Regular Expressions

Basis:

ε is a regular expression. The empty string itself matches the pattern (and nothing else does).

$\emptyset$ is a regular expression. No strings match this pattern.

a is a regular expression, for any $a \in \Sigma$ (i.e. any character). The character itself matching this pattern.

Recursive

If A, B are regular expressions then $(A \cup B)$ is a regular expression
matched by any string that matches A or that matches B [or both]).

If A, B are regular expressions then AB is a regular expression.
matched by any string x such that $x = yz$, y matches A and z matches B.

If A is a regular expression, then A^* is a regular expression.
matched by any string that can be divided into 0 or more strings that match A.

More Practice

You can also go the other way

Write a regular expression for "the set of all binary strings of odd length"

Write a regular expression for "the set of all binary strings with at most two ones"

Write a regular expression for "strings that don't contain 00"

Context Free Grammars

We think of context free grammars as **generating** strings.

1. Start from the start symbol S.
2. Choose a nonterminal in the string, and a production rule $A \to w_1|w_2|\ldots|w_k$ replace that copy of the nonterminal with w_i.
3. If no nonterminals remain, you're done! Otherwise, goto step 2.

A string is in the language of the CFG iff it can be generated starting from S.

Examples

$S \to 0S0|1S1|0|1|\varepsilon$

$S \to 0S|S1|\varepsilon$

$S \to (S)|SS|\varepsilon$

The alphabet here is {(,)} i.e. parentheses are the characters.

$S \to AB$

$A \to 0A1|\varepsilon$

$B \to 1B0|\varepsilon$