
بسم الله الرحمن الرحيم

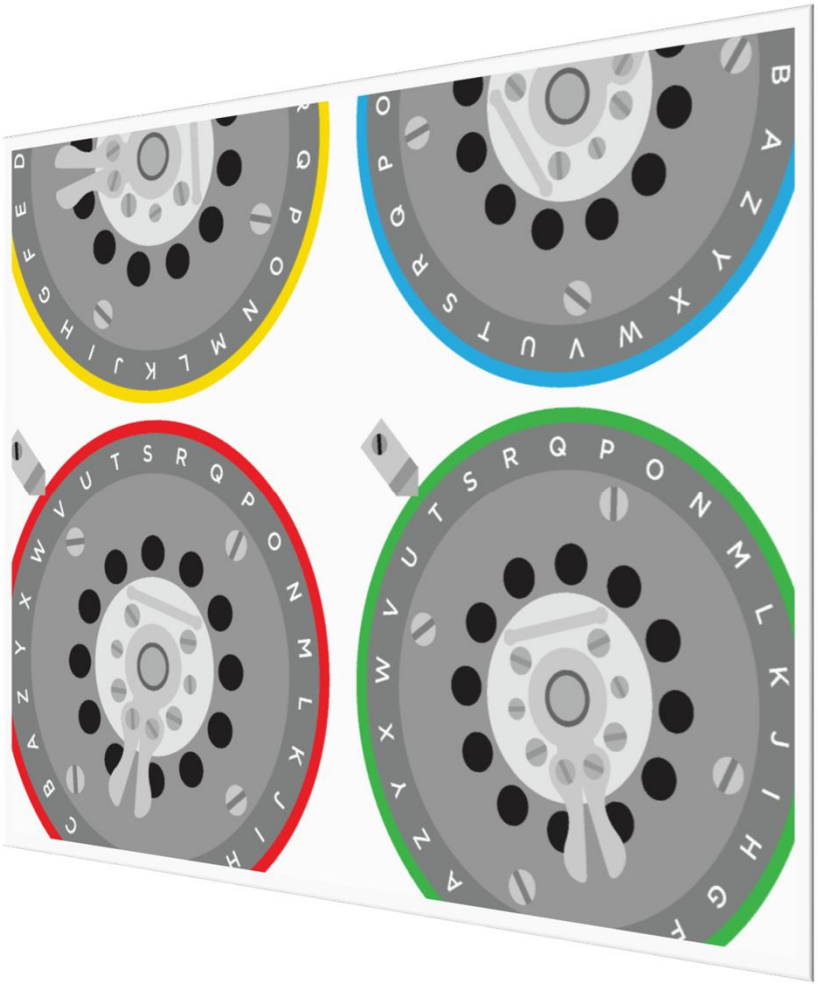
Formal Languages & Automata

Ali Shakiba
Vali-e-Asr University of Rafsanjan
<ali.shakiba@vru.ac.ir>

Goals of the Course

- Understand the fundamental capabilities and ultimate limitations of computation.
  - AKA theory of computation
- Introduction to the theory of computational complexity.

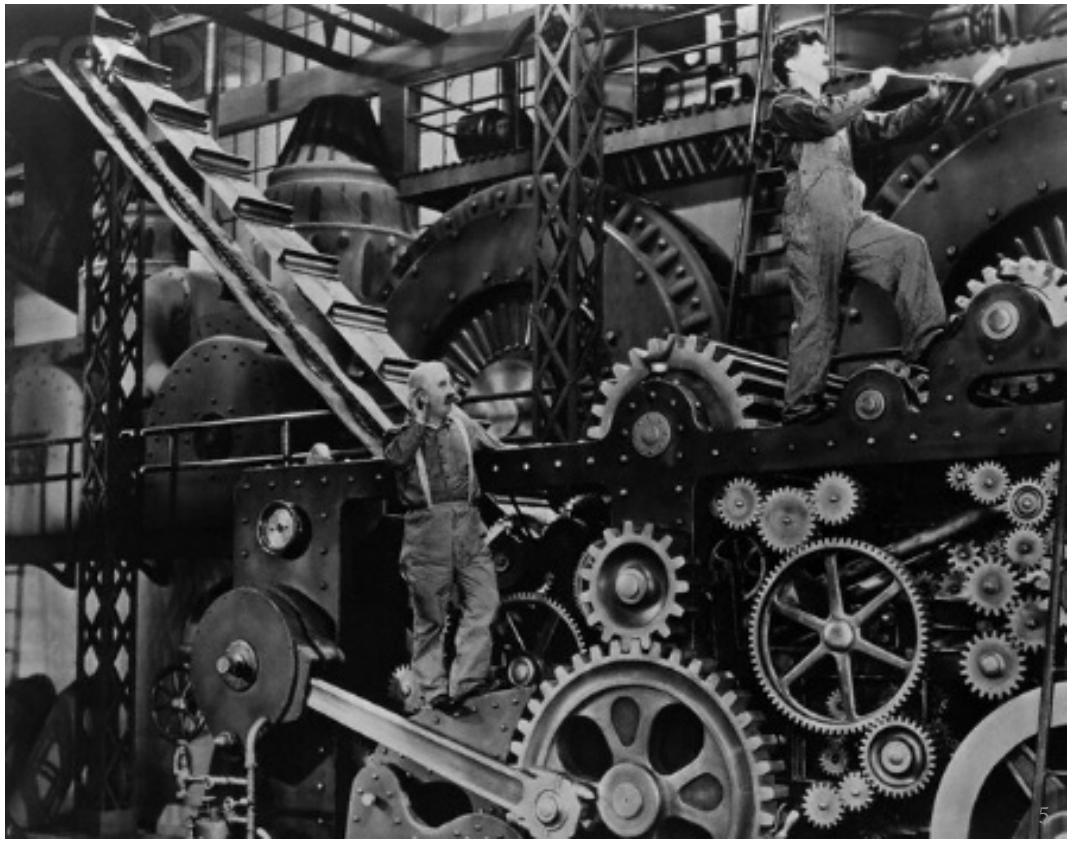
What is Computation?



Transfer of Human labor to Machines

Industrial Revolution

—19th century



Transfer of Human intellectual to Machines

Information Evolution

Questions & Answers

Arithmetic as a mental operation

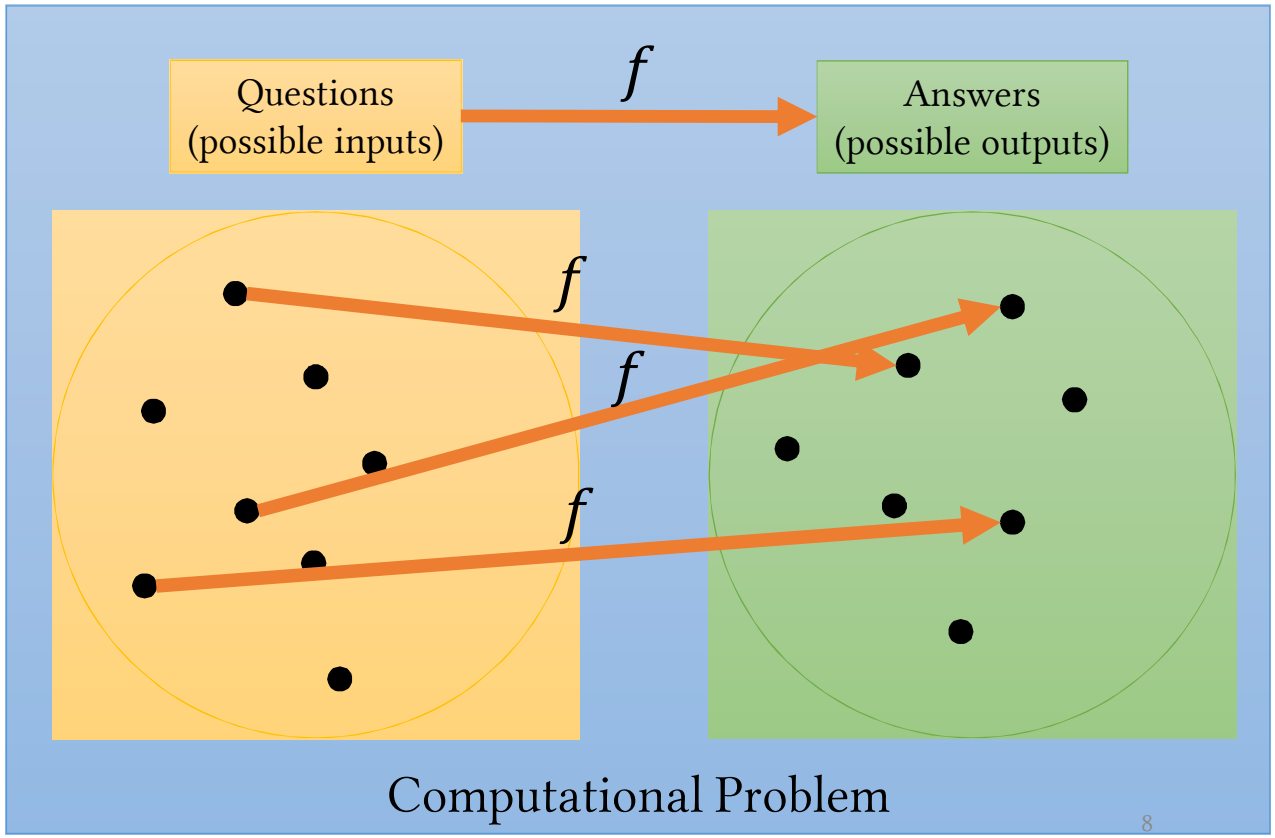
\+ physical operations

PRINCIPLE OF BEHAVIORAL EQUIVALENCE

The Functional Model

Assumes Computations:

1. Read an input, think for a while, write an output, and halt
2. Just the "relation between input and output" is important



The Imperative Model

- What about computations that do not "compute a function"?
  - deleting a file, anti-lock break system, …

sequences of imperatives which manipulate representations

More inclusive than functional model

advantageous

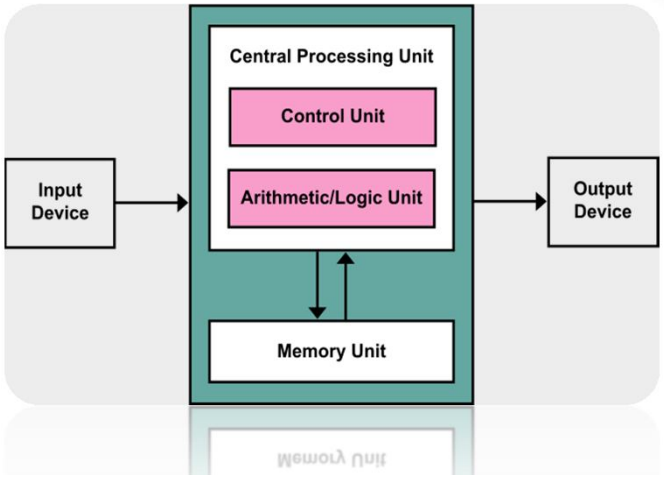
dis-advantageous

Hard to reason about programs at this level

Digital Computers

Defining characteristics:

1. Programmability
2. Uniform meta-representation for all data-types

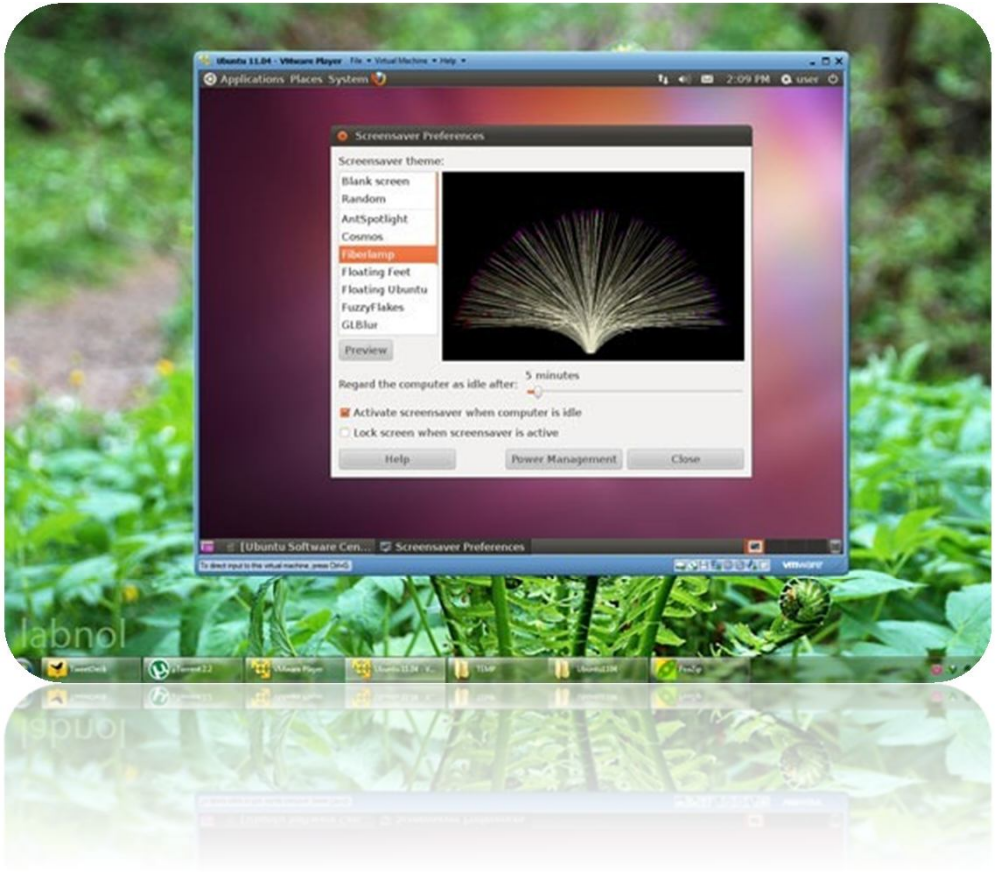


Meta-programming

debuggers, static checkers, profilers, compilers, source-code management systems

"While programmability makes computer hardware viable, meta-programmability makes software economically viable."

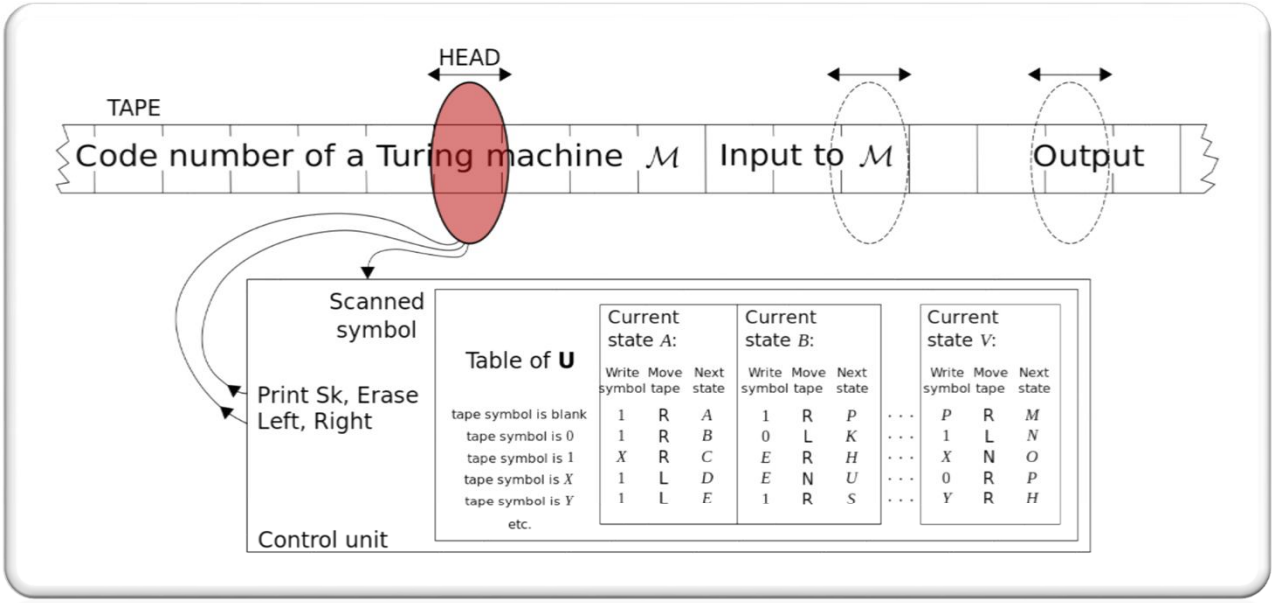
Simulation



- Can one computer always simulate another if it has enough memory?
- Is there an instruction set that is sufficient for simulating any kind of computer with any kind of instruction set?

Universality

Turing-completeness
Turing equivalence



"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine *U* is supplied with a tape on the beginning of which is written the S.D ["standard description" of an action table] of some computing machine *M*, then *U* will compute the same sequence as *M*"

Church-Turing Hypothesis (Church's Thesis)

Any computable function is computable by a Turing machine.

The Limits of Computation

Halting problem: automated detection of infinite loops

Halting problem is un-computable.

```
void contrarian(int input) {
   if(halts(contrarian, input)
      while(true) {
         // loop infinitely
      }
}
```

After all, it is Computer ***Science***

- This class will be about some of the foundational theories of computer science.

- Science is also about experimentation.
  - Try things and see what happens.
  - Learn from the experimental results.

Some Bold Assertions

- The abstract models in computability theory deal with:
  - Computers and software that currently exist
  - Computers and software that will exist
  - Computers and software that we can only imagine
- We are not concerned with **optimization**.
- Rather, we are concerned with the question of possibility.
  - What computers and software can and cannot do.

Overview of the Course

- We will begin with the study of languages.
  - In a very formal way, hence the term *formal languages*.
  - Which sentences belong to a language, and which ones don't?
- We will design small machines called automata.
  - An automaton designed for a particular language will "execute" when given a sentence as input and decide whether or not the sentence belongs to the language.

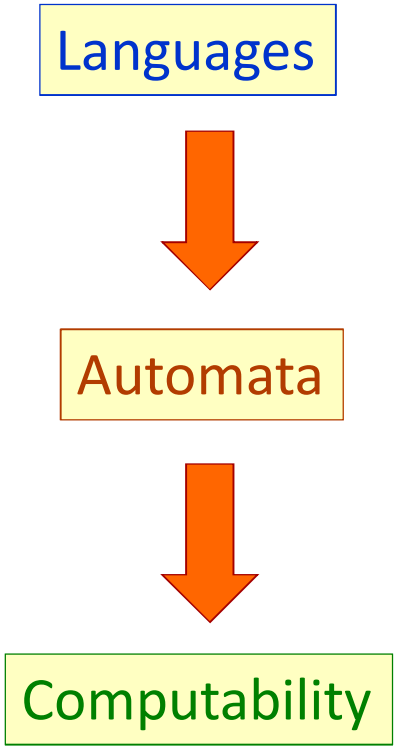
Overview of the Course, *cont'd*

- Our first languages will be very simple, and so will be their automata.
  - For example, an automaton may have very limited memory.
- What operations can we perform on sentences from a language and have the result be in the language?
- As the languages we study become more sophisticated, so will their automata.

Overview of the Course, *cont'd*

- The automata are the models that will enable us to study computation.
- Can we design an automaton that can execute any algorithm that a modern computer can?
  - How complex would this automaton have to be?
  - What are the implications if the automaton cannot execute an algorithm?
- Are there algorithms that are inherently very hard to execute? That are impossible to execute by any computer?

Our Plan (Very High Level Version)



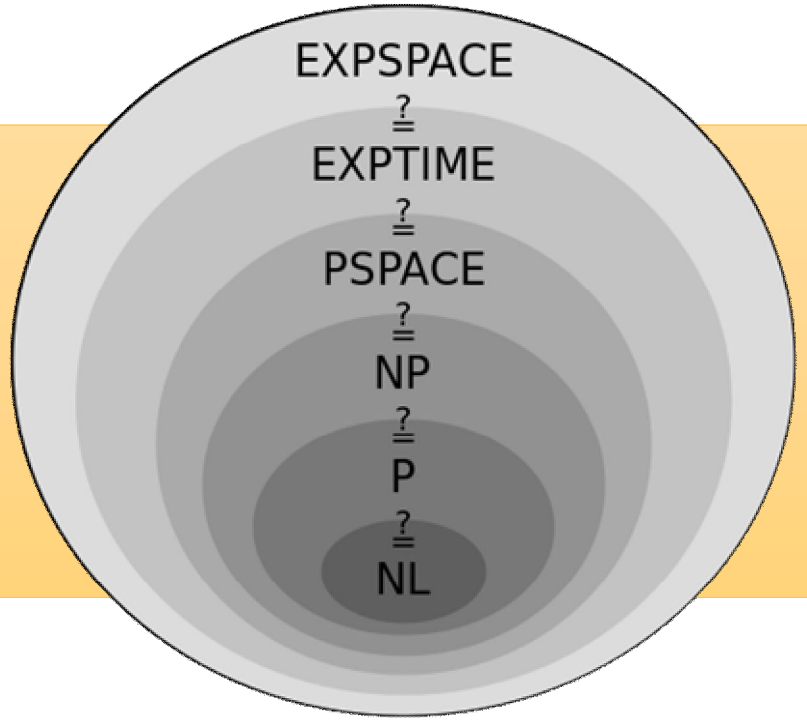
The "Language Game"

Abstract models that enable
us to study computation.

Which algorithms are computable?
Which ones are impossible to compute?

Computational Complexity

general study of what can be achieved within limited time and/or other limitations on natural computational resources

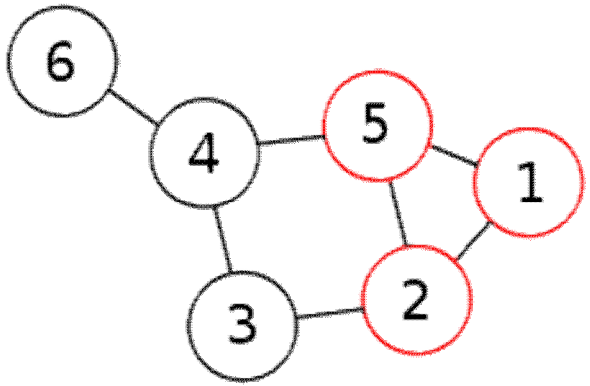


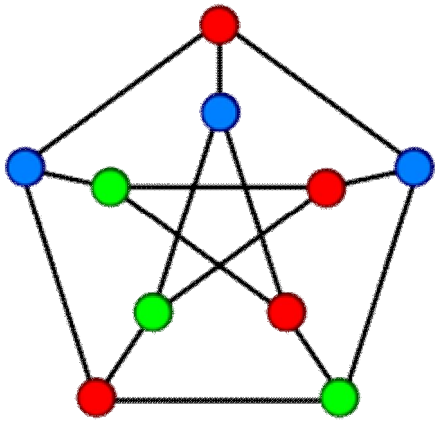
Two Concerns of Complexity

1. determination of the complexity of any well-defined task
2. obtaining an understanding of the relations between various computational phenomena

P, NP, and NP-completeness

$(x \vee x \vee y) \wedge$
$(\neg x \vee \neg y \vee \neg y) \wedge$
$(\neg x \vee y \vee y)$





These three seemingly different computational tasks are computationally equivalent.

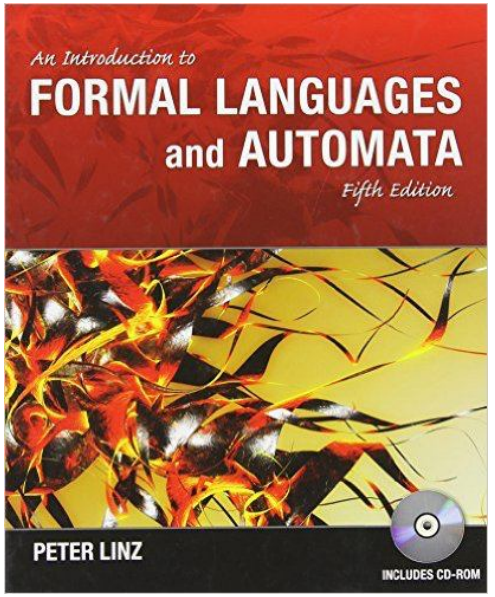
Other advanced topics

- Randomness
- Knowledge
- Interaction
- Secrecy
- Learning
- Approximation
- Average-case complexity
- Space complexity
- ...

Syllabus

Title	Approx. Time (Weeks)	References
Regular Languages and Finite Automaton	4	Chapters 1-4
Context-free Languages and Grammars; Pushdown Machines	4	Chapters 5-8
Turing Machines and Decidability	5	Chapters 9-12
Basic Complexity and Its Modern Applications	3	Chapter 14 + …

An Introduction to Formal Languages and Automata, 5th edition
Peter Linz
Jones & Bartlett Learning
2012 978-1-4496-1552-9



Software to Install Locally

- JFLAP
  - Java Formal Language and Automata Package
  - http://www.jflap.org

- There may be other software packages announced during the semester.

Grades

Title	Grade	Description
Exercises (Written + Programming) (at least 12 series)	5	Weekly
Midterm 1 Chapters 1-4	3	Sunday, 2nd Aban 1395
Midterm 2 Chapters 5-8	3	Tuesday, 16th Azar 1395
Final All the course material	9	See GOLESTAN
Excellence	+2	Extra credit
Total	**20 + 2**	

10% penalty for every late day.
100% penalty after 72 hours.

The Language Game

- Our study of formal languages starts out like a game.
- As with any other game, it has rules.
- The rules determine what sentences belong to the language.
- The goal of the game is simple:
  Given an arbitrary sentence, determine if it belongs to the language.

Some Basic Terms

- Let Σ represent a nonempty set of symbols called an alphabet.
- We can construct finite strings of symbols from the alphabet.

String Examples

- Let alphabet $\Sigma = \{a, b\}$.
- Then $abab$ and $aaabbba$ are strings on Σ.
- If we write

  $$w = abaaa$$

  it means that the string named w has the value $abaaa$.

By convention, we use lowercase letters $a, b, c, \ldots$ for elements in Σ and $u, v, w, \ldots$ for string names.

Some Basic Terms, *cont'd*

- If string $w = a_1a_2 \ldots an$ and string $v = b_1b_2 \ldots bm$ then $wv = a_1a_2 \ldots a_nb_1b_2 \ldots bm$ is the concatenation of strings w and v.
- String $w^R = a_n \ldots a_2a_1$ is the reverse of string w.
- $|w|$ is the length of string w.
  - The number of symbols in the string.
- λ is the empty string.
  - $|\lambda| = 0$
  - $\lambda w = w\lambda$

Substrings

- A substring of string w is any string of consecutive symbols of w.
- If $w = vu$, then the substring v is a prefix of w, and the substring u is a suffix of w.
  - Example: If $w = abbab$, then
    - All prefixes: $\{\lambda, a, ab, abb, abba, abbab\}$
    - All suffixes: $\{\lambda, b, ab, bab, bbab, abbab\}$
- If u and v are strings, then $|uv| = |u| + |v|$.

A Proof by Induction that $|uv| = |u| + |v|$

- Definitions: For all a in Σ and w any string on Σ

$$|a| = 1$$
$$|wa| = |w| + 1$$

- Basis: By definition, $|uv| = |u| + |v|$ is true for all strings v of length 1.
- Inductive hypothesis: Assume that $|uv| = |u| + |v|$ is true for all strings v of lengths $1, 2, 3, \ldots, n$.
- Let v have length $n + 1$ and let $v = wa$, where $|w| = n$.
- Then $|v| = |w| + 1$ by definition, and therefore

$$|uv| = |uwa| = \boxed{|uw|} + 1 = \boxed{|u| + |w|} + 1 = |u| + |v|$$

inductive hypothesis

More Basic Terms

- If w is a string, then w^n is the string obtained by repeating w for n times.
  - Special case: $w^0 = \lambda$ for all w.

- If Σ is an alphabet, then Σ^* is the set of strings obtained by concatenating zero or more symbols from Σ.
  - Σ^* always contains λ
  - $\Sigma^+ = \Sigma^* - \{\lambda\}$ is the set of all nonempty strings

The $*$ operator is known as the Kleene star.

- Even though Σ is finite, Σ^* and Σ^+ are infinite since there is no limit on the string lengths.