بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه سوم: زبان‌های غیر منظم و زبان‌های مستقل از زمینه

Theory of computation - 002 - S03 - Non-Regular Languanges & CFLs

Non-Regular Languages

How do we show a language is not regular?

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w$ has equal numbers of 01 and 10 substrings$\}$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w$ has equal numbers of 01 and 10 substrings$\}$

   0101 $\notin C$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$

$0101 \notin C$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$

   0101 $\notin C$ 0110 $\in C$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$

   0101 $\notin C$ 0110 $\in C$

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w$ has equal numbers of 01 and 10 substrings$\}$

$0101 \notin C$ $0110 \in C$

Intuition: C is not regular because DFAs cannot count unboundedly.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- To show a language is *not* regular, we must give a proof.
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$

$0101 \notin C$ $\quad$ $0110 \in C$

Intuition: C is not regular because DFAs cannot count unboundedly.

However C is regular!

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w \text{ has equal numbers of 01 and 10 substrings}\}$

$0101 \notin C \qquad 0110 \in C$

Intuition: C is not regular because DFAs cannot count unboundedly.

However C is regular! <u>Sometimes intuition works, but it can also be wrong.</u>

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w \mid w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

2. Let $C = \{w \mid w$ has equal numbers of 01 and 10 substrings$\}$

$0101 \notin C$ $0110 \in C$

Intuition: C is not regular because DFAs cannot count unboundedly.

However C is regular! <u>Sometimes intuition works, but it can also be wrong.</u>

Moral: You need to give a proof.

Method for Proving Non-regularity

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ ____________________

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

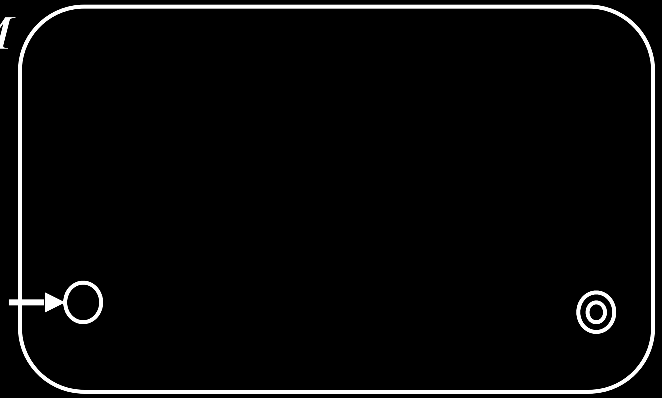
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ ______________________

M



Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

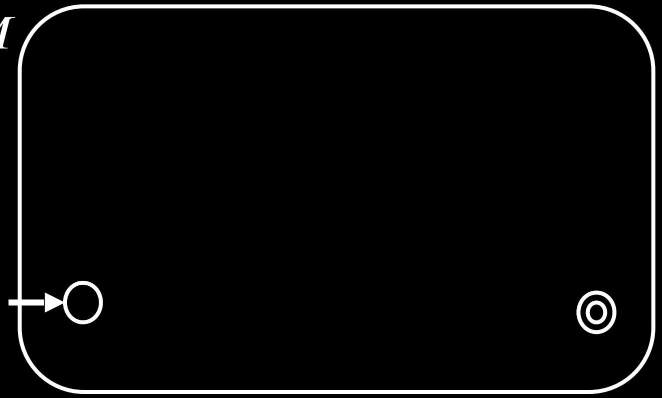
$$y^i = \underbrace{yy\cdots y}_{i}$$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ ______________________

M



M will repeat a state q_j when reading s because s is so long.

Method for Proving Non-regularity

Pumping Lemma: For every regular language A, there is a number p (the "pumping length") such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

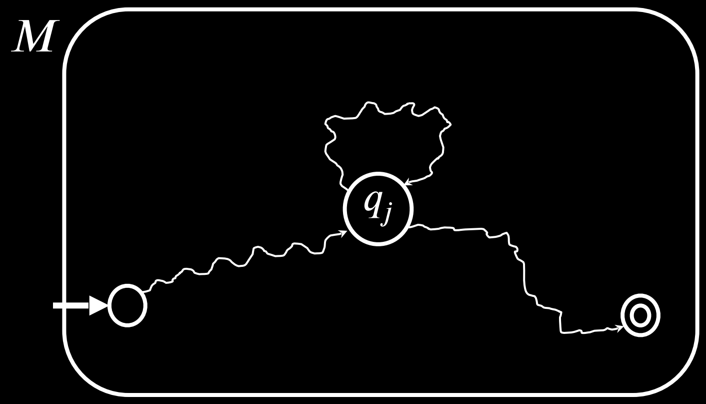
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ ____________________

M will repeat a state q_j when reading s because s is so long.



The path that M follows when reading s.

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

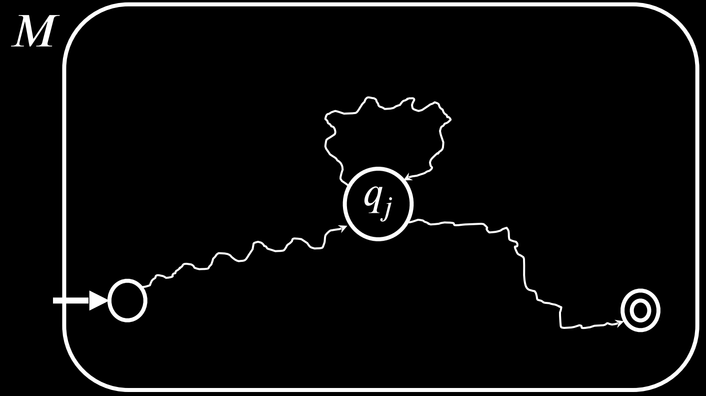
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$

M will repeat a state q_j when reading s because s is so long.



The path that M follows when reading s.

Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

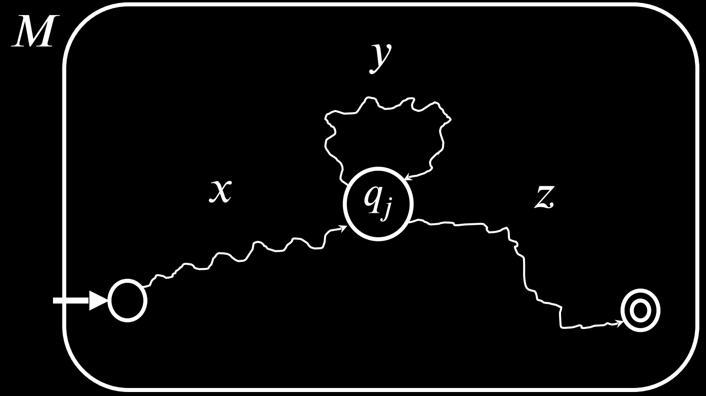
1) $xy^iz \in A$ for all $i \geq 0$ $y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$

M will repeat a state q_j when reading s because s is so long.



The path that M follows when reading s.

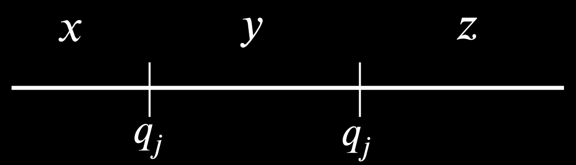
Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

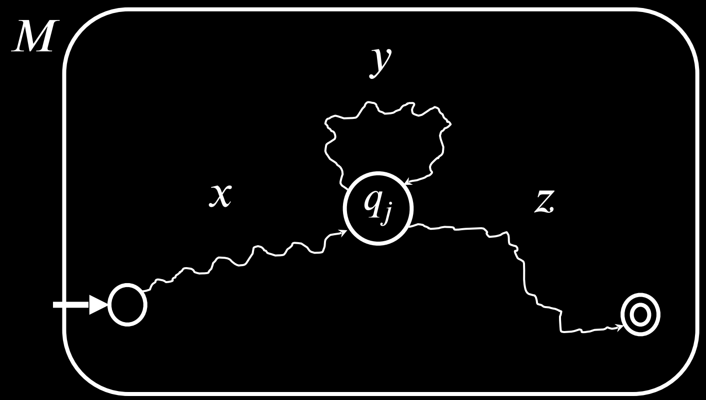
1) $xy^iz \in A$ for all $i \geq 0$ $y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ 

M will repeat a state q_j when reading s because s is so long.



The path that M follows when reading s.

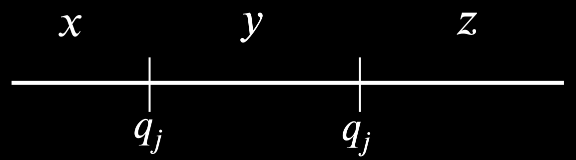
Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

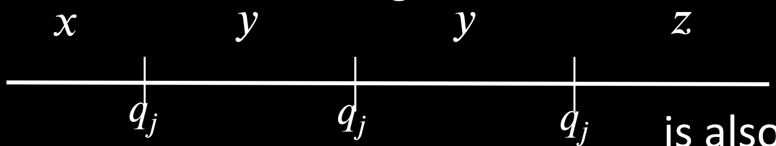
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

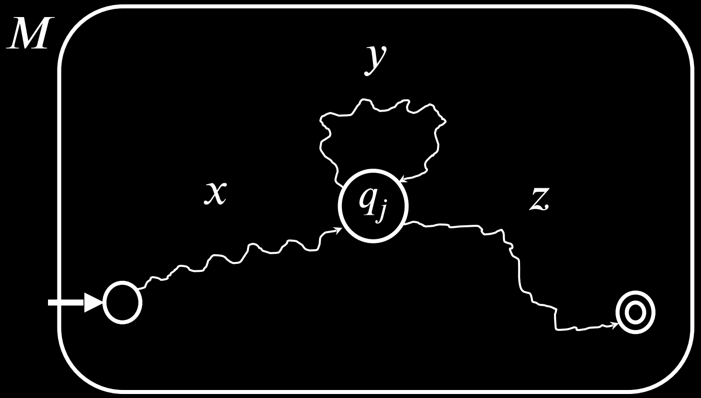
Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ 

M will repeat a state q_j when reading s because s is so long.



is also accepted



The path that M follows when reading s.

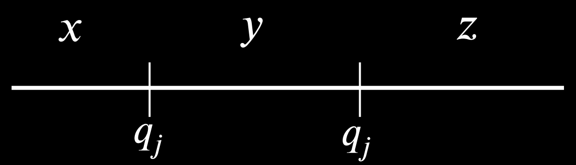
Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

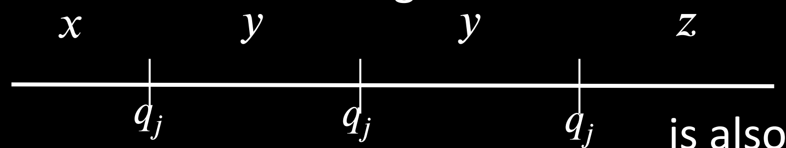
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

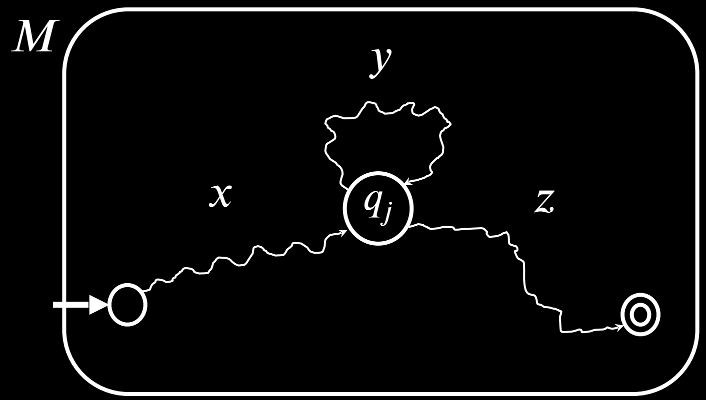
Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.

$s =$ 

M will repeat a state q_j when reading s because s is so long.



is also accepted



The path that M follows when reading s.

Check-in 3.2

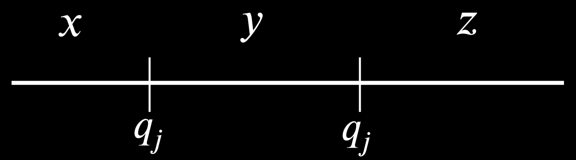
Method for Proving Non-regularity

Pumping Lemma: For every regular language A,
there is a number p (the "pumping length") such that
if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

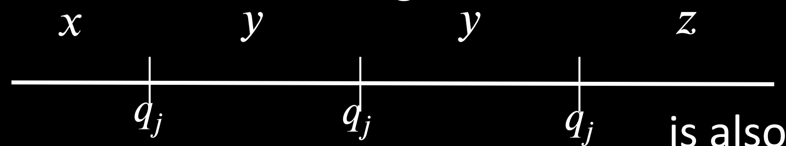
Informally: A is regular → every long string in A can be pumped and the result stays in A.

Proof: Let DFA M recognize A. Let p b

$s =$ 

M will repeat a state q_j when reading s

because s is so long.



is also accepted

Check-in 3.2

The Pumping Lemma depends on the fact that
if M has p states and it runs for more than p steps
then M will enter some state at least twice.
We call that fact:
(a) ?

s

Check-in 3.2

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Proof by Contradiction:

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.

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

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $D = \left\{0^k1^k \middle| \, k \geq 0\right\}$

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.

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

$s = \quad 000\cdots000111\cdots111111$

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^i z \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

$s = \quad 000\cdots000111\cdots111111$

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.

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

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

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Proof by Contradiction:
Assume (to get a contradiction) that D is regular.
The pumping lemma gives p as above. Let $s = 0^p 1^p \in D$.
Pumping lemma says that can divide $s = xyz$ satisfying the 3 conditions.

$s = 000 \cdots 000111 \cdots 111111$

$\quad x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D <u>is</u> regular.

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

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

$s = 000 \cdots 000111 \cdots 111111$

$x \quad y \quad z$

$\leftarrow \leq p \rightarrow$

But $xyyz$ has excess 0s and thus $xyyz \notin D$ contradicting the pumping lemma.

Therefore our assumption (D is regular) is false. We conclude that D is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

$s = \quad 000\cdots000000\cdots000$

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

$s = 000\cdots000000\cdots000$

$x \mid y \mid z$

$\leftarrow \leq p \longrightarrow$

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

$$s = \underbrace{000\cdots000000\cdots000}$$

$x \mid y \mid z$

$\leftarrow \leq p \longrightarrow \qquad y = 00$

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

$s = 000\cdots000000\cdots000$

$\underbrace{x}_{}\ \underbrace{y}_{}\ \underbrace{z}_{}$

$\leftarrow \leq p \longrightarrow \quad y = 00$

Proof by Contradiction:

Assume (for contradiction) that F <u>is</u> regular.

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

Try $s = 0^p0^p \in F$. But that s <u>can</u> be pumped and stay inside F. Bad choice.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$. But that s can be pumped and stay inside F. Bad choice.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

$s = \underbrace{000\cdots000000\cdots000}$

$x \mid y \mid z$

$\leftarrow \leq p \longrightarrow \qquad y = 00$

$s = 000\cdots001000\cdots001$

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\quad y^i = yy\cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

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

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

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

Try $s = 0^p0^p \in F$. But that s can be pumped and stay inside F. Bad choice.

Try $s = 0^p10^p1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

$s = 000\cdots000000\cdots000$

$x \mid y \mid z$

$\leftarrow \leq p \longrightarrow$ $\quad y = 00$

$s = 000\cdots001000\cdots001$

$x \mid y \mid z$

$\leftarrow \leq p \longrightarrow$

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B <u>is</u> regular.

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

But $D = B \cap 0^*1^*$ and we already showed D is not regular. Contradiction!

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w \mid w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

But $D = B \cap 0^*1^*$ and we already showed D is not regular. Contradiction!

Therefore our assumption is false, so B is not regular.

Context Free Grammars

Context Free Grammars

G_1

$$S \rightarrow 0S1$$

$$S \rightarrow R$$

$$R \rightarrow \varepsilon$$

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \to 0S1 \\ S \to R \\ R \to \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Context Free Grammars

G_1

$$\left.\begin{aligned} &S \rightarrow 0S1 \\ &S \rightarrow R \\ &R \rightarrow \varepsilon \end{aligned}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals

Context Free Grammars

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

(Substitution) Rules

Rule: Variable → string of variables and terminals

Variables: Symbols appearing on left-hand side of rule

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals

Variables: Symbols appearing on left-hand side of rule

Terminals: Symbols appearing only on right-hand side

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals

Variables: Symbols appearing on left-hand side of rule

Terminals: Symbols appearing only on right-hand side

Start Variable: Top left symbol

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Context Free Grammars

G_1

$S \rightarrow 0S1$
$S \rightarrow R$
$R \rightarrow \varepsilon$

$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\}$ (Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

Context Free Grammars

G_1

$$S \rightarrow 0S1$$
$$S \rightarrow R$$
$$R \rightarrow \varepsilon$$

(Substitution) Rules

Rule: Variable $\rightarrow$ string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \to 0S1 \\ S \to R \\ R \to \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule

   Repeat until only terminals remain

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals

Variables: Symbols appearing on left-hand side of rule

Terminals: Symbols appearing only on right-hand side

Start Variable: Top left symbol

In G_1:

3 rules

R,S

0,1

S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:
3 rules
R,S
0,1
S

Example of G_1 generating a string

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \to 0S1 \\ S \to R \\ R \to \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Example of G_1 generating a string

S S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

ε

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:
3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions

S

S

Resulting string

ε

Context Free Grammars

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

(Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

In G_1:

3 rules
R,S
0,1
S

Example of G_1 generating a string

Tree of substitutions		Resulting string
S	S	
0 S 1	0S1	

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\}$$ (Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions		Resulting string
S	S	
0 S 1	0S1	
0 S 1	00S11	
ε		

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\}$$ (Substitution) Rules

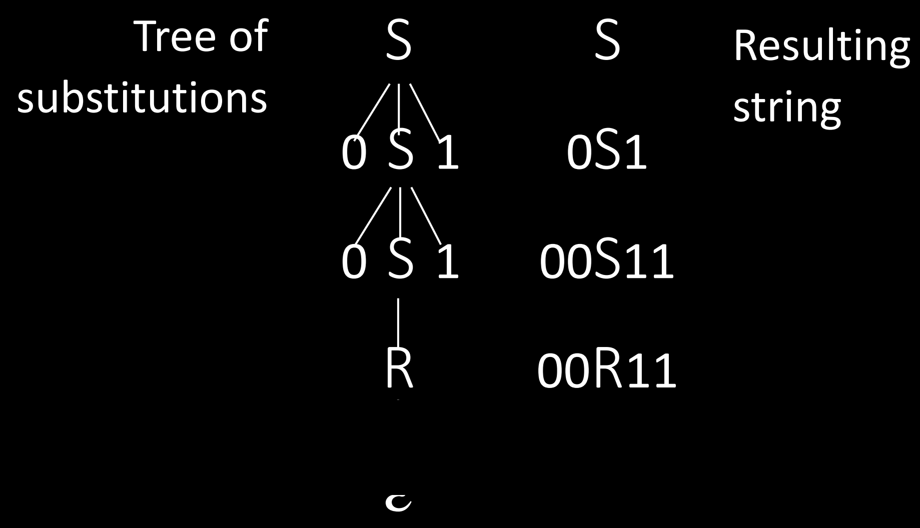
Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:
3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string



Context Free Grammars

G_1

$$S \rightarrow 0S1$$
$$S \rightarrow R$$
$$R \rightarrow \varepsilon$$

(Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions		Resulting string
S	S	
0 S 1	0S1	
0 S 1	00S11	
R	00R11	
ε	0011	

Context Free Grammars

G_1

$S \rightarrow 0S1$
$S \rightarrow R$
$R \rightarrow \varepsilon$

(Substitution) Rules

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

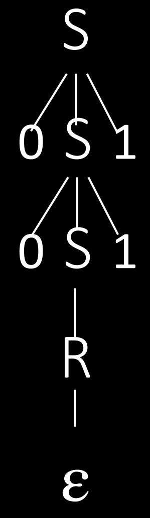
3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions



Resulting string

S
0S1
00S11
00R11
0011 $\in L(G_1)$

Context Free Grammars

G_1

$$\left.\begin{aligned} S &\to 0S1 \\ S &\to R \\ R &\to \varepsilon \end{aligned}\right\} \text{(Substitution) Rules}$$

Rule: Variable $\to$ string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions		Resulting string
	S	S
	0 S 1	0S1
	0 S 1	00S11
	R	00R11
	ε	0011 $\in L(G_1)$

$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

Context Free Grammars

G_1

$S \rightarrow 0S1$
$S \rightarrow R$
$R \rightarrow \varepsilon$

$\Big\}$ (Substitution) Rules

Rule: Variable $\rightarrow$ string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

In G_1:

3 rules
R,S
0,1
S

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Example of G_1 generating a string

Tree of substitutions		Resulting string
S	S	
0 S 1	0S1	
0 S 1	00S11	
R	00R11	
ε	0011 $\in L(G_1)$	

$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

Check-in 3.3

Context Free Grammars

G_1

$$\left.\begin{array}{l} S \rightarrow 0S1 \\ S \rightarrow R \\ R \rightarrow \varepsilon \end{array}\right\} \text{(Substitution) Rules}$$

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

Check-in 3.3

G_2

$$\begin{array}{l} S \rightarrow RR \\ R \rightarrow 0R1 \\ R \rightarrow \varepsilon \end{array}$$

Check all of the strings that are in $L(G_2)$:

(a) 001101
(b) 000111
(c) 1010
(d) ε

Check-in 3.3

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

Context Free Grammars (CFGs)

G_1

$$S \rightarrow 0S1$$
$$S \rightarrow R$$
$$R \rightarrow \varepsilon$$

Shorthand:

$$S \rightarrow 0S1 \mid R$$
$$R \rightarrow \varepsilon$$

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

Example of G_1 generating a string

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

Example of G_1 generating a string

S S

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"	S	S	Resulting string

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"	Resulting string
S	S
0 S 1	0S1

Context Free Grammars (CFGs)

G_1

$$S \rightarrow 0S1$$
$$S \rightarrow R$$
$$R \rightarrow \varepsilon$$

Shorthand:

$$S \rightarrow 0S1 \mid R$$
$$R \rightarrow \varepsilon$$

Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"		Resulting string
	S	S
	0 S 1	0S1
	0 S 1	00S11

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

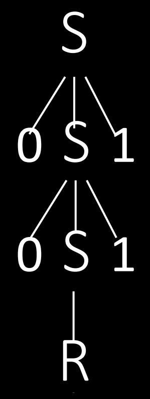
Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"



Resulting string

S

0S1

00S11

00R11

Context Free Grammars (CFGs)

G_1

S → 0S1

S → R

R → ε

Shorthand:

S → 0S1 | R

R → ε

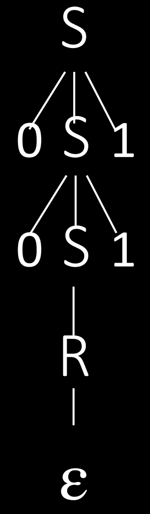
Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"



Resulting string

S

0S1

00S11

00R11

0011

Context Free Grammars (CFGs)

G_1

$S \rightarrow 0S1$

$S \rightarrow R$

$R \rightarrow \varepsilon$

Shorthand:

$S \rightarrow 0S1 \mid R$

$R \rightarrow \varepsilon$

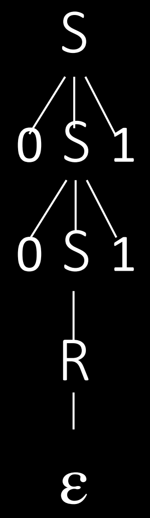
Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"



Resulting string

S

0S1

00S11

00R11

0011 $\in L(G_1)$

Context Free Grammars (CFGs)

G_1

$S \to 0S1$

$S \to R$

$R \to \varepsilon$

Shorthand:

$S \to 0S1 \mid R$

$R \to \varepsilon$

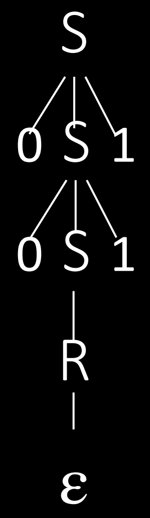
Recall that a CFG has terminals, variables, and rules.

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings
5. We call $L(G)$ a Context Free Language.

Example of G_1 generating a string

Tree of substitutions "parse tree"



Resulting string

S

0S1

00S11

00R11

0011 $\in L(G_1)$

$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

CFG – Formal Definition

CFG – Formal Definition

Defn: A Context Free Grammar (CFG) G is a 4-tuple (V, Σ, R, S)

V finite set of variables

Σ finite set of terminal symbols

R finite set of rules (rule form: $V \to (V \cup \Sigma)^*$)

S start variable

For $u, v \in (V \cup \Sigma)^*$ write

1) $u \Rightarrow v$ if can go from u to v with one substitution step in G

2) $u \overset{*}{\Rightarrow} v$ if can go from u to v with some number of substitution steps in G

$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k = v$ is called a derivation of v from u.

If $u = S$ then it is a derivation of v.

$$L(G) = \{w \mid w \in \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$$

Defn: A is a Context Free Language (CFL) if $A = L(G)$ for some CFG G.

CFG – Formal Definition

Defn: A <u>Context Free Grammar</u> (CFG) G is a 4-tuple (V, Σ, R, S)

V finite set of variables

Σ finite set of terminal symbols

R finite set of rules (rule form: $V \rightarrow (V \cup \Sigma)^*$)

S start variable

For $u, v \in (V \cup \Sigma)^*$ write

1) $u \Rightarrow v$ if can go from u to v with one substitution step in G

2) $u \overset{*}{\Rightarrow} v$ if can go from u to v with some number of substitution steps in G

$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k = v$ is called a derivation of v from u.

If $u = S$ then it is a <u>derivation</u> of v.

$L(G) = \{w \mid w \in \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$

Defn: A is a <u>Context Free Language</u> (CFL) if $A = L(G)$ for some CFG G.

Check-in 4.1

CFG – Formal Definition

Defn: A Context Free Grammar (CFG) G is a 4-tuple (V, Σ, R, S)

V finite set of variables

Σ finite set of terminal symbols

R finite set of rules (rule form: $V \to (V \cup \Sigma)^*$)

S start variable

For $u, v \in (V \cup \Sigma)^*$ write

1) $u \Rightarrow v$ if can go from u to v with one substitution step in

2) $u \overset{*}{\Rightarrow} v$ if can go from u to v with some number of substit

$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k = v$ is called a derivat

If $u = S$ then it is a derivation of v.

$L(G) = \{w \mid w \in \Sigma^* \text{ and } S \overset{*}{\Rightarrow} w\}$

Defn: A is a Context Free Language (CFL) if $A = L(G)$ for

Check-in 4.1

Which of these are valid CFGs?

C_1:
B → 0B1 | ε
B1 → 1B
0B → 0B

C_2:
S → 0S | S1
R → RR

a) C_1 only

b) C_2 only

c) Both C_1 and C_2

d) Neither

CFG – Example

G_2

$$E \rightarrow E{+}T \mid T$$

$$T \rightarrow T{\times}F \mid F$$

$$F \rightarrow (\,E\,) \mid a$$

$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

CFG – Example

G_2

$$\mathrm{E} \longrightarrow \mathrm{E{+}T} \mid \mathrm{T}$$

$$\mathrm{T} \longrightarrow \mathrm{T{\times}F} \mid \mathrm{F}$$

$$\mathrm{F} \longrightarrow (\ \mathrm{E}\ ) \mid \mathrm{a}$$

$V =$ {E, T, F}
$\Sigma =$ {+, ×, (,), a}
$R =$ the 6 rules above
$S =$ E

Generates a+a×a

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree

E

E

Resulting string

Generates a+a×a

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

Parse tree

$$\begin{array}{ccc} & E & \\ / & | & \backslash \\ E & + & T \end{array}$$

Resulting string

E

E+T

$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Generates a+a×a

CFG – Example

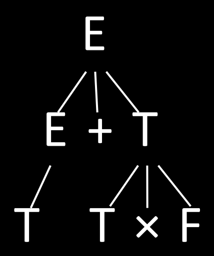
G_2

E → E+T | T

T → T×F | F

F → (E) | a

Parse tree



E

E+T

T+T×F

Resulting string

$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Generates a+a×a

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

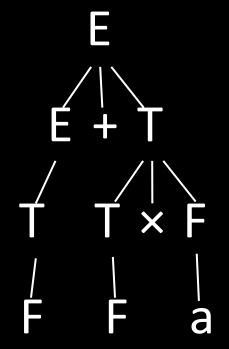
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



E

E+T

T+T×F

F+F×a

Resulting string

Generates a+a×a

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

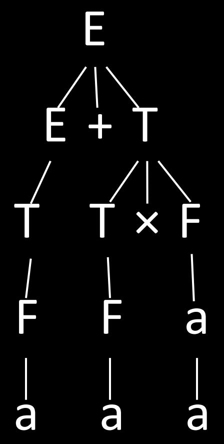
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



Generates a+a×a

E	Resulting string
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

CFG – Example

G_2

$$E \to E{+}T \mid T$$

$$T \to T{\times}F \mid F$$

$$F \to (\,E\,) \mid a$$

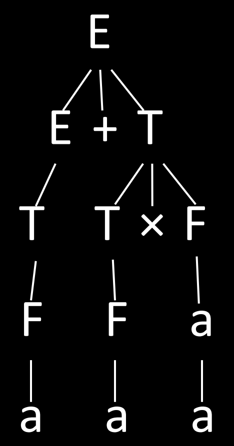
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



	Resulting string
E	
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

Generates a+a×a, (a+a)×a, a, a+a+a, etc.

CFG – Example

G_2

$$E \longrightarrow E+T \mid T$$

$$T \longrightarrow T\times F \mid F$$

$$F \longrightarrow (\,E\,) \mid a$$

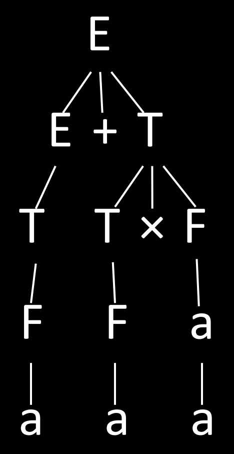
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



	Resulting string
E	
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

Generates a+a×a, (a+a)×a, a, a+a+a, etc.

Observe that the parse tree contains additional information, such as the precedence of × over + .

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

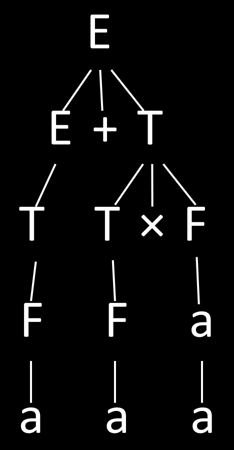
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



E	Resulting string
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

Generates a+a×a, (a+a)×a, a, a+a+a, etc.

Observe that the parse tree contains additional information, such as the precedence of × over + .

If a string has two different parse trees then it is derived ambiguously and we say that the grammar is <u>ambiguous</u>.

CFG – Example

G_2

E → E+T | T

T → T×F | F

F → (E) | a

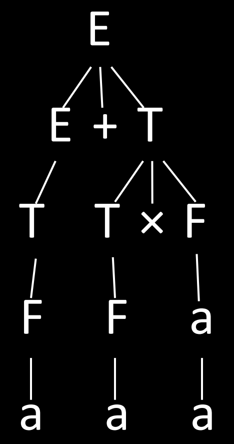
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



E	Resulting string
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

Generates a+a×a, (a+a)×a, a, a+a+a, etc.

Observe that the parse tree contains additional information, such as the precedence of × over + .

If a string has two different parse trees then it is derived ambiguously and we say that the grammar is <u>ambiguous</u>.

Check-in 4.2

CFG – Example

G_2

E $\rightarrow$ E+T | T

T $\rightarrow$ T×F | F

F $\rightarrow$ (E) | a

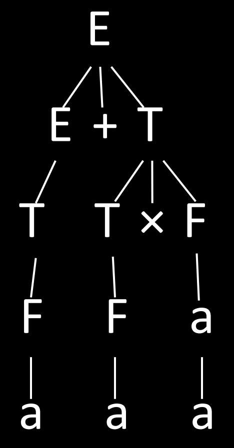
$V =$ {E, T, F}

$\Sigma =$ {+, ×, (,), a}

$R =$ the 6 rules above

$S =$ E

Parse tree



E	Resulting string
E+T	
T+T×F	
F+F×a	
a+a×a	$\in L(G_2)$

Generates a+a×a, (a+

Observe that the parse tree contains additional informatio
such as the precedence of × over + .

If a string has two different parse trees then it is derived a
and we say that the grammar is <u>ambiguous</u>.

Check-in 4.2

How many reasonable distinct meanings does the following English sentence have?

The boy saw the girl with the mirror.

(a) 1

(b) 2

(c) 3 or more

Ambiguity

G_2

$E \rightarrow E{+}T \mid T$

$T \rightarrow T{\times}F \mid F$

$F \rightarrow (\,E\,) \mid a$

G_3

$E \rightarrow E{+}E \mid E{\times}E \mid (\,E\,) \mid a$

Ambiguity

G_2

E $\rightarrow$ E+T | T

T $\rightarrow$ T×F | F

F $\rightarrow$ (E) | a

G_3

E $\rightarrow$ E+E | E×E | (E) | a

Both G_2 and G_3 recognize the same language, i.e., $L(G_2) = L(G_3)$.

Ambiguity

G_2

E $\longrightarrow$ E+T | T

T $\longrightarrow$ T×F | F

F $\longrightarrow$ (E) | a

G_3

E $\longrightarrow$ E+E | E×E | (E) | a

Both G_2 and G_3 recognize the same language, i.e., $L(G_2) = L(G_3)$.

However G_2 is an unambiguous CFG and G_3 is ambiguous.

Ambiguity

G_2

$$E \longrightarrow E+T \mid T$$

$$T \longrightarrow T \times F \mid F$$

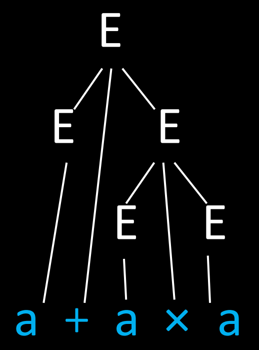
$$F \longrightarrow (\, E \,) \mid a$$

G_3

$$E \longrightarrow E+E \mid E \times E \mid (\, E \,) \mid a$$

Both G_2 and G_3 recognize the same language, i.e., $L(G_2) = L(G_3)$.

However G_2 is an unambiguous CFG and G_3 is ambiguous.



Ambiguity

G_2

E $\longrightarrow$ E+T | T

T $\longrightarrow$ T×F | F

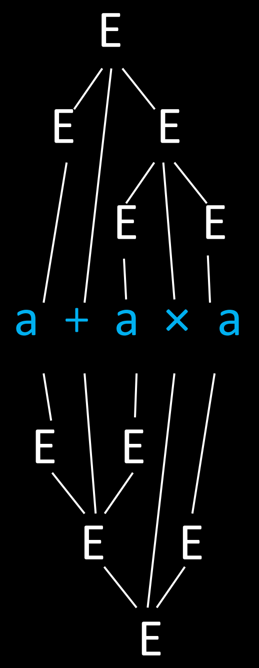
F $\longrightarrow$ (E) | a

G_3

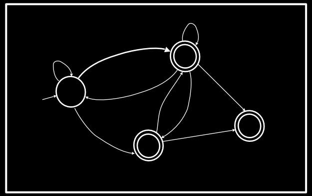
E $\longrightarrow$ E+E | E×E | (E) | a

Both G_2 and G_3 recognize the same language, i.e., $L(G_2) = L(G_3)$.

However G_2 is an unambiguous CFG and G_3 is ambiguous.



Pushdown Automata (PDA)

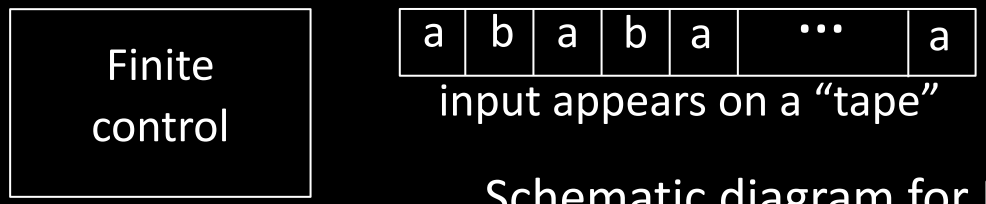


Schematic diagram for DFA or NFA

Pushdown Automata (PDA)

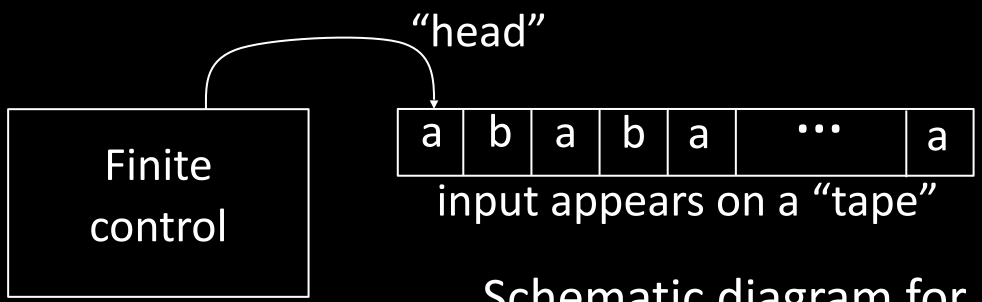
Schematic diagram for DFA or NFA

Pushdown Automata (PDA)



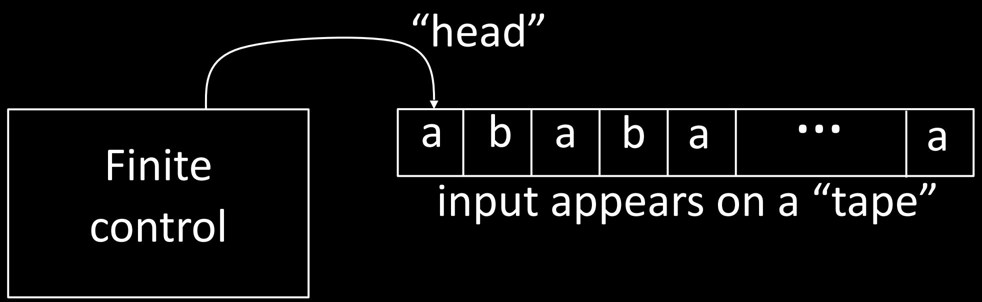
Schematic diagram for DFA or NFA

Pushdown Automata (PDA)



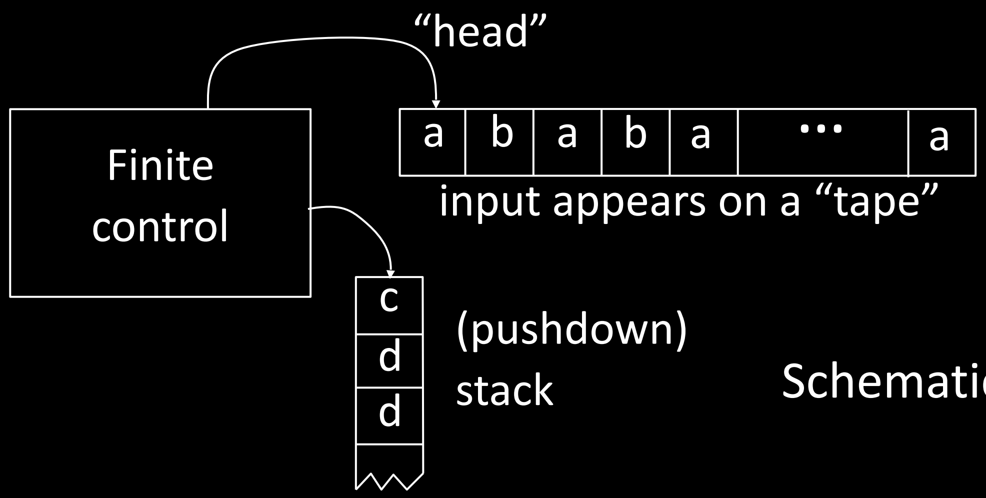
Schematic diagram for DFA or NFA

Pushdown Automata (PDA)



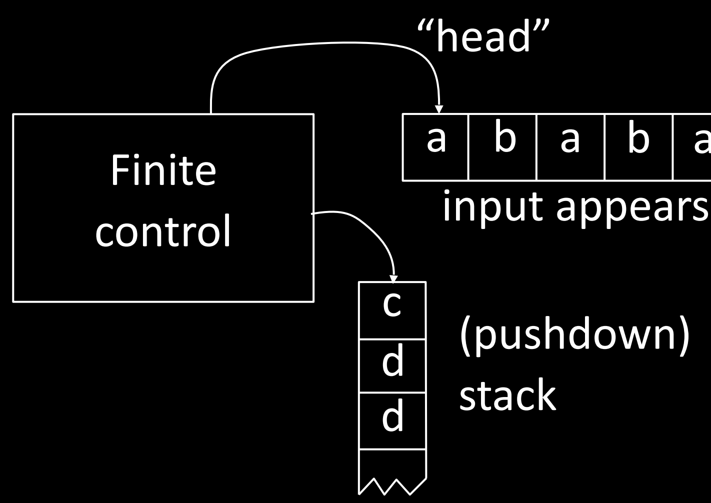
Schematic diagram for PDA

Pushdown Automata (PDA)



Schematic diagram for PDA

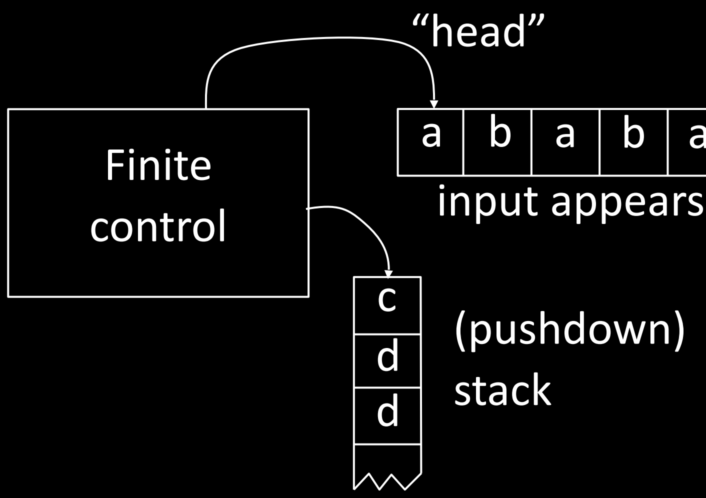
Pushdown Automata (PDA)



Schematic diagram for PDA

Operates like an NFA except can <u>write-add</u> or <u>read-remove</u> symbols from the top of stack.

Pushdown Automata (PDA)

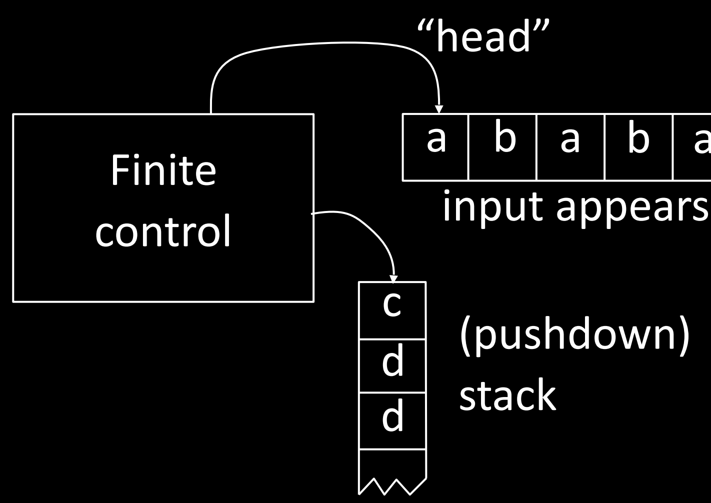


Schematic diagram for PDA

Operates like an NFA except can <u>write-add</u> or <u>read-remove</u> symbols from the top of stack.

push   pop

Pushdown Automata (PDA)



Schematic diagram for PDA

Operates like an NFA except can write-add or read-remove symbols from the top of stack.

push pop

Example: PDA for $D = \left\{ 0^k 1^k \middle|\ k \geq 0 \right\}$

1) Read 0s from input, push onto stack until read 1.
2) Read 1s from input, while popping 0s from stack.
3) Enter accept state if stack is empty. (note: acceptance only at end of input)

PDA – Formal Definition

PDA – Formal Definition

Defn: A Pushdown Automaton (PDA) is a 6-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q0,\ F)$

Σ input alphabet

Γ stack alphabet

δ: $\mathrm{Q} \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathscr{P}(Q \times \Gamma_\varepsilon)$

$$\delta(q, \mathrm{a}, \mathrm{c}) = \left\{ (r_1, \mathrm{d}),\ \ (r_2,\ \mathrm{e}) \right\}$$

PDA – Formal Definition

Defn: A Pushdown Automaton (PDA) is a 6-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q0,\ F)$

Σ input alphabet

Γ stack alphabet

δ: $\mathrm{Q} \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathscr{P}(Q \times \Gamma_\varepsilon)$

$$\delta(q, \mathrm{a}, \mathrm{c}) = \left\{ (r_1, \mathrm{d}),\ \ (r_2,\ \mathrm{e}) \right\}$$

Accept if some thread is in the accept state at the end of the input string.

PDA – Formal Definition

Defn: A <u>Pushdown Automaton</u> (PDA) is a 6-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q0,\ F)$

Σ input alphabet

Γ stack alphabet

δ: $\mathrm{Q} \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathscr{P}(Q \times \Gamma_\varepsilon)$

Accept if some thread is in the accept state at the end of the input string.

$$\delta(q, \mathrm{a}, \mathrm{c}) = \left\{ (r_1, \mathrm{d}),\ \ (r_2,\ \mathrm{e}) \right\}$$

Example: PDA for $B = \{ww^{\mathscr{R}} \mid w \in \{0,1\}^*\}$

1) Read and push input symbols.
   Nondeterministically either repeat or go to (2).
2) Read input symbols and pop stack symbols, compare.
   If ever $\neq$ then thread rejects.
3) Enter accept state if stack is empty. (do in "software")

PDA – Formal Definition

Defn: A Pushdown Automaton (PDA) is a 6-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q0,\ F)$

Σ input alphabet

Γ stack alphabet

δ: $\mathrm{Q} \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathscr{P}(Q \times \Gamma_\varepsilon)$

$$\delta(q, \mathrm{a}, \mathrm{c}) = \left\{ (r_1, \mathrm{d}),\ \ (r_2,\ \mathrm{e}) \right\}$$

Accept if some thread is in the accept state at the end of the input string.

Example: PDA for $B = \{ww^{\mathscr{R}} \mid w \in \{0{,}1\}^*\ \}$ Sample input:

0	1	1	1	1	0

1) Read and push input symbols.
   Nondeterministically either repeat or go to (2).
2) Read input symbols and pop stack symbols, compare.
   If ever $\neq$ then thread rejects.
3) Enter accept state if stack is empty. (do in “software”)

PDA – Formal Definition

Defn: A Pushdown Automaton (PDA) is a 6-tuple

$(Q,\ \Sigma,\ \Gamma,\ \delta,\ q0,\ F)$

Σ input alphabet

Γ stack alphabet

δ: $\mathrm{Q} \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathscr{P}(Q \times \Gamma_\varepsilon)$

$$\delta(q, \mathrm{a}, \mathrm{c}) = \left\{ (r_1, \mathrm{d}),\ \ (r_2,\ \mathrm{e}) \right\}$$

Accept if some thread is in the accept state at the end of the input string.

Example: PDA for $B = \{ww^{\mathscr{R}} \mid w \in \{0,1\}^*\}$

Sample input:

0	1	1	1	1	0

1) Read and push input symbols.
   Nondeterministically either repeat or go to (2).
2) Read input symbols and pop stack symbols, compare.
   If ever $\neq$ then thread rejects.
3) Enter accept state if stack is empty. (do in "software")

The nondeterministic forks replicate the stack.

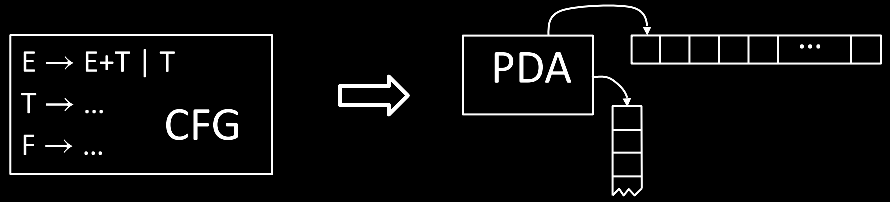
This language requires nondeterminism.
Our PDA model is nondeterministic.

Converting CFGs to PDAs

Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.

Problem! Access below the top of stack is cheating!

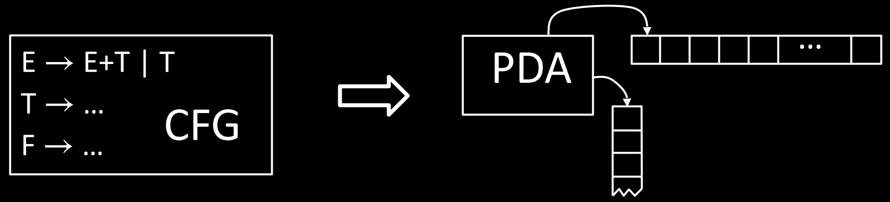
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.

G_2

E → E+T | T

T → T×F | F

F → (E) | a

Input:

a	+	a	×	a

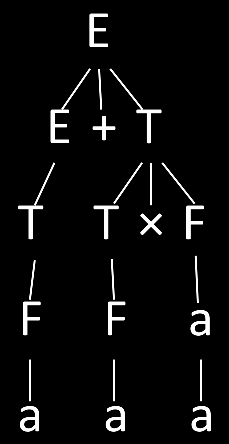
E

E+T

T+T×F

F+F×a

a+a×a



Problem! Access below the top of stack is cheating!

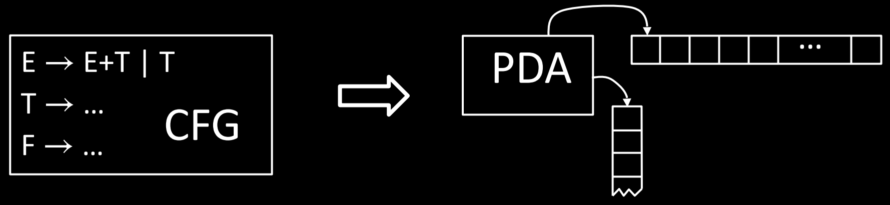
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



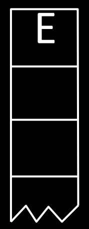
G_2

E → E+T | T

T → T×F | F

F → (E) | a

IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.



Input:

a	+	a	×	a

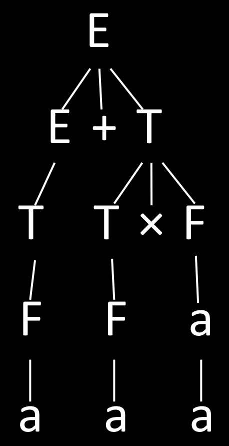
E

E+T

T+T×F

F+F×a

a+a×a



Problem! Access below the top of stack is cheating!

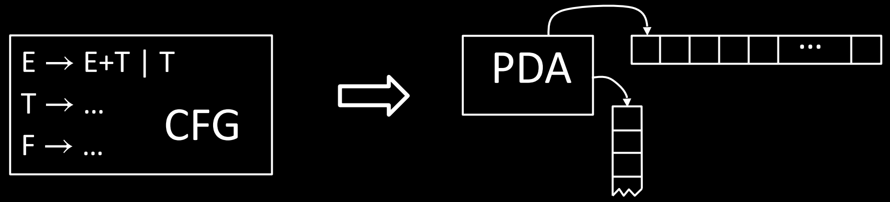
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



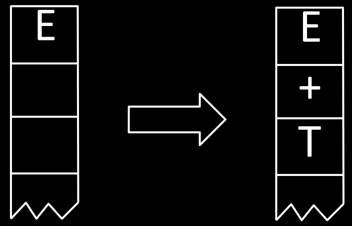
G_2

E → E+T | T

T → T×F | F

F → (E) | a

IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.



Input: a + a × a

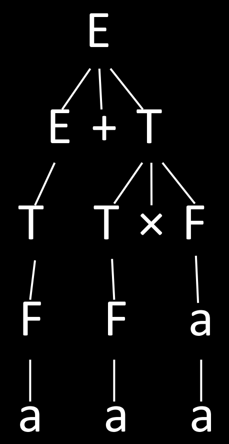
E

E+T

T+T×F

F+F×a

a+a×a



Problem! Access below the top of stack is cheating!

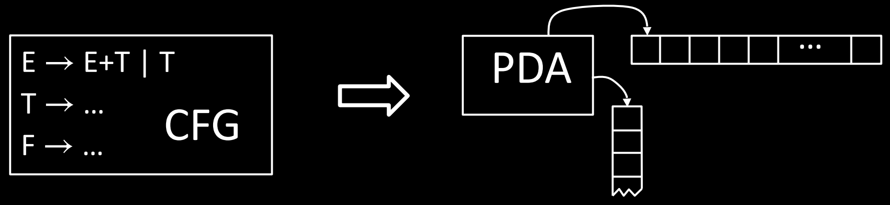
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

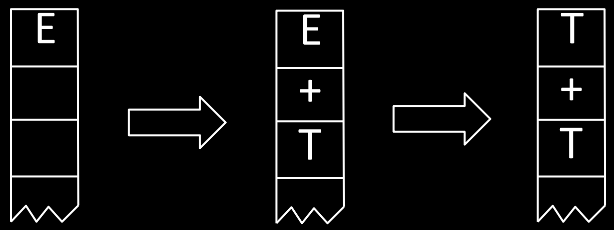
Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.



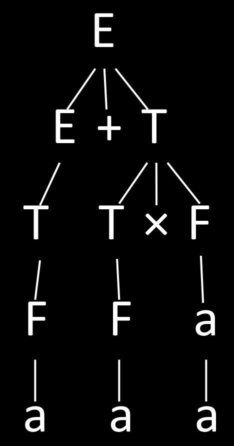
Input:

a	+	a	×	a

G_2

E → E+T | T
T → T×F | F
F → (E) | a

E
E+T
T+T×F
F+F×a
a+a×a



Problem! Access below the top of stack is cheating!

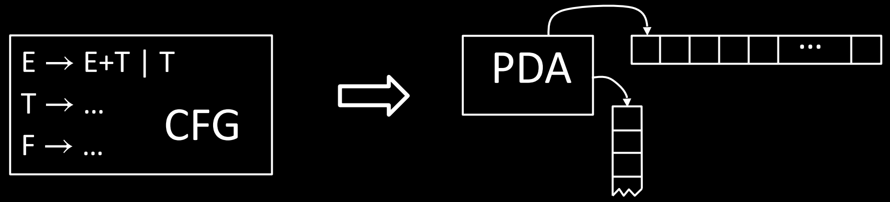
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



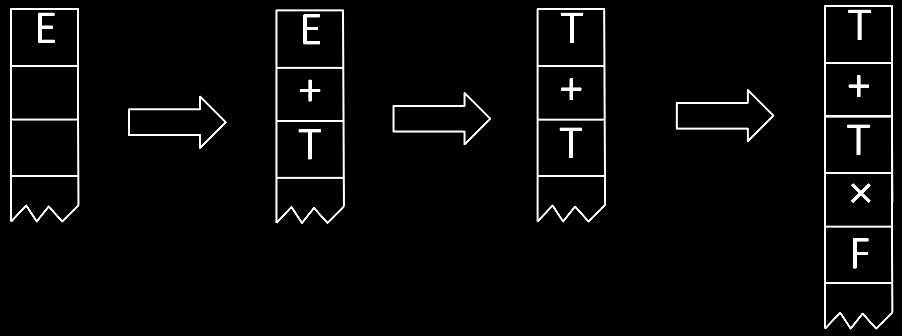
G_2

E → E+T | T

T → T×F | F

F → (E) | a

IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.



Input: a + a × a

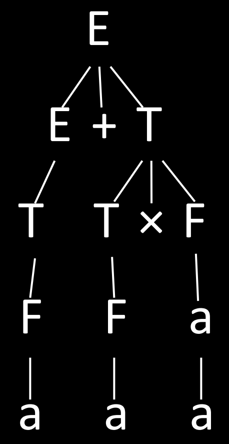
E

E+T

T+T×F

F+F×a

a+a×a



Problem! Access below the top of stack is cheating!

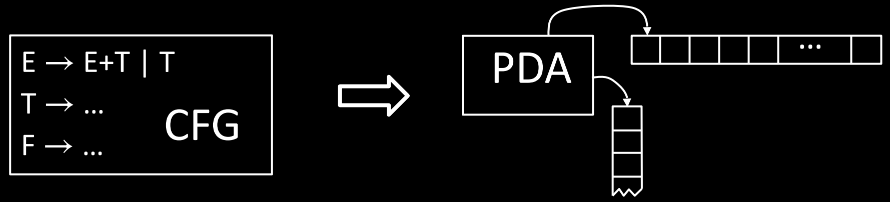
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

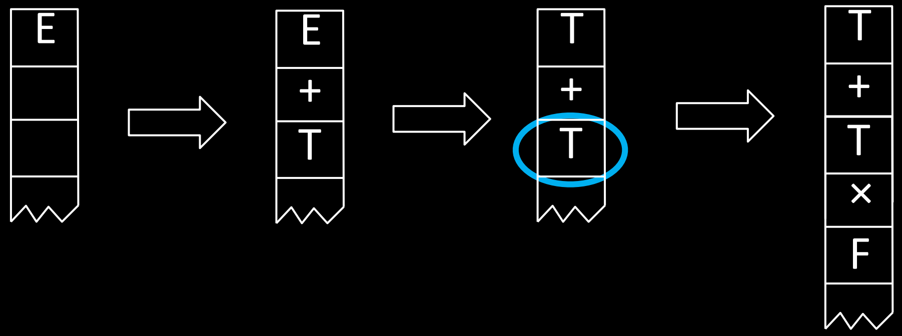
Converting CFGs to PDAs

Theorem: If A is a CFL then some PDA recognizes A

Proof: Convert A's CFG to a PDA



IDEA: PDA begins with starting variable and guesses substitutions.
It keeps intermediate generated strings on stack. When done, compare with input.



Input:

a	+	a	×	a

G_2

E → E+T | T

T → T×F | F

F → (E) | a

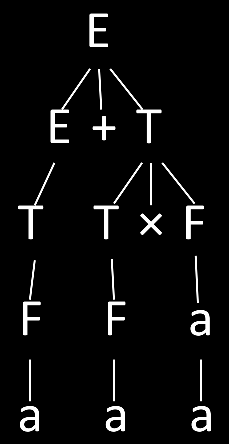
E

E+T

T+T×F

F+F×a

a+a×a



Problem! Access below the top of stack is cheating!

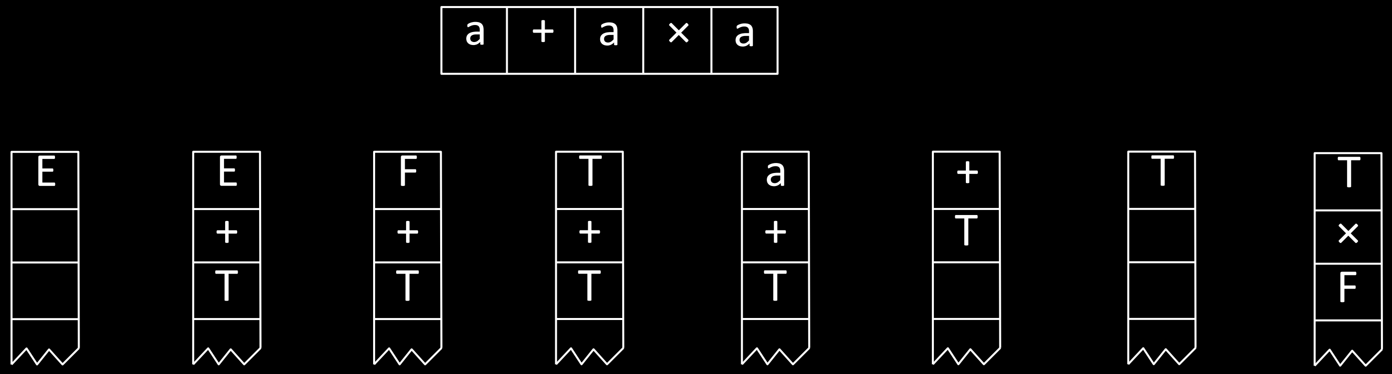
Instead, only substitute variables when on the top of stack.

If a terminal is on the top of stack, pop it and compare with input. Reject if $\neq$.

Converting CFGs to PDAs (contd)

G_2

$$E \to E+T \mid T$$
$$T \to T\times F \mid F$$
$$F \to (\,E\,) \mid a$$



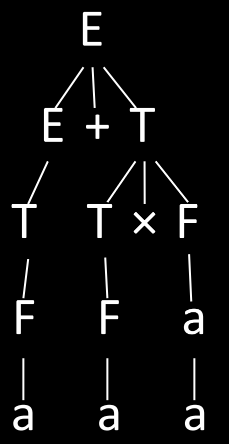
E

E+T

T+T×F

F+F×a

a+a×a



Converting CFGs to PDAs (contd)

G_2

E → E+T | T

T → T×F | F

F → (E) | a

Theorem: If A is a CFL then some PDA recognizes A

Proof construction: Convert the CFG for A to the following PDA.

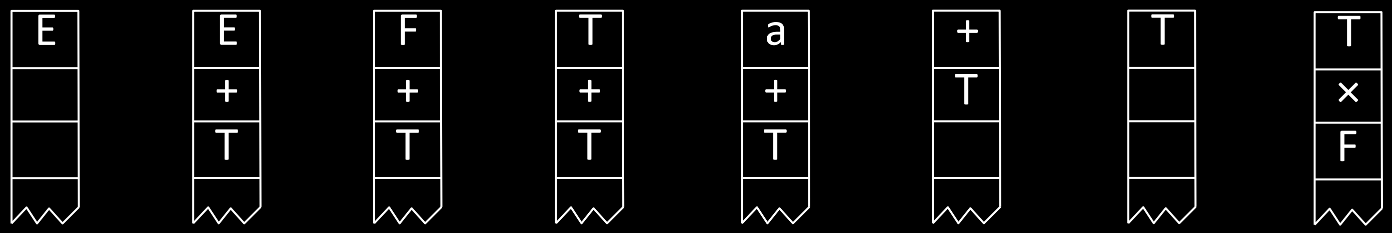
1) Push the start symbol on the stack.
2) If the top of stack is

   Variable: replace with right hand side of rule (nondet choice).

   Terminal: pop it and match with next input symbol.
3) If the stack is empty, *accept.*

Example:

a	+	a	×	a



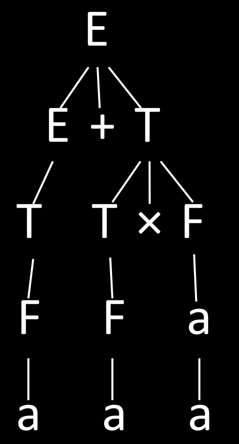
E

E+T

T+T×F

F+F×a

a+a×a



Equivalence of CFGs and PDAs

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

Equivalence of CFGs and PDAs

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

Done.

Equivalence of CFGs and PDAs

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

$\longleftrightarrow$ Done.

Equivalence of CFGs and PDAs

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

$\longleftrightarrow$ Done.

In book. You are responsible for knowing it is true, but not for knowing the proof.

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

* “iff” = “if an only if” means the implication goes both ways.

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

* "iff" = "if an only if" means the implication goes both ways.
So we need to prove both directions: forward ($\rightarrow$) and reverse ($\leftarrow$).

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

* "iff" = "if an only if" means the implication goes both ways.
So we need to prove both directions: forward ($\rightarrow$) and reverse ($\leftarrow$).

Check-in **
Do you know the proof of
PDA —> CFG

(a) Yes

(b) No

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

* "iff" = "if an only if" means the implication goes both ways.
So we need to prove both directions: forward ($\rightarrow$) and reverse ($\leftarrow$).

Check-in **
Do you know the proof of
PDA —> CFG

(a) Yes

(b) No

Check-in 4.3

Equivalence of CFGs and PDAs

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

$\longrightarrow$ Done.

$\longleftarrow$ In book. You are responsible for knowing it is true, but not for knowing the proof.

* "iff" = "if an only if" means the implication goes both ways.
So we need to prove both directions: forward ($\rightarrow$) and reverse ($\leftarrow$).

Check-in **
Do you know the proof of
PDA —> CFG

(a) Yes
(b) No

Check-in 4.3
Is every Regular Language also a Context Free Language?

(a) Yes
(b) No
(c) Not sure

Check-in 4.3

Recap

	Recognizer	Generator
Regular language	DFA or NFA	Regular expression
Context Free language	PDA	Context Free Grammar