Preliminaries

Sets ($S$): union ($\cup$), intersection ($\cap$), difference ($-$), complementation ($\overline{S}$), universal set ($U$), empty set ($\emptyset$), DeMorgan’s Laws, subset ($\subseteq$), proper subset ($\subset$), disjoint, powerset ($2^S$), Cartesian product ($\times$).

Functions ($f$) and Relations: domain, range, total function, partial function, “order at most” aka big-Oh ($O$), “order at least” aka big-Omega ($\Omega$), “same order of magnitude” aka big-Theta ($\Theta$), relation, equivalence ($\equiv$).

Preliminaries Continued

Graphs and Trees: vertices ($V$), edges ($E$), walk, path, simple path, cycle, loop, root, leaves, parent, child, level, height, ordered trees.

Proof Techniques: proof by induction, basis, inductive assumption, inductive step, proof by contradiction.
Languages

Alphabet ($\Sigma$): a finite set of symbols, e.g.,
$\Sigma = \{a, b\}$.

String ($w$): a finite sequence of symbols from $\Sigma$, e.g., $w = babaa$.

concatenation ($vw$), reverse ($w^R$), length ($|w|$), substring, prefix, suffix

Empty String ($\lambda$): the string of length 0.

$w^n$: string $w$ repeated $n$ times, e.g., $a^3 = aaa$.

Languages Continued

$\Sigma^*$: all possible strings using $\Sigma$.

$\Sigma^+$: all possible strings using $\Sigma$ except $\lambda$.

Language ($L$): A subset of $\Sigma^*$, e.g.,
$L = \{a^n b^n : n \geq 0\}$

union, intersection, concatenation of languages

reverse ($L^R$), concatenation with itself ($L^n$),
star-closure ($L^*$), positive closure ($L^+$)
Grammars

A grammar $G = (V, T, S, P)$ where:

$V$ is a set of symbols called *variables*,

$T$ is a set of *terminal symbols*,

$S \in V$ is the *start* variable, and

$P$ is a finite set of *productions*.

All production rules are of the form $x \rightarrow y$, where $x \in (V \cup T)^+$ and $y \in (V \cup T)^*$. This means that $x$ can be replaced with $y$.

Grammars Continued

A string $w$ *derives* $z$ ($w \Rightarrow^* z$) if

- $w = z$, or
- $w \Rightarrow z$, i.e., $w = uxv$ and $z = uyv$ and there is a production $x \rightarrow y$, or
- $w \Rightarrow w_2 \Rightarrow \ldots \Rightarrow z$ in a finite sequence.

The language of a grammar is:

$$\mathcal{L}(G) = \{w \in T^* : S \Rightarrow^* w\}$$
Grammar Example 1

\[ V = \{ S \}, \ T = \{ a, b \}, \text{ and } P \text{ is:} \]

\[
S \rightarrow aSb \\
S \rightarrow aSbb \\
S \rightarrow \lambda \\
S \Rightarrow aSbb \Rightarrow aaSbbb \Rightarrow aabbb
\]

\[ \mathcal{L}(G) = \{ a^m b^n : 0 \leq m \leq n \leq 2m \} \]

Grammar Example 2

Example: \( V = \{ S \}, \ T = \{ a, b \}, \ P \text{ is:} \)

\[
S \rightarrow \lambda \\
S \rightarrow aSb \\
ab \rightarrow ba
\]

\[
S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \Rightarrow abab \Rightarrow baab \Rightarrow baba \Rightarrow bbaa
\]

\[ \mathcal{L}(G) = \{ w : \text{number of } a\text{'s }= \text{number of } b\text{'s} \} \]
Automata

An automaton reads a string from an input file, (possibly) reads/writes to a storage device, and has a control unit in an internal state.

The configuration of an automaton is the position of the input pointer, position of the storage pointer, and the current internal state.

A transition function maps from the current internal state, input symbol, and storage symbol to the next internal state and any changes.

Automata Continued

An automaton is deterministic if each move is unique. An automaton is nondeterministic if there is a choice between several possible moves.

An automaton is an accepter if outputs “yes” or “no”. An automaton is a transducer if it outputs a string.