Session 1

Preliminaries and Languages

Central Concepts

- Alphabets
- Strings
- Languages
- Representation
- Interpretation
- Problems

Alphabets

- A finite (nonempty) set of symbols: Σ
  - Σ = {*a*, *b*, *c*, …, *z*}
  - Σ = {α, β, γ, …, ω}
  - Σ = {0, 1}
  - Σ = {0, 1, 2, 3, …, 9}
  - Σ = {1}
  - Σ = The set of ASCII characters

Strings

- A string (a word) over an alphabet Σ
  - A finite sequence of symbols of Σ
- Length of a string
  - The number (positions) of symbols in a string:
    - *w* = "111" has one symbol in three positions
  - The length of string *w* is |*w*|
    - |*w*| = 3
- The null string: Λ (lambda)
  - Λ may be chosen from any alphabet
    - |Λ| = 0

Strings

- Notation:
  - Lower-case letter at the beginning of the alphabet denote symbols: *a*, *b*, *c*…
  - Lower-case letters at the end of the alphabet denote strings: *w*, *x*, *y*, *z*

Strings

- Powers of an alphabet
  - If Σ is an alphabet Σ^k is the set of strings of length *k*, such that all symbols in *k* are in Σ
  - $\Sigma^0 = \{\Lambda\}$
  - If Σ = {0, 1} then
    - $\Sigma^0 = \{\Lambda\}$
    - $\Sigma^1 = \{0, 1\}$
    - $\Sigma^2 = \{00, 01, 10, 11\}$ and so on
  - $\Sigma \neq \Sigma^1$ (Σ is the alphabet and Σ^1 is the set of strings of length 1)

Strings

- For any alphabet Σ the set of *all* strings over Σ is denoted as Σ^*
  - $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$
    - $\Sigma^* = \{0, 1\}^* = \{\Lambda, 0, 1, 00, 01, 10, 11, \ldots\}$
- And without Σ^0:
  - $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots$
    - $\Sigma^+ = \{0, 1\}^+ = \{0, 1, 00, 01, 10, 11, \ldots\}$

Σ^* is denumerable

- A set is *denumerable* or *enumerably infinite* or *countable* if it can be arranged in a single (perhaps infinite) list:
  - 1, 2, 3, 4... is an infinite list
  - 1, 3, 5, …, 2, 4, 6… is not! (∞ + 1?)
  - We need to be able to tell for each object listed, which one is it on the list
- A set *A* denumerable if there is a function with domain in *N* (the natural numbers) such that each member of *A* is associated to one *n* in *N*

Σ^* is denumerable

N	Σ^*
1	Λ
2	0
3	1
4	00
5	01
6	10
7	11
8	000

N	Σ^*
9	001
10	010
11	011
12	100
13	101
14	110
15	111
16	...

$\Sigma^* = \{0, 1\}^* = \{\Lambda, 0, 1, 00, 01, 10, 11, \ldots\}$

Σ^* is denumerable

- Σ = {a,..,z} (26 letters)
  - Σ^0 has 1 string of length 0 (i.e. Λ)
  - Σ^1 has 26 strings of length 1 (i.e. a,…,z)
  - Σ^2 has 26^2 strings of length 2 (in alphabetic order)
  - Σ^3 has 26^3 strings of length 3 (in alphabetic order)
  - …
- There is a function that for any argument *n* the value is the corresponding string in Σ^*

Strings

- Concatenation of strings
  - If *x* and *y* are strings *xy* denotes the concatenation of *x* and *y*
  - More precisely: if $x = x_1 x_2 \ldots x_i$ and $y = y_1 y_2 \ldots y_j$ then $xy = x_1 x_2 \ldots x_i y_1 y_2 \ldots y_j$
  - i.e: *x* = 01101 and *y* = 110 then *xy* = 01101110
  - $|xy| = i + j$
  - Identity of concatenation: $\Lambda x = x \Lambda = x$
  - Concatenation is associative: $(xy)z = x(yz)$

Strings

- Concatenation of sets of strings
  - If $A, B \subseteq \Sigma^*$ the concatenation of *A* and *B* is

    $AB = \{xy \mid x \in A \text{ and } y \in B\}$
- Concatenation is NOT commutative
  - *A* = {a, b} and *B* = {c, d}
  - *AB* = {ac, ad, bc, bd}
  - *BA* = {ca, cb, da, db}

Concatenation of strings

$AB = \{xy \mid x \in A \text{ and } y \in B\}$

d	ad	bd
c	ac	bc
B \ A	a	b

Different from BA = {ca, cb, da, db}

Strings

- However:
  - $\Sigma^{n+1} = \Sigma\, \Sigma^n = \Sigma^n\, \Sigma \quad (\Sigma = \Sigma^1)$
  - If $\Sigma = \{0, 1\}$ then
    - $\Sigma^0 = \{\Lambda\}$
    - $\Sigma^1 = \{\Lambda\}\{0, 1\} = \{0, 1\}\{\Lambda\} = \{0, 1\}$
    - $\Sigma^2 = \{0, 1\}\{0, 1\} = \{00, 01, 10, 11\}$
    - $\Sigma^3 = \{0, 1\}\{00, 01, 10, 11\} = \{00, 01, 10, 11\}\{0, 1\}$
      $= \{000, 001, 010, 011, 100, 101, 110, 111\}$
    - …

Languages

- A *language* is a set of *strings* made out of symbols of an *alphabet*
- Natural languages (English, Spanish, etc.)
  - Syntactic level: Sentences are made out of words
  - Lexical level: Words are made out of symbols of an alphabet
- Formal languages
  - Syntactic level: Well-formed expressions are made out strings (tokens)
  - Lexical level: tokens are made of alphabet symbols (ASCII)

Definition of Languages

- A language over Σ is a subset of Σ^*
- L is a language over Σ if $L \subseteq \Sigma^*$
- L needs not to include all symbols in Σ, so if L is a language over Σ, it is also a language over a super set of Σ

Examples of Languages

- The language of all strings consisting of n 0's followed by n 1's, for some $n \geq 0$:

  $\{\Lambda, 01, 0011, 000111, \ldots\}$
- The set of strings of 0's and 1's with an equal number each:

  $\{\Lambda, 01, 10, 0011, 0101, 1001, \ldots\}$
- The set of strings representing prime numbers in binary notation:

  $\{10, 11, 101, 111, 1011, \ldots\}$

Examples of Languages

- Σ^* is a language over any alphabet
- Φ, the empty language, is a language over any alphabet
- $\{\Lambda\}$, the language consisting of the empty string
- Note that $\Phi \neq \{\Lambda\}$

Stating a language

- As set formation:
  - $\{w \mid \text{a property of } w\}$
  - Example: $\{w \mid w$ consists of a sequence of n 0's followed by a sequence of n 1's$\}$
- Expressing w with parameters
  - $\{0^n 1^n \mid n \geq 0\}$ where n is the parameter
  - $\{0^i 1^j \mid 0 \leq i \leq j\}$ where i and j are the parameters
- Combining set operators with concatenation
  - $\{ab, bab\}^* \cup \{b\}\{bb\}^*$
- Or even:
  - $\{byb \mid y \in \{a, b\}^*\}$
- It would be nice to have a simple and clear way to do it!

Representation

- Strings in languages *represent* objects in the world:
  - John, Pete and Luis are represented by {*john*, *pete*, *luis*}
  - 2, 3, 5, 7, 11 are represented by {10, 11, 101, 111, 1011, …}
- A representation can be thought of as a function from the world to a language!
- This function is applied by the sender of a message!

John → is represented by → *John*

Interpretations

- Strings in languages are *interpreted* as objects in the world:
  - {*john*, *pete*, *luis*} are interpreted as John, Pete and Luis
  - {10, 11, 101, 111, 1011, …} are interpreted as 2, 3, 5, 7, 11
- A interpretation can be thought of as a function from the the language to the world!
- This functions is applied by the receiver of the message!

John → is interpreted as → John

Problems

- A *problem* is the question of deciding whether a string is a member of some particular language
- If Σ is an alphabet, L is a language over Σ, the problem L is:
  - Giving a string w in Σ^*, decide whether or not w is in L
- Problems and Languages are really the same thing!

A Problem

- The problem of whether a number is prime can be expressed by the language L_p consisting on all monadic strings whose length is a prime number:
  - $L_p = \{11, 111, 11111, 1111111, 11111111111, \ldots\}$
  - $111 \in L_p$
  - $1111 \notin L_p$
- Given a string of 1's say "yes" or "not" depending the string in question represents a prime
- We need to think of the language as a representation; otherwise we cannot define the algorithm

Problems

- Two ways of think of Problems:
  - As decision questions: Deciding whether a string is in a set of strings

    $111 \in \{11, 111, 11111, 1111111, 11111111111, \ldots\}$?
  - As procedures transforming an input into an output:

    111 → Algorithm → yes
- Question of complexity theory:
  - Executing an algorithm will take some computational resources: space and time
  - If a problem is *hard* it can be *reduced* to an equivalent but simpler formulation