

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

**Logos** lō·gōs ‘the Word’ of John i 1. XVI. In gen. sense, account, ratio, reason, argument, discourse, saying, (rarely) word, rel. to *legis*; *logos*, put together, choose, recount, say (see LECTIO).

## IF.05.22 THEORETICAL INFORMATICS

Peter Bauer

# UNIT 01

Overview and Administrivia

# TODAY

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

# CONTENT OUTLINE

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

# WHY TO STUDY THEORETICAL INFORMATICS? – PART I

- One base of computer science

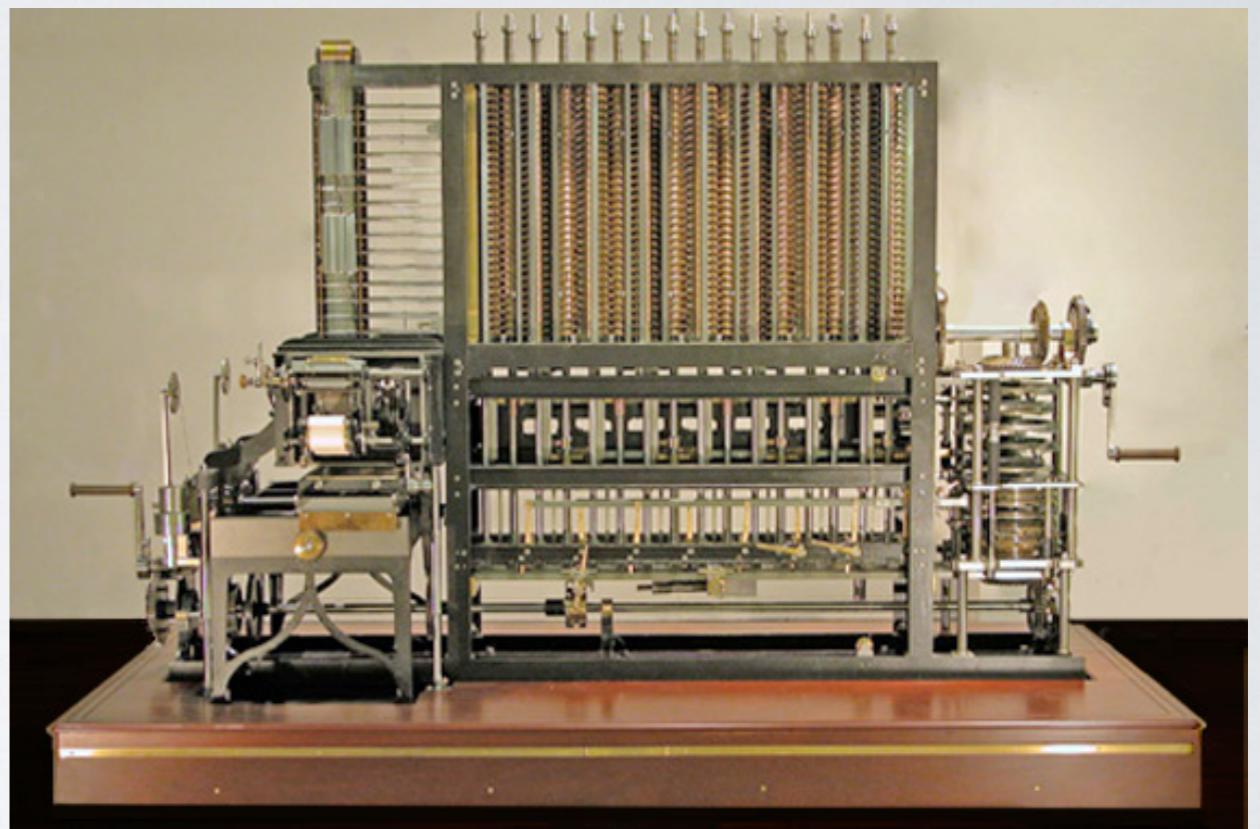
# VERY EARLY HISTORY

# VERY EARLY HISTORY

- 1837: Charles Babbage:  
Analytical engine

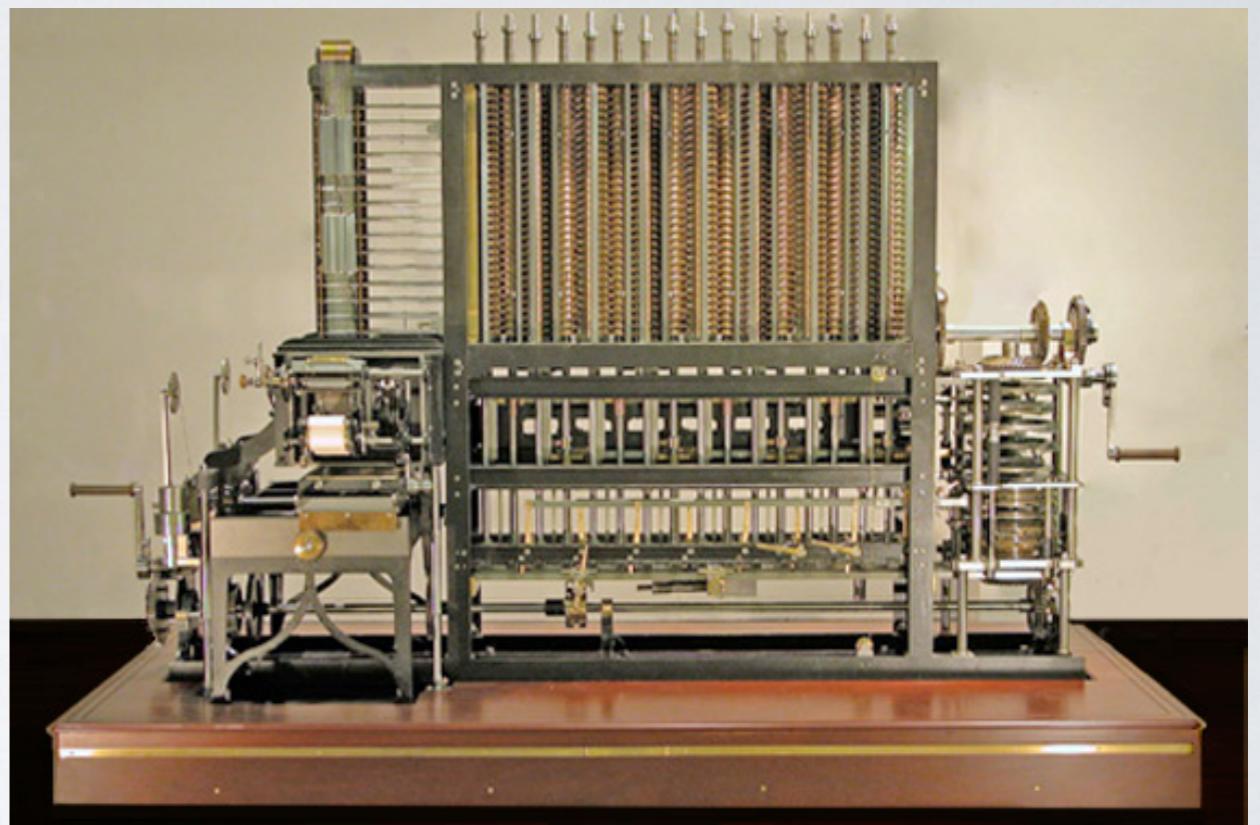
# VERY EARLY HISTORY

- 1837: Charles Babbage:  
Analytical engine



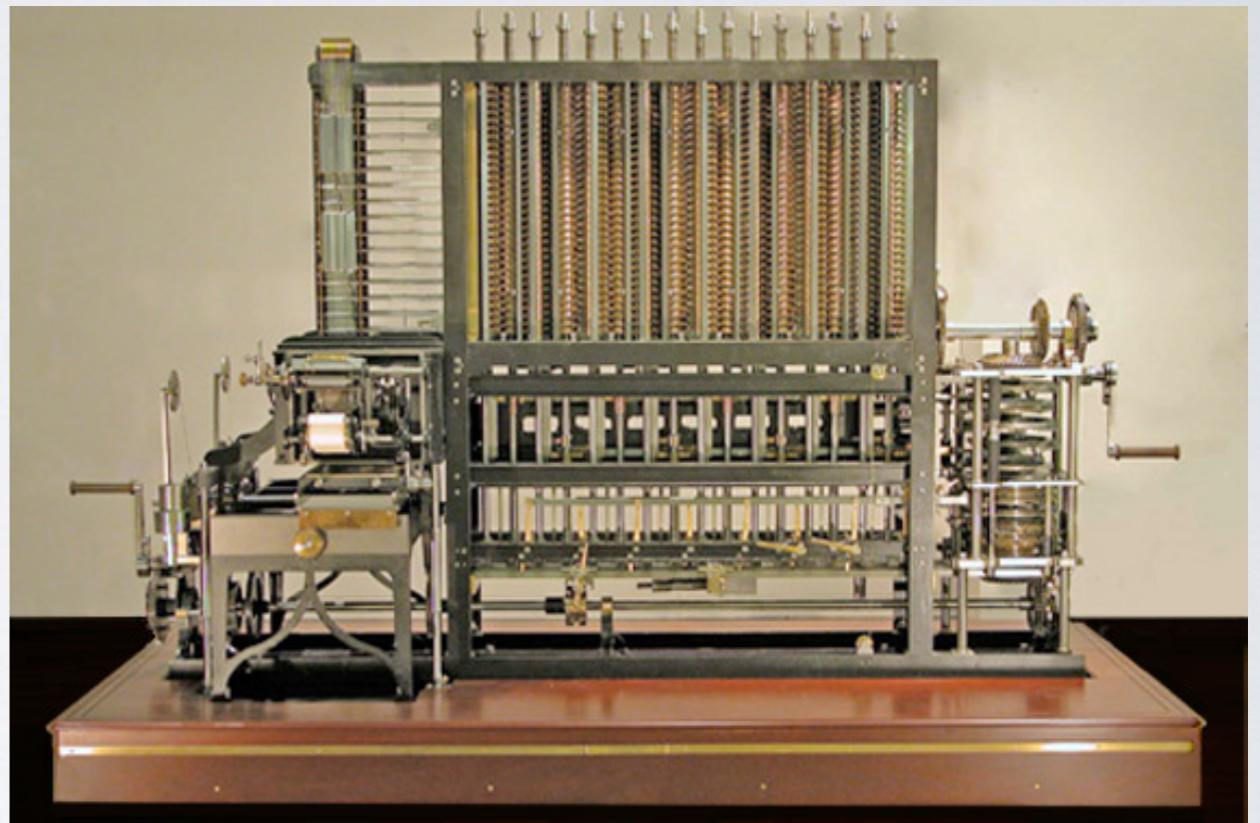
# VERY EARLY HISTORY

- 1837: Charles Babbage:  
Analytical engine
- 1854: Boole: Boolean  
algebra



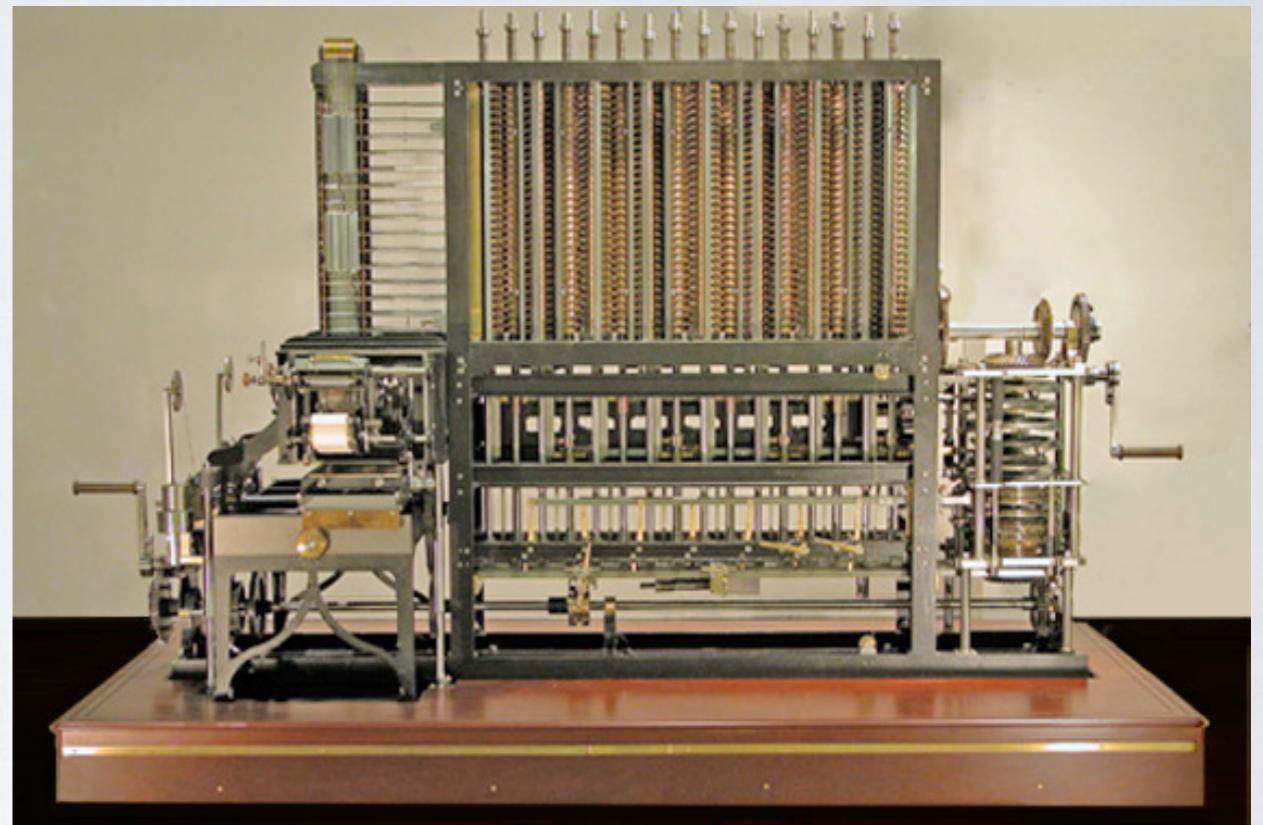
# VERY EARLY HISTORY

- 1837: Charles Babbage:  
Analytical engine
- 1854: Boole: Boolean  
algebra
- 1931: Kurt Gödel:  
Incompleteness theorem



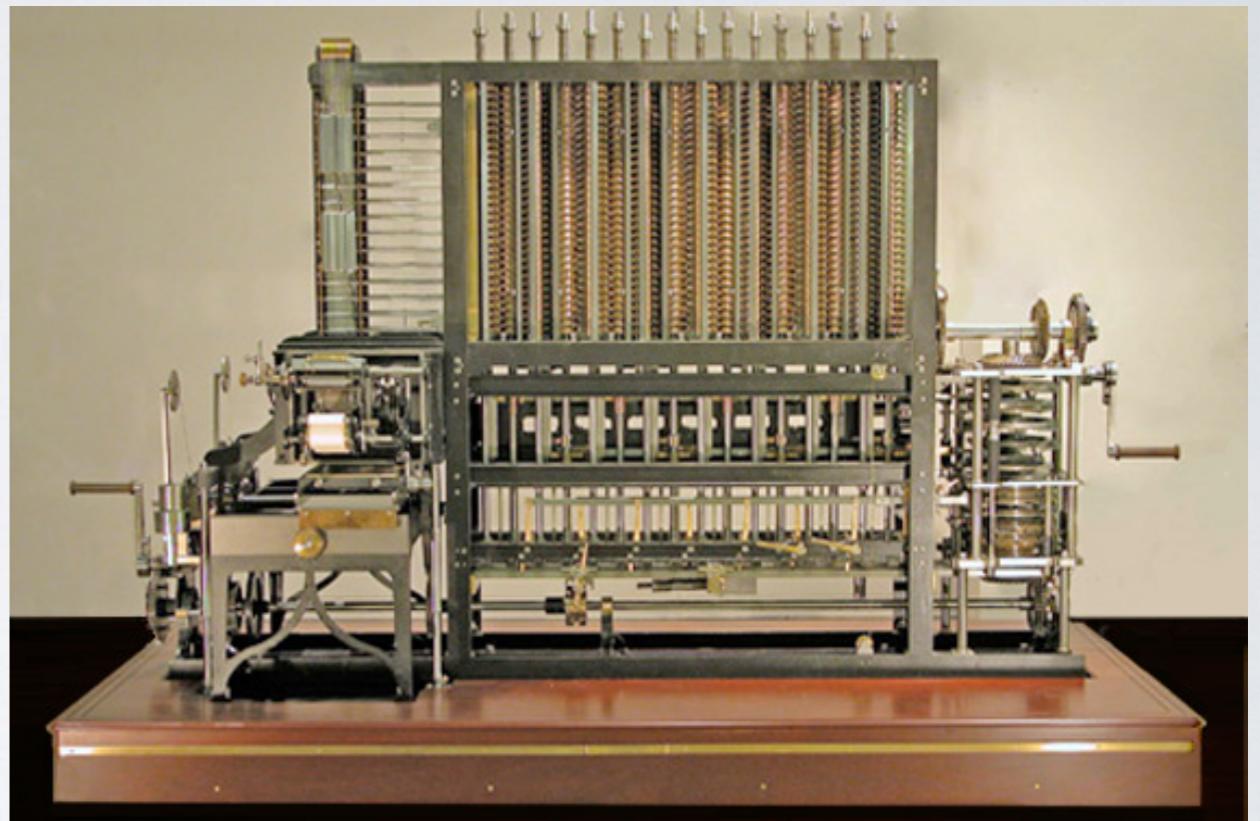
# VERY EARLY HISTORY

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



# 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
  - Village with just one barber, all men clean-shaven.
  - Barber announces: “I shave all those men who do not shave themselves.”
  - Question: “Who shaves the barber?”

# EXAMPLE?

- Difficult, because all “easy understandable” examples do not really catch it
  - Try: Barber paradox
    - Village with just one barber, all men clean-shaven.
    - Barber announces: “I shave all those men who do not shave themselves.”
    - Question: “Who shaves the barber?”
- Either answer leads to a contradiction, therefore the question cannot be answered

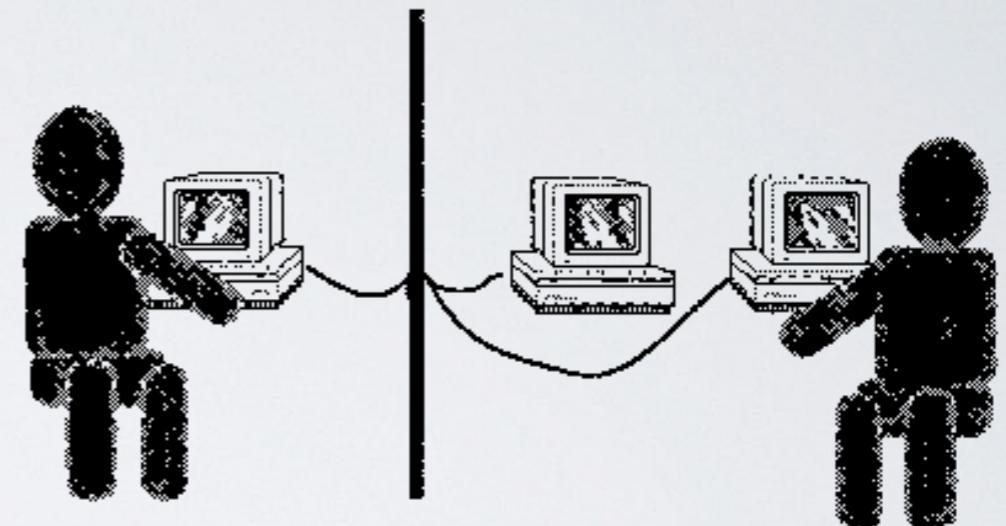
# EARLY HISTORY

# 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>

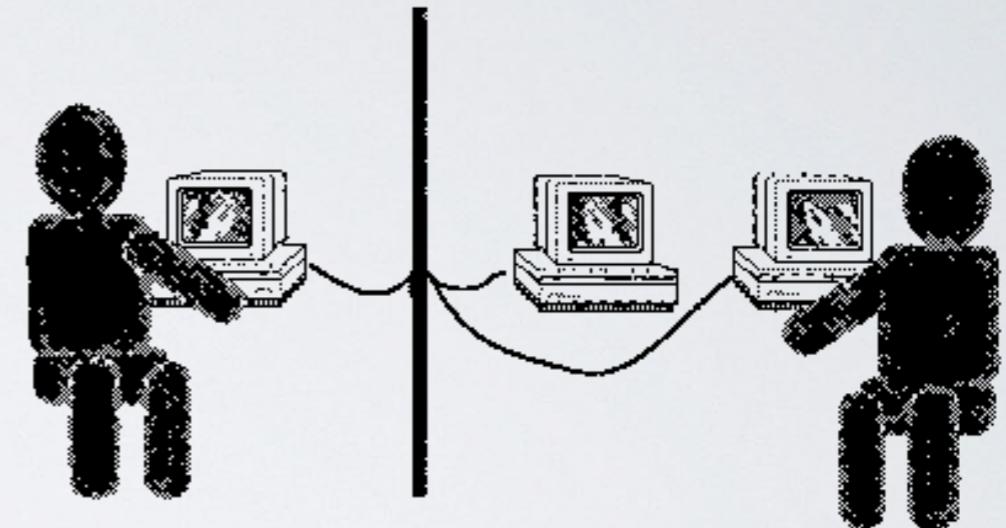
# EARLY HISTORY

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



# EARLY HISTORY

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



The screenshot shows a window titled 'File Edit Options Buffers Tools Help' with a toolbar below it. The main area contains a conversation transcript:

```
I am the psychotherapist. Please, describe your problems.  
Each time  
you are finished talking, type RET twice.  
  
I have a terrible headache.  
Maybe you should consult a doctor of medicine, I am a psychi  
atrist.  
Okay, I'll do that then.  
Can you elaborate on that?  
What are you talking about? No I cannot elaborate. See you l  
ater.  
Why do you say that?  
Now you're just talking nonsense!  
What makes you believe now I am just talking nonsense?  
You are not making any sense.  
  
Perhaps your plans have something to do with this.  
-u:** *doctor* (Doctor Fill)--L1--Top-----  
X
```

# HISTORY

# HISTORY

- 1974: Paul Werbos:  
Backpropagation algorithm

# HISTORY

- 1974: Paul Werbos:  
Backpropagation algorithm
- 1980s: Expert systems,  
(Autonomous) industrial robots,  
Artificial Neural Networks, ...



# 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, ...



# 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, ...

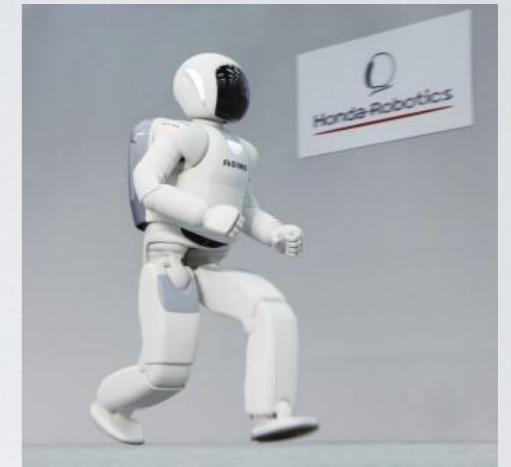
Artificial Neural Networks /  
Deep Learning will be an extra  
Course in grade 4 and 5 (DSAI)



# MORE RECENT DEVELOPMENTS

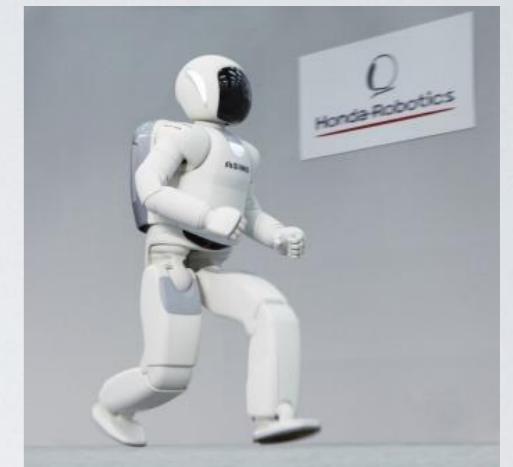
# MORE RECENT DEVELOPMENTS

- 2000s: Asimo, Blue Brain, Web Ontology Language, ...

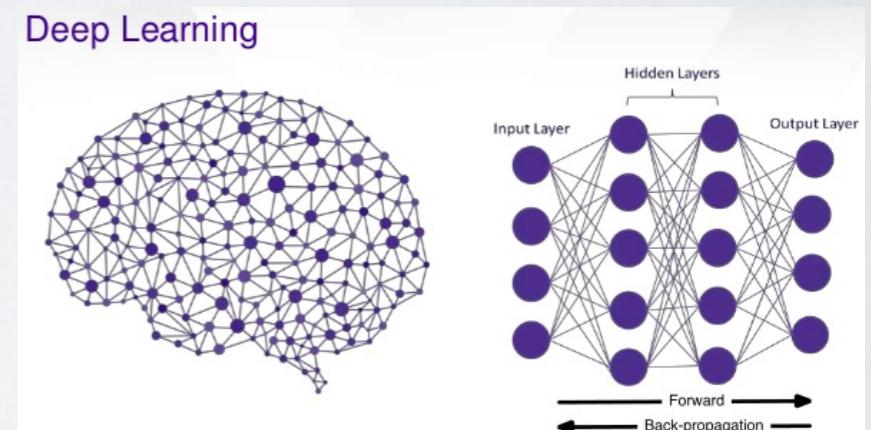


# MORE RECENT DEVELOPMENTS

- 2000s: Asimo, Blue Brain, Web Ontology Language, ...
- 2010s: Machine Learning, Deep Learning, ...

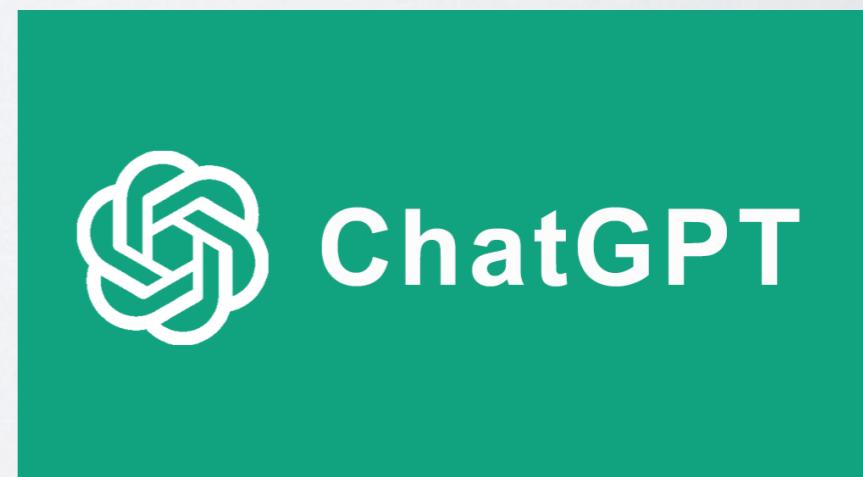
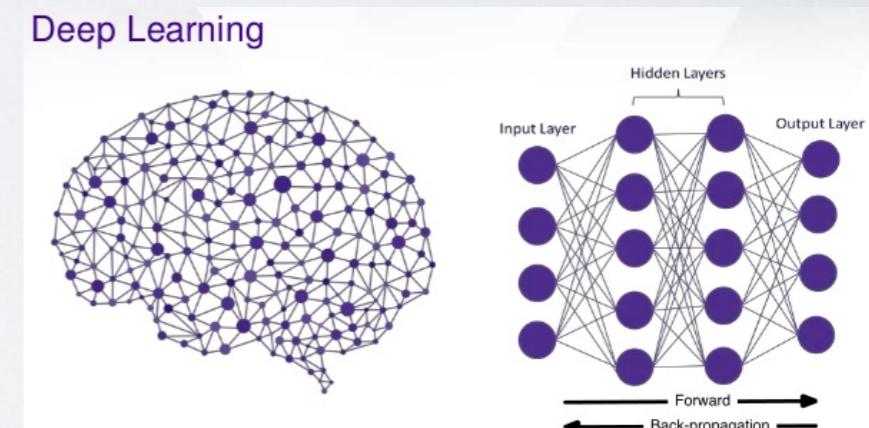


Deep Learning



# MORE RECENT DEVELOPMENTS

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



# ADMINISTRIVIA – PART I

# 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

# 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 %

# WHY TO STUDY THEORETICAL INFORMATICS? – PART 2

- Better mathematical knowledge enables to write better software
- Theoretical informatics applied

# BETTER MATHEMATICAL KNOWLEDGE I

# BETTER MATHEMATICAL KNOWLEDGE I

- Traditional algorithm to calculate the gcd of two integers  $a$  and  $b$  ( $a > 0, b \geq 0$ )
  - Calculate the prime factors of  $a$  and  $b$
  - Multiply the common prime factors both numbers  
-> gcd

# BETTER MATHEMATICAL KNOWLEDGE I

- Traditional algorithm to calculate the gcd of two integers  $a$  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  $b = 16$

# BETTER MATHEMATICAL KNOWLEDGE 2

# BETTER MATHEMATICAL KNOWLEDGE 2

- Euclidean algorithm to calculate the gcd
  - $\gcd(a, 0) = a$
  - $\gcd(a, b) = \gcd(b, a \bmod b)$

# BETTER MATHEMATICAL KNOWLEDGE 2

- 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  $b = 16$

# BETTER MATHEMATICAL KNOWLEDGE 3

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

# THEORETICAL COMPUTER SCIENCE APPLIED

# THEORETICAL COMPUTER SCIENCE APPLIED

- Formal languages → Compiler construction

# THEORETICAL COMPUTER SCIENCE APPLIED

- Formal languages → Compiler construction
- Complexity theory → Efficiency of algorithms

# THEORETICAL COMPUTER SCIENCE APPLIED

- Formal languages → Compiler construction
- Complexity theory → Efficiency of algorithms
- Cryptography → Computer security

# THEORETICAL COMPUTER SCIENCE APPLIED

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

# 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 – PART 2

# 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 – SUMMARY

- Overall Score in LOAL =
  - $\text{Score}(\text{Puzzles and Tests}) * \frac{\text{Score}(\text{Exercises})}{2} + 0.55$
- This score and the Java score is then averaged to get a final grade in POSE

# COMMUNICATION: MSTEAMS

- For general announcements  
and “offline” discussions online

...