

1 Sets

Sets are collections of values of one type with no internal structure. Sets have no repeated items and order does not matter.

1.1 Set Operations

- Union - $A \cup B = \{x : x \in A \text{ or } x \in B\}$
- Intersect - $A \cap B = \{x : x \in A \text{ and } x \in B\}$
- Concatenation - $AB = \{ab : a \in A \text{ and } b \in B\}$
- Cartesian Product - $A \times B = \{(a, b) : a \in A \text{ and } b \in B\}$
- Complement - $A^c = \{x : x \notin A\}$
- Powerset - $P(A) = \{A' : A' \subset A\}$

1.2 Finite and Infinite Sets

Set A is **finite** if it can be put in 1-to-1 correspondence with an initial segment of \mathbb{N} , or is the empty set \emptyset . More intuitively, a set is finite if we can list the elements of A in finite time.

Set A is **countably infinite** if it can be put in 1-to-1 correspondence with *all* of \mathbb{N} . Intuitively, a procedure can be devised to list all elements, but this procedure will never finish.

Set A is **uncountably infinite** if it is not countable.

1.3 Relations and Functions

A **function** can be defined as a binary operation between two sets that given an input in set A produces an output in set B . There are three types of functions:

1. Total function: $A \rightarrow B$ means for every input A there is exactly one output B
2. Partial function: $A \rightarrow B$ means for every input A there is at most one output B
3. Multi function: $A \multimap B$ means for every input A there are 0, 1, or many outputs in B

We also define the **identity function** such that for any type A , $f(A) = A$.

1.4 Cardinality of Sets

The cardinality of a set A , or $|A|$, is the total number of elements in A .

- Union - $|A \cup B| = |A| + |B| - |A \cap B|$
- Intersect - $|A \cap B| = |A| + |B| - |A \cup B|$
- Concatenation -

- Cartesian Product - $|A \times B| = |A| \cdot |B|$
- Powerset - $P(A) = 2^{|A|}$
- Total function - $|A \rightarrow B| = |B|^{|A|}$
- Partial function - $|A \rightharpoonup B| = |B + 1|^{|A|}$
- Multi function - $|A \multimap B| = |P(B)|^{|A|}$

2 Alphabets and Languages

Alphabet A is a set of single characters. **Word** w in alphabet A is a string of characters from A . A^* denotes the set of all words in alphabet A . This operation is called *Kleene star*.

2.1 Regular Languages

A language is **regular** if it can be defined as follows.

- The empty language \emptyset is regular
- For each $a \in \epsilon$ where ϵ is the alphabet in question, the singleton language $\{a\}$ is regular
- If A and B are regular, then $A \cup B$ is regular
- If A and B are regular, then $A \cdot B$ is regular
- If A is regular, then A^* is regular.

Regular languages can be expressed by **regular expressions**. For each rule above, we have a corresponding regular expression.

- $e \rightarrow \emptyset$
- $a \rightarrow \{a\}$
- $a \vee b \rightarrow \{a\} \cup \{b\} = \{a, b\}$
- $ab \rightarrow \{a\} \cdot \{b\} = \{ab\}$
- $a^* \rightarrow \{a\}^* = \{\emptyset, a, a^2, a^3 \dots\}$

2.2 Pumping Lemma

All finite languages are regular. For infinite languages, we use the **Pumping lemma** to determine whether or not they are regular.

3 Finite Automata

A **deterministic finite automata** consists of

- alphabet A
- set of states $S = S_0, S_1, S_2 \dots S_n$
- transition function $f : S \times A \rightarrow S$

The transition function given a state S_i and input a changes the machine to state S_j . By adding

- start state S_0
- final states $F \subset S$

we create a f.a. that can be said to *accept a language*. The language accepted by a fa can be defined as $LM = \{w | S_0 \xrightarrow{w} \text{stops in final state}\}$

3.1 Deterministic versus Non-Deterministic

A deterministic fa is one where the number of arrows leaving each state is exactly equal to $|A|$. In a **non-deterministic** fa the number of arrows leaving a state is ≥ 0 . Additionally, the same word may lead to different states from the same state. In terms of functions, we consider the transition function as $f : S \times A \rightarrow S$ for a dfa and $f : S \times A \rightarrow \mathcal{P}(S)$ for a ndfa.

We describe a procedure for converting a ndfa into a dfa.¹

1. List ES_0 , the set including the initial state and all states reachable without consuming any characters
2. For $a \in A$, list the set of states reachable from ES_0 by consuming a
3. Repeat 2. until no new sets are produced (including \emptyset)
4. Draw the new dfa by treating each unique set as a state, where ES_0 is the initial state and any set containing a final state from the ndfa is a final state

3.2 Regular Expressions to fa

The collection of languages accepted by fa is exactly equal to the regular languages. We can therefore define a simple fa for each of the regular expressions in section 2.1.

- e
- a
- $a \vee b$
- ab
- a^*

These can be used as building blocks to construct fa for more complex regular expressions. When two of these are chained together, a new state must be added where all final states can reach this new state using e .

3.3 Simplifying

3.4 Finding Language of fa

¹See worksheet problem 7 for example.