Faculty of Mathematics, Informatics and Mechanics, UW

Computational Complexity

Problem book

Spring 2011

Coordinator Damian Niwiński

Contributors:

Contents

Chapter 1

Turing Machines

Contributed: Vince Bárány, Bartek Klin, Henryk Michalewski, Damian Niwiński

1.1 Machines in general

1. From Arora and Barak [1], Chap. 0. The sculpture Machine with Concrete by Arthur Ganson consists of 13 gears connected in a row such that the next gear moves 50 times slower than its predecessor. The first gear is supported by an engine and makes 212 rotations per minute. The last gear is fixed to a block of concrete and apparently does not move at all. Explain how it is possible. 

1.2 Coding objects as words and problems as languages

1. Try to find an efficient way to encode a **pair** of binary words as a single binary word.

   Note. Let $C(a_1a_2\dots a_k) = a_10a_20\dots a_k011$. The first approximation codes (u,v) by $C(u)v$, but it can be improved to $C(\alpha)uv$, where α is the binary representation of the length of u. Now this construction can be iterated...

2. The incidence matrix of a directed graph with n vertices, written as a single word of length n^2, has 1 on position $(i-1)n+j$ if there is an arrow $i \to j$, and 0 otherwise. Note that no infinite set of graphs is encoded by a context-free language (why ?).

   But with some care we can find some (not completely trivial) graph-theoretic properties encoded by complements of context-free languages. Give an example.

   Hint. Graphs with only self-loops $i \to i$. But maybe you will find something better ?

3. Propose an alternative encoding of symmetric graphs using the concept of adjacency list (Pol. listy sąsiedztwa).

4. Show that if $\alpha : \mathbb{N} \to \{0,1\}^*$ is a 1:1 function then, for infinitely many n's $|\alpha(n)| \geqslant \lfloor \log_2 n \rfloor$.

   Remark. This means that no notation for natural numbers can be more efficient than the usual binary notation.

   (It is the content of Corollary 1 in `http://www.mimuw.edu.pl/~wjaworski/ TI/notatki_20-12.pdf`.)

1.3 Turing machines as computation models

1. **Robustness.** Show the equivalences between various models of Turing machines: two-way infinite tape vs one-way infinite tape; k tapes vs 1 tape. Estimate the cost of simulation. Show that the

alphabet of auxiliary symbols can be reduced to $\{0,1,B\}$ (in fact, $\{1,B\}$ would suffice as well). *Hint.* Assume M is a TM with 2 tapes. We define N a TM with 1 tape which will accept exactly the words accepted by M. Let A be the alphabet of M. The alphabet of N is $\{a,\dot{a},\sharp : a \in A\}$. We assume that on the first tape of M there is the input word $w_1w_2\dots w_n$ and the second tape of M is empty. In the first step machine N rewrites a given word $w_1w_2\dots w_n$ in the form $\sharp\dot{w}_1w_2\dots w_n\sharp\dot{\sqcup}\sharp$. Dots represent the positions of heads on the two tapes of M. If N reaches $\sharp$ and is forced to write down something in this cell, it moves one position right all cells from$\sharp$ to end of the word.

2. Are the following modifications of Turing machine equivalent to the original?
   (a) The machine's head always moves right. (No, it is equivalent to finite automata).
   (b) The machine can only write on the input word, not in empty cells. (No, the halting problem is decidable.)
   (c) Machine that rejects immediately if it visits the same configuration twice. (Yes: it would loop anyway).

3. **Write–once Turing machine.** A Turing which writes a symbol in a cell only if it is empty (i.e., contains a blank) can simulate an arbitrary Turing machine. We do not put restriction on the number of tapes.

4. If we additionally require that a machine of (3) has only **one tape** then it can recognize (only) a regular language. *Hint.* Let M be a TM and let A be the alphabet of M an Q be the states of M. For $s \in A^*$ assume that s is the input word of M, that is s is the read–only part of the tape, and define $F_s : Q \cup \{f\} \to Q \cup \{r,a\}$ such that
   (a) we start M from the initial state at the right end of s and $F_s(f)$ is the state M assumes when it moves right from s for the first time;
   (b) we start M from the state q at the right end of s and $F_s(q)$ is the state M assumes when it moves right from s for the first time,
   (c) in the above cases it is $F_s(f)$ and $F_s(q)$ is r, if M never leaves the read–only part of the tape and rejects the word s directly or via looping and it is a if M never leaves the read–only part of the tape and accepts s.

   Now define the states of a deterministic automaton $\mathcal{B}$ over the alphabet A as $\{F_s : s \in A^*\}$ and declare the accepting states of $\mathcal{B}$ as $\{F_s : s \text{ accepted by } M\}$. For a state F_s and letter $a \in A$ the automaton moves to the state F_{sa}.

5. ([2]) Each Turing machine can be simulated by a machine with **3** states.

   Hint. That a fixed number can be achieved follows from the construction of universal machine. Note that auxiliary symbols are needed to produce the encoding of the simulated machine. However, to achieve 3 states, another idea is needed. The states of an original machine are encoded by auxiliary tape symbols which most conveniently can be viewed as sequences of bits. Moving of such a "symbol" from one cell to another is performed in many steps, "bit by bit".

6. **Reversible computation.** We call a configuration c of a machine M reversible if there is at most one configuration d, such that c is reached from d in one step. The machine M is weakly reversible if any configuration reachable from some initial configuration is reversible. Note that if it is the case then we can trace the computation back.

   For a given Turing machine M, construct a weakly reversible machine recognizing the same language; estimate the time overhead in your construction.

   Remark. A machine is (strongly) reversible if all its configurations are reversible. Note that this implies that all maximal computation paths are disjoint. The analogous claim for reversible machines is problematic, see discussion on `http://mathoverflow.net/questions/24083/reversible-turing-machines`.

1.4 Turing machines — basic complexity

1. **Constructible functions.** A function $f : \mathbb{N} \to \mathbb{N}$ is space constructible if there is an off-line Turing machine which, for an input of length $n \geqslant 1$, writes in exactly $f(n)$ cells of the auxiliary tapes. Show that the following functions are space constructible: $n, 2n, n^2, n^2 + n, 2^n, \lceil \log n \rceil$.

   A function f is time constructible if there is a Turing machine which, for an input of length $n \geqslant 1$, makes exactly $f(n)$ steps and halts. Show that the following functions are time constructible: $n, 2n, n^2, 2^n, 2^{2^n}$.

2. Assuming that natural numbers k,m,n are given in binary representation, estimate (from above) the time to compute $m + n$, $m \bmod n$, $m \cdot n$, $m^n \bmod k$.

3. Estimate the computation time of the **Euclide algorithm** implemented on Turing machine.

1.5 One-tape Turing machines

1. Construct a 1-tape DTM, which recognizes the language $\{0^n 1^n : n \in \mathbb{N}\}$ in time $\mathcal{O}(n \log n)$.

   Hint. Implement a counter moving along with the head.

2. A one-tape Turing machine which operates in **linear time** can only recognize a regular language[1].

   Hint. Use a concept of a history of a cell similar to that of crossing sequence in two-way finite automata. A history gathers the information how many times the head has visited the cell, in which states, in what direction it moved, etc. Note that the sum of the heights of all histories gives the time of computation which, by assumption, is $K \cdot n$, for some K. Show that there is a common bound K' on the height of all histories. (From this it is routine to construct a finite automaton.) To show the claim suppose that there is an accepting computation with some history of a huge height $\geqslant M$. Choose a shortest word w_M which witnesses this fact. As the "average" height is K, M must be balanced by a sufficient number of histories shorter than K. But there is only a finite number of them. Thus, with an appropriate relation between K and M, we can find two identical histories in two different points lying on the same side of the huge history. Then, by anti-pumping we can make w_M shorter, contradicting its minimality.

3. Show that a one-tape Turing machine recognizing **palindromes** requires $\Omega(n^2)$ steps.

   Hint. Introduce and use Kolmogorov complexity; see Exercise 2.8.5 in Papadimitriou [4] and the hint there.[2]

1.6 Decidability and undecidability

1. Show that the following conditions are equivalent for all languages L:

   (a) L is recursively enumerable (RE),

   (b) L is the domain of some partial computable function,

   (c) L is the range of some partial computable function,

   (d) L is the range of some (total) computable function.

2. Show that a languange is computable iff it is finite or is the range of some computable function that is monotone (wrt. lexicographic ordering).

3. (Turing–Post Theorem) If a language and its complement are RE, the language is decidable.

4. Show that there is a language that is RE, whose complement is infinite but contains no infinite RE set.

[1] I mean a deterministic machine here, although the proof should work for non-deterministic machines as well.
[2] This exercise is very optional.

5. Assume that L is decidable. Is the following language decidable:

$$\{w \mid \text{there is } u \in L \text{ such that } u \text{ and } v \text{ are equal up to at most two positions}\} \text{ ?}$$

6. Let us consider TMs over the input/output alphabet $\{a,b\}$. Are the following problems decidable? Are they RE?

   (a) Does the given machine accept only words that do not contain letter b?

   (b) Does the given machine accept only words that contain letter b?

   (c) Does the given machine accept some word that contains letter b?

   Hint: Give a Karp reduction from the complement of the halting problem to (a) and to (b).

Chapter 2

Circuits

Contributed: Vince Bárány, Henryk Michalewski, Damian Niwiński

2.1 Circuit basics

1. Prove that almost all Boolean functions have exponential circuit complexity.

   Hint: The number of functions $\{0,1\}^n \to \{0,1\}$ that can be computed by a circuit of size (number of edges/wires) s is at most $s2^s$; by circuits of size s and on $v < s$ gates/nodes is at most $\binom{v}{2}^s < v^{2s}$.

2. Show that any function $F : \{0,1\}^* \to \{0,1\}^*$ of time complexity $t(n)$ can be computed by uniform circuits of size $poly(t)$, i.e. $t(n)^{O(1)}$.

   Hint: The circuit C_n computing F on inputs of n-bits has $t(n)$ layers each of $s(n) \leqslant t(n)$ nodes, each layer representing the instantenious description at the corresponding time step of the $t(n)$-time TM computing F. Uniform wiring connecting adjacent layers can emulate the evolution of the TM.

3. *The parity function* $P_n : \{0,1\}^n \to \{0,1\}$ is defined as $P(x_0 \dots x_{n-1}) = \sum_{i=0}^{n-1} x_i \mod 2$. Construct a circuit computing P_3 and do the computation for the input 011.

4. Let $A_n : \{0,1\}^{2n} \to \{0,1\}^{n+1}$ be the mapping which adds two binary sequences $x_0 \dots x_{n-1}$ and $x_n \dots x_{2n-1}$. Construct a circuit of size $O(n)$ with n outputs computing A_n.

5. Define $M_n : \{0,1\}^n \to \{0,1\}$ as 0 if $\sum_{i=1}^{n} x_i < n/2$ and as 1 otherwise. Construct a circuit of size $O(n \log n)$ computing M_n. *Hint.* One can apply Exercise 4.

6. Show that any regular langauge $L \subseteq \{0,1\}^*$ can be recognized by a sequence of circuits of polynomial size and depth $\mathcal{O}(\log n)$, even if we require that fan-in of each gate is at most 2.

   Hint. The proof is easiest if we apply the monoid recognizability of regular languages.

7. A regular language is star free if it can be represented by a star-free regular expression

   $$R := a|\emptyset|(\neg R)|(RR)|(R + R)$$

   Show that a star-free regular language can be recognized by a sequence of circuits of polynomial size and constant depth. (Here we asume arbitrary fan-in of the gates.)

   (Logspace uniformity of the circuit family should be obvious, putting this porblem in AC^0.)

   Remark: notice that the only operation requiring unbounded fan-in is concatenation, and, crucially, only for OR gates. All AND gates in the above are of fan-in 2.

   Motivated by this, one can introduce "semi-bounded fan-in circuts" SAC^i, where only one type of gates (typically AND gates) are required to be of bounded fan-in.

8. More generally, every first-order definable class of graphs (or whatever structures) can be recognized by an AC^0 circuit family.

   Hint: Assume that the formula is in negation normal form. Then the circuits will have one gate for each pair (ϑ,η) for subformulas ϑ and assignments η of elements to free variables of ϑ. The type of the gate matches the type of outermost logical connective of ϑ and the tributaries to the gate are the gates corresponding to the most immediate subformulas of ϑ with assignments compatible with η.

9. Another problem in AC^0 is Boolean matrix multiplication.

   Note: the multiplication scheme $(AB)_{i,j} = \Sigma_k A_{i,k} B_{k,j}$ for matrices in general instantly leads to the circuit consisting of output gates for every pair (i,j) of indices wired as in $\bigvee_k x_{i,k} \wedge x_{k,j}$. (For inputs of size not equal to some n^2 the corresponding circuit is void/undefined.) The circuit family thus defined is obviously logspace uniform.

   (Again, note that only OR gates are of unbounded fan-in.)

10. Consider circuits, where instead of Or and And gates we use only one type of the gate, namely Majority gate. The Majority gate with n inputs is like M_n from Exercise 5. Show that any circuit can be transformed to an equivalent circuit of this new type with at most twice as many gates.

11. Show that the functions P_n of Exercise 3 can be computed by circuits with Majority gate (Exercise 10) of polynomial size and constant depth.

    Remark. A difficult result by Furst, Saxe, and Sipser says that it is not possible for And-Or circuits (see Exercise 2.2.1).

12. Prove that the function M_n is computable using a circuit of size $O(n)$. *Remark.* A circuit is *monotone* if it uses only OR and AND gates. The following strengthening of this exercise was proved by Hoory, Magen and Pitassi: M_n is computable using a mototone circuit of size $O(n)$ and depth $O(\log(n))$.

13. Prove that a Boolean formula induces a monotone circuit if and only if changing one of the inputs from false to true cannot change the value of the formula from true to false.

14. Let $F = \{0,1\}^{\{0,1\}^n}$ and let G consists of $f \in F$ computable by a circuit of size at most $\frac{2^n}{10n}$. Prove that
$$\frac{|G|}{|F|} \to 0.$$
    Remark. This is a variation of a result of Shannon from 1947. An informal interpretation of this result is that a random function $f : \{0,1\}^n \to \{0,1\}$ is not computable by a circuit of size at most $\frac{2^n}{10n}$.

15. Prove that any function $f : \{0,1\}^n \to \{0,1\}$ is computable by a circuit of size $1000\frac{2^n}{n}$.

16. Describe a decidable language L_1 and an undecidable language L_2 which are $\mathbf{P/poly}$ but not in $\mathbf{P}$. *Hint.* Consider a given language L consider
$$U = \{1^n : \text{binary expansion of } n \text{ belongs to } L.\}$$

17. Show that $NC = AC \subseteq P$. Of course, $NC = \bigcup_i NC^i$ and AC is defined similarly, hence their equality follows from the fact that $NC^i \subseteq AC^i \subseteq NC^{i+1}$ for all i. Evaluation via "dynamic programming", i.e., bottom-up calculation of the value at each gate of a given circuit on a given input runs in polynomial time (and uses linear space) in the size of the circuit. Recall that in each of the above circuit classes the circuit families are required to be logspace uniform. Hence for all circuit families in NC or AC both the construction of the appropriate circuit of a family for a given input and its evaluation on the input can be performed in polynomial time.

18. Show that $\mathrm{NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{AC}^1$.

    For $\mathrm{NC}^1 \subseteq \mathrm{L}$ we have to devise an approach to circuit evaluation different from the above. Two improvements need to be made. First, the circuit of the right size cannot be constructed in logarithmic space, instead, it must be calculated over and over again on-the-fly while reusing space. Second, a top-down procedure traversing the circuit from the output gate downward in manner of a depth-first search can be implemented in space proportional to the depth of the circuit, assuming it is of bounded fan-in. (This recursive procedure uses its storage as a call stack, storing the current path leading from the output gate to the current gate under evaluation.)

    For $\mathrm{NL} \subseteq \mathrm{AC}^1$ we need – assuming the notion of completeness has been introduced – to define appropriate circuits solving the s-t connectivity problem in directed graphs.

    A solution is to use Boolean matrix multiplication applied to the adjacency matrix of the graph. More precisely, starting from the matrix $B = A + I$ where A is the adjacency matrix, we iterate squaring calculating $B^2, B^4, \ldots, B^{2^{\lceil \log n \rceil}}$ – the matrix B^{2^m} is represented by a layer of n^2 gates at height $2m$ linked to the previous layer as in Exercise 9 above – and selecting the gate indexed (s,t) from the top-most layer.

    Alternatively, one can directly encode Savitch's approach to reachability leading essentially to the same circuits but without mention of adjacency matrices. Yet another way to think of this is in terms of an alternating algorithm existentially guessing a middle point of a connecting path and universally verifying connectivity of the guessed middle point with both endpoints. This algorithm uses only a logarithmic number of alternations between universal and existential modes, but the existential guesses each involve a logarithmic number of bits in the size of the input, hence the running time is $O(\log^2 n)$. (Note that ALOGTIME is NC^1.)

2.2 More on the parity function

1. Show that if a DNF or CNF formula computes the parity function P_n, then it has to be of exponential size.

2. (Hastad's Switching Lemma) Assume, that a given function $f : \{0,1\}^n \to \{0,1\}$ is representable as $k - DNF$ that is as OR of AND's where each AND involves at most k variables. A *restriction* ρ is a partial assignment of t bits from the input of the function f. Fix $x \geqslant 2$. We say that a restriction ρ is *convenient*, if $f|\rho$ is representable as $s - CNF$, that is as AND of OR's, where each OR involves at most s variables. Prove that probability of choosing an inconvenient ρ is less or equal $(\frac{(n-t)k^{10}}{n})^{s/2}$.

3. Assume, that there are given circuits C_n computing parity functions P_n $(n \in \mathbb{N})$. Prove, that there exists a constant D and circuits C'_n of size at most $D \cdot |C_n|$ such that C'_n also compute parity function P_n $(n \in \mathbb{N})$ and C'_n has the following additional properties:

   (a) all not gates are at the input level,

   (b) on the each level of the tree there are either only $\wedge$ gates or only $\vee$ gates.[1]

4. Assume, that there are given circuits C_n of polynomial size computing parity functions P_n $(n \in \mathbb{N})$. Prove, that C_n has to be of arbitrary depth. *Hint.* Assume to the contrary that C_n are of bounded depth and using Haastad's Lemma reduce the depth to 2. *Remark.* This result was obtained by Ajtai and independently by Furst, Saxe and Sipser. Haastad's Lemma was proved later and the Lemma can be considered a useful combinatorial blackbox.

[1] More ornamental properties may be useful in the next exercise.

2.3 Circuit depth and communication complexity

1. (cf. [Arora-Barak, Problem 12.10] where this is attributed to Karchmer-Wigderson) To every Boolean function $f : \{0,1\}^n \to \{0,1\}$ one can associate the following communication problem – a promise problem, in fact. The two parties Alice and Bob receive input words x and y, respectively, both x and y of length n, with the guarantee that $f(x) = 1$ and $f(y) = 0$. In particular, $x \neq y$. The task of Alice and Bob is to jointly discover a position $1 \leqslant i \leqslant n$ such that $x_i \neq y_i$ with the least possible number of bits of communication between them.

   (As for every communication problem, there is always a trivial solution with communication cost n+1: Alice sends her input to Bob, who computes the answer and subsequently sends it back to Alice. This is in analogy with Kolmogorov complexity, where n+C is a trivial upper bound. The challenge is, of course, to do better than that. Warm-up: demonstrate that the communication complexity of checking equality of the inputs of Alice and Bob is n.)

   Give a communication protocol solving this problem assuming a Boolean circuit computing f is at hand. For simplicity and convenience of illustration we may take fan-in 2 circuits in which negations have been pushed to the input level. How are depth of the circuit and the worst-case communication cost related?

   Hint: Alice and Bob each have knowledge of the circuit. By assumption $f(x) = 1$ and $f(y) = 0$, in other words, the values assumed at the output gate of the circuit differ on inputs x and y. The protocol is aimed at exploiting this fact and maintaining it invariant while descending the circuit from the output gate towards input gates at the lowest level tracing a path of gates at which the values on input x and y are distinct. This way, once an input bit (or its negation) is reached, they will have found their solution.

   Here is the protocol: if the current gate is an AND gate, then whoever has value 0 for their input there is obliged to communicate to the other party which of the two tributary gates (left or right) has the value 0 (breaking ties arbitrarily). Dually, if the current gate is an OR gate, then whoever has value 1 there communicates to the other which of the tributary gates is yielding the value 1. In both cases both Alice and Bob proceed to the chosen tributary gate. The worst-case communication cost of this protocol is equal to the depth of the circuit.

   Moral: lower bounds on communication complexity result in lower bounds on circuit depth!

   How would the protocol and its complexity change if we started with a circuit of unounded fan-in?

   Addendum: Show that every communication protocol (irrespective of the computational (super-)powers Alice and Bob might possess) solving the above promise problem for f can be turned into a circuit computing f. In fact, the above construction can be reversed almost literally.

2. [Arora-Barak, Problem 12.11] taken literally from there:
   "Use the previous question to show that computing the parity of n bits requires depth at least $2\log n$."

   But it cannot be that easy, now can it? Hints needed....

   See [Arora-Barak, Chapter 12], or http://www.cs.uiuc.edu/class/sp08/cs579/slides/CC-S08-Lect23.pdf for a crash course on lower-bound methods in comm.compl. for Boolean functions. Though that is not what we are dealing with here.

2.4 Machines with advice

Say that a TM M computes $F : \{0,1\}^* \to \{0,1\}^*$ with advice of length $l(n)$ if there is a sequence of advice $(a_n)_n$ such that $|a_n| \leqslant l(n)$ for all n and $M(x, a_{|x|}) = f(x)$ for all $x \in \{0,1\}^*$.

The complexity class P/poly is the family of languages computable by a deterministic polynomial-time TM with polynomial length advice.

1. Prove that there exists a function that can be computed with advice of length 1 but not without any advice.

   Hint: Equiavlently, some non-computable functions can be computed by circuits of constant size, outputing 0 or 1 depending on the length n of the input according to a non-computable dependence.

2. Let SPARSE be the class of sparse languges, i.e. those $L \subseteq \Sigma^*$ such that $l(n) = |L \cap \Sigma^n| = n^{O(1)}$. Show that SPARSE $\subseteq$ P/poly

3. Showe that for every language L we have $L \in$ P/Poly if and only if there exists a sparse language S and a polynomial reduction from L to S. *Remark.* One implication follows from the previous exercise and the other implication requires the right definition of the set S — for details consult the file mahaney .pdf in the the svn directory.

4. Show that if there is some NP-complete language L in SPARSE then P=NP.

   Source: http://www.cs.umd.edu/~gasarch/858/sparse.pdf

   The idea is this. Instead of reducing from SAT one can work with LSAT = the set of those pairs (ϕ,y) with $y \in \{0,1\}^n$ an assignment to the variables of ϕ such that there exists a lexicographically not greater ($\preceq$) satisfying assignment for ϕ. It is easy to see that LSAT is NP-complete. Assuming some PTime many-one reduction $f : \text{LSAT} \to L$ we will show that SAT can be solved in polynomial time.

   Note that for every propositional formula ϕ on n variables and $y \in \{0,1\}^n$ we have $|f(\phi,y)| \leqslant p(n)$ for some polynomial p. Let $q(n) = |L \cap \{0,1\}^{\leqslant p(n)}|$, also a polynomial.

   Now when searching for a satisfying assignment for a SAT instance ϕ the idea is to subdivide the set of candidate assignments into $q(n) + 1$-many equal-sized intervals (simply checking a remaining $\leqslant q(n)$-many lex-minimal assignments from among the candidates for satisfaction of ϕ as needed) with endpoints $y_0 \prec y_1 \prec \cdots \prec y_{q(n)} = 1^n$. Applying the reduction consider now $f(\phi,y_i)$ for $0 \leqslant i \leqslant q(n)$. By choice of q either some $f(\phi,y_i)$ is not in L, in which case $f(\phi,y_0) \notin L$ by the definition of LSAT; or there are some $i < j$ such that $f(\phi,y_i) = f(\phi,y_j)$ (hence the LSAT-answer is the same for both (ϕ,y_i) and (ϕ,y_j) depending on whether $f(\phi,y_i) = f(\phi,y_j) \in L$ or not, though we do not know which of these cases hold). In the first case we can drop the interval $[\bar{0},y_0)$ from among the search space of satisfying assignments; in the second case we can drop $[y_i,y_j)$. Either way, we now have only at most $(1 - \frac{1}{q(n)+1}) < e^{-\frac{1}{q(n)+1}}$-times as many candidate assignments to consider as before.

   Iterating this $n(q(n) + 1)$-times reduces the search space from 2^n possible assignments to naught. So if there was a satisfying assignment for ϕ, we must have found it along the way.

2.4.1 P-selective problems

A language L is P-selective if there exists a selector function $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ for L such that

i. f is polynomial-time computable

ii. $f(x, y) \in \{x, y\}$ for all x,y

iii. whenever $\{x, y\} \cap L \neq \emptyset$ then $f(x, y) \in L$

In other words, when given two candidates, one can decide in polynomial time via f, which of the two is more likely to be in L. Obviously, P $\subseteq$ P-sel $\subseteq$ NP.

Example for $r \in (0,1)$ an arbitrary real number the set $L_r = \{bin(q) \mid q \in (r,1) \cap \mathbb{Q}\}$ is P-selective: via the trivial selector $f(x,y) = \max(x,y)$, even if it is not computable (e.g. if the n-th binary digit of r is one precisely if the n-th turing machine halts on empty input).

1. Prove that if there exists an NP-complete P-selective set, then P=NP.

   Here is a thought: assuming SAT can be reduced to a P-selective set L with selector function f via a PTime reduction g we could use this to solve SAT in PTime as follows. Given an instance ϕ pick a variable X occurring in ϕ and compute $f(g(\phi[X = 0]),g(\phi[X = 1]))$ to test which of the two possible assignments for X is more likely to yield a satisfiable reduct of ϕ. This can be computed in polynomial time. By assumption f makes sound judgments and the reduction g is also sound. Hence if one of $\phi[X = 0]$ or $\phi[X = 1]$ is satisfiable then we'll know which one is using the above. We continue with that formula, discarding the other, iterating this until all variables have been eliminated we find our answer to the satisfiability of ϕ.

2. Prove that every P-selective set is in P/poly.

   Hint: To every P-selective set L and every n we can associate a tournament on vertices $L_n = \{x \in L \mid |x| = n\}$ by interpreting $f(\min\{x,y\},\max\{x,y\})$ as the winner of the match between x and y. (Note that now both x and y are in L_n.) As advice we can then take a superloser set of this tournament.

   Remark: this argument actually shows that P-sel $\subseteq$ P/quadratic.

   Recall that a k-tournament is a directed graph on k vertices with no self loops and precisely one edge between any two vertices. Intuitively, such a graph represents the outcomes of a complete tournament among k teams with an edge from v to w representing the fact that w beats v.

   Fact in every k-tournament there exists a set H of vertices such that for every $w \notin H$ there is a $v \in H$ so that w beats v, moreover, $|H| \leqslant \lfloor \log(k + 1) \rfloor$. Such an H is called a superloser set.

   Hint: proved by induction on k: there exists a vertex x who has lost at least $\lceil k - 1/2 \rceil$ matches, put it in H and proceed inductively with all those vertices who lost against x.

   [Another interesting fact: in every tournament there exists a king, i.e. a vertex w such that for every v either w beats v or w beats some z who beats v. Moreover, a tournament with no dominating vertex has at least three kings. Proof using a "median order" and its "feedback property"... median orders also yield Hamilton paths...]

Chapter 3

Basic complexity classes

Contributed: Vince Bárány, Eryk Kopczyński, Henryk Michalewski, Damian Niwiński

3.1 Polynomial time

Resolution is, of course, the well-known method for verifying unsatisfiability of CNF instances based on the following rule, also called resolution: if C and C' are clauses with a unique pair of complementary literals, say, $X \in C$ and $\neg X \in C'$ then their resolvent is the clause $C^* = C \cup C' \setminus \{X, \neg X\}$.

Crucially, we have $C, C' \models C^*$. Hence if we keep adding resolvents to our set of clauses then this does not add or exclude any models. Clearly, if the empty clause can be derived this way, than the original set of clauses was unsatisfiable (soundness). But the converse also holds: if the empty clause cannot be derived then the set of clauses is satisfiable (completeness).

Note that modus ponens is a special case of the resolution rule with $C = X$ and $C' = \neg X \vee Y \ \equiv \ X \to Y$ and $C^* = Y$.

While resolution takes in exponential time in general to complete, in each of the following special cases of CNF-SAT, plain old resolution or a slight adaptation thereof is actually guaranteed to terminate in PTIME.

1. (Horn Clauses) Let X be a finite set of variables and ϕ be a CNF Boolean expression with variables from the set X such that each clause contains only one positive literal, that is all literals, except possibly for one, are negations of variables, e. g. $\phi = (\neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Prove, that the SAT problem for these Boolean expressions is solvable in **P**.

   Hint. Consider a Boolean expression ϕ such that each clause contains only one positive literal and notice, that clauses with at least one non–negative variable one can re-write as implications. In order to find a truth assignment for ϕ start from an empty set T (all assignments are **false**) and for every implication add one variable to T.

2. Let X be a finite set of variables and ϕ be a CNF Boolean expression with variables from the set X such that each clause involves two variables, e. g. $\phi = (x_1 \vee x_2) \wedge (x_3 \vee \neg x_1)$. Let G be a graph with the set of vertices $\{x,\neg x : x \in X\}$ and an edge from x_1 to x_2 if there is a clause in ϕ of the form $\neg x_1 \wedge x_2$ or $x_2 \wedge \neg x_1$. Prove that ϕ is unsatisfiable if and only if there are paths from x to $\neg x$ and from $\neg x$ to x.

3. Prove that 2SAT problem, that is the problem whether a Boolean expression in the form described in the previous exercise is satisfiable, is in **P**.

   Hint. (Krom's method, 1967) Use the previous exercise in order to construct an algorithm in $O(n^4)$; alternatively, use the previous exercise in order to construct an algorithm in **NL** and use the fact that **NL** $\subset$ **P**.

   Remark. As it is noticed in the next exercise, one can solve the problem in $O(n)$ time.

Note: in this case resolution will only ever produce new clauses with at most two literals, whence it is bound to terminate in $O(n^2)$ steps where n is the number of variables.

4. (The Timetable Problem) Let H be a finite set of hours, $T_1,\ldots,T_n \subset H$ be the set of hours during which ith teacher is avaliable for teaching and $C_1,\ldots,C_m \subset H$ be the set of hours during which jth class is avaliable for studying and $R_{ij} \in \mathbb{N}$ $(1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m)$ be the number of hours ith teacher is required to teach jth class. The TT problem is to determine whether there exists a *meeting function* that is a mapping
$$f : \{1,\ldots,n\} \times \{1,\ldots,m\} \times H \to \{0,1\}$$
such that

   (a) $\forall_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}\ f(i,j,h) = 1 \to h \in C_i \cap T_j$,
   (b) $\forall_{1\leqslant i\leqslant n, 1\leqslant j\leqslant m}\ \sum_{h\in H} f(i,j,h) = R_{ij}$,
   (c) $\forall_{1\leqslant i\leqslant n, h\in H} \sum_{i=1}^{n} f(i,j,h) \leqslant 1$,
   (d) $\forall_{1\leqslant j\leqslant m, h\in H} \sum_{j=1}^{m} f(i,j,h) \leqslant 1$.

   Assume, that for every $1 \leqslant i \leqslant n$ we have $|T_i| = 2$ and show that this 2–TT problem is solvable in P.

   Remark. This fact was noticed by S. Even, A. Itai and A. Shamir (1976) alongside a similar observation that 2SAT problem is solvable in $O(n)$ time.

5. Prove that the (CNF) SAT problem, where each variable occurs at most twice is in **P**.

   A variant of resolution offers an easy solution. If a variable occurs only once, or twice but with both occurrences negated or both positive, then the clause(s) containing its occurrence(s) can be safely discarded.

   Now every remaining variable has precisely one positive and one negative occurrence. Choosing any variable X we can form the resolvent of the clauses it occurs in and, crucially, we can discard the two original clauses. (Because any satisfying assignment to the resolvent satisfies at least one of the original clauses and the other can be satisfied by the appropriate choice of X without affecting any of the other clauses, in which X is known not to occur.) Note that we can also discard the resolvent if it contains a pair of complementary literals. This process decreases the number of clauses in each step by at least one.

   Remark: this process is the inverse of what happens in the reduction of CNF to 3CNF.

6. Does the previous claim continues to hold if we replace twice by three times ?

   Presumably no (unless P=NP). Indeed, suppose x occurs k times, $k > 3$. Replace the i th occurrence of x by a fresh variable x_i, and add the cluases $\ldots \wedge (x_1 \to x_2) \wedge (x_2 \to x_3) \wedge \ldots \wedge (x_k \to x_1)$, forcing that the variables x_i get the same value.

7. Consider the satisfiability problem with the assumption that every variable occurs at most twice (like in point 5 above), but allowing arbitrary formulas, not necessarily CNF. Then the problem is NP-complete.

   Hint: Reduce the (unrestricted) CNF-SAT. For any variable x, which occurs positively and negatively in a CNF formula φ, replace its positive occurrences by distinct (fresh) variables $x_1,\ldots,x_k$. Similarly, replace the negative occurrences of x by variables $x'_1,\ldots,x'_\ell$. Add (in conjunction) a formula with single occurrences of variables $x_1,\ldots,x_k,x'_1,\ldots,x'_\ell$, forcing the claim.

   (Solution: $\neg\,((x_1 \vee \ldots \vee x_k)\ \wedge\ (x'_1 \vee \ldots \vee x'_\ell))$.)

8. (After Cook-Levin Theorem) Prove that HORN-SAT is complete for PTIME.

   Hint: We have already seen that HORN-SAT is in PTIME. To prove completeness notice that the formalisation of TM acceptance used in the Cook-Levin theorem is essentially Horn, except for the fact, that non-deterministic transitions are modeled by disjunctions of Horn rules/clauses. For deterministic TM, however, this problem disappears, the formalisation is automatically in Horn-CNF form.

9. Prove that PTIME has no complete problems with respect to linear-time reductions.

   Hint: use the time hierarchy theorem.

10. **NEW** Show that PTIME, NP, and PSPACE are closed under Kleene-$\star$.

3.2 Logarithmic space

1. Show that the class of functions computable in logarithmic space, FL, is closed under composition.

2. Show that the set of well-formed bracket expressions is recognizable in L (logarithmic space).

   Follow-up question: How about if we have two kinds of parantheses, e.g. $(-)$ and $[-]$?

   (Can still be done in L: check well-bracketedness with no regard to parenthesis types at first, then check that all pairs of matching parentheses (found again by counting) have the same type.)

3. Show that the evaluation of Boolean formulas (not circuits!) without variables can be done in L.

   Remarks. You may assume that the parentheses are obligatory. You may first consider the case where negation is applied only to variables.

4. Suppose we have to compute a term τ formed from constants and binary operations. We may use only instructions of the form $x := c$, and $x := f(y,z)$, where x,y,x are variables, c is constant and f a binary operation. How many variables we need w.r.t. the size of τ ?

   Hint. Think of the number of pebbles we need in a pebble game on a binary tree.

5. Let A be a finite algebra over a set of binary operations; we view all elements of A as constants. Considering A as fixed, show that evaluation of any term over A can be done in L. (This generalizes Exercise 2.)

   Hint. Show first that it can be done in DSPACE $(\log^2 n)$, using Exercise 4. Reduction to L is quite tricky.

6. Suppose there is an algorithm which, for a number n (in unary) produces a circuit C_n of polynomial size and logarithmic depth, with the fan-in of Or-gates and And-gates at most 2. Show that the language M recognized by $(C_n)_{n\in\mathbb{N}}$ is in the class L.

   Hint. Note that you cannot store the circuit C_n in the memory of your machine. However, you can call the algorithm generating C_n again and again, whenever needed. Therefore, you can evaluate the circuit keeping in the memory only an identifier of the actual gate (of size $\log n$) and the path from the output gate to the actual gate (as a binary string of size $\log n$ as well).

7. [3] A k-headed finite automaton (k-FA) is a one-tape TM with k two-way heads on its read-only input tape.

   Give a formal definition of these machines, including a definition of acceptance.

   Show that a set is in LOGSPACE (NLOGSPACE) iff it is accepted by a deterministic (nondeterministic) k-FA for some k.

   Hint: the "if" direction is almost obvious, for the other direction one could try to encode the $k \cdot \log n$ bits on the worktape of the TM M to be simulated using the positions of, say, $k+3$ or so heads of a FA (head positions to be used as counters, with atomic operations: increment, decrement, double, half, and parity test. These can be implemented with the help of two or so additional counters (i.e. heads). Trick: to encode the head position l of M on its work tape within the i-th block of $\log n$ bits best use a counter holding the value 2^l while maintaining the block number i in the state. This way the movement of this head can also be simulated with the earlier atomic operations on counters.)

8. Show that Polylogspace $= \bigcup_k \mathrm{DSpace}(log^k n)$ has no complete problems with respect to logspace reductions. (So it isn't a very reasonable class in this sense.)

   Hint: If some L is complete for Polylogspace under logspace reductions and is in $\mathrm{DSpace}(log^k n)$ then that means that Polylogspace collapses to $\mathrm{DSpace}(log^k n)$ contradicting the space hierarchy theorem.

9. [3] Show that 2SAT is complete for co-NL under logspace many-one reductions.

   Hint: For membership, we need to show that 2CNF unsatisfiability is in NL. This is essentially achieved by Krom's solution used in Exercise 3, where we only claimed it to be polynomial.

   Recall that the claim there was that φ is unsatisfiable iff in the associated graph there is a cycle complementary literals X and $\neg X$. This can be tested in NL by guessing the literals and checking reachability of each from the other.

   For hardness we reverse Krom's solution to reduce the NL-complete s-t connectivity problem to 2CNF unsatisfiability. Given (V, E, s, t) consider the 2CNF formula

   $$\varphi = s \wedge \bigwedge_{(u,v)\in E} (u \to v) \wedge \neg t$$

   over V as a set of propositional variables. If there is a path from s to t in G then clearly φ is unsatsfiable. Conversely, if there is no path from s to t, then the assignment mapping each v to 1 iff it is reachable from s and to 0 otherwise, will obviously satisfy φ.

10. [3] A "rectilinear maze" is a finite undirected graph whose vertex set is $\{0, \ldots, N-1\}^2$ and whose edges are all of the form $\{(x, y), (x, y+1)\}$ or $\{(x, y), (x+1, y)\}$. The entry of the maze is (0,0) and the exit is $(N-1, N-1)$. Give an explicit algorithm determining whether in any given rectilinear maze there is an (undirected) path from the entry to the exit is solvable in deterministic logspace.

    Hint: employ the "right hand on the wall" walk heuristic...

11. [3] Show that the non-regular language $\mathrm{BIN} = \{\#b_k(0)\#b_k(1)\#b_k(2)\#\cdots\#b_k(2^k-1)\# \mid k \geqslant 0\}$, where $b_k(i)$ denotes the k-bit binary representation of $i < 2^k$, is recognisable in determinisitc $O(\log\log n)$ space.

    Note that not only accepting computations must pertain to this space bound but also rejecting ones. To that end we successively check membership of the input in the approximating languages $B_j = \{\#w_0\#w_1\#\cdots\#w_{m2^j-1}\# \mid m,j \geqslant 0,\ \bigwedge_{i<2^j} w_i \equiv b_j(i) \pmod{2^j}\}$, i.e., just making sure that the j least significant bits of successive bitstrings separated by #'s follow a cyclic lexicographically incremental pattern starting from all 0's and ending in all 1's. In each such round only $\log j$ bits of storage are needed in order to measure the same distance $< j$ within consecutive segments between successive #'s. Only if the last such round was successful (in which case the input has length at least $j2^j$) do we start the next round (growing the space used by one bit). If a round was unsuccessful then we reject. Even in this case we have not used more than $1 + \log\log n$ bits of storage. Finally, if after some round the space used reaches $\log k$, i.e., if it is large enough to count from the first # to the second # without overflowing, then we do a final round of verifying (using a $\log k$ bit counter) that the #'s are equally spaced.

3.3 Alternation

1. Define alternating finite automata over finite words in one-way, and two-way versions.

   (a) Show that the non-emptiness problem for these automata is decidable.

   (b) Show that these automata recognize only regular languages.

2. Is it true that a single-tape alternating Turing machines working in linear time can recognize only regular languages ?

   Remark. I suppose not, but don't see the proof at the moment. My guess the set of words like $w_1\$w_2\$w_3\$\ldots\$w_{2^n}$, where the w_i's are the words of length n in lexicographic order, should do.

3. Let A be a nonempty finite set, $\cdot$ a binary function on A and G a subset of A. The closure $\langle G\rangle$ of S in $(A,\cdot)$ is the smallest $H \subseteq A$ such that $G \subseteq H$ and H is closed under $\cdot$, i.e., $a \cdot b \in H$ for all $a,b \in H$.

   The problem GEN is to determine whether for given A, $\cdot$, G and $a \in A$ it is the case that $a \in \langle G\rangle$.

   Design an ALogSpace algorithm for GEN. (It is easy to see that GEN is in PTime.)

   Show that GEN is, in fact, P-complete.

   Hint: Prove that HORN-3SAT (is as difficult as HORN-SAT and) is logspace reducable to GEN.

   Remark: Restricted to associative operations $\cdot$ the above problem can be solved in NL. (Show this!)

4. (ALogTime) In order to make the complexity class ALogTime meaningful we must change the model somewhat. (Normally, an ATM running in logtime could only access the first logarithmic number of bits of the input and not the rest.) In a kind-of random access model, our TM will have a special write-only address tape on which to write the position of the bit of the (read-only) input to be retrieved. Thus, in logarithmic time, the TM can write the address of an input bit on the address tape and enter a special "Dorothy" state upon which in the next time step the head on the input tape is thwarted to that position and the machine finds itself in the designated "Kansas" state.

   For such machines it makes sense to discuss logarithmic time bounded complexity classes.

   To make the model more precise the initial configuration should have not only the input tape properly initialised, but also the work tape initialised with a binary representation of the length of the input. (Make the model more precise as needed...)

   By way of example give an ALogTime algorithm for the language of palindromes, PAL.

   Prove that ALogTime = NC^1 [for an appropriate notion of uniformity, stronger than logspace uniformity!!]

3.4 Numbers

1. Given n (in binary), find in polynomial time the smallest a, such that $n = a^k$, for some k.

   Remark. This shows that, in the factorization problem, we can restrict ourselves to the case when $n = p \cdot q$, where $p \perp q$.

2. Show that the following language is in NP:

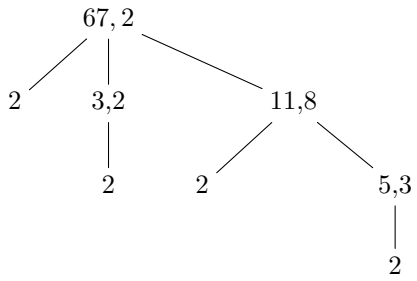
$$\{(p,q) : p \text{ is prime and } q \text{ generates } \mathbb{Z}_p^*\}.$$

   Hint. Use Pratt's certificates of primality:

$$C(p,q) = \begin{cases} 2 & \text{if } p = 2 \wedge q = 1 \\ p_1, \ldots, p_k,\ \alpha_1, \ldots, \alpha_k, & \\ C(p_1,q_1), \ldots, C(p_k,q_k) & \text{otherwise} \end{cases}$$

   where moreover $p - 1 = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$ and $q^{\frac{p-1}{p_i}} \neq 1 \bmod p$, for $i = 1, \ldots, k$.

For example, the following certifies (67,2)



3.5 PSPACE

1. (taken from Lecture 8 of [3])

   In the game of geography, two players alternate thinking of names of countries. The first player may pick any country. Thereafter, each player must think of a country whose name begins with the same letter that the previously named country ends with; for example: Albania, Azerbaijan, Norway, Yemen, Nicaragua, and so on. A country may not be named more than once.

   Consider the following generalisation of this game. An instance of the game is a tuple (V,E,v_0), where (V,E) is a directed graph and $v_0 \in V$. We start with a token on v_0. In the even stages, Player I moves the token from v_{2i} to some adjacent vertex v_{2i+1}, and in the odd stages, Player II moves the token from v_{2i+1} to some adjacent v_{2i+2}. No player may move to a vertex that has already been visited. A player wins by forcing the opponent to a position from which there is no legal next move.

   The decision problem GEOGRAPHY asks: given an instance (V,E,v_0), does Player I have a forced win?

   Problem: Show that this problem is PSPACE-complete.

   Hint: membership in PSPACE is witnessed by the alternating PTIME algorithm described by the imperative: "Play the game!" (technically one must mark the visited vertices thus using $O(n)$ space)

   For PSPACE-completeness devise a reduction from QBF (wlog. in normal form $\exists X_1 \forall X_2 \exists X_3 \forall X_4 \ldots \varphi$ with φ in CNF).

   Follow-up: How does the complexity change if we allow vertices to be revisited?

   Hint: In this case the problem is ALOGSPACE = PTIME -complete. The difference is that we don't need to remember the visited positions, hence only logarithmic space is needed to encode the current position. Playing the game yields an ALOGSPACE algorithm. For PTIME-completeness reduce from the circuit value problem CVP (wlog. on circuits in negation normal form and with alternating AND and OR gates).

2. NEW. Show that the following problem is PSPACE-complete: decide if a given a non-deterministic finite automaton accepts all words.

   Hint: Let a Turing machine M working in space $p(n)$ be fixed. For a word w, construct an automaton accepting those words that are not accepting computations of M on w. The automaton just detects some inaccuracy in a given word (it can count to $p(n)$). Note that in this way we reduce the complement of $L(M)$, but PSPACE is closed under complement.

   Note: For deterministic finite-state automata, an analogous problem (universality) is in P, whereas it is undecidable for non-deterministic pushdown automata.

   Challenge: For deterministic pushdown automata, the universality problem is decidable, but what is the complexity ?

3. NEW. Show that the following problem is PSPACE-complete: given n deterministic finite automata A_i, decide whether $\bigcap_{i=1} L(A_i) \neq \emptyset$ (note that the number of automata is not fixed).

   Hint: Similar to Exercise 2, but this time the automata make sure that a given word is a successful computation of M on w. The number of automata should roughly correspond to $p(n)$, the i-th automaton takes care of the i-th symbol of configuration.

3.6 NP

1. Consider nondeterministic Turing machines that can only move their head to the right, or jump to the beginning of the tape. Are they equivalent to the standard ones?

2. [3] Show that if P=NP then NEXPTIME = EXPTIME.

   Consider some L in NEXPTIME accepted by some 2^{n^c}-time-bounded nondeterministic Turing machine M. Let $L' = \{x\#^k \mid x \in L, k = 2^{|x|^c}\}$. Then L' is in NP, because on some input $x\#^k$ we can verify by counting that the padding is of the correct length, while erasing it, and then run M on x. By the assumption P = NP, so L' is in P. As such it is accepted by deterministic polynomial-time TM N. Then L is in EXPTIME, because on input x we can pad it as in L' and run N all in exponential time.

3. Show that TILING is NP-compelete. Here TILING is the problem to decide, given a set of tiles (the usual square kind), and an initial and terminal row of n tiles each, whether the $n \times n$ board can be tiled with the given inital row on $\{1\} \times [n]$ and the terminal row on $\{n\} \times [n]$. (You can consider other variantions of input constraints.)

4. Show that the following problem is NP-complete: given a family $\mathcal{F}$ of subsets of $[n] = \{1,\ldots,n\}$ is there a subfamily $\mathcal{P} \subseteq \mathcal{F}$ partitioning $[n]$?

5. Let SUBSET-SUM be the problem of determining, given a tuple of naturals $(a_1,\ldots,a_k;s)$ as binary numerals, whether there is a subset $I \subseteq \{1,\ldots,k\}$ such that $s = \Sigma_{i\in I} a_i$. Further let UNARY-SUBSET-SUM be a variant of the above, where all numbers are encoded unary. Prove that SUBSET-SUM is NP-complete and that UNARY-SUBSET-SUM is in PTIME.

6. NEW. Show that the 3-coloring problem remains NP-complete even if restricted to planar graphs.

   Hint: Use an appropriate gadget. (You may find it in svn catalogue in zadanie5.pdf.)

7. NEW. Give an example of a polynomial relation R such that both sets $\exists R = \{x : (\exists y)\, R(x,y)\}$ and $R\exists = \{y : (\exists x)\, R(x,y)\}$ are NP-complete.

3.7 BPP

1. NEW. Show that the class BPP is closed under Kleene-$\star$ (c.f. Exercise 3.1.10).

   Hint: Use a lemma from the lecture saying that if $\Pr(x \in L \not\Leftrightarrow R(x,Y)) \leqslant \frac{1}{4}$ then

   $$\Pr(|\{i : x \in L \Leftrightarrow R(x,Y_i)\}| \leqslant m) \leqslant \left(\frac{3}{4}\right)^m$$

   where $Y_1,\ldots,Y_{2m+1}$ are independent copies of Y.

3.8 Diverse

1. Let $\mathcal{C}$ be a class of circuits. Say that a language L has a proof system in $\mathcal{C}$ if there is a family of circuits $C_n : \{0,1\}^{p(n)} \to \{0,1\}^n$ for each n, with n output bits and $p(n)$ many input bits, where p is a polynomial, such that the range of C_n is precisely the set of length n words in L. In other words,

$$x \in L \iff \exists y,\, |y| = p(|x|) :\ C_{|x|}(y) = x$$

and in such case we say that y is a proof of $x \in L$.

For instance, PARITY (even number of 1's) has NC^0 proofs: for each $x = x_1 x_2 \dots x_n$ let $y = y_0 y_1 y_2 \dots y_{n+1}$ be a proof of $x \in$ PARITY, where $y_0 = y_{n+1} = 0$ and $x_i = y_{i+1} - y_i$ for each i.

   i. Show that every problem in NP has an AC^0 proof system.

   Hint: if M is a nondeterminisitc ptime TM accepting some L then for every x take as proofs the accepting computations (sequences of configurations) of M on x. Each such sequence is a polynomial size proof and can be verified by an AC^0 circuit.

   ii. Show that there is some L in NP that has no P-uniform NC^0 proof system.

   Hint: note that if L has an NC^0 proof system (unifrom or not) then all proofs are of linear size. So if L has an P-uniform NC^0 proof system then L is non-deterministic-linear-time recognisable: given x, guess a proof y of size $O(|x|)$, generate the circuit $C_{|x|}$ and verify that on input y it outputs x. Conclude by the (non-det.) time hierarchy theorem.

- NEW. (A.Niewiarowska) Cat and Mouse occupy some nodes in a (symmetric) graph. They start to play moving in alternation to a neighbour position. There is a distinguished position Hole; Mouse wins if she reaches this position. Cat wins if he reaches a position currently occupied by Mouse. Otherwise, there is a draw. Determine the complexity of solving this game.

  Hint: Should be P (?).

Bibliography

[1] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach, Cambridge University Press, 2009. http://www.cs.princeton.edu/theory/complexity/

[2] John E. Hopcroft and Jeffrey D. Ullman. Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[3] Dexter C. Kozen. Theory of Computation, Text in Computer Science, Springer, 2006.

[4] Christos Papadimitriou. Computational complexity, Addison-Wesley, 1993.