Free University of Bolzano, Prof. J. Nievergelt: Formal Languages, Sem 1, Fall 2006

Oct 02	Models of computation: Ruler and compass, systolic arrays, finite state machines
Oct 02 Lab	Kara: study examples and write a program for a problem of your own design
Oct 03	Various models of finite state machines and their applications
Oct 09	Theory of finite automata (FA) and their languages
Oct 09 Lab	Exorciser: designing finite automata
Oct 10	Regular expressions (RE) and their languages
Oct 16	Equivalence of deterministic (DFA), non-deterministic (NFA) finite automata, and REs
Oct 16 Lab	Exorciser: practice standard FA algorithms
Oct 17	Finite state machines that control external storage. Counter automata
Dec 04	Context-free grammars and languages (CFG, CFL), pushdown automata (PDA)
Dec 04 Lab	Exorciser: Parsing CFLs
Dec 05	Context-sensitive grammars and languages (CSG, CSL). Queue automata
Dec 11	Turing machines (TM), examples, Universal TM
Dec 11 Lab	Kara: design and test TMs
Dec 12	The concepts [un-]decidable, [non-]computable. Halting problem, BusyBeaver function
Dec 18	Complexity: the problem classes P, NP, and NP-complete
Dec 18 Lab	GraphBench: NP-complete problems and problem reductions
Dec 19	Reserve time, and review of the course.

Goal: An intuitive introduction to the theory of computation based on interesting examples.

Lecture notes: **www.jn.inf.ethz.ch "Education"**
Similar topics presented more formally can be found in many text books, e.g.

N. Blum: Theoretische Informatik - eine anwendungsorientierte Einfuehrung, Oldenburg, Muenchen, 1998
U. Schöning: Theoretische Informatik - kurzgefasst, Spektrum Akademischer Verlag, Heidelberg, 1997
I. Wegener: Theoretische Informatik, Teubner, Stuttgart 1 1993
E. Engeler, P. Laeuchli: Berechnungstheorie fuer Informatiker, Teubner, Stuttgart 1988
J.E. Hopcroft, R. Motwani, J.D. Ullman: Introduction to automata theory, languages and computation, 2nd edititon, Addison-Wesley 2001
M. Minsky: Computation: Finite and infinite machines, Prentice-Hall, 1967
H.R. Lewis, C.H. Papadimitriou: Elements of the theory of computation, Prentice-Hall, 1981
M. Sipser: Introduction to the theory of computation, PWS Publ. Co, Boston, 1997
J.E. Savage: Models of computation: Exploring the power of computing, Addison-Wesley 1998
R. Gregory Taylor: Models of computation and formal languages, Oxford Univ. Press, 1998

What is "theory of computation" ?

- Basic question: What can or cannot be computed with given resources:
  primitive operations, bounds on resources (time, memory=space, wire length, fanout, energy consumed, ..)
- Brief survey of the historical development:
  Computability, decidability (1930s, 40s: A. Church, E.L. Post, A.M. Turing, A.A. Markov, S.C. Kleene).
  Church's thesis: the most general concept of automatic computation.
  Formal languages and grammars as models of natural language (50s: N. Chomsky)
  Classes of automata: computing devices of limited power
  Relationship between classes of automata and languages
  Complexity (60s, 70s): how many operations and memory cells are needed.
- Content of our course: Automata, formal languages, computability, complexity.

Various guest lectures will introduce additional self-contained theoretical topics and applications.

Why a theory of computation? -> Nothing is more practical than a good theory!

- What kind of knowledge can you acquire today that will serve for your entire career, that will still be valid in the year 2040? (Hint: concepts that have already survived half a century of CS development.)
- Theory extracts the basic concepts that apply to any conceivable implementation of computing machines. These concepts are of timeless validity, in contrast to technology-specific and product-specific know-how.
- A firm mastery of basic concepts is a great “data compression technique”: many seemingly unrelated facts, presented in time-varying jargon, can be understood intuitively as instances of the same principle.

Tentative list of homework, problems will be explained in class

Models of computation

Hw 1.1: The quadratic equation $x^2 + bx + c = 0$ has roots $x_1, x_2 = (-b \pm \text{sqrt}(b^2 - 4c)) / 2$. Prove that in the ruler and compass construction shown in the notes, the segments x_1 and x_2 are the solutions of this equation.
Hw1.2: Prove: a sorting network using only adjacent comparisons must have ≥ n-choose-2 comparators.
Hw 1.3: Define an interesting model suitable for computing pictures on an infinite 2-d array of pixels. Study its properties. Is your model "universal", in the sense that it is able to compute any "computable picture"?
Hw 1.4: Program a Markov algorithm f that doubles the input string s: f(s) = ss.

Finite state machines, regular languages

Hw2.1: Analyze the behavior of the ticket machines used by some system of public transportation. Draw a state diagram capable of this behavior. Evaluate the design from the user's point of view.
Hw2.2: Design a finite state machine to operate a wrist watch that offers 4 different functions: time and date for 2 time zones, alarm, chronometer. Assume input and output devices typical of today's watches.
Hw2.3: Given 3 FAs over the alphabet A = {a,b,c,...,z}: one to accept 'begin', one to accept 'end', and one to accept any identifier in $A^+ = AA^*$. Construct a FA that distinguishes 'begin', 'end', and identifiers different from these 2 reserved words. Assume the input string is terminated by a special character #. Proceed in three steps: 1) merge them using e-transitions, 2) remove e-transitions, and 3) remove non-determinism.
Hw 2.4: Design a non-deterministic coin changer that can be used to change a 2 euro coin into two 1 euro coins, and vice versa. It has states 0 euro, 1 euro and 2 euro, where 0 euro is the accepting state. In the state 2 euro the machine goes non-deterministically to 0 euro or 1 euro. Construct an equivalent deterministic coin changer. Give an interpretation of the accepting state(s) of the result.
Hw 2.5: Invent another interesting example of a DFA M with equivalent states and apply the dynamic programming algorithm to obtain an equivalent M' with the minimum number of states.
Hw 2.6: Analyze the complexity of this dynamic programming algorithm in terms of the number of states, n, and the size of the alphabet.
Hw 2.7: Prove the following Thm: If states r, s are indistinguishable by words w of length $|w| \leq n = |S|$, then r and s are equivalent.
Hw 2.8: Logic design for the fsm "binary integer mod 3"
Hw 2.9: We saw a 2-state fsm serial adder. Prove: there can be no fsm multiplier for numbers of arbitrary size.

Context free grammars and languages

Hw 3.1: a) For the Algol 60 grammar G (simple arithmetic expressions) discussed, explain the purpose of the rule E -> AT and show examples of its use. Prove or disprove: G is unambiguous.
b) Construct an unambiguous grammar for the language of parenthesis expressions given by the grammar:
S -> S S | (S) | ( ).
c) The ambiguity of the "dangling else". Several programming languages (e.g. Pascal) assign to nested if-then[-else] statements an ambiguous structure. It is then left to the semantics of the language to disambiguate. Let E denote Boolean expression, S statement, and consider the 2 rules:
S -> **if** E **then** S, and S -> **if** E **then** S **else** S. Discuss the trouble with this grammar, and fix it.
d) Design a CFG for $L = \{ 0^i 1^j 2^k \mid i = j \text{ or } j = k \}$. Optional: Try to prove: L is inherently ambiguous.
Hw 3.2 (CSG): Prove that $L = \{w\, w \mid w \in \{0, 1\}^* \}$ is **not** context free, but is context sensitive.

Turing machines, computability

Hw 4.1: Surf the web in search of small universal TMs and Busy Beaver results. Report the most interesting findings, along with their URL.

Hw 4.2: Investigate the values of the Busy Beaver function B(2), B(3), B(4). A.K. Dewdney: The Turing Omnibus, Computer Science Press, 1989, Ch 36: Noncomputable functions, 241-244, asserts:
B(1) = 1, B(2) = 4, B(3) = 6, B(4) = 13, B(5) ≥ 1915. Have you found any new records in the competition to design Turing machines that write a large number of 1s and stop?
Hw 4.3: Design Turing machines UB and BU that convert natural numbers from unary to binary representation, and vice versa. State clearly what the initial and final configurations are.
Hw4.4: Prove that REGULARTM = { M | M is a Turing machine and L(M) is regular } is undecidable.
Hw 4.5: Prove that the two definitions of Turing computable numbers are equivalent. Prove that any rational number x = n/m is computable: sketch a non-halting TM that prints the digits in sequence, and a halting TM that prints the k-th digit b_k and halts, given k.