Theory of Computation, Fall'15

CS422 M. Ziegler

Schedule: Tue.+Thu. 14h30—15h45 in N1 #111

Language: English (except *Piazza* forum)

TA: 김미진, office hours after lecture in N1 #403

Attendance: 10 points for missing <5 lectures, 9 points when missing 5, and so on.

Grading: Homework 20%, Midterm exam 30%, Final exam 40%, Attendance 10%.

Homework: Assigned roughly every 2nd week, 11 days to solve, individual handwritten solutions.

Literature, slides, assignments etc:
`http://theoryofcomputation.asia/15b_CS422/`

Exams: Midterm Oct. 22, Final exam Dec. 17

Students' Background Check

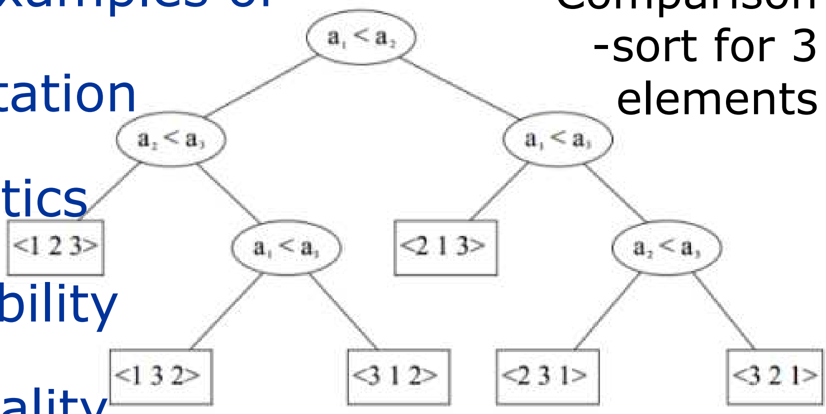
CS422 M. Ziegler

- ? CS204 *Discrete Mathematics*
- ? CS206 *Data Structures*
- ? CS300 *Introduction to Algorithms*
- ? CS320 *Programming Languages*
- ? CS322 *Formal Languages and Automata*
- ? MAS275 *Discrete Mathematics*
- ? MAS365 *Intro. to Numerical Analysis*
- ? MAS477 *Introduction to Graph Theory*
- ? MAS480 *Topics in Mathematics*
- ? graduate courses (at KAIST)
- ? non-KAIST courses

§1 Motivation & Examples

Four elementary examples of

- models of computation
- syntax vs. semantics
- limits of computability
- algorithmic optimality



Comparison -sort for 3 elements

Example 1: *Optimal Sorting Algorithm*

Problem specification:

Model of computation:

First algorithm:

Its cost analysis:

Second algorithm:

Its cost analysis:

Proof of optimality:

Example 2: Finite Automata

- Motivation from practice
- Syntax and semantics
- Example algorithms
- Programming challenges
- Limits of computability
- Equivalent characterizations

States (h,m,q)
where $h \in H=\{0,1,\ldots,23\}$
$m \in M=\{0,1,\ldots,59\}$
$q \in \{$ NIL, setH, setM $\}$
Operations SET and INC:

INC: $(h,m,\text{NIL}) \to (h,m,\text{NIL})$

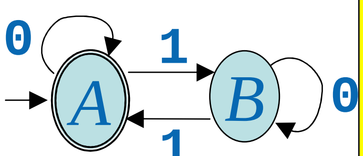
SET: $(h,m,\text{NIL}) \to (h,m,\text{setH})$

SET: $(h,m,\text{setH}) \to (h,m,\text{setM})$

SET: $(h,m,\text{setM}) \to (h,m,\text{NIL})$

INC: $(h,m,\text{setH}) \to (h+1 \bmod 24,m,\text{setH})$

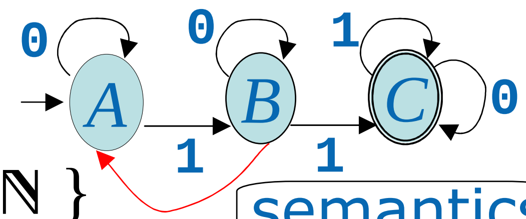
INC: $(h,m,\text{setM}) \to (h,m+1 \bmod 60,\text{setM})$



Computability by Finite Automata

Lemma: Suppose $L\subseteq\{0,1\}^*$ is accepted by a finite automaton. Then there exists some $n\in\mathbb{N}$ s.t. every $\underline{w}\in L$ of length $|\underline{w}|\geq n$ admits a decomposition $\underline{w} = \underline{x}\,\underline{y}\,\underline{z}$ with $|\underline{y}|\geq 1$ and $|\underline{x}\,\underline{y}|\leq n$ such that $\underline{x}\,\underline{y}^j\,\underline{z}\in L$ holds for every $j\in\mathbb{N}$.

$\{\ 0^n\,1^m\,0^k\ : n,m,k\in\mathbb{N}\ \}$



Theorem: a) $\{\ 0^n\,1^n : n\in\mathbb{N}\ \}$ cannot be accepted by a finite automaton.

semantics

b) To every *non*-deterministic finite automaton there is an equivalent deterministic one.

Asymptotic Efficiency

n	$\log_2 n\ \cdot$10s	$n\cdot\log n$ sec	n^2 msec	n^3 μsec	2^n nsec
10	33sec	33sec	0.1sec	1msec	1msec
100	≈1min	11min	10sec	1sec	40 Mrd. Y
1000	≈1.5min	≈3h	17min	17min	
10 000	≈2min	1.5 days	≈1 day	11 days	
100 000	≈2.5min	19 days	4 months	32 years	

- Running times of some sorting algorithms
  - BubbleSort: $O(n^2)$ comparisons and copy instr.s
  - QuickSort: typically $O(n\cdot\log n)$ steps
    but $O(n^2)$ in the *worst-case*
  - HeapSort: always at most $O(n\cdot\log n)$ operations
  - BucketSort: $O(n)$ operations
  - SORT primitive: $O(1)$
- Worst-case vs. average-case vs. best case
- w.r.t. input size (e.g. bit length) =: $n\to\infty$

Example 3: Algebraic Computation

Warmup Problem: Fix $n\in\mathbb{N}$. Given x, calculate x^n.

- Naïve algorithm: n-1 multiplications
- Improve: Calculate $x^2, x^4, x^8, \ldots, x^{2^k}$ for $k:=\lfloor\log_2 n\rfloor$
  Then multiply powers x^{2^j} with b_j=1, where
  $n=b_0+2b_1+4b_2+\ldots+2^k b_k$ is the binary expansion.
- Homework: Improve by a constant factor!
- Asympt. optimality: Each multiplication at most doubles the degree of the intermediate results; so computing x^n requires at least $\log_2 n$ of them.

Example 3: Matrix Multiplication

- Input: entries of $n\times n$-matrices A,B
- Wanted: entries of $n\times n$-matrix $C := A+B$
- High school: n^2 inner products á $O(n)$:

$O(n^3)$, optimal

7 multiplications +18 additions of $(n/2)\times(n/2)$-matrizes

$T_1:=(A_{2,1}+A_{2,2})\cdot B_{1,1}$

$T_2:=(A_{1,1}+A_{1,2})\cdot B_{2,2}$

$T_3:=A_{1,1}\cdot(B_{1,2}-B_{2,2})$

$T_4:=A_{2,2}\cdot(B_{2,1}-B_{1,1})$

$$\begin{pmatrix}C_{1,1}&C_{1,2}\\C_{2,1}&C_{2,2}\end{pmatrix}=\begin{pmatrix}A_{1,1}&A_{1,2}\\A_{2,1}&A_{2,2}\end{pmatrix}\cdot\begin{pmatrix}B_{1,1}&B_{1,2}\\B_{2,1}&B_{2,2}\end{pmatrix}$$

$C_{1,1}=T_5+T_4-T_2+T_7$, $C_{1,2}=T_3+T_2$

$L(n) = 7\cdot L(\lceil n/2\rceil)$

asymptotics dominated by #multiplications

World record: $O(n^{2.37})$ [Coppersmith&Winograd'90, François Le Gall'14]

$L(n) = O(n^{\log_2 7})$, $\log_2 7\approx 2{,}81$

Some mathematical background

- Sets: $\{0,1\}$, $\{0,1,2,\ldots\}=\mathbb{N}$, $\mathbb{Z}=\{0,-1,1,-2,2,\ldots\}$
- Cartesian products $X\times Y$, X^n, X^*; subset, powerset
- properties, relations; e.g. prime numbers, <
- functions $f{:}\subseteq X\to Y$, total, injective, surjective; well-definition

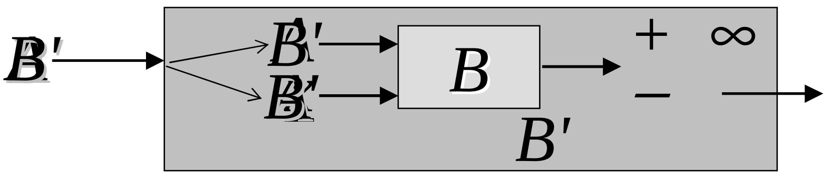
```
s='s=%r;print s%%s';print s%s
```

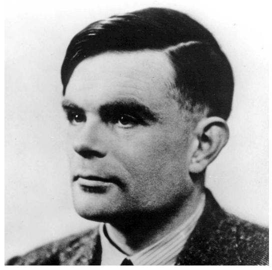
Im-/Possibility results & techniques

- $\mathbb{N}^2 \ni (x,y) \to 2^x\cdot(2y+1)-1 \in \mathbb{N}$
- space-filling curve, fractals
- NFAs equivalent to DFAs
- There is a Python program printing it's own source (w/o file access)
- $\sqrt{2}$ is no fraction
- $2^{\mathbb{N}}$ is uncountable
- and so is $[0;1]$

Alan M. Turing 1936

- first scientific calculations on digital computers
- *What are its fundamental limitations?*





- Undecidable Halting Problem H: **No** algorithm B can always correctly ans ~~simulator/interpreter B~~ ?

Given $\langle A,\underline{x}\rangle$, does algorithm A terminate on input $\underline{x}$?

Proof by contradiction: Consider algorithm B' that, on input A, executes B on $\langle A,A\rangle$ and, upon a positive answer, loops infinitely. How does B' behave on B' ?

Summary of §1

KAIST
CS422 M. Ziegler

The *Theory of Computation*

- considers mathematical models of computers
- (often separating their syntax from semantics),
- explores their capabilities and limitations
- as well as optimal asymptotic algorithmic cost.

We have seen four examples:

- comparison-based branching trees
- finite automata
- unit-cost algebraic / Blum-Shub-Smale machine
- some (unspecific/generic) programming system

§2 Computability Theory

KAIST
CS422 M. Ziegler

- Computability, semi-/decidability, enumerability
- Examples of undecidable problems
- Reduction: comparing problems
- Busy Beaver function
- LOOP programs
- Ackermann function
- WHILE programs

Un-/Semi-/Decidability I

Definition: a) An 'algorithm' $\mathcal{A}$ computes a partial function $f :\subseteq \{0,1\}^* \to \{0,1\}^*$ if it

- on inputs $\underline{x} \in \mathrm{dom}(f)$ prints $f(\underline{x})$ and terminates,
- on inputs $\underline{x} \notin \mathrm{dom}(f)$ does not terminate.

Cmp. [Papadimitriou §3.3], [Sipser §3.2+§4.2]

b) $\mathcal{A}$ decides set $L \subseteq \{0,1\}^*$ if it computes its total char. function: $\mathrm{cf}_L(\underline{x}) := 1$ for $\underline{x} \in L$, $\mathrm{cf}_L(\underline{x}) := 0$ for $\underline{x} \notin L$.

c) $\mathcal{A}$ semi-decides L if terminates precisely on $\underline{x} \in L$

d) $\mathcal{A}$ enumerates L if $L = \mathrm{range}(f)$ for some computable total injective $f : \{0,1\}^* \to \{0,1\}^*$.

Un-/Semi-/Decidability II

KAIST CS422 M. Ziegler

Example: The Halting problem H, considered as subset of $\{0,1\}^*$, is semi-decidable, not decidable.

Theorem: a) Every finite L is decidable.
b) L is decidable iff its complement L^C is.
c) L is decidable iff both L, L^C are semi-decidable.
d) L is enumerable iff infinite and semi-decidable.

b) $\mathcal{A}$ decides set $L \subseteq \{0,1\}^*$ if it computes its total char. function: $\mathrm{cf}_L(\underline{x}) := 1$ for $\underline{x} \in L$, $\mathrm{cf}_L(\underline{x}) := 0$ for $\underline{x} \notin L$.

c) $\mathcal{A}$ semi-decides L if terminates precisely on $\underline{x} \in L$

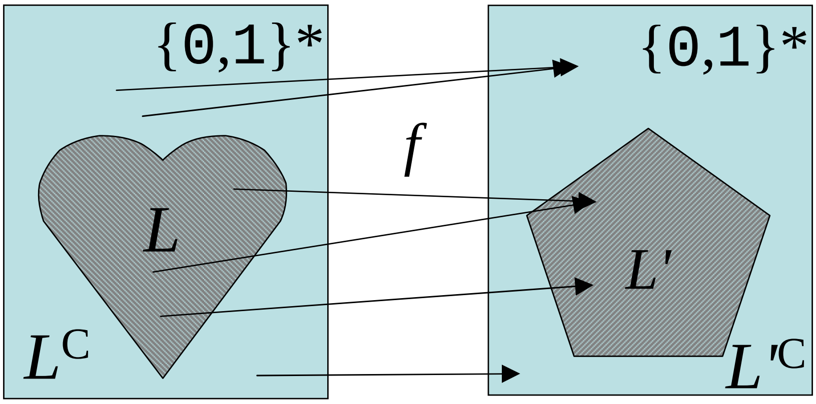
d) $\mathcal{A}$ enumerates L if $L = \mathrm{range}(f)$ for some computable total injective $f : \{0,1\}^* \to \{0,1\}^*$.

Comparing Problems

Halting problem $H = \{ \langle \mathcal{A},\underline{x}\rangle : \mathcal{A}(\underline{x})$ terminates $\}$

Nontriviality $N = \{ \langle \mathcal{A}\rangle : \exists \underline{y}: \mathcal{A}(\underline{y})$ terminates $\}$

Totality problem $T = \{ \langle \mathcal{A}\rangle : \forall \underline{z}\ \mathcal{A}(\underline{z})$ terminates$\}$



- $H \preccurlyeq N$ undecidable
- $H \preccurlyeq T$ undecidable
- $N \preccurlyeq H$
- $T \not\preccurlyeq H$

For $L,L'\subseteq\{0,1\}^*$ write $L\preccurlyeq L'$ if there is a computable $f:\{0,1\}^*\to\{0,1\}^*$ such that $\forall \underline{x}:\ \underline{x}\in L \Leftrightarrow f(\underline{x})\in L'$.

a) L' decidable $\Rightarrow$ so L. b) $L\preccurlyeq L'\preccurlyeq L'' \Rightarrow L\preccurlyeq L''$

LOOP Programs

Syntax in Backus—Naur Form:

$P := (\ x_j := 0 \mid x_j := x_i + 1 \mid P\,;P \mid$
$\quad \text{LOOP } x_i \text{ DO } P \text{ END })$

$\in\mathbb{N}$

Semantics:

- $x_1,\ldots x_k$ contain input $\in\mathbb{N}^k$
- LOOP executed x_j times
- Body must not contain x_j

Example: simulate "if $x_j\neq 0$ then P else Q"

Example: simulate "$x_j := \max(0, x_i-1)$" :

$x_j := 0\ ;\ x_k := 0\ ;$
LOOP x_i DO
$\quad x_j := x_k\ ;\ x_k := x_k + 1$
END

$x_k := 0$; LOOP x_j DO $x_k := 1$ END ; $x_\ell := 1$;
LOOP x_k DO P ; $x_\ell := 0$ END ; LOOP x_ℓ DO Q END

Capabilities of LOOP Programs

Examples: simulate addition "$x_k := x_j + x_i$"

$x_k := 0$; LOOP x_j DO $x_k := x_k + 1$ END;
LOOP x_i DO $x_k := x_k + 1$ END

Simulate multiplication "$x_k := x_j \times x_i$"

$x_k := 0$; LOOP x_i DO $x_k := x_k + x_j$ END

Now recall Ackerman's function (Problem 1d):

$$A(1,n)=2n,\quad A(\ell,0)=1,\quad A(\ell+1,n+1) = A(\ell,A(\ell+1,n))$$

Theorem: • To every LOOP program $P=P(x_1,\ldots x_k)$ there exists some $\ell=\ell(P)\in\mathbb{N}$ s.t. P on input $\underline{x}\in\mathbb{N}^k$ makes at most $A(\ell,n)<\infty$ steps where $n:=\max(1,\|\underline{x}\|_1)$

• $A(n,n)$ is <u>not</u> computable by <u>any</u> LOOP program!

Proof by Structural Induction

P := ($x_j := 0$ | $x_j := x_i + 1$ | $P ; P$ | LOOP x_i DO P END)

Lemma: $A(\ell+1,n+m) = A(\ell,A(\ell,A(\ldots A(\ell,A(\ell+1,n)))))$

Proof, induction: $x_j := 0 \mid x_j := x_i + 1$: $1 \le A(1,1)$ steps

$P ; P$: $A(\ell,n) + A(\ell,A(\ell,n)) \le A(\ell,n) + A(\ell+1,n+1)$
$\le A(\ell+2,n)$ steps

LOOP x_i DO P END:

$A(\ell,n-x_i)+A(\ell,A(\ell,n-x_i))+A(\ell,A(\ell,A(\ell,n-x_i)))+\ldots$ [x_j-iter.] steps
$\le A(\ell+1,n)+A(\ell+1,n)+\ldots \qquad \le A(\ell+2,n)$

Theorem: • To every LOOP program $P=P(x_1,\ldots x_k)$ there exists some $\ell=\ell(P)\in\mathbb{N}$ s.t. P on input $\underline{x}\in\mathbb{N}^k$ makes at most $A(\ell,n)<\infty$ steps where $n:=\max(1,\|\underline{x}\|_1)$

$$A(1,n)=2n,\quad A(\ell,0)=1,\quad A(\ell+1,n+1) = A(\ell,A(\ell+1,n))$$

Power of LOOP Programs 2

Def: Recall bijective $\mathbb{N}^2 \ni (x,y) \to \langle x,y\rangle := 2^x \cdot (2y+1)-1 \in \mathbb{N}$ and write $\langle x,y,z\rangle := \langle\langle x,y\rangle,z\rangle$, $\langle x,y,z,w\rangle := \langle\langle x,y,z\rangle,w\rangle$ etc.

Lemma: a) There exists a LOOP program that, given $x,y\in\mathbb{N}$, returns $\langle x,y\rangle\in\mathbb{N}$.

b) There exists a LOOP program that, given $\langle x,y\rangle\in\mathbb{N}$, returns x and $y\in\mathbb{N}$.

c) There exists a LOOP program that, given integers $n\leq N$ and $\langle x_1,x_2,\ldots,x_n,\ldots,x_N\rangle$, returns x_n.

d) There exists a LOOP program that, given $n\leq N$ and y and $\langle x_1,x_2,\ldots,x_n\ldots,x_N\rangle$, returns $\langle x_1,x_2,\ldots,y,\ldots,x_N\rangle$.

array of integers with indirect addressing

WHILE Programs

Syntax in Backus—Naur Form:

$$P := (\ x_j := 0 \mid x_j := x_i + 1 \ \mid \ P\,;P \mid \text{WHILE } x_j \text{ DO } P \text{ END }\ )$$

body better modify x_j

Semantics: loop executed as long as $x_j \neq 0$

Observation: a) To every LOOP program P there exists an equivalent WHILE program P'.

b) As opposed to LOOP programs, a WHILE program might not terminate (on some inputs).

Question: Does every WHILE program P admit a bound $t(P,n)$ such that P, on inputs $\underline{x}\in\mathbb{N}^k$ on which it does terminate, makes at most $t(P,\|\underline{x}\|_1)$ steps?

First UTM Theorem

KAIST
CS422 M. Ziegler

UTM-Theorem: There exists a LOOP program U' that, given $\langle P\rangle\in\mathbb{N}$ and $\langle x_1,\dots,x_n\rangle\in\mathbb{N}$ and $N\in\mathbb{N}$, simulates P on input $(x_1,\dots,x_n)$ for N steps.

Proof (Sketch): Use one variable y for $\langle x_1,\dots,x_n\rangle$, and z to store the current program counter of P:

Case $\langle P\rangle(z)$:

„x_j:=0" :	$\langle x_1,\dots x_j,\dots,x_n\rangle := \langle x_1,\dots 0,\dots,x_n\rangle$; z:=z+1
„x_j:=x_k+1" :	$\langle x_1,\dots x_j,\dots x_n\rangle := \langle x_1,\dots x_k+1\dots x_n\rangle$; z:=z+1
„WHILE x_j DO" :	if x_j=0 then z:=1+#of corresponding END
„END" :	z := #of corresponding WHILE

Definition: Let $\langle P\rangle\in\mathbb{N}$ denote the encoding of WHILE program P (e.g. as ascii sequence).

Normalform Theorem

KAIST
CS422 M. Ziegler

UTM-Theorem: There exists a LOOP program U' that, given $\langle P\rangle\in\mathbb{N}$ and $\langle x_1,\dots,x_k\rangle\in\mathbb{N}$ and $N\in\mathbb{N}$, simulates P on input $(x_1,\dots,x_k)$ for N steps.

Normalform-Thm: To every WHILE program P there exists an equivalent one P' containing only one WHILE command (and several LOOPs).

Corollary: A WHILE program U can semi-decide the *Halting problem* for WHILE programs, but no WHILE program can decide it.

$$H = \{\ (\langle P\rangle,\langle x_1,\dots x_k\rangle) : P \text{ terminates on input } (x_1\dots x_k)\ \}$$

SMN Theorem: Currying

Definition: Let $\underline{P} = \langle P\rangle \in \mathbb{N}$ denote the encoding of WHILE program P and $P = \rangle\underline{P}\langle$ its inverse/decoding.

Example (Calculus): imagine $f(x,y) = \sin(x)\cdot e^y$

SMN-Theorem a) There exists a WHILE program C that, given $\langle P\rangle \in \mathbb{N}$ and $x \in \mathbb{N}$, returns $\langle P(x,\cdot)\rangle$, where $P(x,\cdot)(x_2,\ldots x_k) :\equiv P(x,x_2,\ldots x_k)$

SMN-Theorem b) There exists a WHILE program D that, given $\langle P\rangle \in \mathbb{N}$, returns $\langle Q\rangle \in \mathbb{N}$ with $Q(x,x_2,\ldots x_k) = \rangle P(x)\langle\ (x_2,\ldots x_k)$ for all $x,x_2,\ldots x_k$

Summary of §2

- Computability, semi-/decidability, enumerability
- Examples of undecidable problems
- Reduction: comparing problems
- LOOP programs
- simulating +, -, ×, ÷, IF-THEN-ELSE
- "implementing" a stack/array
- Ackermann's function and runtime bounds
- WHILE programs
- UTM, Normalform, SMN Theorem

§3 Complexity Theory

- Model of computation with cost
- Complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$
- and their inclusion relations
- Encoding graphs/non-integer data
- Example problems: **EC, HC, VC, ILP, IS, Clique**
- Comparing difficulty: polynom. reduction
- $\mathcal{NP}$ and completeness
- Time hierarchy, $\mathcal{UP}$ and cryptography

Model of Computational Cost

WHILE takes expon. time to add two n-bit integers

Now WHILE+ programs: Input $x_1 \in \mathbb{N}$, output $x_0 \in \mathbb{N}$

$x_j := 0 \mid x_j := 1 \mid x_j := x_i + x_k \mid x_j := x_i - x_k \mid$
$x_j := x_i \div 2 \mid \quad P;P \mid \text{WHILE } x_i \text{ DO } P \text{ END}$

Definitions: binary *length* of $x \in \mathbb{N}$: $\ell(x) := 1 + \lfloor \log_2 x \rfloor$

- time of a WHILE+ program P on input $\underline{x} = (x_1, \ldots x_k)$
- space (=memory) used: $\max_t \ell(\underline{x}) := \ell(x_1) + \ldots + \ell(x_k)$
- asymptotic time/space $t(n)/s(n)$:
  worst-case over all inputs $\underline{x}$ with $\ell(\underline{x}) < n$
- *better* pairing function $\langle x,y \rangle := x + (x+y) \cdot (x+y+1)/2$

Some Complexity Classes

Definition: a) A WHILE+ program computes the function $f:\mathbb{N}\to\mathbb{N}$ if on input x it prints $f(x)$ and terminates in time $t(n)$ / space $s(n)$, $n:=\ell(\underline{x})$

Polynom.growth: $\exists k\ t(n)\leq O(n^k)$; exponential: $2^{O(n^k)}$

Def: For decision problems $L\subseteq\mathbb{N}$ or $L\subseteq\{0,1\}^*$

- $\mathcal{P}$ = { L decidable in polynomial time }
- $\mathcal{NP}$ = { L verifiable in polynomial time }, i.e.

$L = \{\ x\in\mathbb{N} : \exists y\in\mathbb{N},\ \ell(y)\leq\text{poly}(\ell(x)),\ \langle x,y\rangle\in V\ \},\ \ V\in\mathcal{P}$

- $\mathcal{PSPACE}$ = { L decidable in polynomial space }
- $\mathcal{EXP}$ = { L decidable in exponential time }

Theorem: $\mathcal{P}\subseteq\mathcal{NP}\subseteq\mathcal{PSPACE}\subseteq\mathcal{EXP}$

Non-Deterministic WHILE+

Theorem: $L\subseteq\mathbb{N}$ is accepted by a *non*-deterministic polynomial-time WHILE+ program iff $L\in\mathcal{NP}$.

$x_j := 0 \mid x_j := 1 \mid x_j := x_i + x_k \mid x_j := x_i - x_k \mid$
$x_j := x_i \div 2 \mid \text{guess } x_j \mid P;P \mid \text{WHILE } x_i \text{ DO } P \text{ END}$

Definition: A *non*-deterministic WHILE+ program may (repeatedly) guess a bit (0/1).

- Its runtime is $\leq t(n)$ if it makes no more than $t(\ell(x_1))$ steps, regardless of the guesses.
- It accepts input x_1 if there <u>exists</u> some choice of guessed values such as to return x_0=1.
- It rejects x_1 if <u>no</u> choice of guesses returns x_0=1.

Preliminaries: Graphs and Coding

- A *directed* graph $G=(V,E)$ is a finite set V of *vertices* and a set $E\subseteq V\times V$ of *edges*
- Call G *undirected* if it holds $(u,v)\in E \Leftrightarrow (v,u)\in E$
- sometimes $c:E\to\mathbb{N}$ assigning *weights* to edges.

For input to a WHILE+ program:

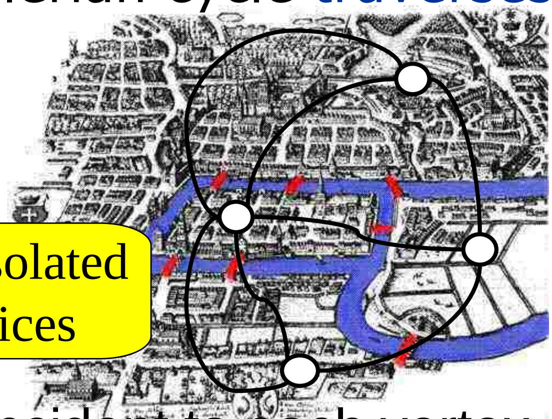
- Represent (G,c) as adjacency matrix $A\in\mathbb{N}^{V\times V}$
  - $A[u,v] := c(i,j)$ for $(u,v)\in E$,
  - $A[u,v] := "\infty"$ for $(u,v)\notin E$
- Undirected case: only upper triangular matrix.
- Encoding $\langle G,c\rangle\in\mathbb{N}$ has $|\langle G,c\rangle| \geq |V|$

Example Problems (I)

KAIST
CS422 M. Ziegler

In an undirected graph *G*, Eulerian cycle traverses each <u>edge</u> precisely once;

Hamiltonian cycle visits each <u>vertex</u> precisely once.



G admitting a Eulerian cycle is connected and has an even number of edges incident to each vertex

save isolated vertices

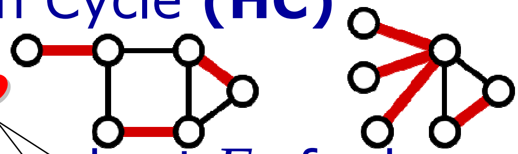
Theorem: Conversely every connected graph with an even number of edges incident to each vertex admits a Eulerian cycle.

EC := { $\langle G\rangle$ | G has a Eulerian cycle} $\mathcal{NP}$

HC := { $\langle G\rangle$ | G has Hamiltonian cycle} $\mathcal{NP}$

Example Problems (II)

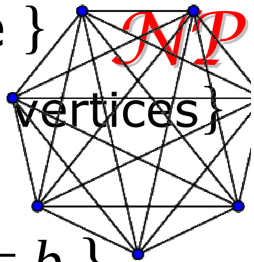
- Eulerian **(EC)** vs. Hamiltonian Cycle **(HC)**
- (Minimum) **Edge Cover** $\mathcal{NP}$



"To graph G, find a smallest subset F of edges s.t. any vertex v is adjacent to at least one $e\in F$."

Greedily extend a maximum matching

- vs. **Vertex Cover (VC)** $\mathcal{NP}$
- **CLIQUE** = { $\langle G,k\rangle$ | G contains a k-clique } $\mathcal{NP}$
- **IS**={$\langle G,k\rangle$: G has k pairwise non-adjacent vertices}



- Integer Linear Programming $\mathcal{NP}$?

ILP = { $\langle \underline{A},\underline{b}\rangle : \underline{A}\in\mathbb{Z}^{n\times m}, \underline{b}\in\mathbb{Z}^m, \exists\underline{x}\in\mathbb{Z}^n: \underline{A}\cdot\underline{x}=\underline{b}$ }

VC = $\{\langle V,E,k\rangle: \exists U\subseteq V, |U|=k, \forall(x,y)\in E: x\in U \vee y\in U\}$

$\mathcal{NP} \ni \{x\in\mathbb{N}: \exists y, \ell(y)\leq \text{poly}(\ell(x)), \langle x,y\rangle\in V\}, \; V\in\mathcal{P}$

Example Problems (III)

Def: A Boolean term $\Phi(Y_1,\ldots Y_n)$ is composed from variables $Y_1,\ldots Y_n$, constants 0 and 1, and operations $\vee, \wedge, \neg$.

Examples:
- 0
- $(\neg x \vee y) \wedge (x \vee \neg y)$
- $(\neg x \vee y) \wedge (x \vee y) \wedge \neg y$
- $(\neg x \vee y) \wedge (x \vee \neg z) \wedge (z \vee \neg y) \wedge x \wedge (\neg y)$

clause

literals

Φ in 3-CNF if $\Phi = \bigwedge\big((\neg)y_i \vee (\neg)y_j \vee (\neg)y_\ell\big)$

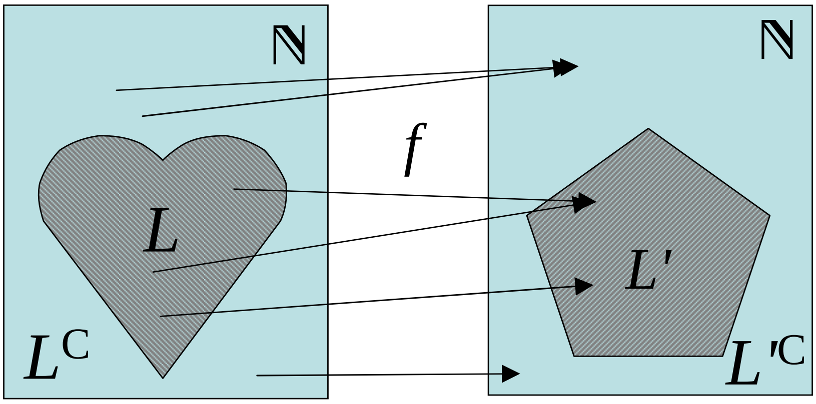
EVAL: Given $\langle\Phi(Y_1,\ldots Y_n)\rangle$ and $y_1,\ldots y_n\in\{0,1\}$, does $\Phi(y_1,\ldots y_n)$ evaluate to 1 ? $\in\mathcal{P}$

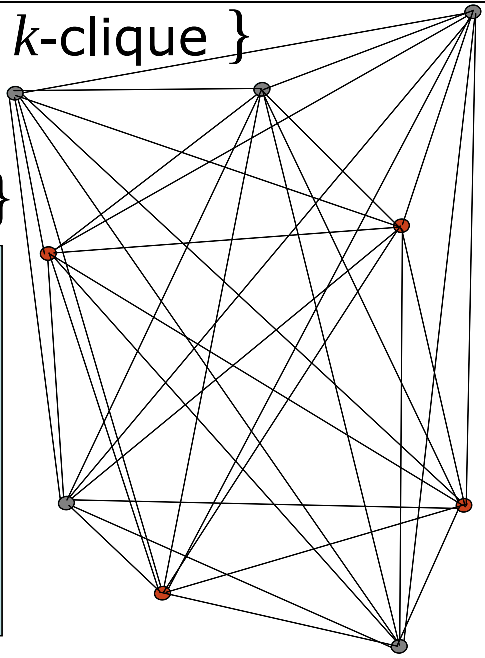
[k-] SAT: Given $\Phi(Y_1,\ldots Y_n)$ [in k-CNF], does it hold $\exists y_1,\ldots y_n\in\{0,1\}: \Phi(y_1,\ldots y_n)=1$?

Comparing Problems, again

CLIQUE = $\{ \langle G,k\rangle \mid G$ contains a k-clique $\}$

$\equiv_p$ **IS**$=\{ \langle G,k\rangle : G$ has k pairwise non-connected vertices$\}$





For $L,L'\subseteq\mathbb{N}$ write $L\leqslant_p L'$ if exists a polynomial-time computable $f:\mathbb{N}\to\mathbb{N}$ such that $\forall \underline{x}:\ \underline{x}\in L \Leftrightarrow f(\underline{x})\in L'$.

a) $L'\in\mathcal{P} \Rightarrow L\in\mathcal{P}$ decidable $\Rightarrow$ so L. b) $L\leqslant_p L'\leqslant_p L'' \Rightarrow L\leqslant_p L''$

Reduction IS $\leqslant_p$ SAT

Goal: Upon input of (the encoding of) a graph G and $k\in\mathbb{N}$, produce in polynomial time a CNF formula Φ such that:
Φ satisfiable iff G contains $\geq k$ independent vertices

Let G consist of vertices $V=\{1,..,n\}$ and edges E.

- Consider Boolean variables $x_{v,i},\ v\in V,\ i=1...k$
  Vertex *v* is #*i* among the *k* independent.
- and clauses $K_i := \bigvee_{v\in V} x_{v,i},\ i=1...k$
  There is an *i*-th vertex
- and $\neg x_{v,i}\vee\neg x_{v,j},\ v\in V,\ 1\leq i<j\leq k$
  Vertex *v* cannot be both #*i* and #*j*.
- and $\neg x_{u,i}\vee\neg x_{v,j},\ \{u,v\}\in E,\ 1\leq i<j\leq k$
  No adjacent vertices are independent.
- Length of Φ: $O(k\cdot n+n\cdot k^2+n^2k^2)=O(n^2k^2)$ since $k\leq n$.
- Computational cost of $(G,k)\to\Phi$: polyn. in $n+\log k$

Example Reduction: 4SAT vs. 3SAT

4-SAT: Is formula $\Phi(\underline{Y})$ in 4-CNF satisfiable?
3-SAT: Is formula $\Phi(\underline{Y})$ in 3-CNF satisfiable?

Given $\Phi = (a \vee b \vee c \vee d) \wedge (p \vee q \vee r \vee s) \wedge \ldots$

with literals $a,b,c,d, p,q,r,s,\ldots.$ (variables, possibly negated)

Introduce new variables $u,v,\ldots$ and consider

$\Phi' := (a \vee b \vee u) \wedge (\neg u \vee c \vee d)$
$\quad \wedge (p \vee q \vee v) \wedge (\neg v \vee \ \vee r \vee s) \wedge \ldots$

$f\!: \langle\Phi\rangle \to \langle\Phi'\rangle$

For $L,L' \subseteq \mathbb{N}$ write $L \preccurlyeq L'$ if exists a computable $f\!: \mathbb{N}\to\mathbb{N}$ such that $\forall \underline{x}$: $\underline{x} \in L \Leftrightarrow f(\underline{x}) \in L'$.

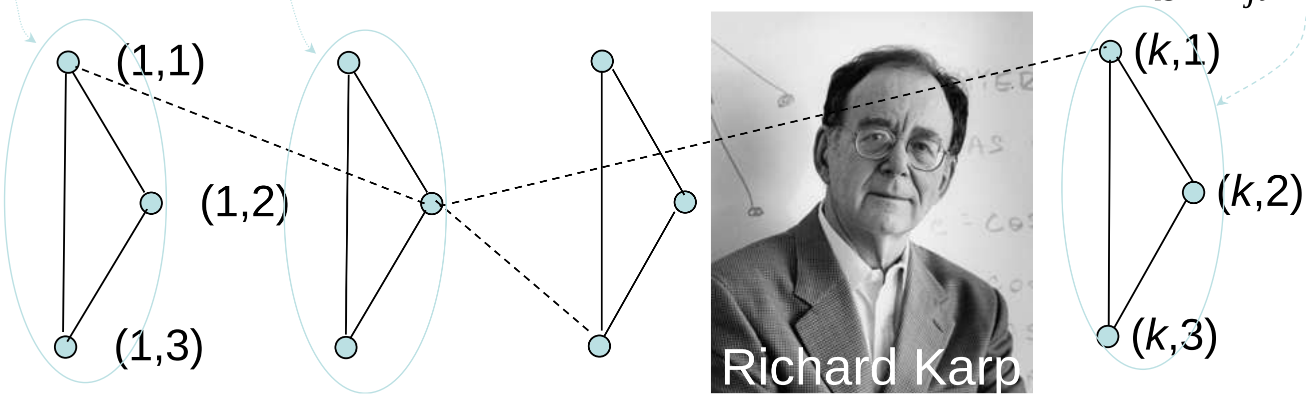
Reduction 3SAT $\preccurlyeq_p$ IS

Produce, given a 3-CNF term Φ, within polynomial time a graph G and integer k such that it holds: *Φ is satisfiable iff G contains k pairwise non-adjacent vertices.*

e.g. $(u \vee .. \vee ..) \wedge (.. \vee \neg u \vee ..) \wedge (.. \vee .. \vee u) \wedge (u \vee .. \vee ..)$

$\Phi = C_1 \wedge C_2 \ldots \wedge C_k$, $C_i = x_{i1} \vee x_{i2} \vee x_{i3}$, x_{is} literals

$V := \{ (i,1),\ldots(i,3) : i \le k \}$, $E := \{ \{(i,s),(j,t)\} : i=j$ or $\overline{x}_{is} = x_{jt} \}$



Problems of similar complexity

unknown yet

- Showed: **CLIQUE** $\equiv_p$ **IS** $\leqslant_p$ **SAT** $\equiv_p$ **3SAT** $\leqslant_p$ **IS.**
- These 4 problem have about same complexity:
  - Either all are belong to $\mathcal{P}$, or none of them.
- We will show: Also **TSP**, **HC**, **VC** and many further problems in $\mathcal{NP}$ belong to this class called $\mathcal{NP}$c.
- **And** will show: These are ‘hardest‘ problems in $\mathcal{NP}$.
  <u>*Cook–Levin Theorem*</u>: Every $L \in \mathcal{NP}$ has $L \leqslant_p$ **SAT.**
- That is, if someone finds a polynomial time algorithm for <u>any</u> problem in $\mathcal{NP}$c, this would prove $\mathcal{P}=\mathcal{NP}$:
- A deterministic **`WHILE+`** program could simulate <u>any</u> *non*-deterministic one with polynomial slowdown!
- And, conversely, a proof that <u>any</u> of these probleme cannot be solved in polynomial time implies that <u>no</u> problem in $\mathcal{NP}$c can be solved in polynomial time!

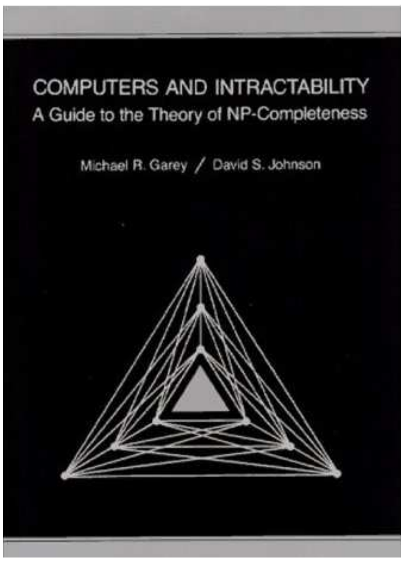
Complexity Class Picture

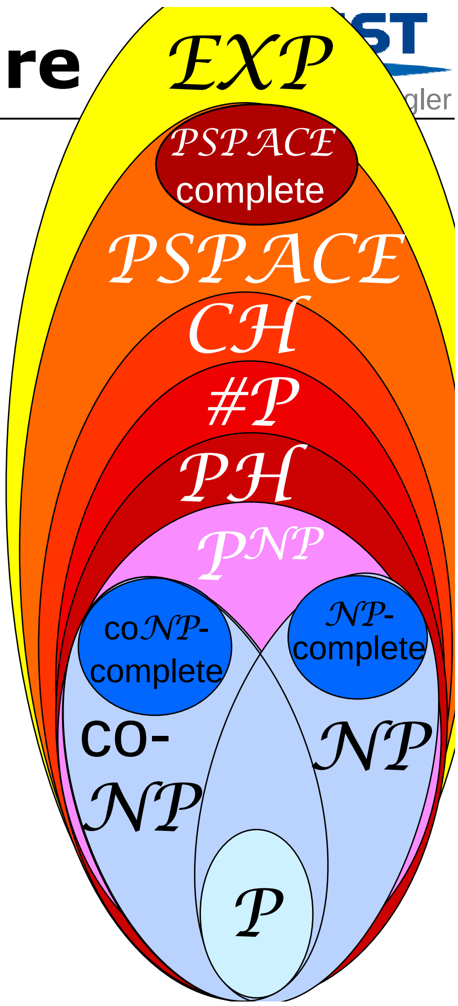
Def: $A \in \mathcal{NP}$ is $\mathcal{NP}$-complete if $L \leqslant_p A$ holds for every $L \in \mathcal{NP}$.

Theorem (Cook'72/Levin'71): **SAT** is $\mathcal{NP}$-complete!

Lemma: For A $\mathcal{NP}$-complete and $A \leqslant_p B \in \mathcal{NP}$, B is also $\mathcal{NP}c$.

Now know ≈500 natural problems $\mathcal{NP}$-complete…





Master Reduction

The following problem **UNP** is $\mathcal{NP}$-complete:

$\{\ \langle\mathcal{A},x,2^N\rangle$: nondetermin. **`WHILE+`** program $\mathcal{A}$ accepts input x within at most N steps $\}$

Proof: **UNP**$\in\mathcal{NP}$: √

Let $L\in\mathcal{NP}$ be arbitary but fixed.

There exists a nondeterministic **`WHILE+`** prog. $\mathcal{A}$ accepting L in time $p(n)$ for some polynomial p.

Reduction $x\to\langle\mathcal{A},\ x,\ 2^{p(\ell(x))}\rangle$. ∎

$\mathcal{NP}\ni\{x\in\mathbb{N}:\ \exists y,\ \ell(y)\leq\text{poly}(\ell(x)),\ \langle x,y\rangle\in V\},\ V\in\mathcal{P}$

SubsetSum is $\mathcal{NP}$-complete

$\{\ \langle a_1,\ldots a_N,b\rangle \mid a_1,\ldots a_N,b\in\mathbb{N},\ \exists\alpha_1,\ldots\alpha_N\in\{0,1\}:\ b=\sum_i a_i\cdot\alpha_i\ \}$

- **SubsetSum**$\in\mathcal{NP}$ √ Show: **3SAT** $\leqslant_p$ **SubsetSum**
- In polyn.time: 3CNF $\Phi\to A\subseteq\mathbb{N}$ and $b\in\mathbb{N}$ s.t.
- $\exists$satisf. assignm. of Φ $\Leftrightarrow$ $\exists B\subseteq A$: $b=\sum_{a\in B} a$

Eg. $\Phi = (x_1\vee\neg x_3\vee x_5)\wedge(\neg x_1\vee x_5\vee x_4)\wedge(\neg x_2\vee\neg x_2\vee\neg x_5)$

v_1 := 100 10000	v_1' := 010 10000	b := 444 11111
v_2 := 000 01000	v_2' := 002 01000	c_1:= 100 00000
v_3 := 000 00100	v_3' := 100 00100	d_1:= 200 00000
v_4 := 010 00010	v_4' := 000 00010	c_2:= 010 00000
v_5 := 110 00001	v_5' := 001 00001	d_2:= 020 00000
		c_3:= 001 00000

m clauses in n var.s $\to$ $2n+2m+1$ values à $n+m$ dec.digits

Time Hierarchy Theorem

The following problem **UTIME³** can be decided in time $O(n^5)$ but not in time $O(n^2)$:

$\{ \langle \mathcal{A}, 2^N \rangle$: deterministic **WHILE+** program $\mathcal{A}$ does not accept input $\langle \mathcal{A}, 2^N \rangle$ within at most $(|\langle \mathcal{A} \rangle| + N)^3$ steps $\}$

Proof: **UTIME³** decidable in time $O(n^5)$. √

Suppose $\mathcal{B}$ decides **UTIME³** in $\leq K \cdot n^2$ steps, $K \in \mathbb{N}$.

$N \gg K$

Case $\langle \mathcal{B}, 2^N \rangle \in$ **UTIME³**: contradiction.

Case $\langle \mathcal{B}, 2^N \rangle \notin$ **UTIME³**: contradiction.

$\mathcal{U}$ simulates $\mathcal{A}$ on input $\underline{x}$ in time $|\langle \mathcal{A} \rangle|^2 + |\langle \underline{x} \rangle|^2$ per step

Complexity and Cryptography

A Public-Key System with key-pair $(\underline{e}, \underline{d})$ consists of two functions $E(\underline{e}, \underline{x})$ and $D(\underline{d}, \underline{y})$ such that $D(\underline{d}, E(\underline{e}, \underline{x})) = \underline{x}$ holds for all $\underline{x}$.

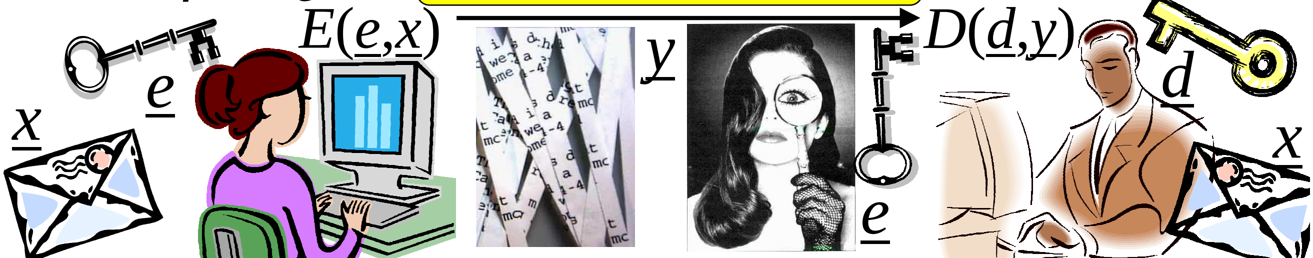
RSA

Call $f: \mathbb{N} \to \mathbb{N}$ a one-way function if

i) injective and $\ell(\underline{x})^k \geq \ell(f(\underline{x})) \geq \ell(\underline{x})^{1/k}$ for some k

ii) computable in polynomial time (i.e. $f \in \mathcal{FP}$)

iii) but $f^{-1} \notin \mathcal{FP}$ impossible if $\mathcal{P} = \mathcal{NP}$ $\Rightarrow f^{-1} \in \mathcal{FNP}$



encrypt with private key $\underline{e}$, decrypt with public key $\underline{d}$.

One-Way Functions and $\mathcal{UP}$

Definition: Call a nondeterm. **WHILE+** program unambiguous if, for any input x, it has at most one accepting computation.

$\mathcal{P} \subseteq \mathcal{UP} \subseteq \mathcal{NP}$

$\mathcal{UP}$ = { decision problems accepted by unambiguous polynomial-time nondetem. **WHILE+** programs}

Theorem: $\mathcal{P} \neq \mathcal{UP}$ iff one-way functions exist.

Proof $\Leftarrow$: For one-way f let $L := \{ (x,y) \mid \exists z \leq x: f(z)=y \}$
Then $L \in \mathcal{UP}$. Binary search with polynomially many queries for $L \in \mathcal{P}$ would imply $f^{-1} \in \mathcal{FP}$.

$\Rightarrow$: Let $\mathcal{UP} \setminus \mathcal{P} \ni L = \{ x \mid \exists y : (y) \leq \ell(x)^k, \langle x,y \rangle \in V \}$
and define $f(\langle x,y \rangle) := 2x+1$ for $x \in L$; else $f(z) := 2z$.
This is one-way!

Summary of §3

- Model of computation *with* (bit) cost
- Complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{PSPACE}$, $\mathcal{EXP}$
- and their inclusion relations
- Encoding graphs/non-integer data
- Example problems: **EC, HC, VC, ILP, IS, Clique**
- Comparing difficulty: polynom. reduction
- $\mathcal{NP}$ and completeness
- Time hierarchy, $\mathcal{UP}$ and cryptography

Conclusion

The *Theory of Computation*

- considers mathematical models of computers
- (often separating their syntax from semantics),
- explores their capabilities and limitations
- as well as optimal asymptotic algorithmic cost.

§1 Motivation and Examples

§2 Computability Theory

§3 Complexity Theory

Perspectives

CS500 Design and Analysis of Algorithms (M.Z.)

CS520 Theory of Programming Languages

CS522 Theory of Formal Languages and Automata

CS548 Advanced Information Security

CS610 Parallel Processing

CS624 Program Analysis

CS700 Topics in Computation Theory

CS712 Topics in Parallel Processing

Theory of Computation Seminar