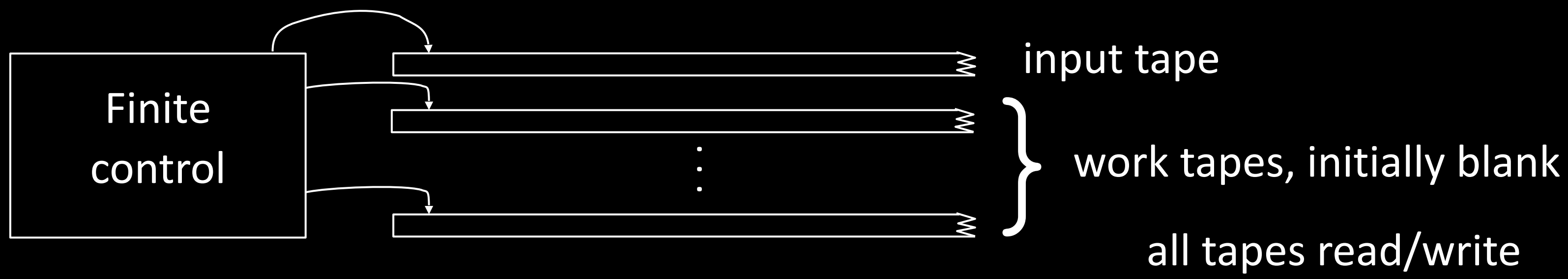


S simulates M by storing the contents of multiple tapes on a single tape in "blocks". Record head positions with dotted symbols.

Some details of S:

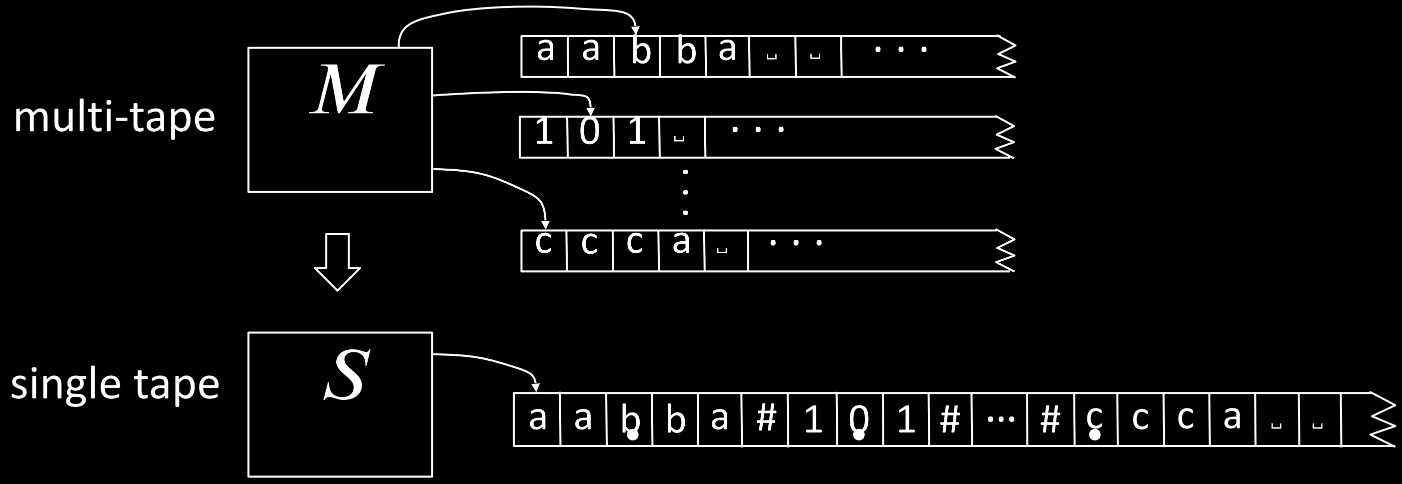
1) To simulate each of M's steps
   a. Scan entire tape to find dotted symbols.
   b. Scan again to update according to M's δ.
   c. Shift to add room as needed.

Multi-tape Turing machines



Theorem: A is T-recognizable iff some multi-tape TM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert multi-tape to single tape:



S simulates M by storing the contents of multiple tapes on a single tape in “blocks”. Record head positions with dotted symbols.

Some details of S:

1) To simulate each of M’s steps
   a. Scan entire tape to find dotted symbols.
   b. Scan again to update according to M’s δ.
   c. Shift to add room as needed.
2) Accept/reject if M does.

Nondeterministic Turing machines

NTM accepts if there is a series of proper actions leading to qacc

Nondeterministic Turing machines

A Nondeterministic TM (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

NTM accepts if there is a series of proper actions leading to qacc

Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

Theorem: A is T-recognizable iff some NTM recognizes A

NTM accepts if there is a series of proper actions leading to qacc

Nondeterministic Turing machines

A Nondeterministic TM (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.

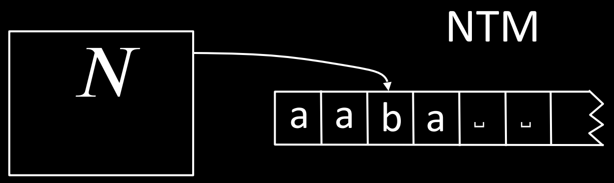
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(\, Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}\,)$.

NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.



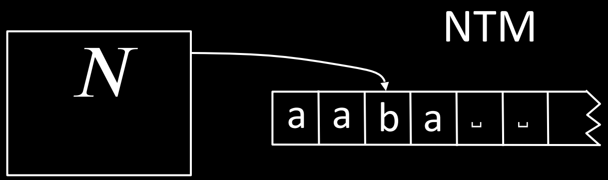
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

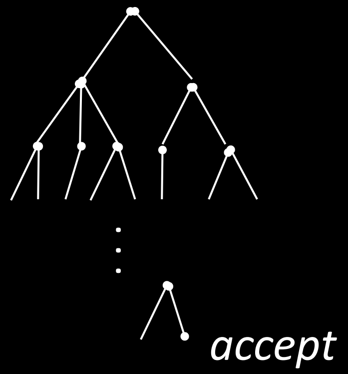
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



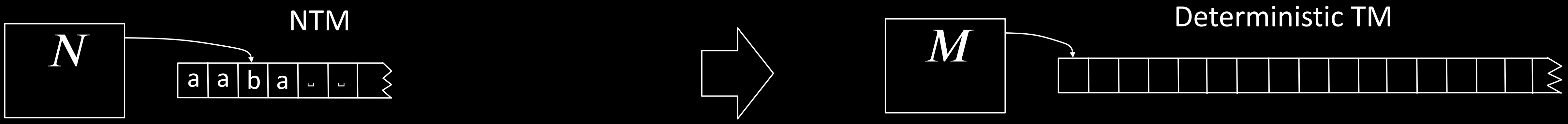
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

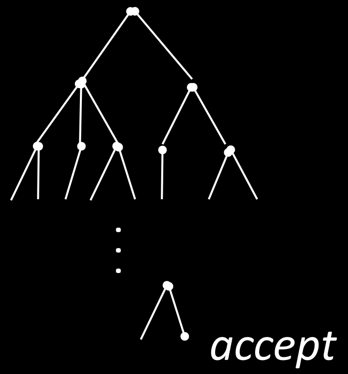
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



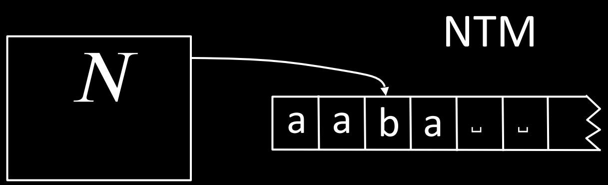
Nondeterministic Turing machines

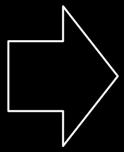
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

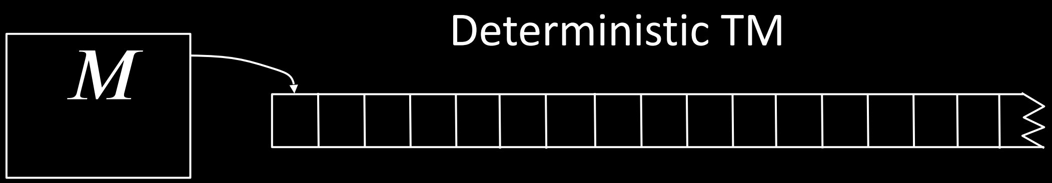
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

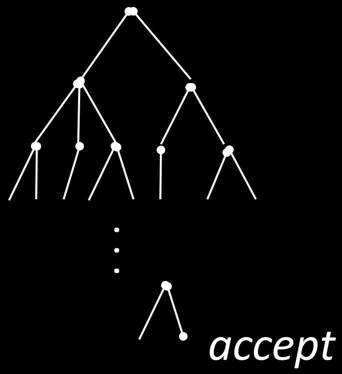
Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.







Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.

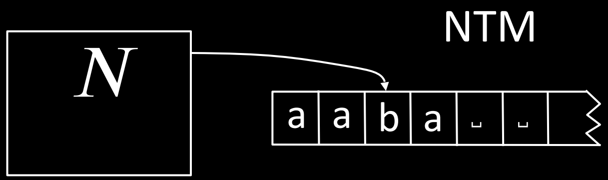
Nondeterministic Turing machines

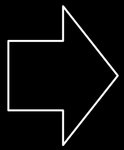
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\,Q \times \Gamma \times$ {L, R}).

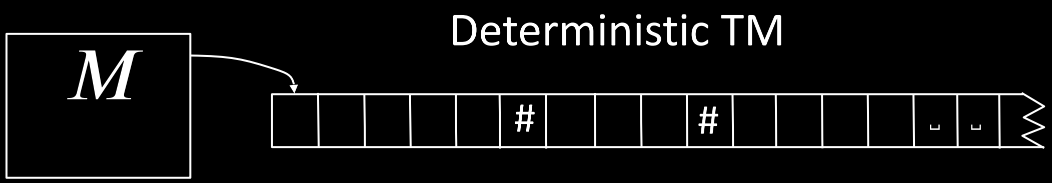
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

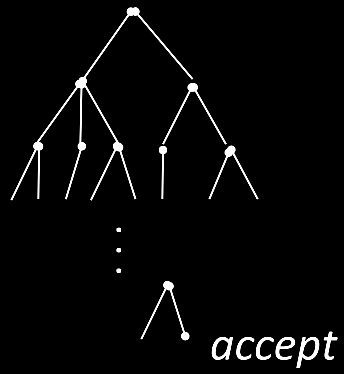
Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.







Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.

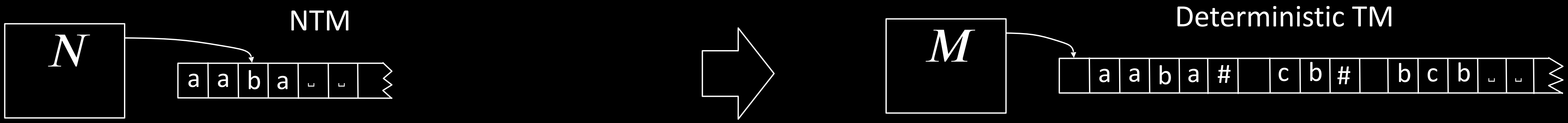
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

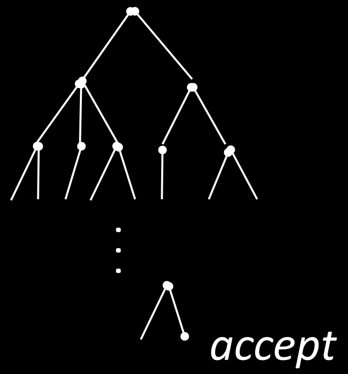
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.

Nondeterministic Turing machines

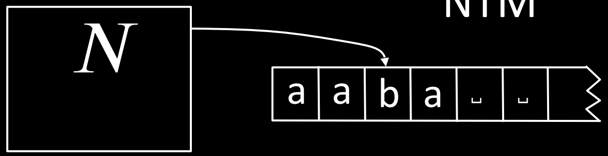
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(\,Q \times \Gamma \times$ {L, R}).

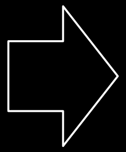
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.

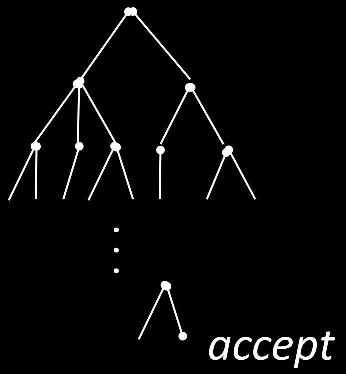
NTM





Deterministic TM

Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,

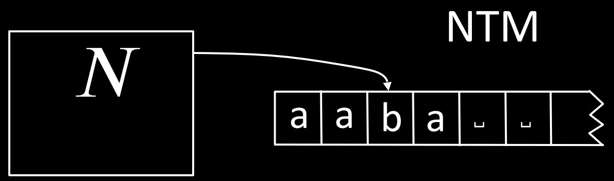
Nondeterministic Turing machines

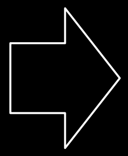
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(Q \times \Gamma \times$ {L, R}).

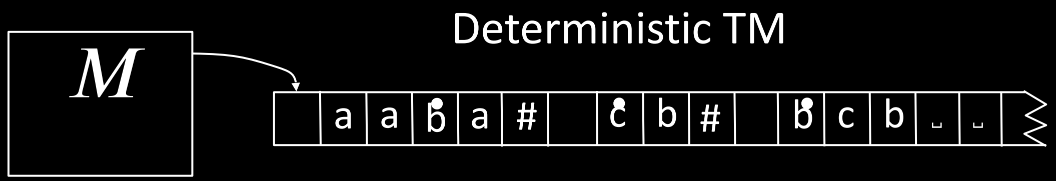
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

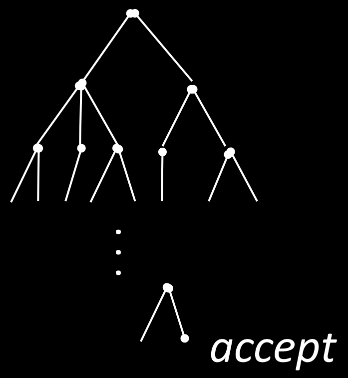
Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.







Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,

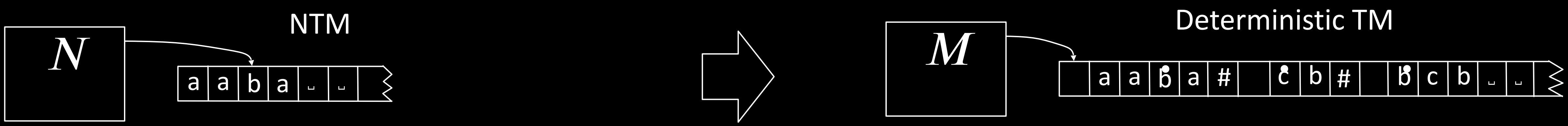
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\, Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}\,)$.

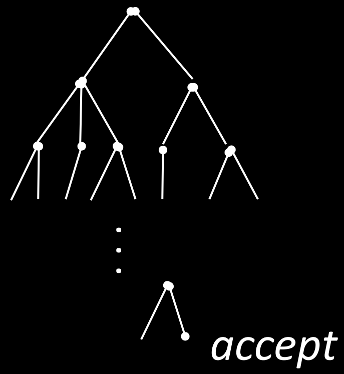
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,
and the state for each thread, in the block.

Nondeterministic Turing machines

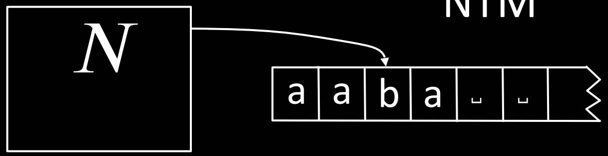
A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \rightarrow \mathscr{P}(\,Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}\,)$.

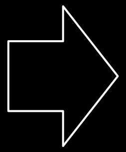
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

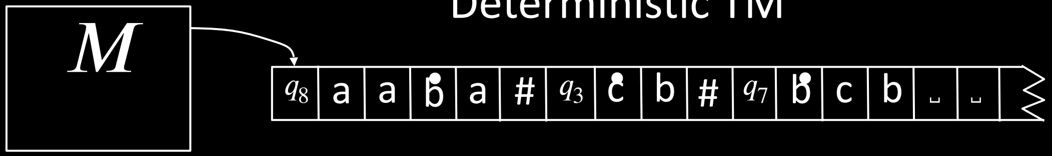
Proof: $(\rightarrow)$ immediate. $(\leftarrow)$ convert NTM to Deterministic TM.

NTM

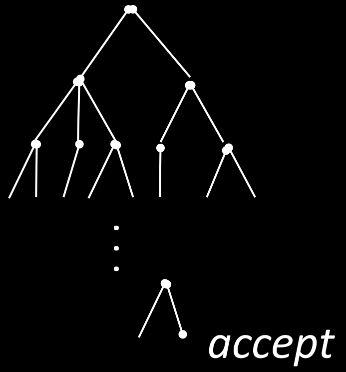




Deterministic TM



Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,
and the state for each thread, in the block.

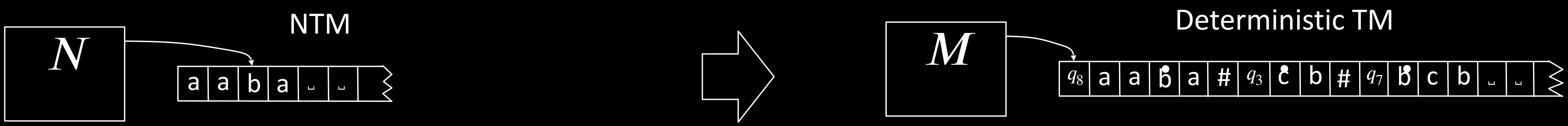
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(\,Q \times \Gamma \times$ {L, R}).

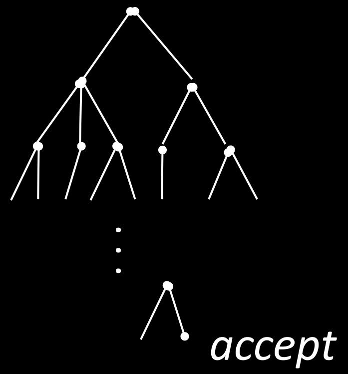
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,
and the state for each thread, in the block.

If a thread forks, then M copies the block.

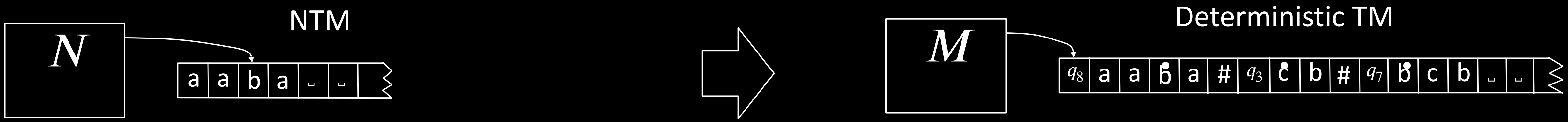
Nondeterministic Turing machines

A <u>Nondeterministic TM</u> (NTM) is similar to a Deterministic TM except for its transition function $\delta\colon \mathrm{Q} \times \Gamma \to \mathscr{P}(\, Q \times \Gamma \times$ {L, R}).

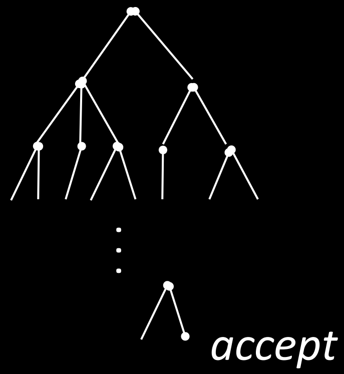
NTM accepts if there is a series of proper actions leading to qacc

Theorem: A is T-recognizable iff some NTM recognizes A

Proof: ($\rightarrow$) immediate. ($\leftarrow$) convert NTM to Deterministic TM.



Nondeterministic computation tree for N on input w.



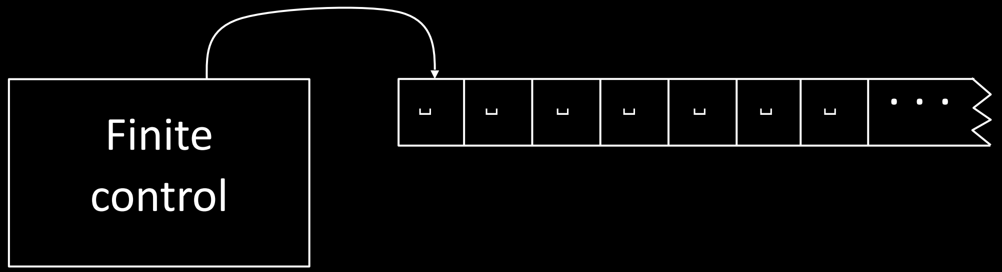
M simulates N by storing each thread's tape in a separate "block" on its tape.
Also need to store the head location,
and the state for each thread, in the block.

If a thread forks, then M copies the block.

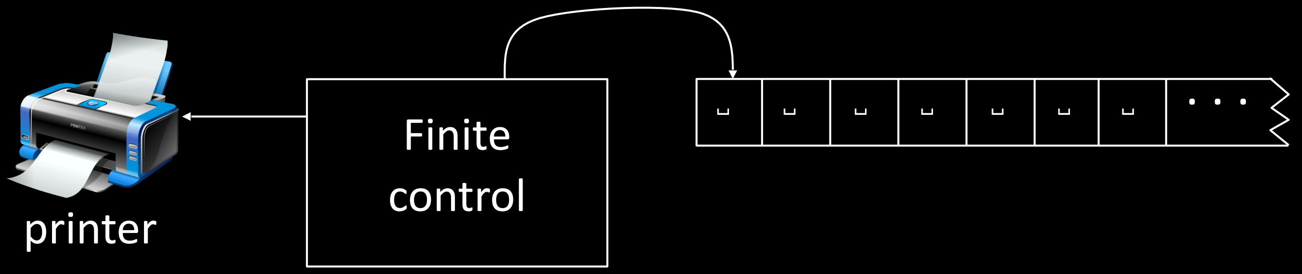
If a thread accepts then M accepts.

Turing Enumerators

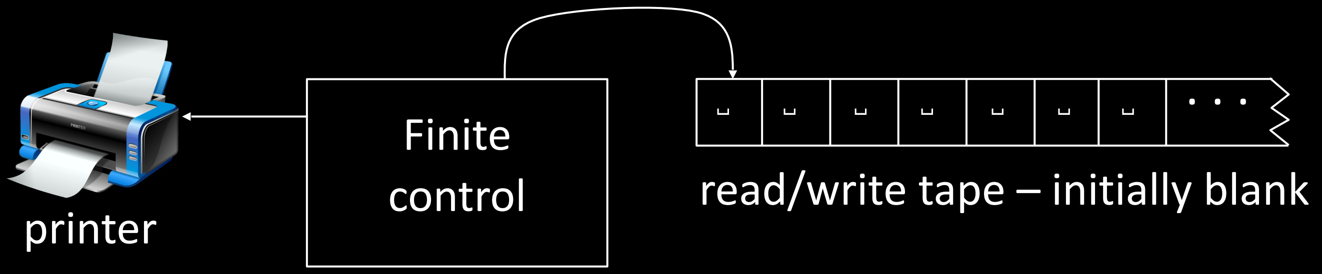
Turing Enumerators



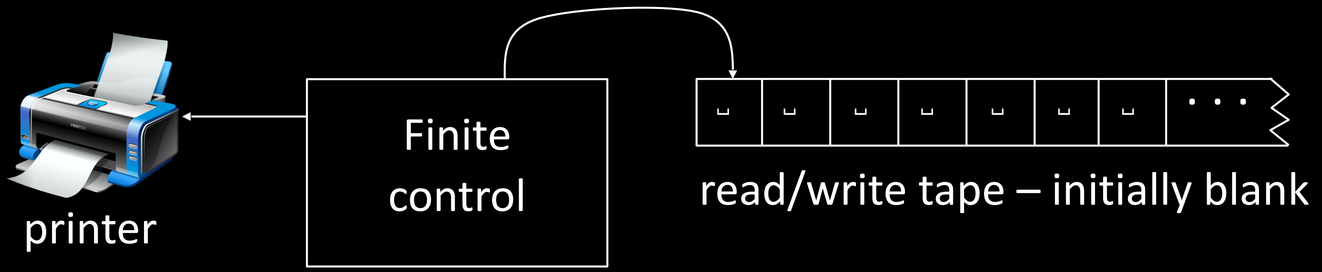
Turing Enumerators



Turing Enumerators



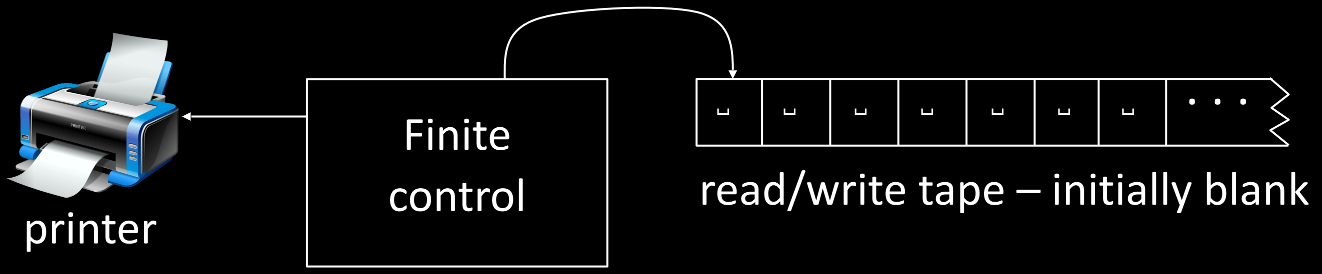
Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Turing Enumerators

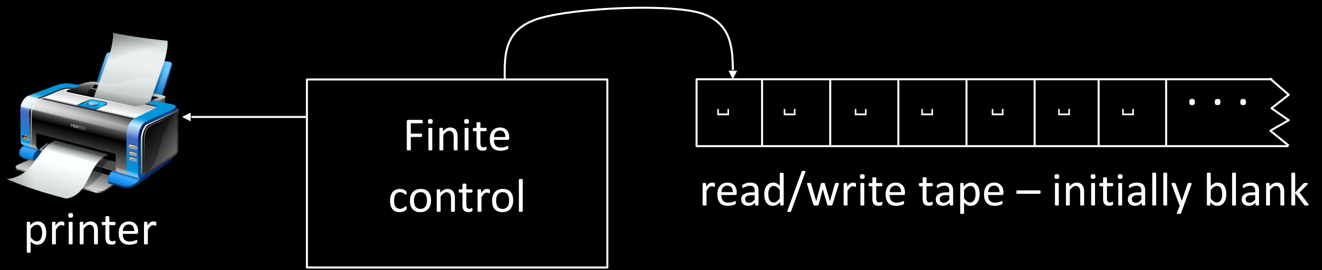


Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1 , w_2 , w_3 , \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

Turing Enumerators



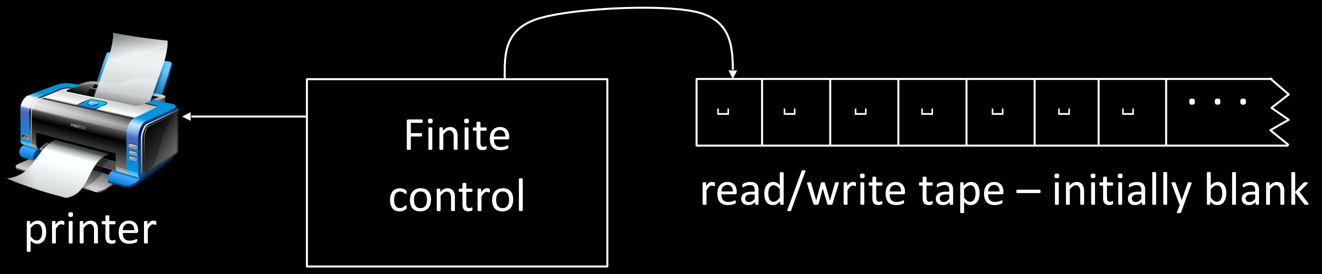
Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

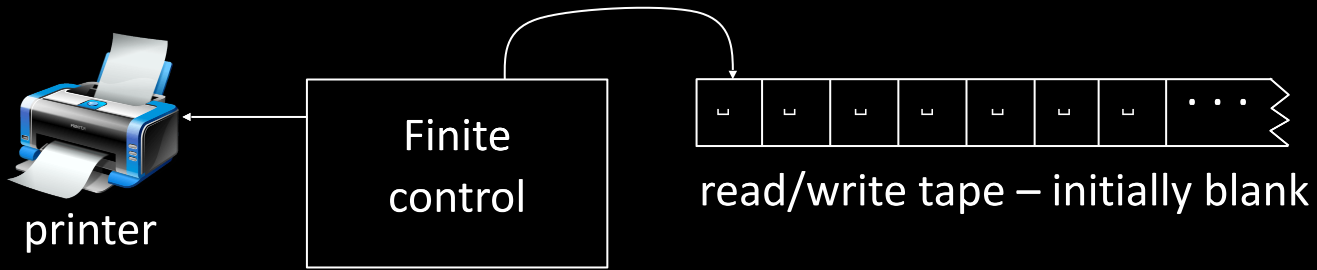
It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

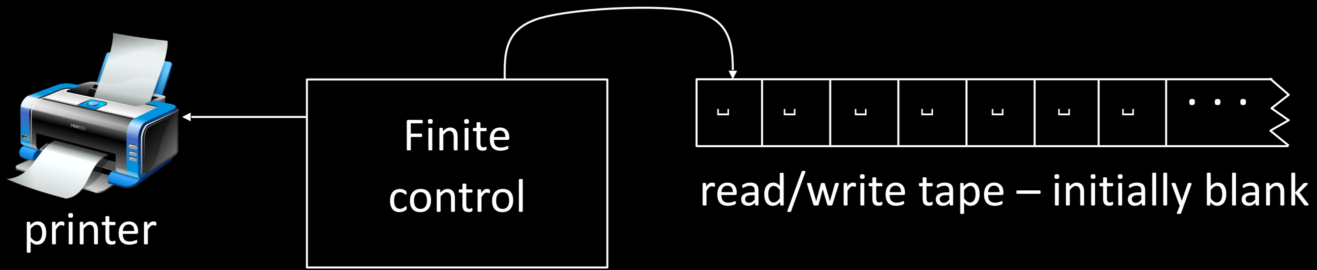
Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E$ prints $w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

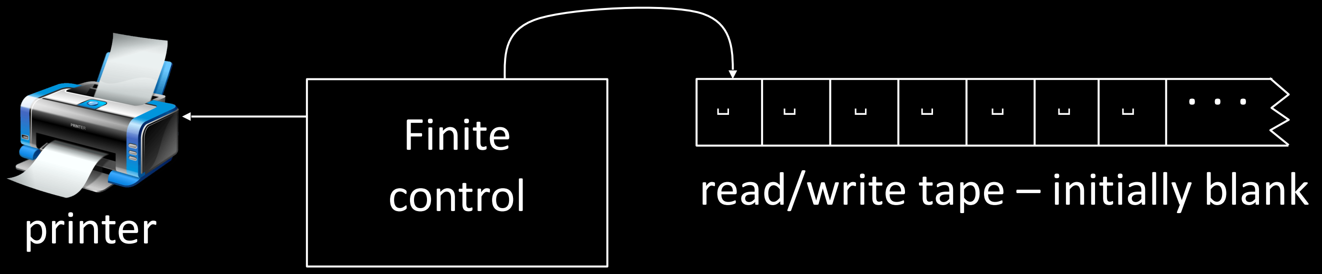
For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: (←) Convert E to equivalent TM M.

$M =$ for input w:

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

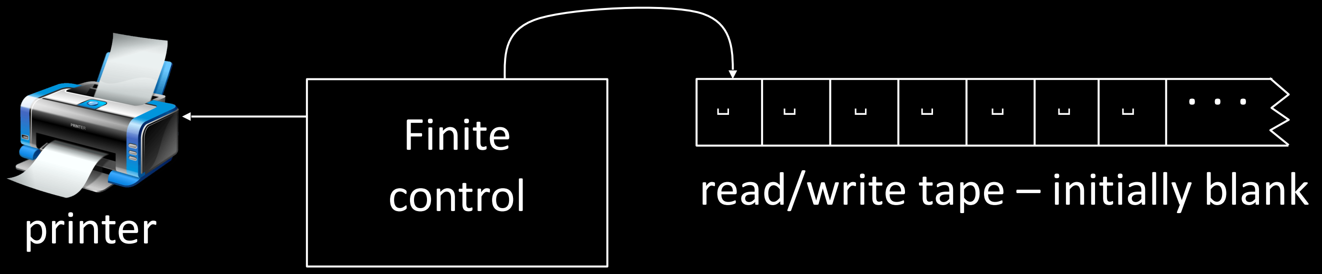
Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E$ prints $w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

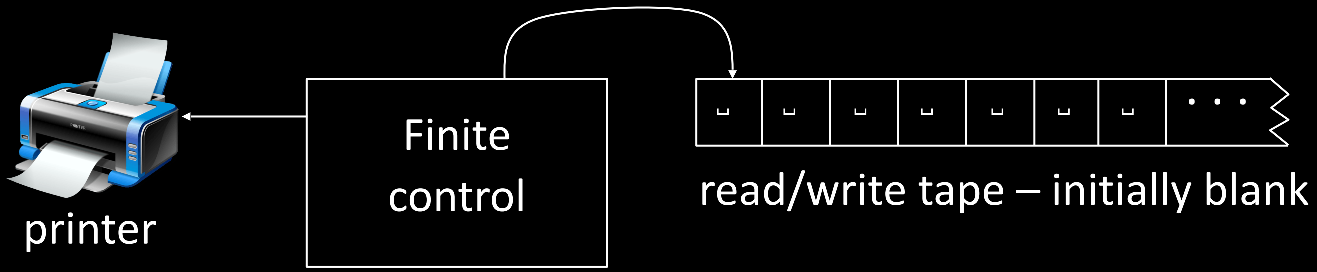
Proof: (←) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E$ prints $w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

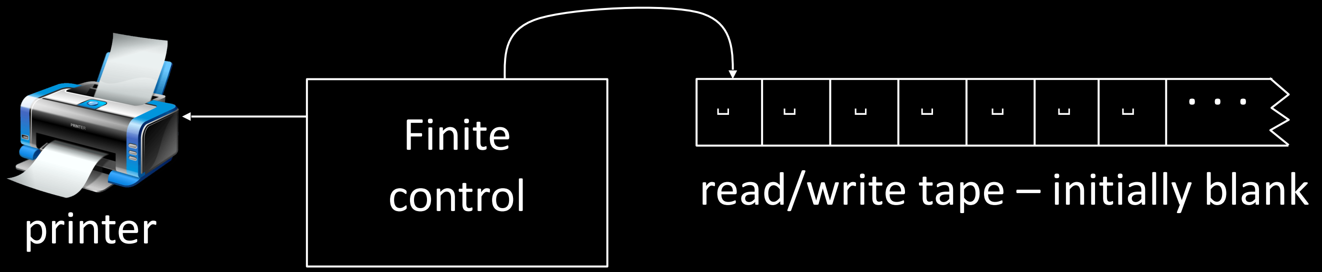
$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: (←) Convert E to equivalent TM M.

$M =$ for input w:

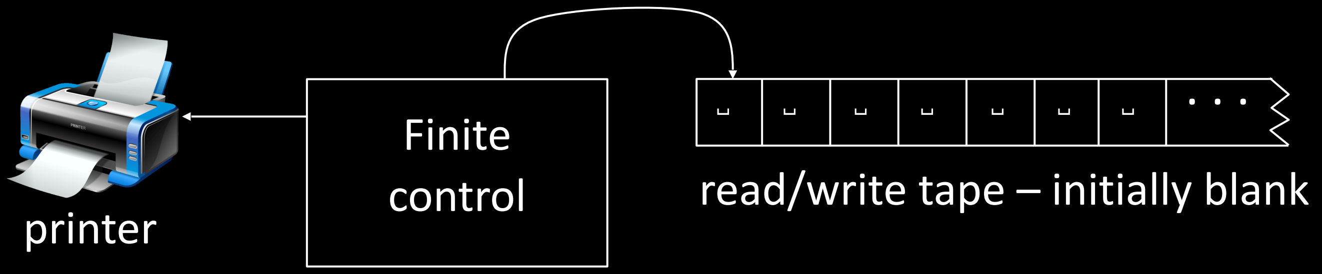
Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: (→) Convert TM M to equivalent enumerator E.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

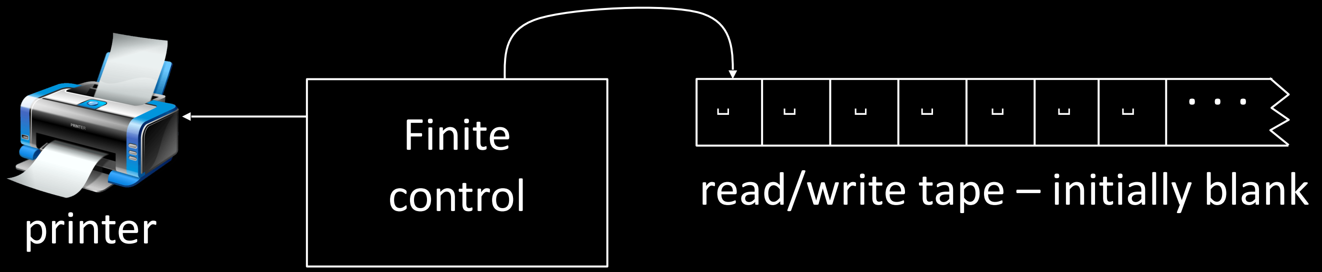
Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0,1, 00,01, 10,\ldots\}$

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E$ prints $w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

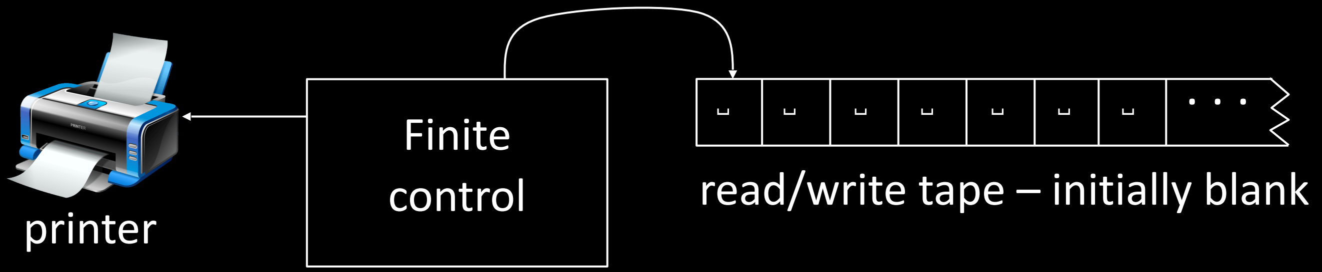
Accept if $=$ and continue otherwise.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon,\ 0,1, 00,01, 10, \ldots\}$

If M accepts w_i then print w_i .

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

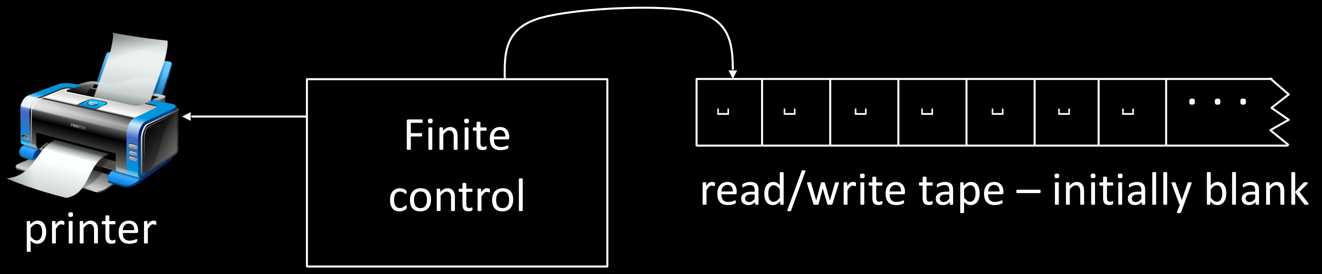
Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0,1, 00,01, 10, \ldots\}$

If M accepts w_i then print w_i.

Continue with next w_i.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

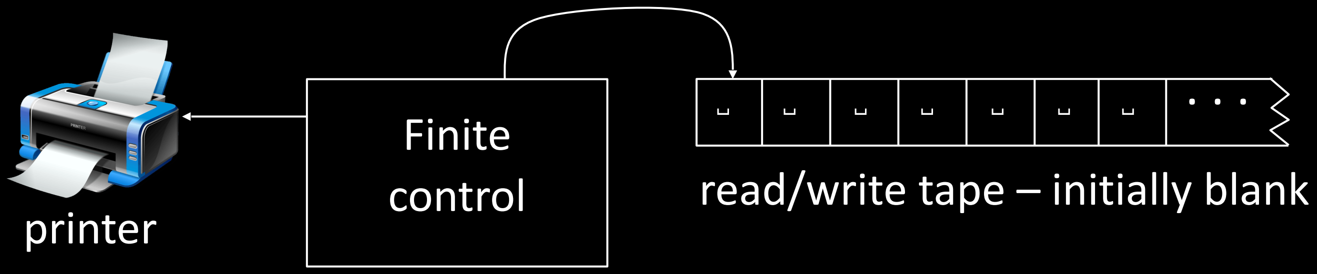
$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, \ldots\}$

If M accepts w_i then print w_i.

Continue with next w_i.

Problem: What if M on w_i loops?

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, \ldots\}$

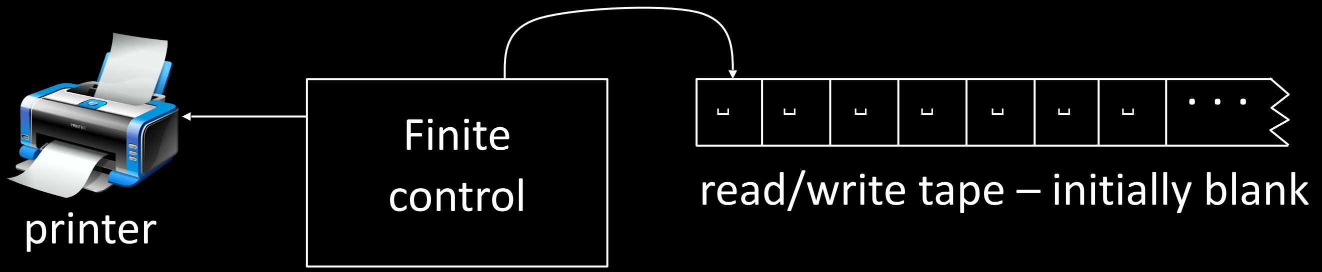
If M accepts w_i then print w_i.

Continue with next w_i.

Problem: What if M on w_i loops?

Fix: Simulate M on $w_1, w_2, \ldots, w_i$ for i steps, for $i = 1, 2, \ldots$

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0,1, 00,01, 10,\ldots\}$

If M accepts w_i then print w_i.

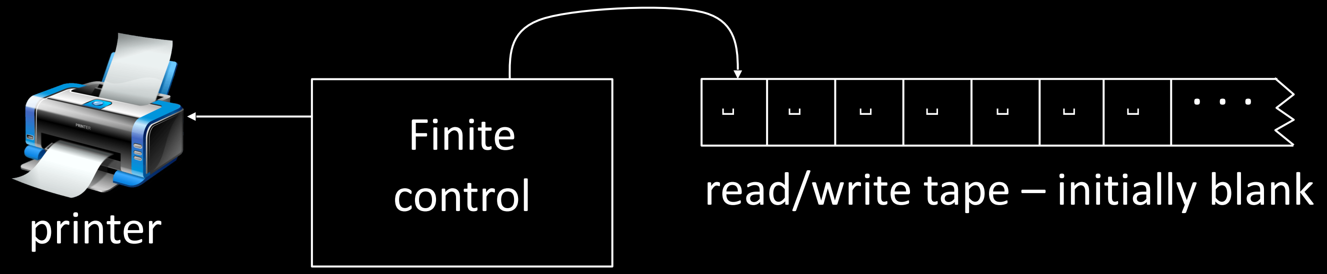
Continue with next w_i.

Problem: What if M on w_i loops?

Fix: Simulate M on $w_1, w_2, \ldots, w_i$ for i steps, for $i = 1,2,\ldots$

Print those w_i which are accepted.

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1, w_2, w_3, \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E \text{ prints } w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: (←) Convert E to equivalent TM M.

$M =$ for input w:

Simulate E (on blank input).

Whenever E prints x, test $x = w$.

Accept if $=$ and continue otherwise.

Proof: (→) Convert TM M to equivalent enumerator E.

$E =$ Simulate M on each w_i in $\Sigma^* = \{\varepsilon, 0,1,00,01,10,\ldots\}$

If M accepts w_i then print w_i.

Continue with next w_i.

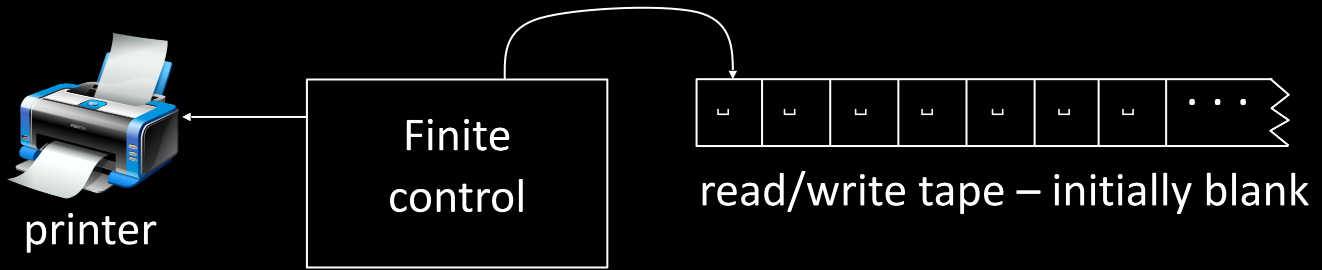
Problem: What if M on w_i loops?

Fix: Simulate M on $w_1, w_2, \ldots, w_i$ for i steps, for $i = 1,2,\ldots$

Print those w_i which are accepted.

Check-in 6.1

Turing Enumerators



Defn: A <u>Turing Enumerator</u> is a deterministic TM with a printer.

It starts on a blank tape and it can print strings $w_1 , w_2 , w_3 , \ldots$ possibly going forever.

Its language is the set of all strings it prints. It is a generator, not a recognizer.

For enumerator E we say $L(E) = \{w \mid E$ prints $w\}$.

Theorem: A is T-recognizable iff $A = L(E)$ for some T-enumerator E.

Proof: ($\leftarrow$) Convert E to equivalent TM M.

M =

Check-in 6.1

When converting TM M to enumerator E,
does E always print the strings in ***string order***?

a) Yes.
b) No.

Proof: ($\rightarrow$) Convert TM M to equivalent enumerator E.

Simulate M on each w_i in $\Sigma^* = \{\varepsilon,\ 0,1,00,01,10,\ldots\}$

M accepts w_i then print w_i .

ontinue with next w_i .

oblem: What if M on w_i loops?

x: Simulate M on $w_1 , w_2 , \ldots, w_i$ for i steps, for $i = 1,2,\ldots$

Print those w_i which are accepted.

Check-in 6.1

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm

Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm

Intuitive

Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm

Intuitive

Turing machine

Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm	Turing machine
Intuitive	Formal

Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm	=	Turing machine
Intuitive		Formal

Alan Turing
1912–1954

Church-Turing Thesis ~1936


Alonzo Church
1903–1995

Algorithm	=	Turing machine
Intuitive		Formal

Instead of Turing machines,
can use any other "reasonable" model
of unrestricted computation:

λ-calculus, random access machine,
your favorite programming language, …


Alan Turing
1912–1954

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm	=	Turing machine
Intuitive		Formal

Instead of Turing machines,
can use any other "reasonable" model
of unrestricted computation:

λ-calculus, random access machine,
your favorite programming language, …

Alan Turing
1912–1954

Big impact on mathematics.

Church-Turing Thesis ~1936

Alonzo Church
1903–1995

Algorithm	=	Turing machine
Intuitive		Formal

Alan Turing
1912–1954

Big impact on mathematics.

Hilbert's 10th Problem

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $\left|\mathbb{N}\right| < \left|A\right| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where

$\left|\mathbb{N}\right| < \left|A\right| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where

$\left|\mathbb{N}\right| < \left|A\right| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

Diophantine equations:

David Hilbert
1862—1943

Hilbert's 10^{th} Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

Diophantine equations:

Equations of polynomials where solutions must be integers.

David Hilbert
1862—1943

Hilbert's 10^{th} Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

Diophantine equations:

Equations of polynomials where <u>solutions must be integers</u>.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

David Hilbert
1862—1943

Diophantine equations:

Equations of polynomials where solutions must be integers.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

Let $D = \{p \mid$ polynomial $p(x_1, x_2, \ldots, x_k) = 0$ has a solution in integers$\}$

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where

$\left|\mathbb{N}\right| < \left|A\right| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

Diophantine equations:

Equations of polynomials where solutions must be integers.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

Let $D = \{p \mid$ polynomial $p(x_1,\ x_2,\ \ldots,\ x_k) = 0$ has a solution in integers)

Hilbert's 10th problem: Give an algorithm to decide D.

David Hilbert
1862—1943

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

David Hilbert
1862—1943

Diophantine equations:

Equations of polynomials where <u>solutions must be integers</u>.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

Let $D = \{p \mid$ polynomial $p(x_1, x_2, \ldots, x_k) = 0$ has a <u>solution in integers</u>)

Hilbert's 10th problem: Give an algorithm to decide D.

Matiyasevich proved in 1970: D is not decidable.

Hilbert's 10th Problem

In 1900 David Hilbert posed 23 problems

#1) Problem of the continuum (Does set A exist where $|\mathbb{N}| < |A| < |\mathbb{R}|$?).

#2) Prove that the axioms of mathematics are consistent.

#10) Give an algorithm for solving *Diophantine equations.*

David Hilbert
1862—1943

Diophantine equations:

Equations of polynomials where solutions must be integers.

Example: $3x^2 - 2xy - y^2z = 7$ solution: $x = 1,\ y = 2,\ z = -2$

Let $D = \{p \mid$ polynomial $p(x_1, x_2, \ldots, x_k) = 0$ has a solution in integers)

Hilbert's 10th problem: Give an algorithm to decide D.

Matiyasevich proved in 1970: D is not decidable.

Note: D is T-recognizable.

Notation for encodings and TMs

Notation for encodings and TMs

Notation for writing Turing machines

We will use high-level English descriptions of algorithms when we describe TMs, knowing that we could (in principle) convert those descriptions into states, transition function, etc. Our notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

Notation for encodings and TMs

Notation for writing Turing machines

We will use high-level English descriptions of algorithms when we describe TMs, knowing that we could (in principle) convert those descriptions into states, transition function, etc. Our notation for writing a TM M is

$M =$ "On input w

[English description of the algorithm]"

Check-in 6.3

Notation for encodings and TMs

Check-in 6.3

If x and y are strings, would xy be a good choice for their encoding $\langle x, y\rangle$ into a single string?

a) Yes.

b) No.

Check-in 6.3

TM – example revisited

TM – example revisited

TM M recognizing $B = \{\mathrm{a}^k\mathrm{b}^k\mathrm{c}^k \mid k \geq 0\}$

$M =$ "On input w

1. Check if $w \in \mathrm{a}^*\mathrm{b}^*\mathrm{c}^*$, *reject* if not.
2. Count the number of a's, b's, and c's in w.
3. *Accept* if all counts are equal; *reject* if not."

High-level description is ok.
You do not need to manage tapes, states, etc…

Problem Set 2

Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid\ \ \exists y \ \ \langle x, y \rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y \rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.

Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid \ \exists y \ \ \langle x, y \rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y \rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.

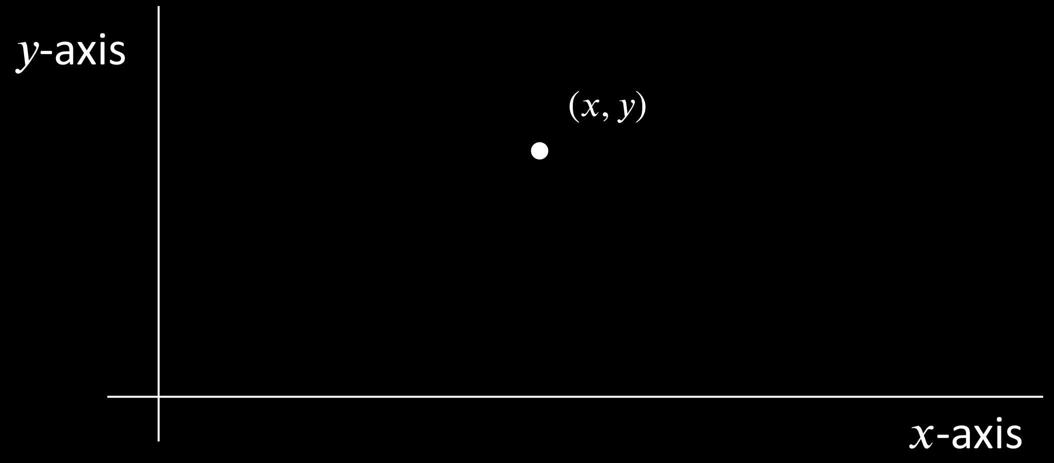
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid \ \exists y \ \ \langle x, y\rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y\rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



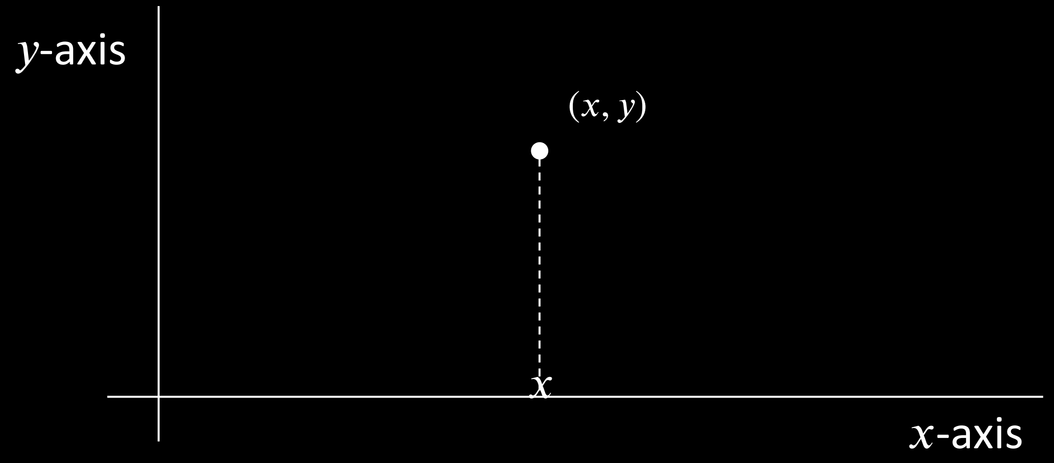
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid\ \exists y\ \ \langle x, y\rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y\rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



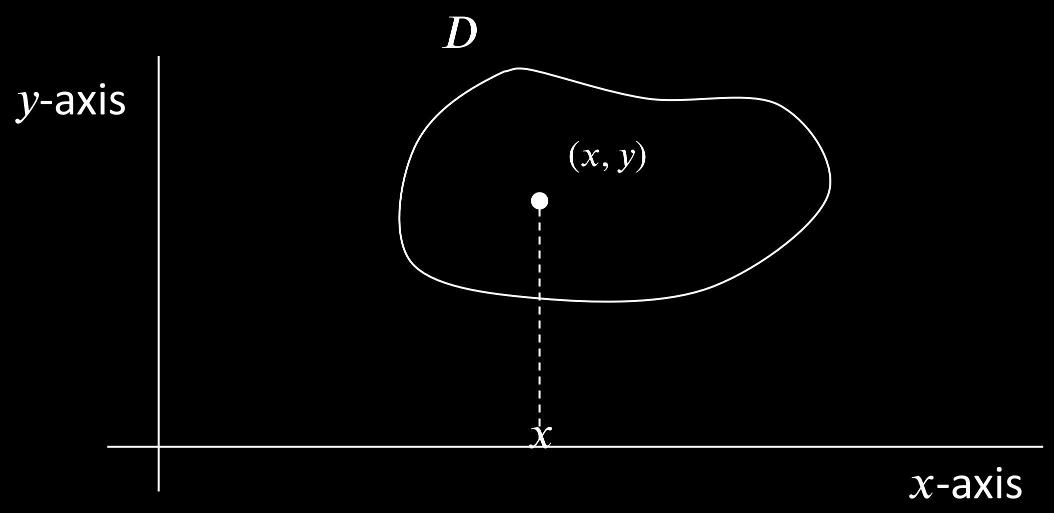
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid\ \exists y\ \ \langle x, y \rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y \rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



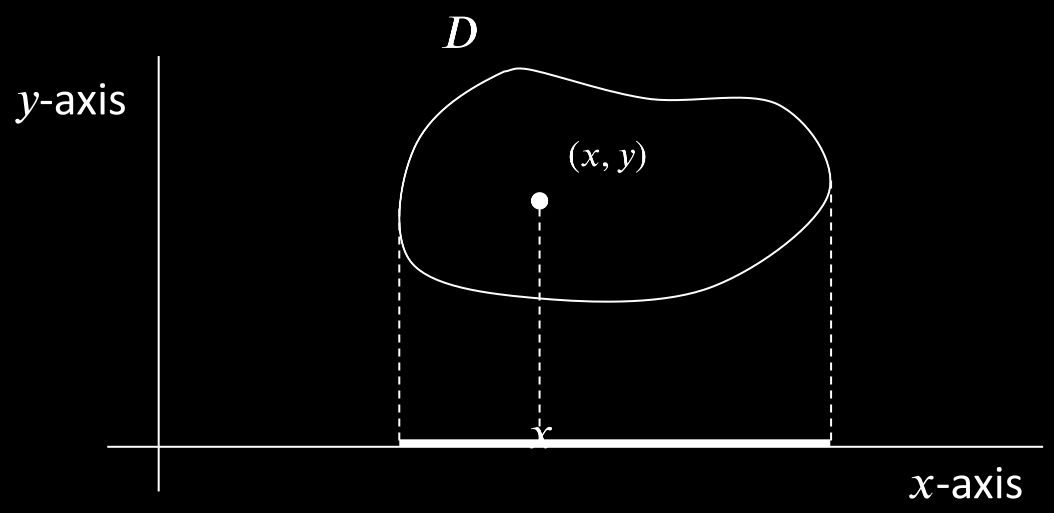
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid \ \exists y \ \ \langle x, y\rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y\rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



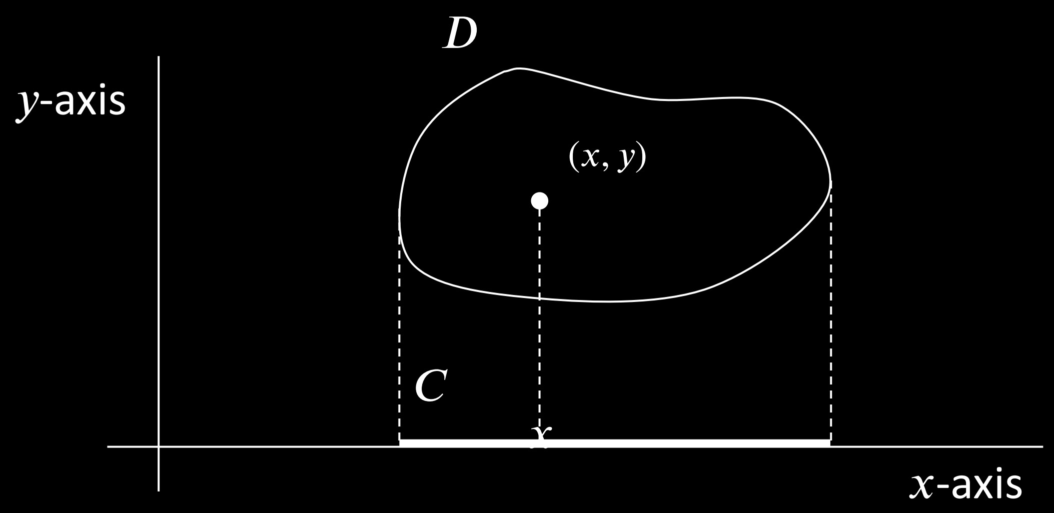
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid\ \ \exists y\ \ \langle x, y\rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y\rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



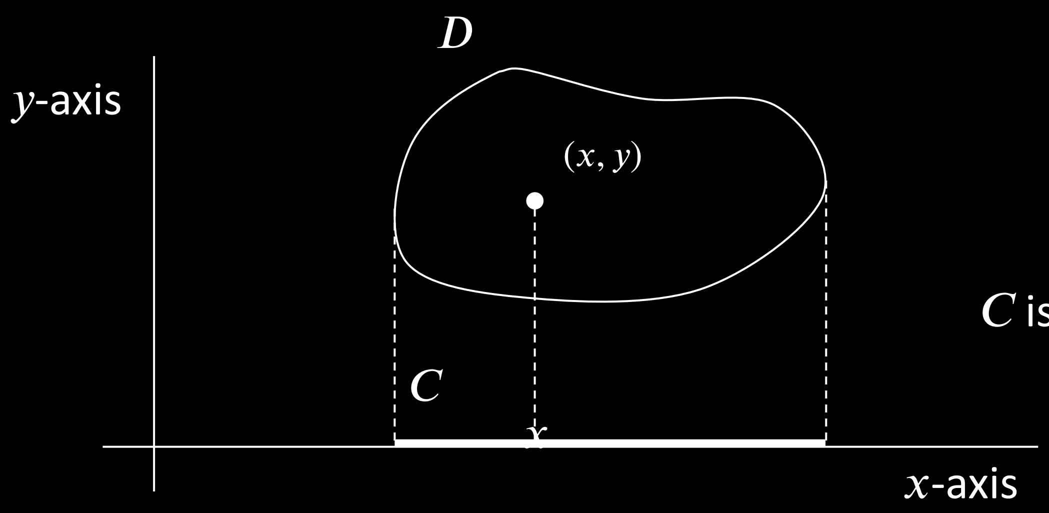
Problem Set 2

#5) Show C is T-recognizable iff there is a decidable D where

$$C = \{\ x \mid \ \exists y \ \ \langle x, y \rangle \in D\ \} \qquad x, y \in \Sigma^*$$

$\langle x, y \rangle$ is an encoding of the pair of strings x and y into a single string.

Think of D as a collection of pairs of strings.



C is a "projection" of D

Quick review of today

Quick review of today

1. CFG pumping lemma

Quick review of today

1. CFG pumping lemma
2. We showed that various TM variants (multi-tape, nondeterministic, enumerator) are all equivalent to the single-tape model.

Quick review of today

1. CFG pumping lemma
2. We showed that various TM variants (multi-tape, nondeterministic, enumerator) are all equivalent to the single-tape model.
3. Concluded that all “reasonable” models with unrestricted memory access are equivalent.

Quick review of today

1. CFG pumping lemma
2. We showed that various TM variants (multi-tape, nondeterministic, enumerator) are all equivalent to the single-tape model.
3. Concluded that all “reasonable” models with unrestricted memory access are equivalent.
4. Discussed the Church-Turing Thesis: Turing machines are equivalent to “algorithms”.

Quick review of today

1. CFG pumping lemma
2. We showed that various TM variants (multi-tape, nondeterministic, enumerator) are all equivalent to the single-tape model.
3. Concluded that all "reasonable" models with unrestricted memory access are equivalent.
4. Discussed the Church-Turing Thesis:
   Turing machines are equivalent to "algorithms".
5. Notation for encoding objects and describing TMs.

Quick review of today

1. CFG pumping lemma
2. We showed that various TM variants (multi-tape, nondeterministic, enumerator) are all equivalent to the single-tape model.
3. Concluded that all "reasonable" models with unrestricted memory access are equivalent.
4. Discussed the Church-Turing Thesis: Turing machines are equivalent to "algorithms".
5. Notation for encoding objects and describing TMs.
6. Discussed Pset 2 Problem 5.