Introduction to the Theory of Computing

Bilal Alqudah , PhD

Lecture Outline

- Course Overview
- Languages
- Models of Computation

Lecture Outline

- Course Overview
- Languages
  - Human Languages
  - Programming Languages
  - Formal Languages
- Models of Computation

Human Languages

English, French, Danish, Hungarian, Urdu, Cantonese, . . .

Which sentences below are true, meaningful, grammatical?

- vgrlum qp#d*n aoiuiui brubrubrubru 3jc6r
- dog homework ate my. My
- Erpa shumblers groffed dulky brubrus.
- Iron is denser than styrofoam.
- The textbook for this class has exactly ten pages.
- Two is less than three.
- The loneliness sat for cast iron subtraction.
- George W. Bush is smarter than a dead slug.

Programming Languages

C, Java, Python, Prolog, Pascal, . . .

When is a program:

- syntactically correct?
- compilable?
- free from fatal exceptions at runtime?
- free from deadlock or infinite loops?
- a correct implementation of its specification?

Grading

- First exam 20%
- Second exam 20%
- Homework 10%
- Final 50%

Formal Languages

- An alphabet, Σ, is a finite set of symbols, e.g. $\{\clubsuit, \dagger, \oplus, \nabla\}$.
- A string is a sequence of zero or more symbols from Σ, e.g. $\clubsuit \oplus \oplus$ or $\dagger \dagger \dagger$.
- We'll write ϵ to denote the empty string (the string consisting of zero characters).
- We'll write Σ^* to denote all strings consisting of symbols from Σ.
- A language, L, is a subset of Σ^*.

Formal Language, example

- let $\Sigma = \{\mathtt{a}, \mathtt{b}, \wedge, \vee, \neg, \mathtt{(}, \mathtt{)}\}$.
- We could define L_0 to be the set of all strings that represent syntactically correct boolean formulas.
- We could define L_1 to be the set of all strings that represent boolean tautologies.

In logic, a **tautology** is a formula that is true in every possible interpretation

- Example strings:

  $\mathtt{a} \wedge \mathtt{b}$ is in L_0 but not L_1.

  $\mathtt{a} \vee \neg\mathtt{a}$ is in L_0 and L_1.

  $\mathtt{(a} \vee \mathtt{b} \vee \mathtt{(}\neg\mathtt{a} \wedge \neg\mathtt{b))}$ is in L_0 and L_1.

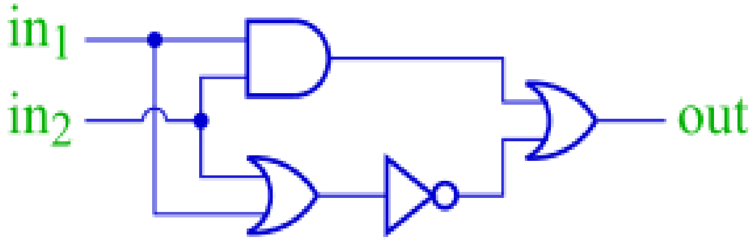
  $\mathtt{(a} \vee \wedge \mathtt{b}$ is not in L_0 and not in L_1.

- We can write a computer program that determines whether or not an arbitrary string is in L_0 or in L_1.

Lecture Outline

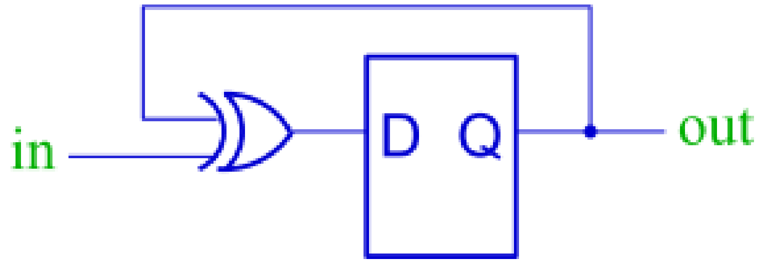
- Course Overview
- Languages
- Models of Computation
  - Logic gates
  - Finite automata
  - Push-down automata
  - Turing machines

Logic Gates



- “Language” is set of all inputs that produce a true output value.
- Any circuit only accepts fixed number of bits for input – not a true language in the sense described above.

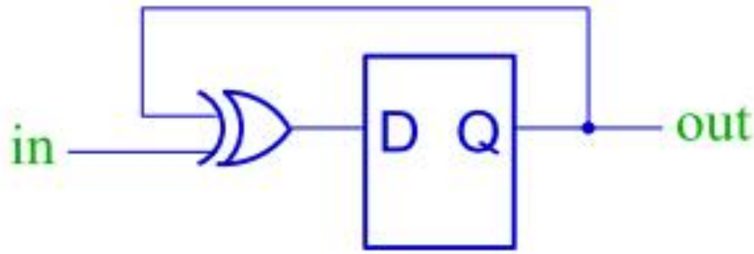
Finite Automata



Initially: out = 0

- Logic gates plus a fixed number of bits of storage.
- Can process an arbitrarily long strings.
  The example circuit accepts all strings with an odd number of ones.
- The languages that can be recognized by finite automata are very restricted.
  - For example, finite automaton can't recognize inputs that have more 1's than 0's or mathematical formulas where the parentheses balance properly.

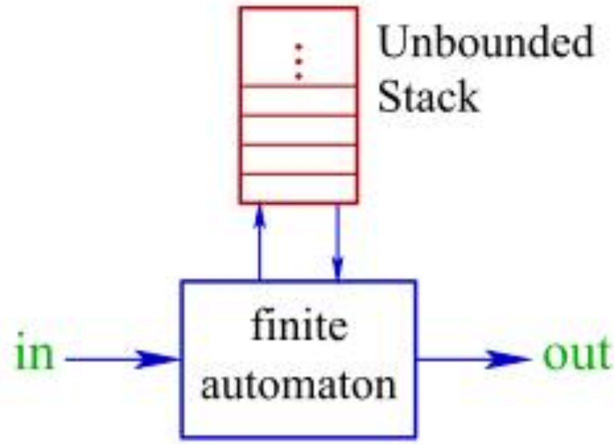
Finite Automata



Initially: out = 0

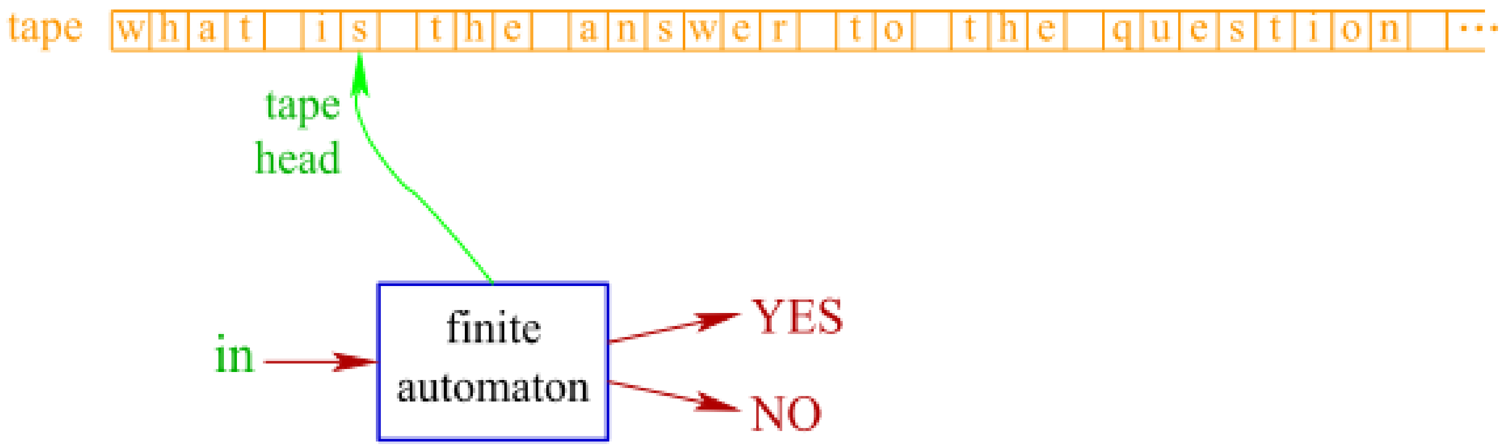
- Logic gates plus a fixed number of bits of storage.
- Can process an arbitrarily long strings.
  The example circuit accepts all strings with an odd number of ones.
- The languages that can be recognized by finite automata are very restricted.
  - For example, finite automaton can't recognize inputs that have more `1`'s than `0`'s or mathematical formulas where the parentheses balance properly.
  - Intuitvely, this is because a machine with a fixed number, k, bits of storage can only count to 2^k. After reading $2^k + 1$ `1`'s, the machine must be in a state that it was in before.

Push-Down Automaton



- A finite automaton with an unbounded stack.
- Can recognize properly balanced parantheses and other languages with nesting structures.
- Most programming languages have syntaxes with this kind of nesting structure.
- More general than finite automata, but still limited.
  - Cannot recognize the language of all strings whose lengths are prime numbers.

Turing Machines



- A finite automaton with an unbounded read/write tape.
- Can recognize any language that is recognizable by ANY computer!
- Yet, there are problems that a Turing machine cannot solve.

What's the "Theory of Computing"?

Here's the kinds of questions we consider:

- **1.** What problems are possible/impossible to solve with a computer?
- **2.** What problems are easy/hard to solve with a computer?
- **3.** What is a computer?
- **4.** Do do the answers to 1 and 2 depend on the answer to 3?

What is a computer?

- Finite state machines:
  A fixed amount of memory.
- Pushdown automata:
  An infinite amount of memory, arranged as a stack.
- Turing machines:
  An infinite amount of memory, arranged as a tape with a "head" that can read, write, and move left or right.
  A Turing machine is very simple but can perform any computation that a conventional comptuter can do. In fact, we don't know of anything that can compute something that a Turing machine cannot.