logic lɔ·dʒik science that treats of forms of thinking XIV; logical argumentation XVII. – (O)F. *logique* – late L. *logica* – Gr. *logikē* (Cicero), for *hē logikē tékhnē* the art of reasoning; *logikē*, fem. of *logikós*, f. *lógos* reasoning, discourse (see LOGOS).

logos lɔ·gɔs 'the Word' of John i 1. XVI. Gr. *lógos* account, ratio, reason, argument, discourse, saying, (rarely) word, rel. to *légein* gather, choose, recount, say (see LECTION).

IF.05.22 THEORETICAL INFORMATICS

Peter Bauer

UNIT 01

Overview and Administrivia

CONTENT

- Motivation
  - Why to study theoretical informatics?
- Administrivia
  - Grading
  - Communication
  - Working

COURSE OUTLINE

- Formal Logic
- Production Systems
- Formal Languages
- Compiler Construction

THREE REASONS WHY TO STUDY THEORETICAL INFORMATICS?

- History view: former theoretical topics are now applied
- Practice view: better mathematical knowledge enables to write better software
- Technology view: one base of computer science

VERY EARLY HISTORY

- 1837: Charles Babbage: Analytical engine
- 1854: Boole: Boolean algebra
- 1931: Kurt Gödel: Incompleteness theorem
- 1936: Alan Turing / Alonzo Church: Turing machine

EXCURSUS – A GLANCE ON THE INCOMPLETENESS THEOREM

- "Any formal system either is inconsistent or incomplete."
- Inconsistent: A false sentence can be proven.
- Incomplete: There is no proof for some sentence in the system.

EXAMPLE?

- Difficult, because all "easy understandable" examples do not really catch it
- Try: Barber paradox (Bertrand Russel,
- Village with just one barber, all men clean-shaven.
- Barber (he/his) announces: "I shave all those men and only those who do not shave themselves."
- Question: "Does the barber shave himself?"

Either answer leads to a contradiction, therefore the question cannot be answered

EARLY HISTORY

- 1950: Alan Turing: Turing test
- 1950: Claude Shannon: Chess as search
- 1964: Joseph Weizenbaum: Eliza: http://www.med-ai.com/models/eliza.html

HISTORY

- 1974: Paul Werbos: Backpropagation algorithm
- 1980s: Expert systems, (Autonomous) industrial robots, Artificial Neural Networks, …
- 1990s: Deep Blue, Semantic Web Roadmap, Web Crawlers, Search Engines, Robotics, …

Artificial Neural Networks / Deep Learning will be an extra Course in grade 4 and 5 (DSAI)

RECENT DEVELOPMENT

- 2000s: Asimo, Blue Brain, Web Ontology Language, …
- 2010s: Machine Learning, Deep Learning, …
- 2020s: Generative AI, Large Language Models, …

ChatGPT

THEORETICAL COMPUTER SCIENCE APPLIED

- Formal languages → Compiler construction
- Complexity theory → Efficiency of algorithms
- Cryptography → Computer security
- Production Systems → Game playing, expert systems, Planning, Robotics
- ANN → Face detection, image analysis

ADMINISTRIVIA

HOW IS THE COURSE ORGANIZED?

- Lecture (~ 1 unit)
- Present and discuss necessary concepts in class
- Exercise (~ 1 unit)
- Discuss home works and other examples

HOW TO GET A GRADE – EXERCISES

- Every week some small examples
- To be solved at home
- Tick your examples next week
- Present them in the exercise hour
- In case of illness: contact the teacher right after you are back and present the examples in a f2f-session

HOW TO GET A GRADE – MINI PUZZLES

- Approx. two to three per semester
- Announced a week in advance
- Results are averaged
- Weight: 50 %

HOW TO GET A GRADE – TESTS

- Date 1 (winter term): Announced on our Teams chat
- Date 2 (summer term): Announced on our Teams chat
- Weight: 50 %

HOW TO GET A GRADE – SUMMARY

- Overall Score in LOAL =
  - Score(Quizzes and Tests) *
  - (Score(Exercises) / 2 + 0.55)
- This score and the Java score is then averaged to get a final grade in POSE

COMMUNICATION: MS TEAMS

- For general announcements and “offline” discussions online …

THREE REASONS WHY TO STUDY THEORETICAL INFORMATICS?

- History view: former theoretical topics are now applied
- Practice view: better mathematical knowledge enables to write better software
- Technology view: one base of computer science

GREATEST COMMON DIVISOR (GCD) – DEFINITION

- $\gcd(a, b) = c$ if and only if
  - c divides a and c divides b and
  - for all numbers i which divide a and b must hold that i is smaller or equal to c

GCD – FIRST APPROACH

- Traditional algorithm to calculate the gcd of a and b ($a > 0, b \geq 0$)
- Calculate the prime factors of a and b
- Multiply the common prime factors both numbers -> gcd
- Try it out with $a = 24$ and $b = 16$

OBSERVATION

- $\gcd(a, b) = \gcd(a - b, b)$ if $a > b$
- Try it out with two arbitrary but well known numbers, e.g., 84 and 70
- Be aware that this is no general proof

ONE FURTHER IMPROVEMENT

- Euclidean algorithm to calculate the gcd
- $\gcd(a, 0) = a$
- $\gcd(a, b) = \gcd(b, a \bmod b)$
- Again try it out with $a = 24$ and $b = 16$

FIGURE IT OUT

- Calculate the gcd of 1989 and 867
- Using prime factors
- Using Euclid's algorithm with modulo
- Compare the efficiency of these algorithms

THREE REASONS WHY TO STUDY THEORETICAL INFORMATICS?

- History view: former theoretical topics are now applied
- Practice view: better mathematical knowledge enables to write better software
- Technology view: one base of computer science

CORE DISCIPLINES COMPUTER SCIENCE EMERGED FROM

- Mathematics
- Logic
- Electrical engineering and Physics

THEORETICAL INFORMATICS COVERS

- Logic
- Mathematics