Informatics III
The Foundations of the Informatics

Peter Sanders & Alexandra Soskova

Seminars:

Thomas Käufl & Blagowest Kirilov

Sergej Petkov

Institut für theoretische Informatik, Algorithmik II

Sofia university

Materials

Slides, Exercises

Books: main Schöning

Wikipedia

Books

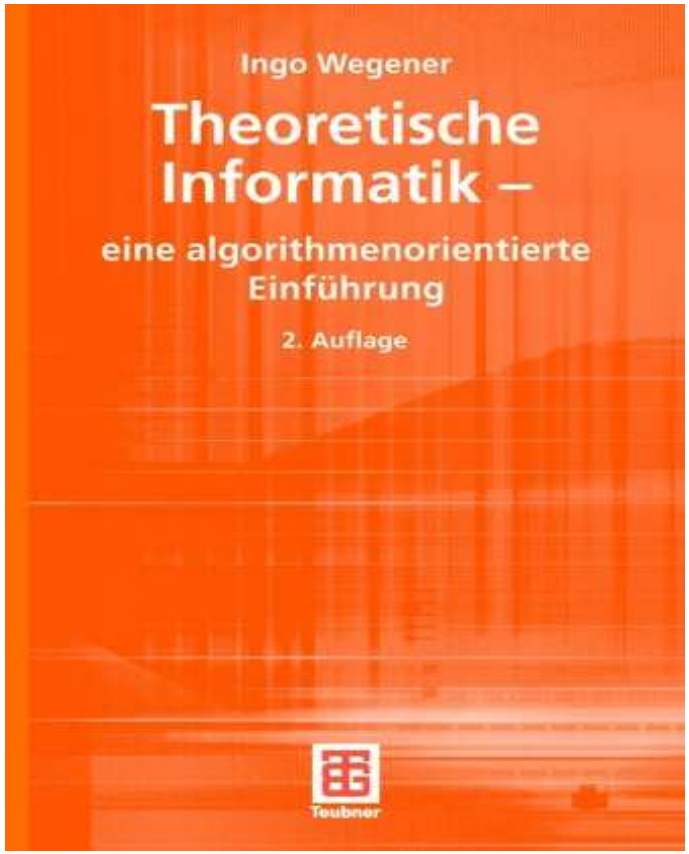
- Comparable contents
- Good for reading and additional motivation

Schöning: Theoretische Informatik — kurzgefasst

- 4. Auflage, Spektrum Verlag, 2001,
- Lectures: The closest content.
  Another chronological order
- Short but understandable
- The proofs are not so detailed

Wegener: Theoretische Informatik — eine algorithmenorientierte Einführung

- 2. Auflage, Teubner, 1999,
- Good application-oriented motivation
- Additional, non formal representation:



Hopcroft, Motwani, Ullman Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie

- 2. Auflage, Addison-Wesley, 2002,
- Good application-oriented motivation
- Not enough material so short explained
- Suitable for self study?
- $\neq$ old Hopcroft Ullman book

Lewis and Papadimitiou

Elements of the theory of computation

- 2. edition, Prentice-hall, 1998
- Good theoretical motivation
- Enough material and very good explained
- Suitable for self study

M. Sipser

Introduction the theory of computation

- PWS publ. comp, MIT, 1997
- Non formal description, good inution
- A lot of examples, a lot of material

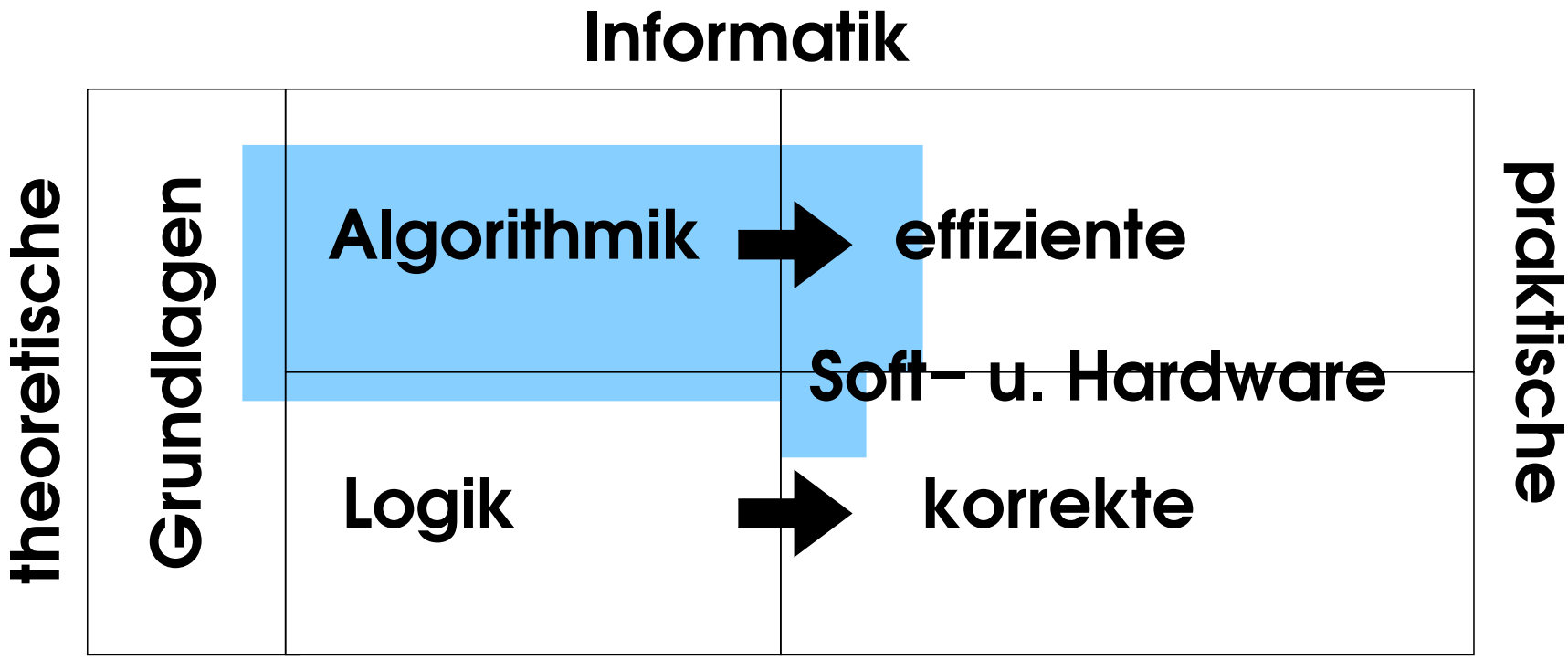
K. Manev

Introduction to discrete mathematics

- In Bulgarian, 2003
- Automata and grammars - good
- Computability and complexity- poor

- 300 BC Euclid and 9th century AD Al Khwarizmi
- 1646-1716 Gottfried Leibniz
- 1832 Charles Babbage - Analytical Engine (Ada Byron)
- 1900 David Hilbert (Entscheidungsproblem) Hilbert's Programm
- 1933 Gödel's - Incompleteness Theorem
- 1936 Alan Turing - Turing machines, Stephen Kleene - recursive function theory
- A. Church - lambda-computable functions
- Shepherdson and Sturgis — Register Computable Functions
- Chomski, Myhill, Nerode, D. Scott, Rabin - automata theory
- S. Cook - complexity

Informatics



Basic paradigms

The foundation of Computer science

Formal languages: How the computer communicates?

Automata theory: Formal model of computer.

Computability: What kind of problems could (not) be solved by computers?

Complexity: What we could (not) solve by efficient algorithms?

The reason of all of this?

- (in)direct use

  (algorithms, ideas, methods)

  in the programm languages, compilers, text manipulation,

  (linguistic), hardware engineering, software engineering,. . .

- Experience with typical discrete proofs

  $\rightsquigarrow$ theory of algorithms, logic, theory of computation, . . .

Formal Languages

Notations

Alphabet: a finite set of symbols (Σ)

Words (in Σ): strings (finite sequence) of symbols from Σ (w)

Formal language: a set of words in Σ (L)

Empty word: ε ($\{\varepsilon\} \neq \emptyset$!)

Concatenation of strings: '$\cdot$' associative.

Example: $ac \cdot bab = acbab$.

Concatenation of languages:

$L_1 \cdot L_2 := \{w_1 \cdot w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}$.

Example: $\{ab, ba\} \cdot \{aa, bb\} = \{abaa, abbb, baaa, babb\}$

Notation

The kth product (Powering): $w^0 := \varepsilon$, $w^n := w \cdot w^{n-1}$ for $n \geq 1$.

$L^0 := \{\varepsilon\}$, $L^n := L \cdot L^{n-1}$ for $n \geq 1$.

Example $a^3 = aaa$, $\{a, bb\}^2 = \{aa, abb, bba, bbbb\}$

Kleene star(hull): $L^* := \bigcup_{i=0}^{\infty} L^i$

(each particular word is finite!)

Example: $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

Σ^*: The set of all words in Σ.

$(\Sigma^*, \cdot, \varepsilon)$ is a monoid.

Positive hull: $L^+ := \bigcup_{i=1}^{\infty} L^i$

Complement language: $L^c := \Sigma^* \setminus L$

$|w|$(length): The number of the symbols in w

Notations

Let $w = u \cdot v \cdot x$

Prefix: u

Substring: v

Suffix: x

Reversal: $(c_1 c_2 \cdots c_k)^R = c_k \cdots c_2 c_1$

An Example of a formal Language $\Sigma = \{0,1\}$

- $L^{=} := \{0^n 1^n : n \geq 1\} = \{01, 0011, 000111, \ldots\}$
- $\{a^n b^n c^n : n \in \mathbb{N}\} = \{abc, aabbcc, aaabbbccc, \ldots\}$
- $\{ww : w \in \Sigma^*\} = \{\varepsilon, 00, 11, 0000, 0101, 1010, 1111, \ldots\}$
- $\{ww^R : w \in \Sigma^*\} = \{\varepsilon, 00, 11, 0000, 0110, 1001, 1111, \ldots\}$
- $L_P := \{w : w = w^R\} = \{\varepsilon, 0, 1, 00, 11, 000, 010, 101, 111, \ldots\}$

  Palindrome

  English: Ana, star, dog,lived, stop,war;

  German: gnudung, hangnah, kajak, lagerregal, reliefpfeiler;

  Bulgarian: Az obicham mach i boza; Nasila zakaraha svinete ni v sahara! - kaza Lisan ; "

  Latin: Sator Arepo Tenet Opera Rotas .

Example of a Formal Language

Balanced left and right parentheses $L_{()}$:

- $\varepsilon \in L_{()}$,
- if $u \in L_{()}$, $v \in L_{()}$ then $uv \in L_{()}$,
- if $u \in L_{()}$ then $(u) \in L_{()}$.

Algorithmic problems

Word problem: $w \in L$?

Equivalence: $L_1 = L_2$?

Specification: mathematical definition (see up), grammars,

acceptors=automata (machine), when the word $w \in L$

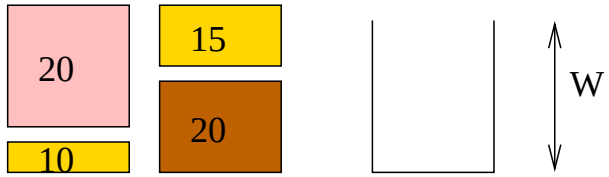
transformation between different specifications of L.

Way Formal Languages?

- The program languages, data formats,. . . are formal languages.
- Simply formulate the problems
- Many algorithmic problems one could assume as word problems.

„When we understand the formal languages then we understand the informatics.“

Example: Knapsack problem



- n objects with weight $w_i \in \mathbb{N}$ and profit p_i
- to select a subset $\mathbf{x}$ of the objects
- such that $\sum_{i \in \mathbf{x}} w_i \leq W$ and
- to maximize the profit $\sum_{i \in \mathbf{x}} p_i$

Example: Knapsack problem

Define $L \subseteq \{0,1,\text{‘,’}\}^*$:

$w \in L$ if w is a list separated by commas of $2n+2$ binary numbers

$P, W, w_1, p_1, \ldots, w_n, p_n$, so that

$\exists \mathbf{x} \subseteq \{1,\ldots,n\} : \sum_{i \in \mathbf{x}} p_i \geq P$ and $\sum_{i \in \mathbf{x}} w_i \leq W$

Word problem for $L \rightsquigarrow$ Knapsack problem:

- rate the optimal P through a binary search
- find $\mathbf{x}$ for the chosen element.

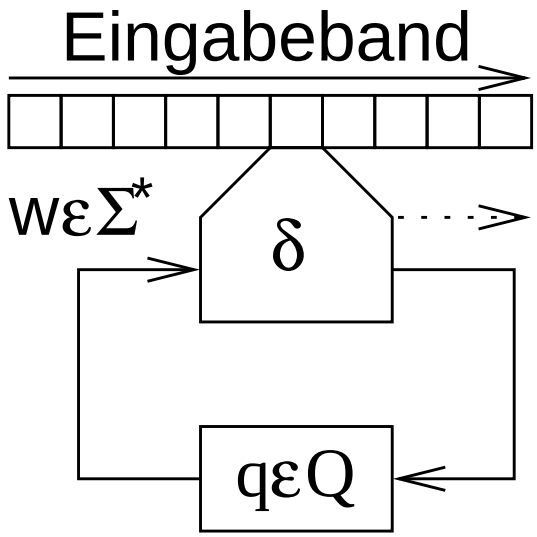
Altogether $\leq n + \log \sum_i p_i$ to solve the word problem „little“

Automata theory

Finite Automata

A deterministic finite automaton (or acceptor) consists of:

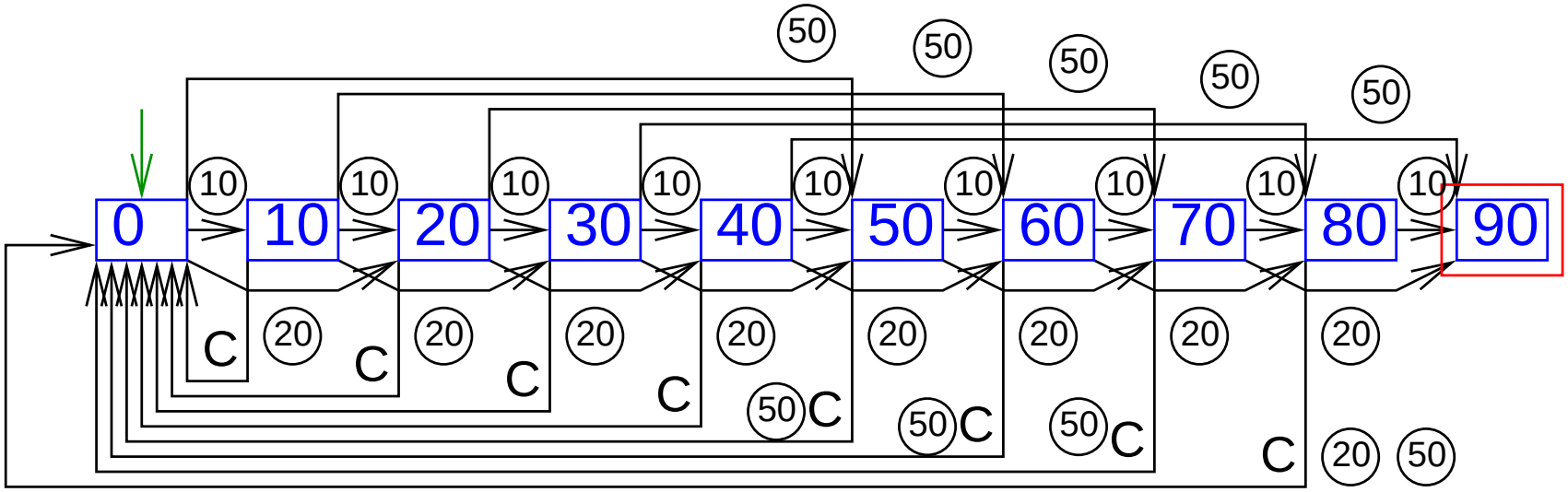
- Q, a finite set of states;
- Σ, a finite set of input symbols, (alphabet);
- $\delta : Q \times \Sigma \rightarrow Q$, a transition function;
- $s \in Q$, the input state;
- $F \subseteq Q$, a finite set of the final states.



Example: a simple ticket automat

- Standard price 90 cents
- 10, 20 and 50 coins will be accepted
- Stop by button C or too much money
- 90 cents exactly are thrown $\rightsquigarrow$ ready

$$(\{10,20,50,C\},\{0,10,20,30,40,50,60,70,80,90\},\delta,0,\{90\})$$



More powerful languages and machines

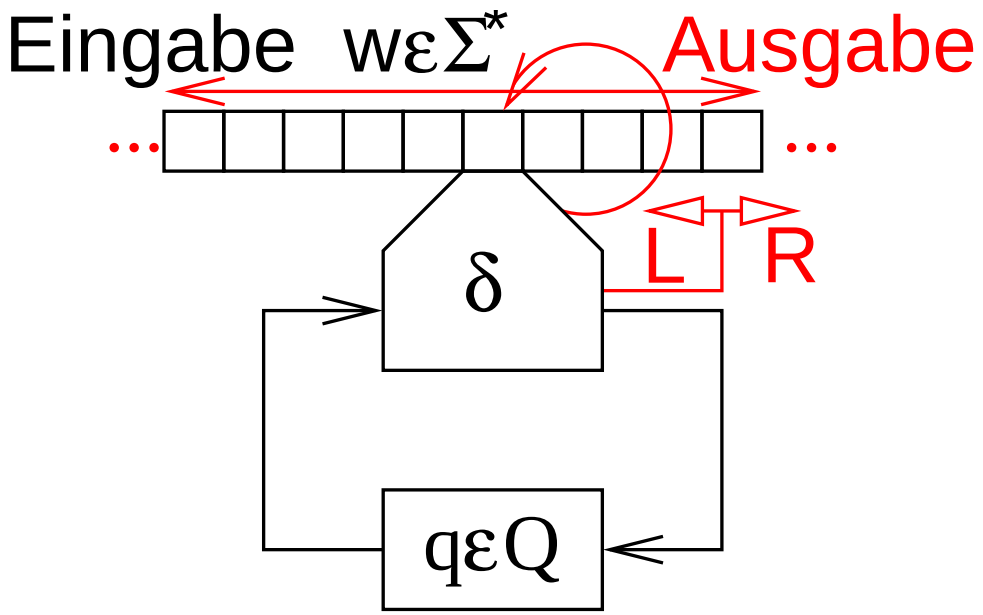
Chomsky-Hierarchy: Grammars, Automata models, and more suitable for families of formal languages.

In particular: context-free languages. For example for the purpose of specification of the syntax of the programming languages.

- Context-free languages we can recognize effectively
- Which kind of languages we can recognize in linear time?

More powerful machines:

Turing machines and Computability



- The Turing machines could solve no more and no less problems than all others enough powerful machine models
- There are (important) non computable functions. In particular could we solve anything by an arbitrary Turing machine?

Theory of complexity: Which problems we could compute efficiently?

- Again are the Turing machines. We neglect polynomial big factors in the runtime(as functions of the length of the input).

$$n \approx n^2 \approx \cdots n^{42} \approx \cdots \ll \quad 2^{0.134n}$$

- We know a little about the lower bound of the running time
- But there are big classes of important algorithmic problems for which all others, less efficient problems are excellent.