Introduction

What lies ahead:

- formalization of computation
- various models of computation (increasing difficulty/power)
- what can / cannot be done?
Introduction

What lies ahead:
- formalization of computation
- various models of computation (increasing difficulty/power)
- what can / cannot be done?

Motivation?
Discrete Math

Models are mathematical
- need basic discrete math knowledge [Part I of the book]
- we will go over Sections 1.5, 2.4, and 2.5

Notation in the book
- $A'$ is the complement of a set $A$ \hspace{1cm} \overline{A} \hspace{1cm} A^c$
- $\mathbb{N}$ is the set of natural numbers (these slides use $\mathbb{N}$)

Quiz

Th 2 - 2:25 Sep 6
1) discrete math reading/writing/reasoning skill
2) proof by induction
Languages
Languages

A **language** is a set of strings involving symbols from some **alphabet**.

An **alphabet** is a finite set of symbols, typically denoted by $\Sigma$.

Examples:

- **English** = { "Hello world!", ... } , $\sum = \{ a, ..., z, ..., !, ? \}$
- **Java** = { "f=1; for (i=1; i<=n; i++) f*=i;", ... } , $\sum = \{ \text{for, int, 1, 0, ...} \}$ (finite)
Languages

A **language** is a set of strings involving symbols from some **alphabet**.

An **alphabet** is a finite set of symbols, typically denoted by \( \Sigma \).

A **string** over \( \Sigma \) is a finite (possibly empty) sequence of elements of \( \Sigma \).

Examples of strings over \{0,1\}:

- \(101\)
- \(1\)
- \(\Lambda\)
- \(1000\)
- \(0\)

These are *not* strings over \{0,1\}:

- \(0151\)
- \(000 \ldots\)
Languages

A **language** is a set of strings involving symbols from some **alphabet**.

An **alphabet** is a finite set of symbols, typically denoted by $\Sigma$.

A **string** over $\Sigma$ is a finite (possibly empty) sequence of elements of $\Sigma$.

$\Lambda$ denotes the **null string** - the empty sequence of elem. of $\Sigma$.

If $x$ is a string over $\Sigma$, $|x|$ denotes the **length** of $x$, i.e. the number of symbols of the alphabet in the string.

\[
x = 0100 \quad |x| = 4 \quad x = 1 \quad |x| = 1 \quad x = \Lambda \quad |x| = 0
\]
Languages

$\Sigma^*$ denotes the language of all strings over $\Sigma$.

Examples:

$\Sigma = \{0,1\}$  \hspace{1cm} $\Sigma^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots \}$

$\Sigma = \{0\}$  \hspace{1cm} $\Sigma^* = \{\lambda, 0, 00, 000, \ldots \}$
Languages

\( \Sigma^* \) denotes the language of all strings over \( \Sigma \).

A **language** over \( \Sigma \) is a subset of \( \Sigma^* \).

Examples:

- \( L_1 = \{ 0, 001, \Lambda \} \) \( \Sigma = \{ 0, 1 \} \)
- \( L_2 = \{ x \mid x \in \Sigma^* \text{ and } |x| \text{ is a prime} \} \)
- \( L_3 = \{ \Lambda \} \)
- \( L_4 = \emptyset = \{ \} \)

\(|L_1| = 3\)
\( |L_2| = \infty \)
\( |L_3| = 1 \)
\( |L_4| = 0 \)

What are the sizes of these languages?
Operations on languages

Set operations:
- union $L_1 \cup L_2 = \{0, 10, 110\}$
- intersection $L_1 \cap L_2 = \{0\}$
- difference $L_1 - L_2 = \{110\}$
- complement $L' = \Sigma^* - L$

$L_1 = \{0, 110\}$
$L_2 = \{10, 0\}$

symmetric dif.
$L_1 \setminus L_2 = \{110, 10\}$
Let $x, y \in \Sigma^*$.

- $xy$ is the **concatenation** of $x$ and $y$

Examples:

- $x = 0110$  \hspace{1cm} $y = 11$  \hspace{1cm} $xy = 011011$
- $x = 01$  \hspace{1cm} $y = \Lambda$  \hspace{1cm} $xy = 01$
Operations on languages

Operations on strings

Let \( x, y \in \Sigma^* \).

- \( xy \) is the **concatenation** of \( x \) and \( y \)
- for an integer \( k \), \( x^k \) is the concatenation of \( k \) copies of \( x \)

Examples:

\[
\begin{align*}
\chi &= 00 \\
\kappa &= 4 \\
\chi^k &= \underbrace{00000000}_8 = 0^8 \\
\chi^0 &= \varnothing
\end{align*}
\]
Operations on languages

Operations on strings

Let \( x, y \in \Sigma^* \).

- \( xy \) is the **concatenation** of \( x \) and \( y \)
- for an integer \( k \), \( x^k \) is the concatenation of \( k \) copies of \( x \)
- \( x \) is a **substring** of \( y \) iff there exist \( w, z \in \Sigma^* \) such that \( y = wxz \)

Examples:

\[
\begin{align*}
y &= 1001011 \\
x &= 101 \\
x &= 11 \\
x &= 111 \quad \text{NOT a substring of } y
\end{align*}
\]
Operations on strings

Let $x, y \in \Sigma^*$.

- $xy$ is the **concatenation** of $x$ and $y$
- for an integer $k$, $x^k$ is the concatenation of $k$ copies of $x$
- $x$ is a **substring** of $y$ iff there exist $w, z \in \Sigma^*$ such that $y = wxz$
- $x$ is a **prefix** of $y$ iff there exists $z \in \Sigma^*$ such that $y = xz$

Examples:

\[ y = 10010 \]
\[ x = \lambda \delta 1 \text{ or } 10 \text{ or } 100 \text{ or } 1001 \text{ or } 10010 \]
Operations on languages

Operations on strings

Let $x, y \in \Sigma^*$.
- $xy$ is the **concatenation** of $x$ and $y$
- for an integer $k$, $x^k$ is the concatenation of $k$ copies of $x$
- $x$ is a **substring** of $y$ iff there exist $w, z \in \Sigma^*$ such that $y = wxz$
- $x$ is a **prefix** of $y$ iff there exists $z \in \Sigma^*$ such that $y = xz$
- $x$ is a **suffix** of $y$ iff there exists $z \in \Sigma^*$ such that $y = zx$

Examples:
Operations on languages

Operations on strings

Let $x, y \in \Sigma^*$.

- $xy$ is the **concatenation** of $x$ and $y$
- for an integer $k$, $x^k$ is the concatenation of $k$ copies of $x$
- $x$ is a **substring** of $y$ iff there exist $w, z \in \Sigma^*$ such that $y = wxz$
- $x$ is a **prefix** of $y$ iff there exists $z \in \Sigma^*$ such that $y = xz$
- $x$ is a **suffix** of $y$ iff there exists $z \in \Sigma^*$ such that $y = zx$
- $x^r$ is the **reverse** of $x$ iff $x^r$ is “$x$ written backwards”

Examples:

\[
x = 10010 \\
x^r = 01001
\]
Operations on languages

Operations on strings extended to languages

Let $L, L_1, L_2 \subseteq \Sigma^*$.

- $L_1L_2$ is the **concatenation** of languages $L_1L_2$, i.e.

  $$L_1L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$$

Examples:

- $L_1 = \{ 0, 10 \}$
- $L_2 = \{ 11, 01 \}$

  $$L_1L_2 = \{ 011, 001, 1011, 1001 \}$$

**Question:** $|L_1L_2| = |L_1| \cdot |L_2|$ ?  **Not True But**

  $$|L_1L_2| \leq |L_1| \cdot |L_2|$$
Operations on languages

Operations on strings extended to languages

Let $L, L_1, L_2 \subseteq \Sigma^*$.

- $L_1 L_2$ is the **concatenation** of languages $L_1 L_2$, i.e.

$$L_1 L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$$

- for an integer $k$, $L^k$ is the concatenation of $k$ copies of $L$

Examples:

$L = \{ \Lambda, 0 \}$

$L^2 = \{ \Lambda, 0, 00 \}$

$L^3 = \{ \Lambda, 0, 00, 000 \}$

$L^0 = \{ \Lambda \}$

$L = \{ 0, 1 \}$

$L^2 = \{ 00, 01, 10, 11 \}$

$L^3 = \{ \}$

$L^k = \{ x \in \{ 0, 1 \}^* \mid |x| = k \}$
Operations on languages

Operations on strings extended to languages

Let $L, L_1, L_2 \subseteq \Sigma^*$.

- $L_1L_2$ is the **concatenation** of languages $L_1L_2$, i.e.
  
  $$L_1L_2 = \{ xy \mid x \in L_1, y \in L_2 \}$$

- for an integer $k$, $L^k$ is the concatenation of $k$ copies of $L$

- $L^* = \bigcup_{k=0}^{\infty} L^k$ (this operation is called **Kleene's star**)

Examples:
Operations on languages

Let \( L, L_1, L_2 \subseteq \Sigma^* \).

- \( L_1L_2 \) is the **concatenation** of languages \( L_1L_2 \), i.e.
  \[
  L_1L_2 = \{ xy \mid x \in L_1, y \in L_2 \}
  \]

- for an integer \( k \), \( L^k \) is the concatenation of \( k \) copies of \( L \)

- \( L^* = \bigcup_{k=0}^{\infty} L^k \) (this operation is called **Kleene's star**)

- \( L^+ = \bigcup_{k=1}^{\infty} L^k \)

**Examples:**

\[
L = \{ x \mid x \in \{0,1\}^* \text{ and } |x| \text{ is a prime} \}
\]
Operations on languages

Operations on strings extended to languages

Let $L, L_1, L_2 \subseteq \Sigma^*$.

- $L_1L_2$ is the **concatenation** of languages $L_1L_2$, i.e.

$$L_1L_2 = \{ xy | x \in L_1, y \in L_2 \}$$

- for an integer $k$, $L^k$ is the concatenation of $k$ copies of $L$

- $L^* = \bigcup_{k=0}^{\infty} L^k$ (this operation is called **Kleene's star**)

- $L^+ = \bigcup_{k=1}^{\infty} L^k$

- $L^r$ is the **reverse** of $L$, i.e.

$$L^r = \{ x^r | x \in L \}$$

Examples:
Recursive Definitions

A well-known recursive definition:

1) $0! = 1$

2) for every $n \in \mathbb{N}$, $(n+1)! = (n+1) \cdot n!$

Compute $5!$

$$5! = 5 \cdot 4! = 5 \cdot 4 \cdot 3! = 5 \cdot 4 \cdot 3 \cdot 2! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 0! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1$$
Recursive Definitions

Recursive definition of $\Sigma^* S$

1) $\Lambda \in \Sigma^* S$

2) for every $\sigma \in \Sigma$ let $\sigma \in \Sigma^*$ S ← redundant

3) for every $\sigma \in \Sigma$ and every $x \in \Sigma^*$ let $\sigma x, x\sigma \in \Sigma^* S$

4) nothing else in $\Sigma^* S$

$\Sigma = \{0, 1\}$

$\Sigma^* = \{\Lambda, 0, 1, 00, 01, 10, \ldots\}$

$\Sigma^* = \{\Lambda, 0, 1, 00, 11, 000, 101, 010, 111, \ldots\}$
Recursive Definitions

Recursive definition of the **length of a string**

1) let $|\Lambda| = 0$

2) for every $x \in \Sigma^*$ and every $\sigma \in \Sigma$, let $|x\sigma| = |x| + 1$
Recursive Definitions

Recursive definition of the reverse of a string

1) let \( \Lambda^r = \Lambda \)

2) for every \( x \in \Sigma^* \) and every \( \sigma \in \Sigma \) let \( (x\sigma)^r = \sigma x^r \)

Compute \( (abcb)^r \).

\[
(abcb)^r = b(abcb)^r = bc(ab)^r = bcb(a)^r = bcba(\Lambda)^r = \\
= bcba \Lambda = bcba
\]
Recursive Definitions

Recursive definition of a language

A palindrome is a string that reads the same forward and backward.

Examples:
Recursive Definitions

Recursive definition of a language

A palindrome is a string that reads the same forward and backward.

\textit{abpal} is the language of all palindromes over \{a,b\}.

Recursive definition of \textit{abpal}

1) \(\Lambda \in \textit{abpal}\)

2) for every \(\sigma \in \{a,b\}\) let \(\overline{\sigma} \in \textit{abpal}\)

3) nothing else is in \textit{abpal}

Is the definition correct?
Recursive Definitions

Recursive definition of a language \( L_{0=1} \subseteq \{0,1\}^* \):

1) \( \Lambda \in L_{0=1} \)

2) for every \( x \in L_{0=1} \), \( 0x1, 1x0, 01x, 10x, x01, \) and \( x10 \in L_{0=1} \)

3) no other strings are in \( L_{0=1} \)

Can you describe \( L_{0=1} \) in words?

\( L_{0=1} \) contains strings (over \( \{0,1\} \)) which have equal number of 0's and 1's.
We will prove that every string \( z \) in the (corrected) language \( L_{0=1} \) contains the same number of zeros and ones.

**BASE CASE:** \( z \) is constructed by rule 1: \( z = \lambda \) and \( \#0's \text{ in } z = 0 = \#1's \text{ in } z \)

**INDUCTIVE CASE:**
\( z \) is constructed by rules 2 or 2.5

**INDUCTIVE HYPOTHESIS:** every \( x \) constructed earlier than \( z \) has the same number of 0's and 1's.

**CASE 1:** \( z \) is constructed by rule 2, part 1: \( z = 0x1 \) for some previously constructed \( x \)

by IH: \( \#0's \text{ in } x = \#1's \text{ in } x = k \)

thus \( \#0's \text{ in } z = k + 1 = \#1's \text{ in } x \)

(similarly other 5 parts of rule 2)

**CASE 2:** \( z \) is constructed by rule 2.5: \( z = xy \) for some previously constructed \( x, y \)

DO: finish the proof of CASE 2.
We proved: $L_{01} \subseteq \{ x \mid x \in \{0,1\}^* \text{ and } \#0's \text{ in } x = \#1's \text{ in } x \}$

To prove equality, we need to say that every string $x$ with the same number of 0's and 1's can be generated by rules 1)-3).
Another Exercise

Recursive definition of a language \( L_{a \geq b} \subseteq \{a,b\}^* \):

1) \( a \in L_{a \geq b} \)
2) for every \( x \in L_{a \geq b} \), \( ax \in L_{a \geq b} \)
3) for every \( x,y \in L_{a \geq b} \), \( bxy, xby, \) and \( xyb \in L_{a \geq b} \)
4) no other strings are in \( L_{a \geq b} \)