## Computer Theory

Background

This topic deals with things you can compute mechanically, how fast and how much space it takes to do so.

People like Aslonzo Church, Stephen Kleene, Emi l Post, Andrei Andreevich Markov, John von Neumann, and Alan Turing independently came up with building blocks for mathematical algorithms.

This topic deals with machines that use rules to accept or reject input.

**FSM**: Finite State Machine

Simplest computation model

Limited memory (cannot store or count strings)

**CFL:** Context Free Language

Higher level computation model

**Turing machine:**

High level computation model

**Undecidable:**

Problems that cannot be solved mechanically

Undecidable

Turing Machine

CFL

FSM

**Lesson 0**

Languages

Finite State machine/ Finite Automata (Prerequisites) **S-A-S-L**

**Symbol**

Example a,b,c,0,1,2,3

**Alphabet**: Collection of symbols

Example

**String**: Sequence of symbols

Example aa,bb,ab

**Language**: Set of strings

Example

Set of all strings of length 2

Example

Set of all strings that begin with a

infinite language

Words

These are any nonempty strings of alphabet characters

Example

In , etc are words.

Powers of Σ

Example: considering the alphabet

Set of all strings of length 0

or empty set

Set of all strings of length 1

Set of all strings of length 2

Set of all strings of length n

Infinite set

Cardinality: Number of elements in a set

1 element

2 elements

4 elements

based on 2 elements in alphabet

(Sigma star/Kleene Closure/Kleene Star)

set of all possible strings of all lengths over

infinite set

(infinitely many words, each of finite length.)

**Lesson 1**

Recursive Definitions

This is a method to define a language

Recursive Enumerable (RE) Languages

Can be accepted/recognized by Turing Machine (Turing Recognizable language)

It will enter the final state for the strings of language.

It may/may not enter rejecting state for strings not in the language.

Recursive Language (REC)

Can be decided by Turing Machine (Turing Decidable Language)

It will enter the final state for the strings of language.

It will not enter rejecting state for strings not in the language.

RE

REC

Why do we need to define sets using a recursive definition?

Because sometimes the number of elements in a set are infinite, and a recursive definition gives us a finite definition of **infinite sets**

**How to write a recursive definition**

1. Specify some of the basic elements in the set.
2. Give some rules for how to construct more elements in the set from the elements that we know are already there.
3. Say that there are no other elements in the set except those constructed using steps 1 and 2.

**Example: EVEN**

1. 2 is in EVEN
2. If x is in EVEN, and y is in EVEN, x+y is in EVEN.
3. The only elements in the set EVEN are those that can be produced from RULEs 1 and 2 above

**Lesson 2**

Regular languages

A language is said to be regular if and only if some finite state machine recognizes it.

A language is not regular if it not recognized by a finite state machine or require memory.

Example

The language has a repeating pattern of ababb.

We cannot store this pattern.

This language is not regular.

Example

The language has the same number of a’s as it does b’s.

We cannot store the number of a’s.

This language is not regular.

Operations on regular languages

Example

**Union**

**Concatenation**

**Star**

infinite set

Theorems

The class of Regular languages is closed under UNION.

i.e. is also a regular language

The class of Regular languages is closed under UNION.

i.e. is also a regular language

**Lesson 3**

Regular Expressions

This is a method to define a language

|  |  |
| --- | --- |
| **Character** | **Meaning** |
| **\*** | Kleene Star. Unknown or undetermined power. Can be empty |
| **+** | Or. |
|  | Empty set. |
| **()** | Group. Can be empty |
| **[]** | Range. Cannot be empty |

Notes:  
 can always be

can always be

**Lesson 4**

Finite Automata

This is a method to define a language

FA without output

FA without output

Moore

Machine

Mealy

Machine

NFA

NFA

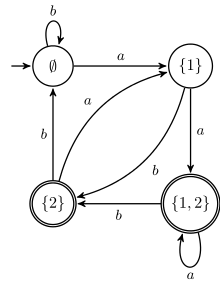
DFA

**FSM**: Finite State Machine

Simplest computation model

Limited memory

How to draw Finite Automata



Circles: States

Double-Circles: Terminating State

Edges: Transitions

Edge-labels: Inputs

How to represent Finite Automata

Q Set of all states

Σ Inputs

Start state/initial state

Start state/initial state

Transition function, maps

Example

**A close up of a blackboard

Description automatically generated**

Q

Σ

Transition function, maps

Transition function

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| A | C | B |
| B | D | A |
| C | A | D |
| D | B | C |

**Lesson 2**

Languages, Recursive Definitions, Regular Expressions

**Lesson 2**

Use ctrl + r to execute (build + run) your code.

#include <iostream> // Standard c++ library.

using std //Included in iostream.

#include “fact.h” //User-defined header

int main() {

int factArg = 0 ;

cout << "Factorial of: ";

cin >> factArg;

findFact(factArg);

return 0;

}