COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

January 4, 2023

Introduction, Logistics, Motivation etc.

What is this course about?

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.
- **COL 351** – *Design and Analysis of Algorithms:* Abstracted computational problems as algorithmic tasks, Algorithm design paradigms.

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.
- **COL 351** – *Design and Analysis of Algorithms:* Abstracted computational problems as algorithmic tasks, Algorithm design paradigms.
- **COL 352** - What is an algorithm? (Eg. Euclid's GCD) What does it mean to compute something?

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.
- **COL 351** – *Design and Analysis of Algorithms:* Abstracted computational problems as algorithmic tasks, Algorithm design paradigms.
- **COL 352** - What is an algorithm? (Eg. Euclid's GCD) What does it mean to compute something?
  1. Can we solve all problems using computers?

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.
- **COL 351** – *Design and Analysis of Algorithms:* Abstracted computational problems as algorithmic tasks, Algorithm design paradigms.
- **COL 352** - What is an algorithm? (Eg. Euclid's GCD) What does it mean to compute something?
    1. Can we solve all problems using computers?
    2. If we can design a program to solve a problem, how do we verify that it is indeed doing what it is supposed to do?

What is this course about?

- **COL 226** - *Programming Languages:* understand major paradigms of programming, implementation in languages, desgin/architecture considerations.
- **COL 351** – *Design and Analysis of Algorithms:* Abstracted computational problems as algorithmic tasks, Algorithm design paradigms.
- **COL 352** - What is an algorithm? (Eg. Euclid's GCD) What does it mean to compute something?
  1. Can we solve all problems using computers?
  2. If we can design a program to solve a problem, how do we verify that it is indeed doing what it is supposed to do?
  3. If we can solve a problem in theory, can it still be solvable in practice?

Automata and Theory of Computation

Dictionary definition

noun (plural automata)

1. A moving mechanical device made in imitation of a human being.
2. A machine that performs a function according to a predetermined set of coded instructions

Automata and Theory of Computation

Dictionary definition

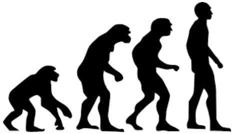
noun (plural automata)

1. A moving mechanical device made in imitation of a human being.
2. A machine that performs a function according to a predetermined set of coded instructions

Goals for this course: Study and explore a series of abstracted models of computation.

Automata and Theory of Computation

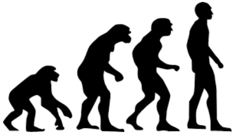
Goals for this course: Study and explore a series of abstracted models of computation.



- Start with the most limited
- Move towards the most versatile*
- Prove their power and limitations
- Determine the relationships between models

Automata and Theory of Computation

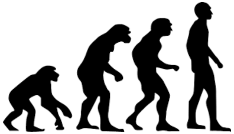
Goals for this course: Study and explore a series of abstracted models of computation.



- Start with the most limited
- Move towards the most versatile*
- Prove their power and limitations
- Determine the relationships between models

Automata and Theory of Computation

Goals for this course: Study and explore a series of abstracted models of computation.



- Start with the most limited
- Move towards the most versatile*
- Prove their power and limitations
- Determine the relationships between models

Main Goal: Learn to formally reason about computation!

Why theory of computation?

Why theory of computation?

- New ways of thinking about computing, different models (Finite automata, grammars, pushdown systems, Turing Machines), different perspectives, build a vocabulary to state your problem!

Why theory of computation?

- New ways of thinking about computing, different models (Finite automata, grammars, pushdown systems, Turing Machines), different perspectives, build a vocabulary to state your problem!
- Theory often drives practice.

Why theory of computation?

- New ways of thinking about computing, different models (Finite automata, grammars, pushdown systems, Turing Machines), different perspectives, build a vocabulary to state your problem!
- Theory often drives practice.
- Mathematical models of computation predated computers (present-day example: we "know" a lot about quantum computing, but no large-scale quantum computers have been built yet!)

Why theory of computation?

- New ways of thinking about computing, different models (Finite automata, grammars, pushdown systems, Turing Machines), different perspectives, build a vocabulary to state your problem!
- Theory often drives practice.
- Mathematical models of computation predated computers (present-day example: we "know" a lot about quantum computing, but no large-scale quantum computers have been built yet!)
- Math is good for you! ☺

"Computer Science is no more about computers than astronomy is about telescopes"

Edgar Dijkstra *(Turing award 1972)*

Logistics

Quizzes	30%
Minors	35%
Major	35%

Attendance: 75% (to qualify for audit pass and remajor).
Audit Pass: 40% total with at least 50% in Major+Minors.

Teaching Assistants: Sourav Bansal (cs5180421), Suraj Patni (csy217550), Sanaa Siddiqui (csy227516), Surbhi Rajput (csy227519), Harsh Singh Chauhan (mcs222052)

Forums: Piazza (Q & A), Moodle (?), MS Teams (Announcements).

Textbooks:

- Automata and Computability - Dexter Kozen.
- Introduction to Theory of Computation - Michael Sipser.

Logistics contd.

Web: https://sites.google.com/view/nikhilbalaji/toc2023

Prerequisites: COL 202, COL 351 or equivalent (if you don't satisfy these, email me!)

- Proof techniques - construction, contradiction, induction, counting techniques.
- Sets and set operations, relations Functions – injections, surjections, and bijections, cardinality.
- Countability and uncountability, diagonalization.
- Equivalence relations, partial orders

Logistics contd.

Web: https://sites.google.com/view/nikhilbalaji/toc2023

Prerequisites: COL 202, COL 351 or equivalent (if you don't satisfy these, email me!)

- Proof techniques - construction, contradiction, induction, counting techniques.
- Sets and set operations, relations Functions – injections, surjections, and bijections, cardinality.
- Countability and uncountability, diagonalization.
- Equivalence relations, partial orders

Let's get started!

Modelling computation

Modeling a computing device: an automaton

- Input
- Processing
- Output

Modelling computation

Modeling a computing device: an automaton

- Input
- Processing
- Output

Example:

- Consider a bulb with a switch.
- Its input are: on/off.
- Behavior during the day is: “on off on off”
- When light is off, I can turn it on and vice versa.

Inputs

Inputs

- Domain of inputs/actions $\Sigma = \{\text{on}, \text{off}\}$.
- Σ is called the Alphabet.
- Elements of the alphabet are Letters.
- Inputs or behaviors are strings or sequences of letters, called Words.
- Example: "on off on off on off" is a word in Σ^*.

Inputs

- Domain of inputs/actions $\Sigma = \{\text{on}, \text{off}\}$.
- Σ is called the Alphabet.
- Elements of the alphabet are Letters.
- Inputs or behaviors are strings or sequences of letters, called Words.
- Example: "on off on off on off" is a word in Σ^*.

Set of words over Σ

- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$

Inputs

- Domain of inputs/actions $\Sigma = \{\text{on}, \text{off}\}$.
- Σ is called the Alphabet.
- Elements of the alphabet are Letters.
- Inputs or behaviors are strings or sequences of letters, called Words.
- Example: “on off on off on off” is a word in Σ^*.

Set of words over Σ

- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$
- What is Σ^0?

Inputs

- Domain of inputs/actions $\Sigma = \{\text{on, off}\}$.
- Σ is called the Alphabet.
- Elements of the alphabet are Letters.
- Inputs or behaviors are strings or sequences of letters, called Words.
- Example: "on off on off on off" is a word in Σ^*.

Set of words over Σ

- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$
- What is Σ^0? $= \{\epsilon\}$

Inputs

- Domain of inputs/actions $\Sigma = \{\text{on}, \text{off}\}$.
- Σ is called the Alphabet.
- Elements of the alphabet are Letters.
- Inputs or behaviors are strings or sequences of letters, called Words.
- Example: "on off on off on off" is a word in Σ^*.

Set of words over Σ

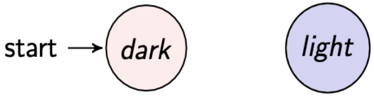
- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$
- What is Σ^0? $= \{\epsilon\}$
- $\Sigma^+ = \Sigma^* \setminus \Sigma^0$

Processing

- States: current information which allows to process the next letter/input.

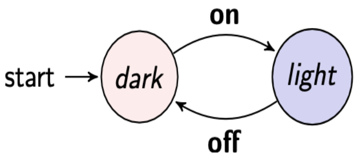
Processing

- States: current information which allows to process the next letter/input.

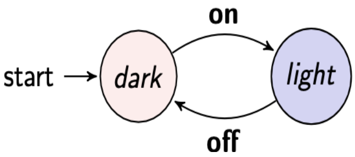


Processing

- States: current information which allows to process the next letter/input.
- Transitions: read an input and move from a state to another



Processing



Which state is reached after reading the following words?

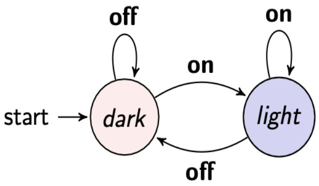
- "on off on"

Processing

Which state is reached after reading the following words?

- “on off on”
- “off off”

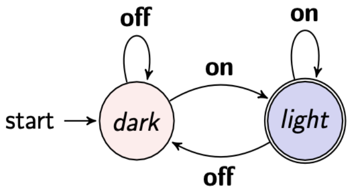
Processing



Which state is reached after reading the following words?

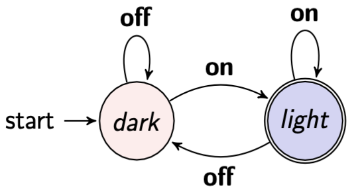
- "on off on"
- "off off"
- "on (off on)*"

Outputs



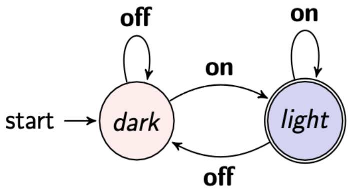
- Some states are designated "accept" or "final".
- If an input word moves to accept state, then its accepted, else rejected.

Outputs



- Some states are designated "accept" or "final".
- If an input word moves to accept state, then its accepted, else rejected.
- What are the words accepted by above automaton?

Language of an automaton



All words accepted by an automaton form its language.

Languages

Languages

Let Σ be a finite alphabet. A language over Σ is a subset of Σ^*.

Languages

Let Σ be a finite alphabet. A language over Σ is a subset of Σ^*.

Examples:

1. L_1: words that end with 1

Languages

Let Σ be a finite alphabet. A language over Σ is a subset of Σ^*.

Examples:

1. L_1: words that end with 1
2. L_2: words that have n 0's followed by n 1's, for each non-negative integer n.

Languages

Let Σ be a finite alphabet. A language over Σ is a subset of Σ^*.

Examples:

1. L_1: words that end with 1
2. L_2: words that have n 0's followed by n 1's, for each non-negative integer n.
3. L_3: words of length a prime number.

Languages

Let Σ be a finite alphabet. A language over Σ is a subset of Σ^*.

Examples:

1. L_1: words that end with 1
2. L_2: words that have n 0's followed by n 1's, for each non-negative integer n.
3. L_3: words of length a prime number.

Exercise:

- Give an example of a finite and infinite language.
- Give an example of a language that contains any language over Σ.

Languages as Problems

Computing problems can be seen as languages!

Languages as Problems

Computing problems can be seen as languages!

Consider

- is $n \in \mathbb{N}$ a prime?

Languages as Problems

Computing problems can be seen as languages!

Consider

- is $n \in \mathbb{N}$ a prime?
- with $\Sigma = \{a\}$, ask if $a^n \in L_3$.

Languages as Problems

Computing problems can be seen as languages!

Consider

- is $n \in \mathbb{N}$ a prime?
- with $\Sigma = \{a\}$, ask if $a^n \in L_3$.
- That is, all the yes instances of the problem are in the language and only these are!

Languages as Problems

Computing problems can be seen as languages!

Consider

- is $n \in \mathbb{N}$ a prime?
- with $\Sigma = \{a\}$, ask if $a^n \in L_3$.
- That is, all the yes instances of the problem are in the language and only these are!
- Can there be an automaton for this?

Languages as Problems

Computing problems can be seen as languages!

Consider

- is $n \in \mathbb{N}$ a prime?
- with $\Sigma = \{a\}$, ask if $a^n \in L_3$.
- That is, all the yes instances of the problem are in the language and only these are!
- Can there be an automaton for this?

We will go back-and-forth between languages and problems.