Week 9 Tutorial

Context-Free Grammars, TMs

Part 1: ***CFGs Warmup***

Let $\Sigma = \{$**y**, **d**$\}$ and let $DOGWALK = \{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Let $\Sigma = \{$y, d$\}$ and let *DOGWALK* = $\{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \to YSD \mid DSY \mid \varepsilon$$
$$Y \to yY \mid \varepsilon$$
$$D \to dD \mid \varepsilon$$

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Let Σ = {y, d} and let *DOGWALK* = {$w \in \Sigma^*$ | w describes a series of steps where you and your dog arrive at the same point }

Here are some *incorrect* CFGs for *DOGWALK:*

1

$S \rightarrow YSD \mid DSY \mid \varepsilon$
$Y \rightarrow yY \mid \varepsilon$
$D \rightarrow dD \mid \varepsilon$

2

$S \rightarrow ySd \mid dSy \mid \varepsilon$

3

$S \rightarrow ydS \mid dyS \mid \varepsilon$

4

$S \rightarrow ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$

1) Explain why none of these grammars are correct by identifying an example string in the language of the grammar but not in *DOGWALK* or a string that's in *DOGWALK* that's not in the language of the grammar.

Fill in answer on Gradescope!

Let $\Sigma = \{\mathbf{y}, \mathbf{d}\}$ and let *DOGWALK* = $\{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \to YSD \mid DSY \mid \varepsilon$$
$$Y \to \mathbf{y}Y \mid \varepsilon$$
$$D \to \mathbf{d}D \mid \varepsilon$$

2

$$S \to \mathbf{y}S\mathbf{d} \mid \mathbf{d}S\mathbf{y} \mid \varepsilon$$

3

$$S \to \mathbf{yd}S \mid \mathbf{dy}S \mid \varepsilon$$

4

$$S \to \mathbf{y}S\mathbf{d} \mid \mathbf{d}S\mathbf{y} \mid \mathbf{yd}S \mid \mathbf{dy}S \mid \varepsilon$$

Let Σ = {**y**, **d**} and let *DOGWALK* = { $w \in \Sigma^*$ | w describes a series of steps where you and your dog arrive at the same point }

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$\begin{aligned} S &\to YSD \mid DSY \mid \varepsilon \\ Y &\to yY \mid \varepsilon \\ D &\to dD \mid \varepsilon \end{aligned}$$

This grammar generates the string **dd**, which is not in *DOGWALK*.

Takeaway: related quantities can't be built independently. If two parts of your string have to match up, they need to be built together.

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Let $\Sigma = \{$**y**, **d**$\}$ and let *DOGWALK* $= \{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \to YSD \mid DSY \mid \varepsilon$$
$$Y \to yY \mid \varepsilon$$
$$D \to dD \mid \varepsilon$$

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Let $\Sigma = \{\mathbf{y}, \mathbf{d}\}$ and let $DOGWALK = \{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \to YSD \mid DSY \mid \varepsilon$$
$$Y \to yY \mid \varepsilon$$
$$D \to dD \mid \varepsilon$$

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

This grammar can't generate the string **yddy**, which is in *DOGWALK*.

Takeaway: make sure you don't unintentionally impose additional restrictions. While we need the number of **y**s and **d**s to be the same, it doesn't matter what order they come in.

Let $\Sigma = \{\mathbf{y}, \mathbf{d}\}$ and let *DOGWALK* = $\{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$\begin{aligned} S &\rightarrow YSD \mid DSY \mid \varepsilon \\ Y &\rightarrow \mathbf{y}Y \mid \varepsilon \\ D &\rightarrow \mathbf{d}D \mid \varepsilon \end{aligned}$$

2

$$S \rightarrow \mathbf{y}S\mathbf{d} \mid \mathbf{d}S\mathbf{y} \mid \varepsilon$$

3

$$S \rightarrow \mathbf{yd}S \mid \mathbf{dy}S \mid \varepsilon$$

4

$$S \rightarrow \mathbf{y}S\mathbf{d} \mid \mathbf{d}S\mathbf{y} \mid \mathbf{yd}S \mid \mathbf{dy}S \mid \varepsilon$$

Let $\Sigma = \{$**y**, **d**$\}$ and let *DOGWALK* = $\{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$\begin{aligned} S &\to YSD \mid DSY \mid \varepsilon \\ Y &\to yY \mid \varepsilon \\ D &\to dD \mid \varepsilon \end{aligned}$$

This grammar can't generate the string **yydd**, which is in DOGWALK.

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

Takeaway: similar to the previous option, this grammar restricts the ordering of **y**s and **d**s.

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Let $\Sigma = \{y, d\}$ and let $DOGWALK = \{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \rightarrow YSD \mid DSY \mid \varepsilon$$
$$Y \rightarrow yY \mid \varepsilon$$
$$D \rightarrow dD \mid \varepsilon$$

2

$$S \rightarrow ySd \mid dSy \mid \varepsilon$$

3

$$S \rightarrow ydS \mid dyS \mid \varepsilon$$

4

$$S \rightarrow ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Let $\Sigma = \{$**y**, **d**$\}$ and let *DOGWALK* = $\{w \in \Sigma^* \mid w$ describes a series of steps where you and your dog arrive at the same point $\}$

Here are some *incorrect* CFGs for *DOGWALK:*

1

$$S \to YSD \mid DSY \mid \varepsilon$$
$$Y \to yY \mid \varepsilon$$
$$D \to dD \mid \varepsilon$$

This grammar can't generate the string **yydddyy**, which is in *DOGWALK*.

Takeaway: don't try to patch up a CFG by adding in more productions. In CFG design, you're looking for a general rule that captures the language.

2

$$S \to ySd \mid dSy \mid \varepsilon$$

3

$$S \to ydS \mid dyS \mid \varepsilon$$

For this particular example, simply listing off all permutations of **y**, **d**, and **S** isn't a great approach because you can't be sure that you've covered everything.

4

$$S \to ySd \mid dSy \mid ydS \mid dyS \mid \varepsilon$$

Part 2: ***Designing CFGs***

Storing Information in Nonterminals

- ***Key idea:*** Different non-terminals should represent different states or different types of strings.
  - For example, different phases of the build, or different possible structures for the string.
  - Think like the same ideas from DFA/NFA design where states in your automata represent pieces of information.

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Examples:

$\varepsilon \in L$	$\texttt{a} \notin L$
$\texttt{abb} \in L$	$\texttt{b} \notin L$
$\texttt{bab} \in L$	$\texttt{ababab} \notin L$
$\texttt{aababa} \in L$	$\texttt{aabaaaaaa} \notin L$
$\texttt{bbbbbb} \in L$	$\texttt{bbbb} \notin L$

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Examples:

ε $\in L$	**a** $\notin L$
a bb $\in L$	**b** $\notin L$
b ab $\in L$	**ab abab** $\notin L$
aa baba $\in L$	**aab aaaaaa** $\notin L$
bb bbbb $\in L$	**bbbb** $\notin L$

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

2) Create a CFG for the language above.

Fill in answer on Gradescope!

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	**bab**
abb	**bbb**
aaabab	**bbabbb**
aababa	**bbbaaaaaa**
aaaaaaaaa	**bbbbbabaa**

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Observation 1:

Strings in this language are either: the first third is **a**s or the first third is **b**s.

Storing Information in Nonterminals

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider this language:

$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Observation 2:

Among these strings, for every **a** I have in the first third, I need two other characters in the last two-thirds.

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa bab

abb bbb

aaabab bbabbb

aaaaa

babaa

Observation 2:

Among these strings, for every **a** I have in the first third, I need two other characters in the last two-thirds.

This pattern of "for every x I see here, I need a y somewhere else in the string" is very common in CFGs!

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

Observation 3:

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

a	a	a	a	b	a	b	b	a	b	b	b

Observation 3:

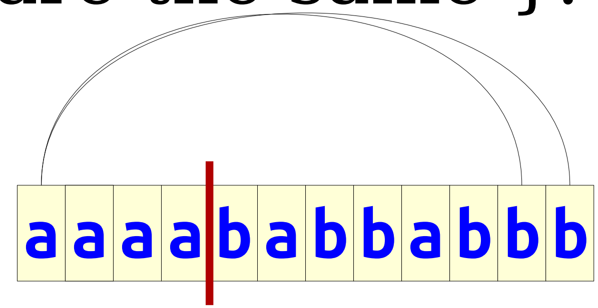
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

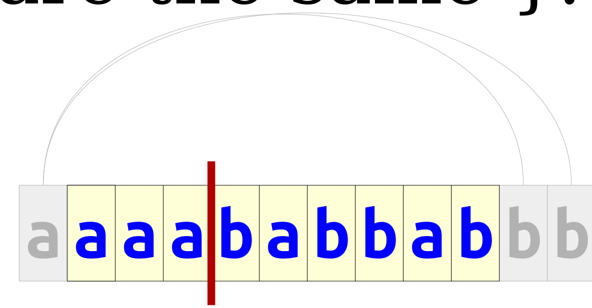
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

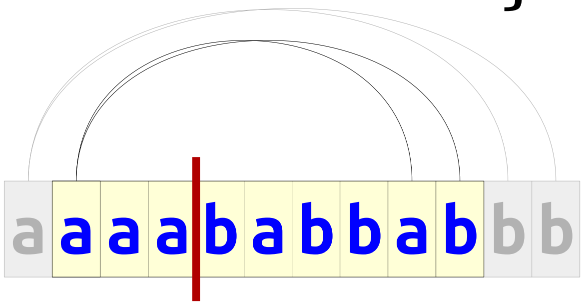
Storing Information in Nonterminals

- Let Σ = {**a**, **b**} and consider this language:

  L = { $w \in \Sigma^*$ | $|w| \equiv_3 0$ and all characters in the first third of w are the same }.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

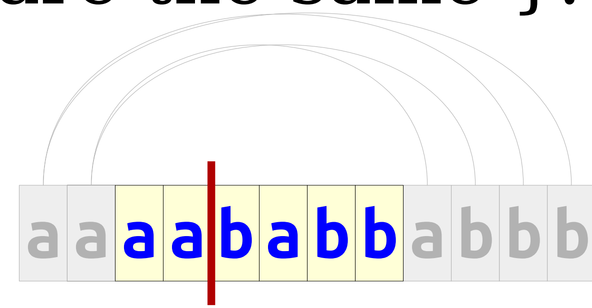
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

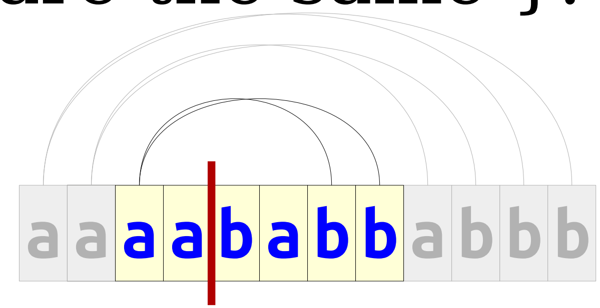
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

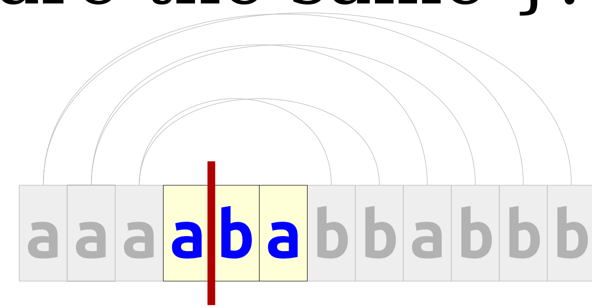
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

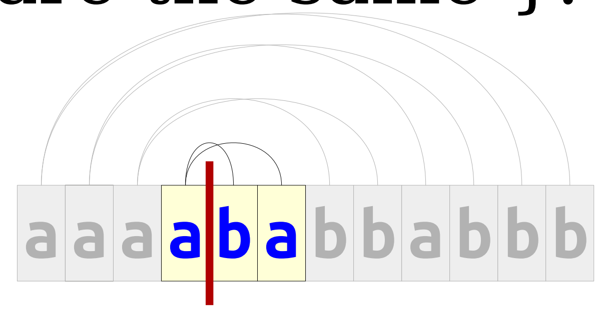
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

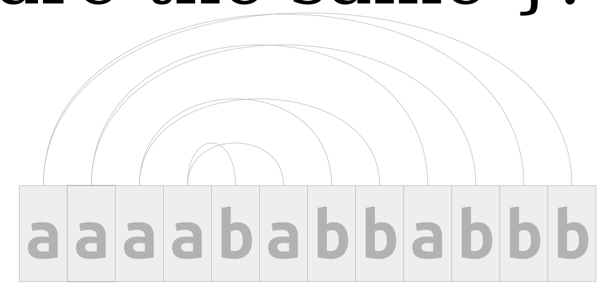
Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

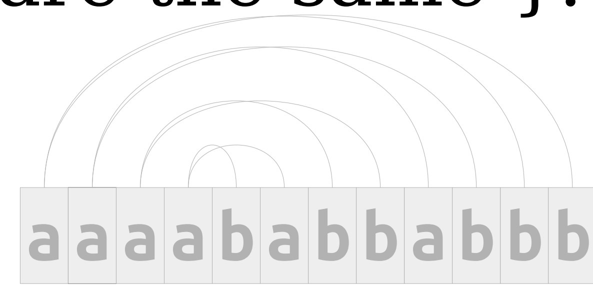
Storing Information in Nonterminals

- Let Σ = {a, b} and consider this language:

$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa



Observation 3:

Crossing off the first character and last two characters leaves a string in L.

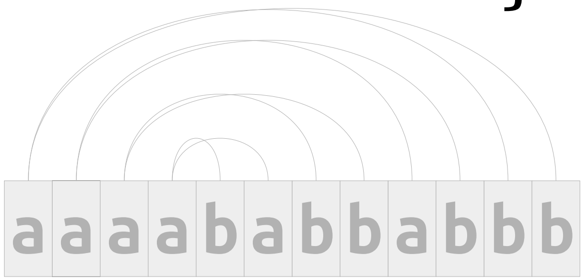
Base case: $\varepsilon \in L$.

Storing Information in Nonterminals

- Let $\Sigma = \{$a, b$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:



aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

$A \rightarrow$ aA_ _ | ε

Observation 3:

Crossing off the first character and last two characters leaves a string in L.

Base case: $\varepsilon \in L$.

Storing Information in Nonterminals

- Let $\Sigma = \{$**a**, **b**$\}$ and consider this language:

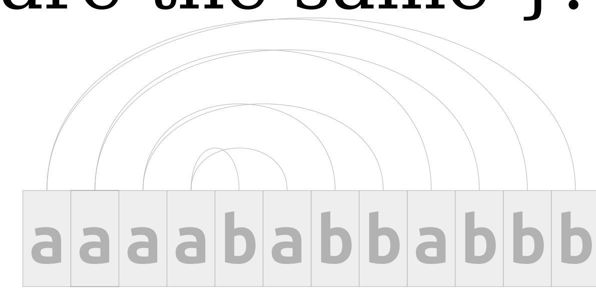
$$L = \{ w \in \Sigma^* \mid |w| \equiv_3 0 \text{ and all characters in the first third of } w \text{ are the same } \}.$$

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

A → **a**AXX | **ε**

X → **a** | **b**



Observation 3:

Crossing off the first character and last two characters leaves a string in L.

Base case: $\varepsilon \in L$.

Storing Information in Nonterminals

- Let $\Sigma = \{$a, b$\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- One approach:

aaa	bab
abb	bbb
aaabab	bbabbb
aababa	bbbaaaaaa
aaaaaaaaa	bbbbbabaa

B → b**BXX** | ε   **X** → a | b

Storing Information in Nonterminals

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Tying everything together:

  $\mathbf{S} \to \mathbf{A} \mid \mathbf{B}$

  $\mathbf{A} \to \mathtt{a}\mathbf{AXX} \mid \varepsilon$

  $\mathbf{B} \to \mathtt{b}\mathbf{BXX} \mid \varepsilon$

  $\mathbf{X} \to \mathtt{a} \mid \mathtt{b}$

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Tying everything together:

  $\mathbf{S} \rightarrow \mathbf{A} \mid \mathbf{B}$

  $\mathbf{A} \rightarrow \texttt{a}\mathbf{AXX} \mid \varepsilon$

  $\mathbf{B} \rightarrow \texttt{b}\mathbf{BXX} \mid \varepsilon$

  $\mathbf{X} \rightarrow \texttt{a} \mid \texttt{b}$

Overall strings in this language either follow the pattern of **A** or **B**.

Storing Information in Nonterminals

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Tying everything together:

$$\begin{aligned} \mathbf{S} &\to \mathbf{A} \mid \mathbf{B} \\ \mathbf{A} &\to \mathtt{a}\mathbf{AXX} \mid \varepsilon \\ \mathbf{B} &\to \mathtt{b}\mathbf{BXX} \mid \varepsilon \\ \mathbf{X} &\to \mathtt{a} \mid \mathtt{b} \end{aligned}$$

A represents "strings where the first third is **a**'s"

Storing Information in Nonterminals

- Let $\Sigma = \{\texttt{a}, \texttt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Tying everything together:

$$
\begin{aligned}
\mathbf{S} &\to \mathbf{A} \mid \mathbf{B} \\
\mathbf{A} &\to \texttt{a}\mathbf{AXX} \mid \varepsilon \\
\mathbf{B} &\to \texttt{b}\mathbf{BXX} \mid \varepsilon \\
\mathbf{X} &\to \texttt{a} \mid \texttt{b}
\end{aligned}
$$

B represents "strings where the first third is **b**'s"

Storing Information in Nonterminals

- Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider this language:

  $L = \{ w \in \Sigma^* \mid |w| \equiv_3 0$ and all characters in the first third of w are the same $\}$.

- Tying everything together:

$$
\begin{aligned}
\mathbf{S} &\rightarrow \mathbf{A} \mid \mathbf{B} \\
\mathbf{A} &\rightarrow \mathtt{a}\mathbf{AXX} \mid \varepsilon \\
\mathbf{B} &\rightarrow \mathtt{b}\mathbf{BXX} \mid \varepsilon \\
\mathbf{X} &\rightarrow \mathtt{a} \mid \mathtt{b}
\end{aligned}
$$

X represents "either an **a** or a **b**"

Part 3: ***Turing Machines***

TMs and Programs

- Though TMs are formally defined using states, transitions, and a tape, we can describe the behavior of *what* TMs can do by writing pseudocode and abstract away the details of *how* exactly it's operating.
- Throughout the rest of the course, we'll switch back and forth between these two different models of TM behavior.

3. Each of the following programs is a decider for some language. Tell us what each of those languages is.

```
/* Program One */
int main() {
  string input = getInput();
  for (char ch: input) {
    if (ch != 'a') reject();
  }
  accept();
}
```

```
/* Program Two */
int main() {
  string input = getInput();
  int n = input.size();
  if (n == 0) reject();

  while (n != 1) {
    if (n % 2 != 0) {
      reject();
    }
    n /= 2;
  }
  accept();
}
```

```
/* Program Three */
int main() {
  string input = getInput();
  if (input == "") accept();

  int left = 0;
  int right = input.size() - 1;

  while (left < right) {
    if (input[left] != input[right]) {
      reject();
    }
    left++; right--;
  }
  accept();
}
```

```
/* Program One */
int main() {
  string input = getInput();
  for (char ch: input) {
    if (ch != 'a') reject();
  }
  accept();
}
```

```
/* Program Two */
int main() {
  string input = getInput();
  int n = input.size();
  if (n == 0) reject();

  while (n != 1) {
    if (n % 2 != 0) {
      reject();
    }
    n /= 2;
  }
  accept();
}
```

```
/* Program Three */
int main() {
  string input = getInput();
  if (input == "") accept();

  int left = 0;
  int right = input.size() - 1;

  while (left < right) {
    if (input[left] != input[right]) {
      reject();
    }
    left++; right--;
  }
  accept();
}
```

4. Each of the following programs is a decider for some language. Tell us what each of those languages is.

```
/* Program One */
int main() {
  string input = getInput();
  string me = mySource();

  if (input != "" && input[0] == me[0]) {
    accept();
  } else {
    reject();
  }
}
```

```
/* Program Two */
int main() {
  string input = getInput();
  string me = mySource();

  if (me == me + input) {
    accept();
  } else {
    reject();
  }
}
```

```
/* Program Three */
int main() {
  string input = getInput();
  string me = mySource();

  if (me == "quokka") {
    reject();
  } else {
    accept();
  }
}
```

```
/* Program One */
int main() {
  string input = getInput();
  string me = mySource();

  if (input != "" && input[0] == me[0]) {
    accept();
  } else {
    reject();
  }
}
```

```
/* Program Two */
int main() {
  string input = getInput();
  string me = mySource();

  if (me == me + input) {
    accept();
  } else {
    reject();
  }
}
```

```
/* Program Three */
int main() {
  string input = getInput();
  string me = mySource();

  if (me == "quokka") {
    reject();
  } else {
    accept();
  }
}
```

Thanks for Calling In!

Stay safe, stay healthy,
and have a good week!

See you next time.