Complexity Theory & Philosophy

Abraham Hinteregger

Vienna University of Technology

2016-06-27

Chapter

Some historical background

- 85 years ago Gödel introduced incompleteness theorems[1]
- 5 years later Turing generalized this to limitations of computation (Halting)[2]
- When "computing machines" became reality, time and space requirements of algorithms were of interest [3]

[1] [Gödel, 1931]
[2] [Turing, 1936]
[3] [Hartmanis and Stearns, 1965]: On the comp. complexity of algorithms

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?

Computability vs. Complexity

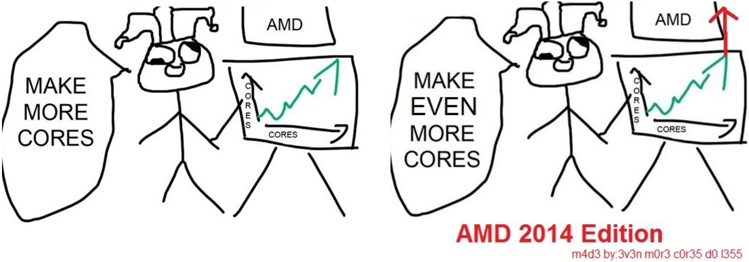
- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?



Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs
  - Use GPUs

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs
  - Use GPUs
  - Hire TU graduates

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs
  - Use GPUs
  - Hire TU graduates
- Optimizing constant factors/reducing exponent concern of engineers

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs
  - Use GPUs
  - Hire TU graduates
- Optimizing constant factors/reducing exponent concern of engineers
- What about 100 seconds versus 2^{100} seconds?

Computability vs. Complexity

- Difference between something being *computable* or *uncomputable* obviously of philosophical importance
- What about *computable* and *efficiently computable*?
- 10 seconds versus 20 seconds?
  - Buy more RAM/CPUs
  - Use GPUs
  - Hire TU graduates
- Optimizing constant factors/reducing exponent concern of engineers
- What about 100 seconds versus 2^{100} seconds?
  - Multiplication vs. prime factoring
  - Reading a book with 400 pages vs. reading all possible 400p books

Relevance of polynomial time

- Computability Theory: Truncated HALTING problem

  Is there an F-proof of S in n or fewer symbols?

- Decidable by exponential bruteforce algorithm
- If there was a polynomial (with sane constants and exponent) algorithm Hilbert's dream (almost) possible

---

Mentioned in a letter from Gödel to von Neumann in 1956

Relevance of polynomial time II

- Number Theory: largest "known" prime: $p = 2^{74,207,281} - 1$
  - expression picks out unique positive integer
  - that integer has to be a prime number

Relevance of polynomial time II

- Number Theory: largest "known" prime: $p = 2^{74,207,281} - 1$
  - expression picks out unique positive integer
  - that integer has to be a prime number
- Why p and not $p' =$ first prime s.t. $p' > p$?

Relevance of polynomial time II

- Number Theory: largest "known" prime: $p = 2^{74,207,281} - 1$
  - expression picks out unique positive integer
  - that integer has to be a prime number
- Why p and not $p' =$ first prime s.t. $p' > p$?
- What criterion is used to call a number a "known prime"?

Relevance of polynomial time II

- Number Theory: largest “known” prime: $p = 2^{74,207,281} - 1$
  - expression picks out unique positive integer
  - that integer has to be a prime number
- Why p and not $p' =$ first prime s.t. $p' > p$?
- What criterion is used to call a number a “known prime”?
  - Is it possible to store the decimal expansion somewhere? Would $2^{2^{2^{1231}}} - 1$ not be a known prime (if it is prime) because the universe is too small?

Relevance of polynomial time II

- Number Theory: largest "known" prime: $p = 2^{74,207,281} - 1$
  - expression picks out unique positive integer
  - that integer has to be a prime number
- Why p and not $p' =$ first prime s.t. $p' > p$?
- What criterion is used to call a number a "known prime"?
  - Is it possible to store the decimal expansion somewhere? Would $2^{2^{2^{1231}}} - 1$ not be a known prime (if it is prime) because the universe is too small?
  - Maybe the existence of polynomial ($n =$ number of digits) procedure to output the decimal digits of p would explain "knowing"

Chapter

Complexity Theory

Turing test & AI

Turing Test

Simulating humans

Can a computer think?

Time & Space

The Turing Test I

Evaluator converses with machine that pretends to be human and a real human. If evaluator can't distinguish the two the machine passes the test.

- Additional requirements could be added, e.g.: vision, hearing, handwriting, drawing
- The test allows a maximum amount of information to be exchanged, e.g. 2^{30} bits.

Chinese Room Argument

Argument against “strong AI”, presented by John Searle in 1980

- Store all things the evaluator could ask
- Also store an answer for every possible question/statement
- Output predefined answers for each question

Chinese Room Argument

Argument against "strong AI", presented by John Searle in 1980

- Store all things the evaluator could ask
- Also store an answer for every possible question/statement
- Output predefined answers for each question

$\implies$ Passing the Turing test is not a question of computability (as e.g. Roger Penrose argued in *The Emperor's New Mind*) but of complexity

Chinese Room Argument II

- Chinese room argument initially made to counter "Strong AI" claim
  - Chinese conversation
  - Person in huge library executes moves of the computer program

Chinese Room Argument II

- Chinese room argument initially made to counter “Strong AI” claim
  - Chinese conversation
  - Person in huge library executes moves of the computer program
- Nobody would claim that this person understands Chinese, therefore the program cannot give a computer “mind” or “understanding” either

Chinese Room Argument II

- Chinese room argument initially made to counter “Strong AI” claim
  - Chinese conversation
  - Person in huge library executes moves of the computer program
- Nobody would claim that this person understands Chinese, therefore the program cannot give a computer “mind” or “understanding” either
- Widely considered a bad argument
- Flood of refutations inspired Pat Hayes to quip:

  The field of cognitive science should be redefined as "the ongoing research program of showing Searle's Chinese Room Argument to be false"

Simulating humans

- Would a compact & efficient program passing the Turing test be intelligent? If yes, is this possible?

Simulating humans

- Would a compact & efficient program passing the Turing test be intelligent? If yes, is this possible?
- Can humans solve NP-complete problems efficiently?

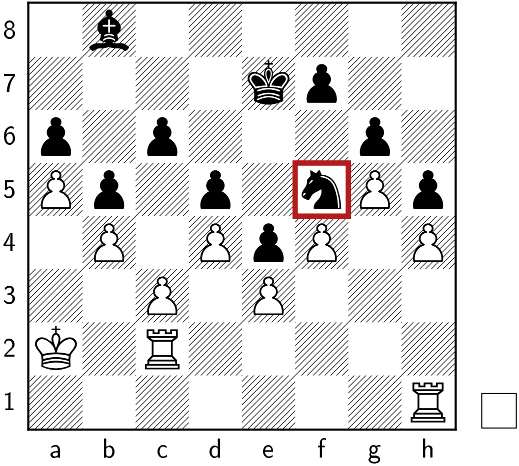
Simulating humans

- Would a compact & efficient program passing the Turing test be intelligent? If yes, is this possible?
- Can humans solve NP-complete problems efficiently? No

Simulating humans

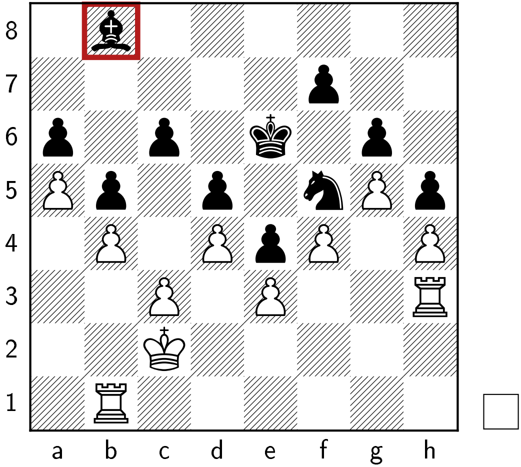
- Would a compact & efficient program passing the Turing test be intelligent? If yes, is this possible?
- Can humans solve NP-complete problems efficiently? No
- Humans are usually superior to machines at search problems with *higher-level structure*, *semantics*, *global patterns* or *symmetries*
  - Proving Fermat's Last Theorem (still took us a while)
  - Pigeonhole Principle (put 1001 objects in 1000 slots)
  - Determining if a chess position is "dead"

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



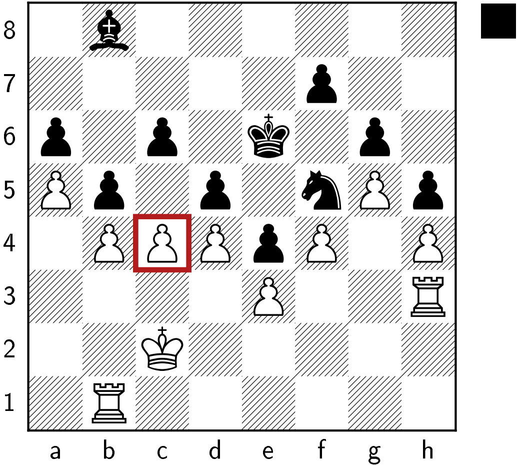
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



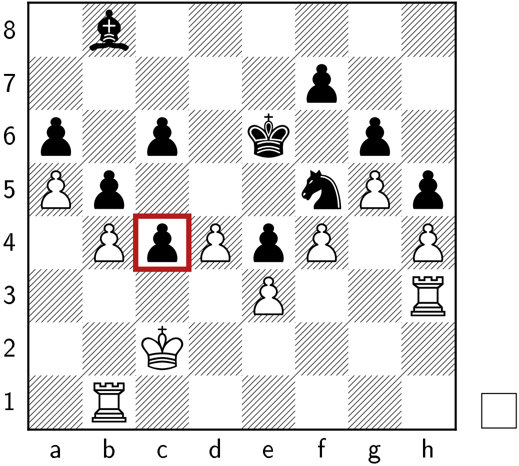
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



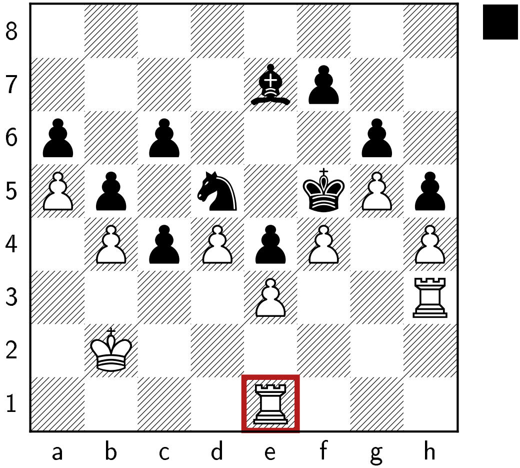
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



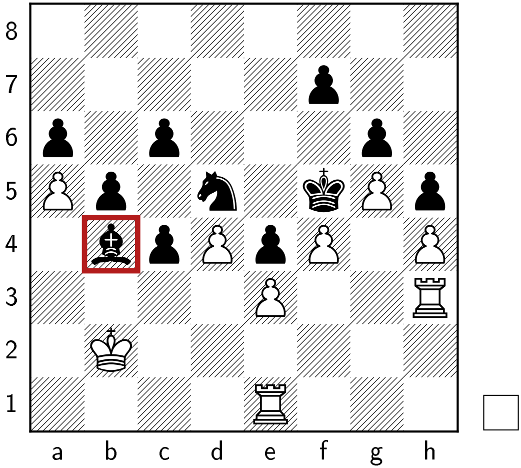
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



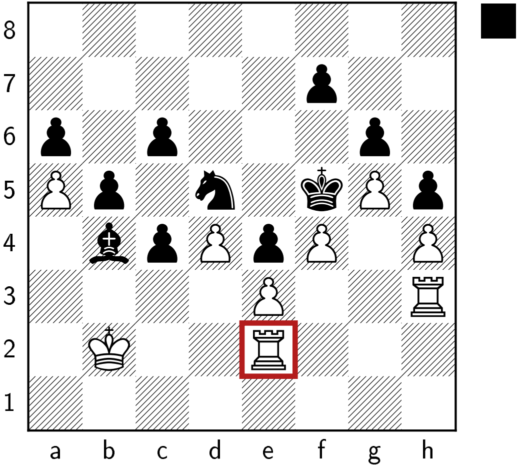
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



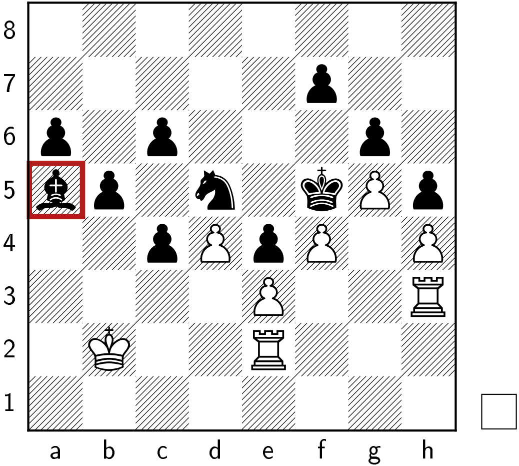
http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



http://www.chessgames.com/perl/chessgame?gid=1497429

Rybka vs Hikaru Nakamura (ICC, 3+0, 2008)



http://www.chessgames.com/perl/chessgame?gid=1497429

Can a computer think?

- Metaphysical question: When does *intelligence* or *understanding* arise? Most certainly not when outputting predefined sentences (current state of Siri & Cortana)
  - Underlying theme of the game *The Talos Principle* (2014)
- Practical question: Are there *patterns* that a computer can't recognize efficiently? If yes – why? (lower bounds on complexity of algorithms[4])

[4] Currently most algorithms require exponential time for proofs using pigeon hole principle [Beame and Pitassi, 2001]

Chapter

Closed Timelike Curves I

- Possibility derived from general relativity by Gödel in 1949
- Since then some other more or less practical ways to achieve CTC discovered, e.g. Tipler cylinder & wormholes
- Famous example:

  Time traveller goes back in time to kill his grandfather to prevent existence of time travellers parents. Then time traveller does not exist and can't kill his grandfather.
- Possible solutions for grandfather paradox:

Closed Timelike Curves II

- Universe ensures consistency and he somehow fails to kill his grandfather
- Quantum mechanic solution: S maps quantum state before time travel to quantum state after time travel. Universe finds fixed point, e.g.: time traveller is born with $p = 1/2$ and may or may not be born and kill his grandfather.[5]

[5] [Deutsch, 1991]

Fixed point time travelling

You go back in time with a copy of Othello, give it to Shakespeare to save him the trouble of writing it in the first place. Shakespeare thanks you for the effort and publishes it verbatim. Centuries later you can buy a copy and travel back in time to give it to him

Fixed point time travelling

You go back in time with a copy of Othello, give it to Shakespeare to save him the trouble of writing it in the first place. Shakespeare thanks you for the effort and publishes it verbatim. Centuries later you can buy a copy and travel back in time to give it to him

- No logical paradox

Fixed point time travelling

You go back in time with a copy of Othello, give it to Shakespeare to save him the trouble of writing it in the first place. Shakespeare thanks you for the effort and publishes it verbatim. Centuries later you can buy a copy and travel back in time to give it to him

- No logical paradox
- But one of computational complexity: Knowledge comes into existence without causal process

Fixed point time travelling

> *You go back in time with a copy of Othello, give it to Shakespeare to save him the trouble of writing it in the first place. Shakespeare thanks you for the effort and publishes it verbatim. Centuries later you can buy a copy and travel back in time to give it to him*

- No logical paradox
- But one of computational complexity: Knowledge comes into existence without causal process (*Evolutionary Principle*)

Closed Timelike Curve Computation [Brun, 2003]

- Take any NP-complete problem p and encode all possible solutions s to binary representation $E : S \rightarrow \{0, 1, \ldots, |S-1|\}$
- Let $f(E(s)) = 1$ iff s is a valid solution for p (runs in polynomial time).
- Run the following program inside a CTC with it's own output:
  - If $f(x) = 1$ then output x
  - Else output $(x + 1) \mod |S|$

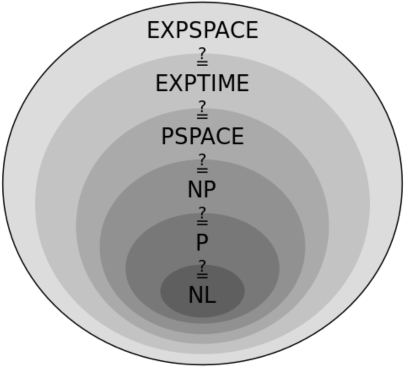
Closed Timelike Curve Computation [Brun, 2003]

- Take any NP-complete problem p and encode all possible solutions s to binary representation $E : S \rightarrow \{0, 1, \ldots, |S - 1|\}$
- Let $f(E(s)) = 1$ iff s is a valid solution for p (runs in polynomial time).
- Run the following program inside a CTC with it's own output:
  - If $f(x) = 1$ then output x
  - Else output $(x + 1) \mod |S|$
- It was later shown (by [Aaronson and Watrous, 2009]) that a CTC computer (assuming causal consistency) could solve all problems in PSPACE

Closed Timelike Curve Computation [Brun, 2003]

- Take any NP-complete problem p and encode all possible solutions s to binary representation $E : S \rightarrow \{0, 1, \ldots, |S - 1|\}$
- Let $f(E(s)) = 1$ iff s is a valid solution for p (runs in polynomial time).
- Run the following program inside a CTC with it's own output:
  - If $f(x) = 1$ then output x
  - Else output $(x + 1) \mod |S|$
- It was later shown (by [Aaronson and Watrous, 2009]) that a CTC computer (assuming causal consistency) could solve all problems in PSPACE
- PSPACE would be the largest class solvable independent from classical or quantum computation

Complexity Classes



$$LSPACE \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$$
$$P \subseteq PSPACE$$

Image from Wikipedia

References I

MathOverflow - Philosophical Question related to Largest Known Primes.

Aaronson, S. (2008).
Shtetl-Optimized » Time: Different from space.

Aaronson, S. (2011).
Why Philosophers Should Care About Computational Complexity.
arXiv: 1108.1791.

Aaronson, S. (2013).
Quantum Computing since Democritus.
Cambridge University Press.
Cambridge Books Online.

References II

Aaronson, S. and Watrous, J. (2009).
Closed timelike curves make quantum and classical computing equivalent.
In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*.

Beame, P. and Pitassi, T. (2001).
Current Trends in Theoretical Computer Science.
pages 42–70. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Brun, T. A. (2003).
Computers with closed timelike curves can solve hard problems efficiently.
Foundations of Physics Letters, 16(3):245–253.

References III

Deutsch, D. (1991).
Quantum mechanics near closed timelike lines.
Physical Review D, 44(10):3197.

Gödel, K. (1931).
Some metamath. results on completeness and consistency...
From Frege to Gödel: A source book in mathematical logic.

Hartmanis, J. and Stearns, R. E. (1965).
On the computational complexity of algorithms.
Transactions of the American Mathematical Society, 117:285–306.

References IV

Immerman, N. (2016).
Computability and Complexity.
In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Spring 2016 edition.

Turing, A. M. (1936).
On computable numbers, with an application to the Entscheidungsproblem.
J. of Math, 58(345-363):5.

Turing, A. M. (1950).
Computing machinery and intelligence.
Mind, 59(236):433–460.