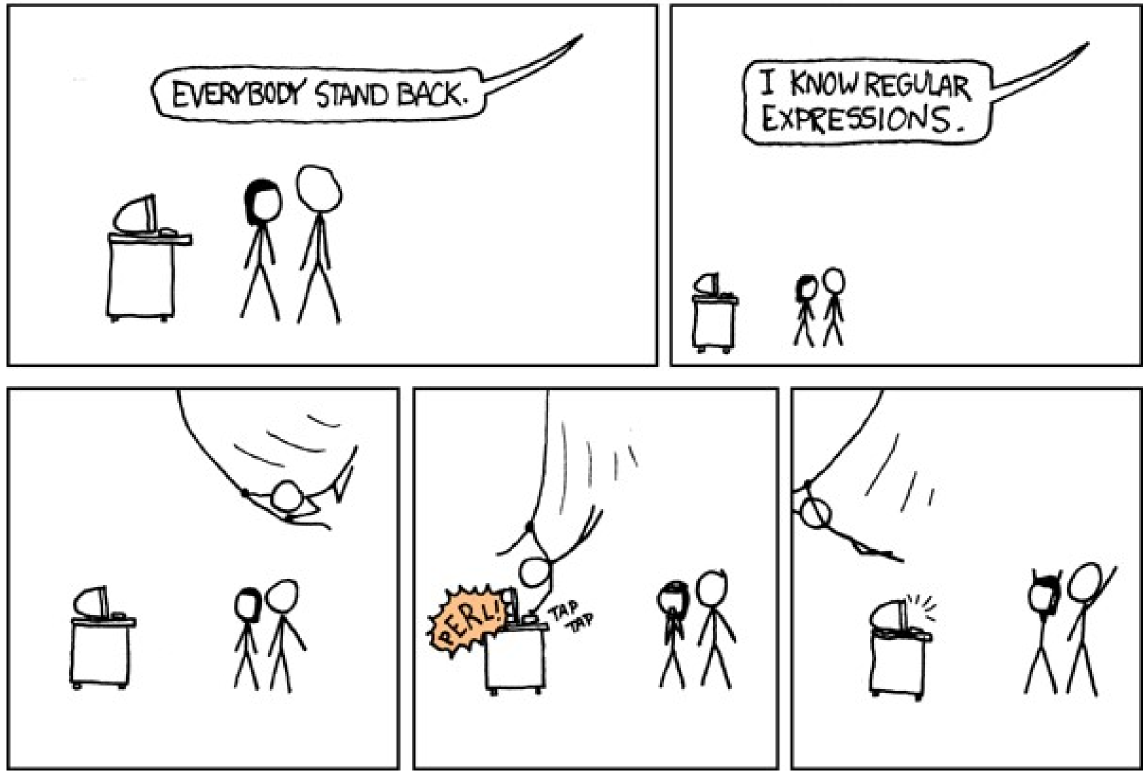


Regular Expressions, Context Free Grammars

CSE 311: Foundations of Computing I
Lecture 19

Announcements

- Homework 6 is due Friday 11:59pm
- Midterm grades tomorrow, 7/30 evening

Theoretical Computer Science

Recall: Course Goals

1. Learn to make & clearly communicate rigorous formal arguments
   - Mathematical Proofs
2. Understand mathematical objects that are widely used in CS
   - Number Theory, Set Theory, Recursively-Defined Functions
3. Explore and analyze models of computation
   - Regular Expressions, Context-Free Grammars, Finite Automata

Languages

Definition:
A language is a set of strings.

For Example:

- “The set of all valid English sentences”
- “The set of all binary strings of even length”
- “The set of all syntactically correct Java programs”

Languages in Theoretical Computer Science

- We want to study different models of computation, and the strengths & limitations of each.
- A computer is said to recognize a language if it can distinguish which strings are in a language vs. which are not.
- One way to evaluate how powerful a model of computation is is to determine which languages it can recognize.

Regular Languages

One class of languages

Regular Expressions

Basis Step:

- ε is a regular expression
- a is a regular expression for any $a \in \Sigma$

Recursive Step: If A and B are regular expressions, then…

- $A \cup B$ is a regular expression
- AB is a regular expression
- A^*

Regular Expressions

Each regular expression **matches** a set of strings (a language).

ε matches only the empty string

a matches only the single-character string a

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

Can Regex express any language?

All binary strings with <u>an equal number </u>of 0's and 1's

Can Regex represent any pattern?

All binary strings with an equal number of 0's and 1's

Does this work? (01 U 10)*

🤔

Can Regex represent any pattern?

All binary strings with <u>an equal number</u> of 0's and 1's

Does this work? (01 U 10)*

Can Regex represent any pattern?

All binary strings with an equal number of 0's and 1's

Does this work? (01 U 10)*

Sometimes! 010110

But what about: 0011

Can Regex represent any pattern?

All binary strings with an equal number of 0's and 1's

Does this work? (01 U 10)*

Sometimes! 010110

But what about: 0011

Would this cover that case: 0* 1*

Can Regex represent any pattern?

All binary strings with an equal number of 0's and 1's

Does this work? (01 U 10)*

Sometimes! 010110

But what about: 0011

Would this cover that case: 0* 1*

Careful! 0*1* would allow 00011

which should not be produced

Is this even possible?!

A regex cannot represent “All binary strings with an equal number of 0’s and 1’s” because this is an irregular language

Regex: regular expression for regular languages

Regular Languages

Definitions:

Regular Languages are languages that can be specified by a regular expression.

Irregular Languages are languages that are not regular.

Irregular Languages

It turns out a lot of useful languages are irregular.

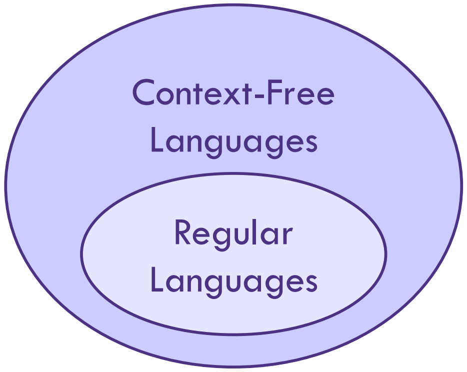
- Binary strings with an equal number of 0s and 1s
- Palindromes (strings that read the same forwards and backwards)
- Matched parentheses, e.g. ((())())
- Properly formed arithmetic expressions

Context Free Languages

Another class of languages

Context-Free Languages

- We just saw some limitations of Regular Languages
- Context-Free Languages are a strictly larger class of languages



- Context-Free Languages are generated by Context-Free Grammars (just like Regular Languages are specified by Regular Expressions)

Context-Free Grammars (CFGs)

- A production rule for a nonterminal A takes the form:

$$\mathrm{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

  where each w_i is a string of terminals and nonterminals

  For example:

$$\mathrm{S} \rightarrow \mathrm{Ab} \mid \mathrm{c}$$

$$\mathrm{A} \rightarrow \mathrm{Aa} \mid \varepsilon$$

Context-Free Grammars (CFGs)

- A production rule for a nonterminal A takes the form:

$$\mathrm{A} \rightarrow w_1 \mid w_2 \mid \cdots \mid w_k$$

where each w_i is a string of terminals and nonterminals

For example:

$$\mathrm{S} \rightarrow \mathrm{Ab} \mid \mathrm{c}$$

$$\mathrm{A} \rightarrow \mathrm{Aa} \mid \varepsilon$$

Context-Free Grammars (CFGs)

For example:

Non-terminal terminal

$$S \rightarrow \boxed{Ab} \,\|\, \boxed{c}$$

Start from S

$$A \rightarrow \boxed{Aa} \,\|\, \boxed{\varepsilon}$$

- A Non-terminal symbol means you can still select another symbol to concatenate
- A terminal symbol is the <u>last</u> symbol you can select

Context-Free Grammars

- A Context-Free Grammar is a finite set of production rules, involving:
  - Alphabet of *terminal* symbols (e.g. $0, 1, \mathrm{a}, \mathrm{b}, \varepsilon$)
  - A finite set of *nonterminal* symbols (e.g. $\mathrm{A}, \mathrm{B}, \mathrm{S}, \mathrm{T}, \mathrm{R}$)
  - One special nonterminal called the start symbol, usually S

Context-Free Grammars

For Example:
$S \to Ab \mid c$
$A \to Aa \mid \varepsilon$

We think of Context-Free Grammars as **generating** strings.

1. Start from the start symbol S.
2. Choose a nonterminal, e.g. S, in the string, and replace it by one of the w's in the rules for S
$$S \to w_1 \mid w_2 \mid \ldots \mid w_k$$
3. Repeat step 2 until there are no nonterminals left.

The language that the CFG describes is the set of all strings that it generates.

Regex to CFGs?

Union

(1 U 0) $S \rightarrow 1 \mid 0$

Kleene Star

(1 U 0)* $S \rightarrow 1\mathbf{S} \mid 0\mathbf{S} \mid \boldsymbol{\varepsilon}$

Concatenation

(1*0*) $S \rightarrow \boldsymbol{AB}$

$\mathbf{A} \rightarrow 1\mathbf{A} \mid \boldsymbol{\varepsilon}$

$\mathbf{B} \rightarrow 0\mathbf{B} \mid \boldsymbol{\varepsilon}$

Context-Free Grammars (CFGs)

For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \varepsilon$

Context-Free Grammars (CFGs)

For example:

$$S \rightarrow Ab \mid c$$

$$A \rightarrow Aa \mid \varepsilon$$

Can you create:

- "cb"
- "aa"
- "aab"
- "caab"

Context-Free Grammars (CFGs)

For example:

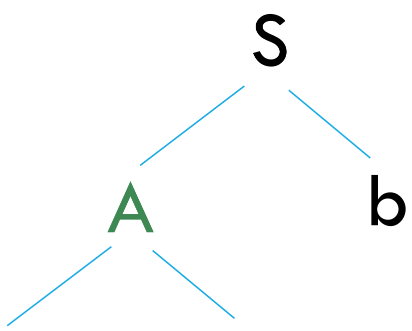
$S \rightarrow \boxed{Ab} \mid c$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- **“cb”**

Ab



- “aa”
- “aab”
- “caab”

Context-Free Grammars (CFGs)

For example:

$S \rightarrow Ab \mid \boxed{c}$

$A \rightarrow Aa \mid \varepsilon$

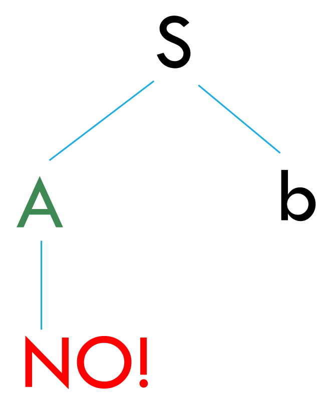
Can you create:

- **"cb"**

Ab c?

c can only be picked from the

A non terminal



- "aa"
- "aab"
- "caab"

Context-Free Grammars (CFGs)

For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- "cb"
- **"aa"**

No! You must start from **S** where we are required to use at least one b or have one c

- "aab"
- "caab"

Context-Free Grammars (CFGs)

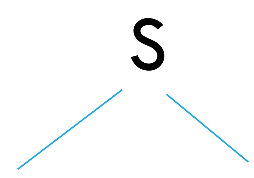
For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- “cb”
- “aa”
- **“aab”**



- “caab”

Context-Free Grammars (CFGs)

For example:

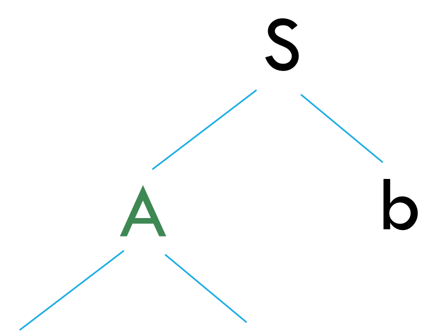
$S \rightarrow \boxed{Ab} \mid c$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- “cb”
- “aa”
- **“aab”**

Ab



- “caab”

Context-Free Grammars (CFGs)

For example:

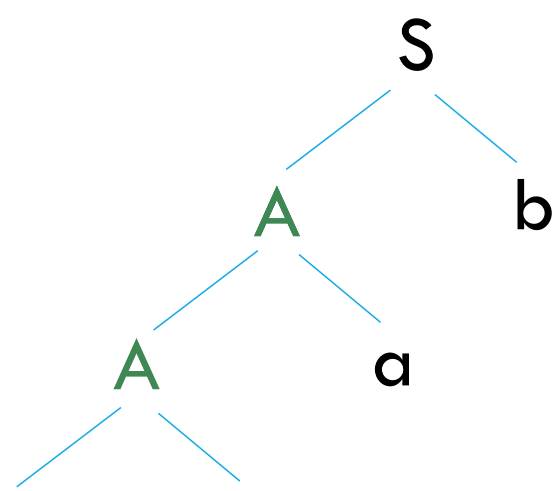
$S \rightarrow Ab \mid c$

$A \rightarrow \boxed{Aa} \mid \varepsilon$

Can you create:

- "cb"
- "aa"
- **"aab"**

Ab Aa



- "caab"

Context-Free Grammars (CFGs)

For example:

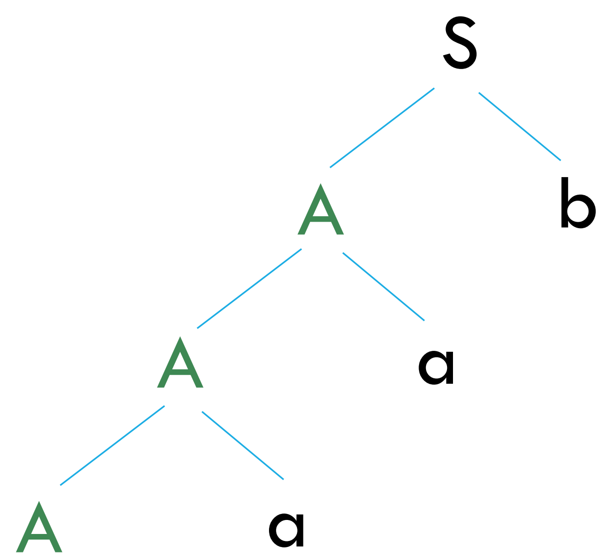
$S \to Ab \mid c$

$A \to \boxed{Aa} \mid \varepsilon$

Can you create:

- "cb"
- "aa"
- **"aab"**

Ab Aa Aa



- "caab"

Context-Free Grammars (CFGs)

For example:

$S \to Ab \mid c$

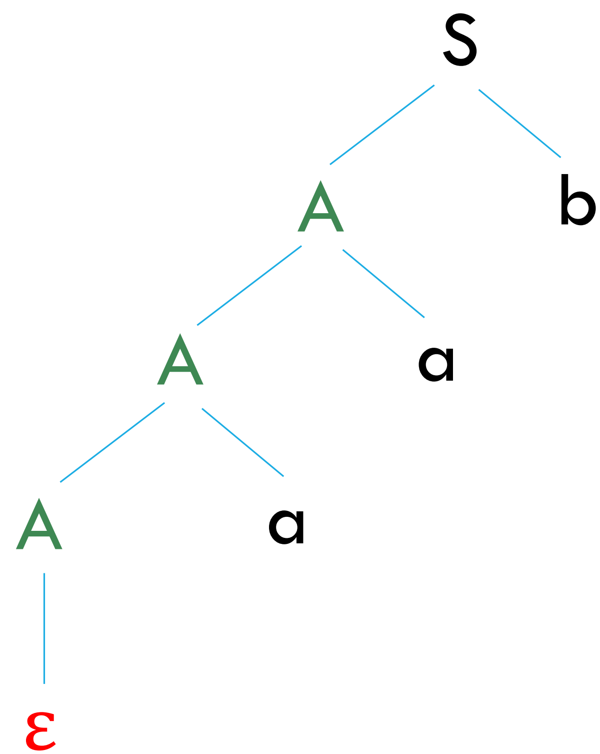
$A \to Aa \mid \boxed{\varepsilon}$

Can you create:

- "cb"
- "aa"
- **"aab"**

$Ab\ Aa\ Aa\ \varepsilon$

- "caab"



Context-Free Grammars (CFGs)

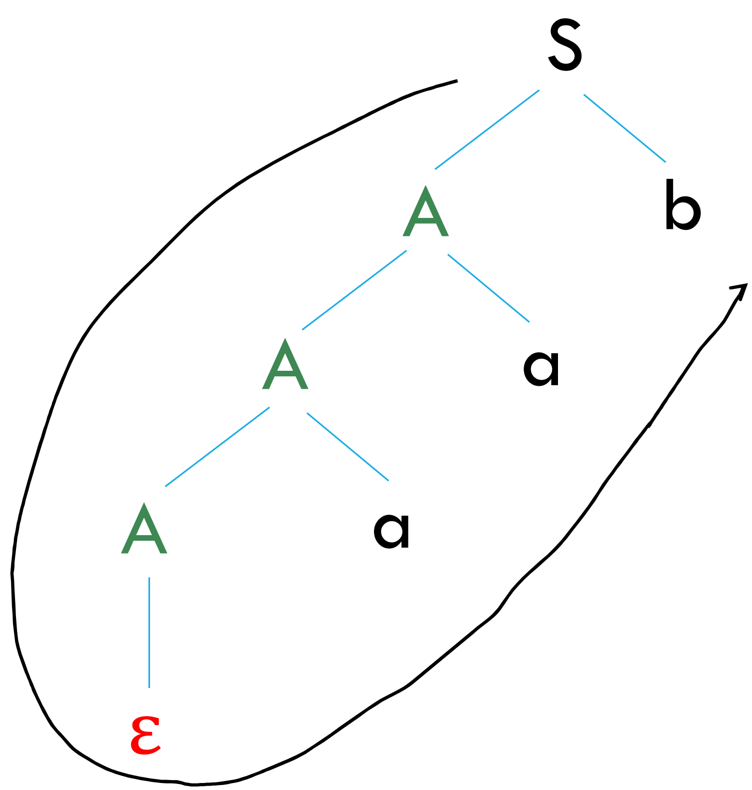
For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \boxed{\varepsilon}$

Can you create:

- "cb"
- "aa"
- **"aab"**
- "caab"



Context-Free Grammars (CFGs)

For example:

$$S \rightarrow Ab \mid c$$

$$A \rightarrow Aa \mid \varepsilon$$

Can you create:

- "cb"
- "aa"
- "aab"
- **"caab"**

Context-Free Grammars (CFGs)

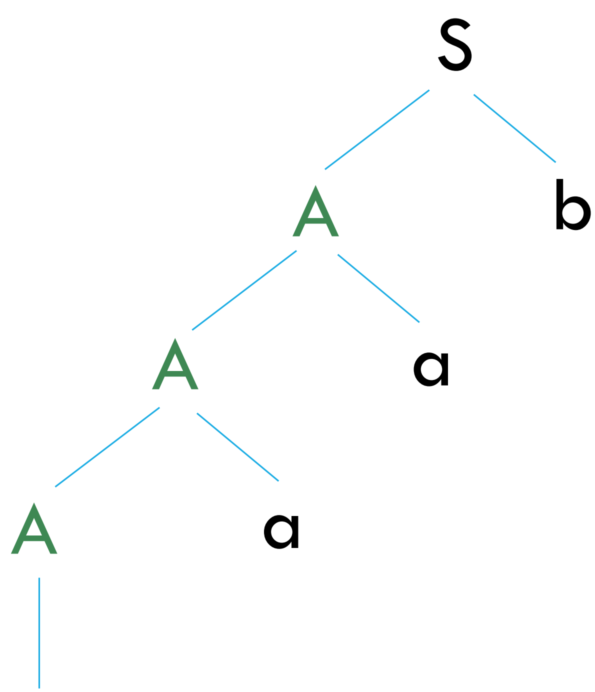
For example:

$S \rightarrow Ab \mid c$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- “cb”
- “aa”
- “aab”
- **“caab”**



Context-Free Grammars (CFGs)

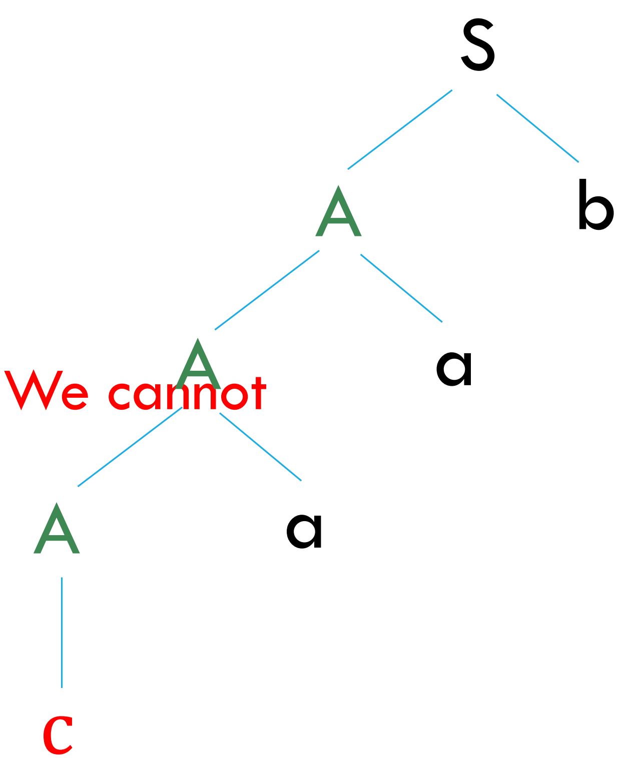
For example:

$S \rightarrow Ab \mid \boxed{c}$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- "cb"
- "aa"
- "aab"
- **"caab"**



This does not work! We cannot select **A**b

Context-Free Grammars (CFGs)

For example:

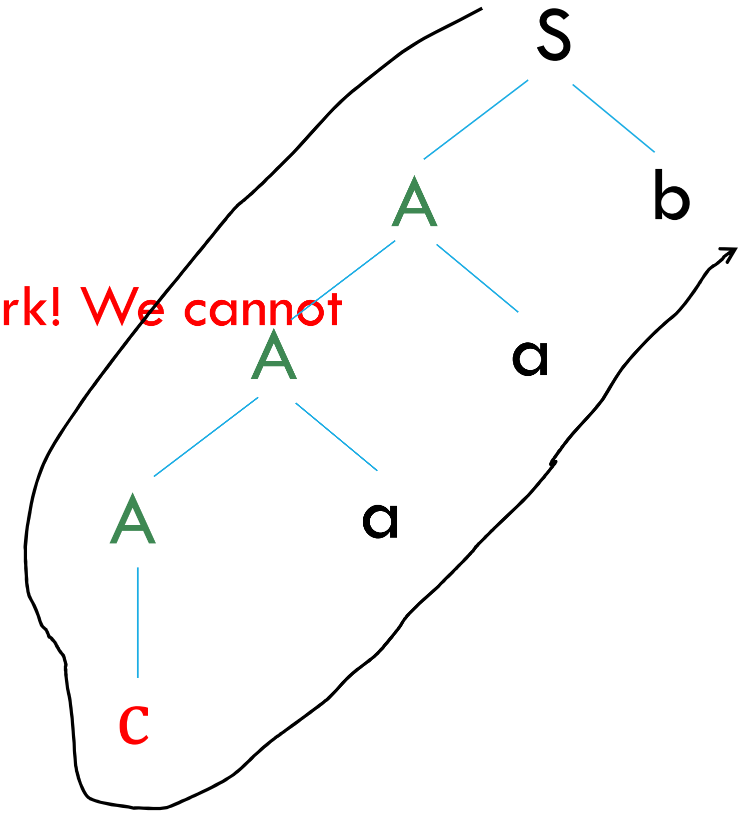
$S \rightarrow Ab \mid \boxed{c}$

$A \rightarrow Aa \mid \varepsilon$

Can you create:

- "cb"
- "aa"
- "aab"
- **"caab"**

This does not work! We cannot select **A**b



Example

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

Example

$S \to 0S \mid S1 \mid \varepsilon$

The set of all binary strings with any number of 0s followed by any number of 1s

Example

$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

Example

$S \rightarrow 0S0 \mid 1S1 \mid 0 \mid 1 \mid \varepsilon$

The set of all binary palindromes.

Example

CFG for the language $\{0^n 1^n : n \geq 0\}$

Example

CFG for the language $\{0^n 1^n : n \geq 0\}$

$\mathrm{S} \rightarrow 0\mathrm{S}1 \mid \varepsilon$

Example

CFG for the language $\{0^n 1^n 23 : n \geq 0\}$

Example

CFG for the language $\{0^n 1^n 23 : n \geq 0\}$

$\mathrm{S} \rightarrow \mathrm{A}23$

$\mathrm{A} \rightarrow 0\mathrm{A}1 \mid \varepsilon$

Exercises

CFG for the set of binary strings with the same number of 0s as 1s.

CFG for the set of balanced parentheses. E.g. ((())())

Exercises

CFG for the set of binary strings with the same number of 0s as 1s.

$S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$

CFG for the set of balanced parentheses. E.g. ((())())

Exercises

CFG for the set of binary strings with the same number of 0s as 1s.

$S \rightarrow SS \mid 0S1 \mid 1S0 \mid \varepsilon$

CFG for the set of balanced parentheses. E.g. ((())())

$S \rightarrow SS \mid (S) \mid \varepsilon$

Simple Arithmetic Expressions

$$E \to E+E \mid E*E \mid (E) \mid x \mid y \mid z \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Generate $(2*x) + y$

Simple Arithmetic Expressions

$$E \rightarrow E+E \mid E*E \mid (E) \mid x \mid y \mid z \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Generate $(2*x) + y$

$$E \Rightarrow E+E \Rightarrow (E)+E \Rightarrow (E*E)+E \Rightarrow (2*E)+E \Rightarrow (2*x)+E \Rightarrow (2*x)+y$$

Exercises

All binary strings that start with 11.

Exercises

All binary strings that start with 11.

Thinking back to regular expressions…

Exercises

All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Exercises

All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Now generate the CFG…

Exercises

All binary strings that start with 11.

Thinking back to regular expressions…

$11\ (0 \cup 1)^*$

Now generate the CFG…

$\mathbf{S} \rightarrow 11\mathbf{T}$
$\mathbf{T} \rightarrow 1\mathbf{T} \mid 0\mathbf{T} \mid \varepsilon$

Exercises

All binary strings that contain at most one 1.

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \, (1 \cup \boldsymbol{\varepsilon}) \, 0^*$

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \, (1 \cup \varepsilon) \, 0^*$

Now generate the CFG…

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \, (1 \cup \varepsilon) \, 0^*$

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \, (1 \cup \varepsilon) \, 0^*$

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Alternative solution:

Exercises

All binary strings that contain at most one 1.

Thinking back to Regular expressions…

$0^* \ (1 \cup \varepsilon) \ 0^*$

Now generate the CFG…

$S \rightarrow ABA$
$A \rightarrow 0A \mid \varepsilon$
$B \rightarrow 1 \mid \varepsilon$

Alternative solution:

$S \rightarrow 0S \mid S0 \mid 1 \mid 0 \mid \varepsilon$