数理解析研究所講究録 1361 巻 2004 年 8-18

Resource Bounded Unprovability of Computational Lower Bounds (Part 1) (Extended Abstract)

岡本 龍明 * 鹿島 亮 **
Tatsuaki Okamoto* Ryo Kashima**

* 日本電信電話 (株)
NTT Laboratories, Nippon Telegraph and Telephone Corporation
1-1 Hikarino-oka, Yokosuka-shi, Kanagawa, 239-0847 Japan
** 東京工業大学
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
1-12-1 Ookayama Meguro-ku, Tokyo, 152-8552 Japan

Abstract. This paper shows that the *proof complexity* (minimum computational complexity of proving formally or asymptotically) of "P≠NP" is *super-polynomial-time* with respect to a theory T, which is a consistent extension of Peano Arithmetic (PA), and PTM-ω-consistent, where the PTM-ω-consistency is a polynomial-time Turing machine (PTM) version of ω-consistency. In other words, *to prove* "P≠NP" (by any way) *requires super-polynomial-time computational power* over T. This result is a kind of generalization of the result of "Natural Proofs" by Razborov and Rudich [20], who showed that to prove "P≠NP" by a class of techniques called "Natural" implies computational power that can break a typical cryptographic primitive, a pseudo-random generator. This result implies that P≠NP is formally unproven in T with PTM-ω-consistency. We also show that *to prove the independence* of P vs NP from T by proving the PTM-ω-consistency of T *requires super-polynomial-time computational power.* This seems to be related to the results of Ben-David and Halevi [4] and Kurz, O'Donnell and Royer [16], who showed that to prove the independence of P vs NP from PA using any currently known mathematical paradigm implies an extremely-close-to-polynomial time algorithm that can solve NP-complete problems. Based on this result, we show that *the security of any computational cryptographic scheme is unprovable* in the standard setting of modern cryptography, where an adversary is modeled as a polynomial-time Turing machine.

Key Words: computational complexity, computational lower bound, P vs NP, natural proofs, cryptography, unprovability, proof theory, incompleteness theorem

1 Background

It looks very mysterious that proving computational lower bounds is extremely difficult, although many people believe that there exist various natural intractable problems that have no efficient algorithms that can solve them. A classical technique,

diagonalization, can separate some computational classes like P $\neq$ EXP, but it fails to separate computational classes between P and PSPACE, which includes almost all practically interesting computational problems. Actually we have very few results on the lower bounds of computational natural problems between P and PSPACE. The best known result of computational lower bounds (in standard computation models such as Turing machines and Boolean circuits) of a computational natural problem is about $5n$ in circuit complexity [15], where n is problem size. Therefore, surprisingly, it is still very hard for us to prove even the $6n$ lower bound of TQBF, a PSPACE complete problem, which is considered to be much more intractable than NP complete problems.

Considering this situation, it seems natural to think that there is some substantial reason why proving computational lower bounds is so difficult. An ultimate solution to this question would be to show that such computational lower bounds are impossible to prove, e.g., showing its independence from a formal proof system like Peano Arithmetic (a formal system for number theory) and ZFC (a formal system for set theory).

This paper gives a new type of impossibility result, *resource bounded* impossibility, in the proof of computational lower bounds.

2 Our Results

This paper presents:

1. Let theory T, on which we are assumed to try to prove P$\neq$NP, be an extension of Peano Arithmetic (PA) and consistent, throughout this paper (or hereafter in this section).
   We formalize P$\neq$NP in two ways: one is formalized by a sentence $\overline{\text{P}\neq\text{NP}}_T$ in PA, which depends on T, and the other is by a sentence $\overline{\text{P}\neq\text{NP}}$ in PA, which does not depend on any theory T.
   We introduce a concept of *proof complexity* that is the minimum computational complexity of (either formally or asymptotically) proving a statement.
2. $\overline{\text{P}\neq\text{NP}}_T$ cannot be *formally* proven in T.
   No polynomial-time Turing machine can *asymptotically* produce a proof of $\overline{\text{P}\neq\text{NP}}_T$ over T.
   As a result, the *proof complexity* of $\overline{\text{P}\neq\text{NP}}_T$ is *super-polynomial-time* with respect to T.
3. $\overline{\text{P}\neq\text{NP}}$ cannot be *formally* proven in T, under an additional assumption, PTM-ω-consistency of T for Δ_2^P.
   No polynomial-time Turing machine can *asymptotically* produce a proof of $\overline{\text{P}\neq\text{NP}}$ over T.

As a result, the *proof complexity* of $\overline{\text{P}\neq\text{NP}}$ is *super-polynomial-time* with respect to T, under an additional assumption, PTM-ω-consistency of T for Δ_2^P.

4. There exists a Δ_2^P-formula $\varphi(\mathbf{x})$ such that
   - PTM-ω-consistency of T for $\varphi(\mathbf{x})$ cannot be *formally* proven in T.
   - No polynomial-time Turing machine can *asymptotically* produce a proof of PTM-ω-consistency of T for $\varphi(\mathbf{x})$.
   - As a result, the *proof complexity* of PTM-ω-consistency of T for $\varphi(\mathbf{x})$ is *super-polynomial-time* with respect to T.
   - If T is PTM-ω-consistent for $\varphi(\mathbf{x})$, $\overline{\text{P}\neq\text{NP}}$ cannot be *formally* proven in T.

   Thus, the *proof complexity* of the independence of $\overline{\text{P}\neq\text{NP}}$ from T by proving PTM-ω-consistency of T for $\varphi(\mathbf{x})$ is *super-polynomial-time* with respect to T.
5. Under an additional assumption, PTM-ω-consistency of T for Δ_2^P, the one-wayness of any function family is (formally and asymptotically) *unprovable* over T, in the standard setting of the modern cryptography, where an adversary (and prover) is modeled to be a polynomial-time Turing machine.
   In other words, the security of any computational cryptographic scheme is (formally and asymptotically) unprovable.

3 Related Works

3.1 Self-defeating results

Our result is considered to be a kind of generalization of or a close relation to the previously known self-defeating results as follows:

- Our result that the proof complexity of $\overline{\text{P}\neq\text{NP}}$ and $\overline{\text{P}\neq\text{NP}}_T$ is *super-polynomial-time* with respect to T under an assumption of T implies a *self-defeating* property such that to prove a super-polynomial-time lower bound (P$\neq$NP) requires super-polynomial-time computational power (or implies a super-polynomial-time upper bound).
  "Natural Proofs" by Razborov and Rudich [20] showed that to prove a computational lower bound (e.g., P$\neq$NP) by a class of techniques called "Natural" implies a comparable level of computational upper bound (e.g., computational power sufficient to break a typical cryptographic primitive, a pseudo-random generator).
- Our results imply another *self-defeating* property such that the proof complexity of the independence of $\overline{\text{P}\neq\text{NP}}$ from T by proving PTM-ω-consistency of T for a Δ_2^P-formula is *super-polynomial-time* with respect to T. In other words, to prove the independence of "P$\neq$NP" from T through proving PTM-ω-consistency

(i.e., to prove $T \nvdash \overline{\text{P}\neq\text{NP}}$ by proving PTM-ω-consistency of T and to prove $T \nvdash \overline{\text{P}=\text{NP}}$ by some way) requires super-polynomial-time computational power (or implies a super-polynomial-time upper bound).
Ben-David and Halevi [4] and Kurz, O'Donnell and Royer [16] showed that to prove the independence of a computational lower bound, P≠NP, from PA using any currently known mathematical paradigm implies a comparable level of computational upper bound, an extremely-close-to-polynomial time algorithm to solve NP-complete problems.

3.2 Relativizable proofs

This paper shows that there is no formal proof for "P≠NP" in theory T under an assumption of T. This is considered to be a generalization of the result by Baker, Gill and Solovay [1], who showed that there is no relativizable proof for "P≠NP", and the result by Hartmanis and Hopcroft [12, 13], who showed that for any reasonable theory T we can effectively construct a TM M such that relative to oracle $L(M)$, "P≠NP" cannot be proven in T.

This result might be related to the result by da Costa and Doria [6], but the relationship between their result and ours is unclear for us.

3.3 Mathematical logic approaches

The results of this paper are constructed on the theory and techniques of mathematical logic, especially proof theory. Several mathematical logic approaches to solve the P versus NP problem have been investigated such as bounded arithmetic [5, 17], propositional proof length [3, 17, 19] and descriptive complexity [8].

Bounded arithmetic characterizes an analogous notion of PH (polynomial hierarchy of computational complexity), which is a hierarchy of weak arithmetic theories, so-called bounded arithmetic classes, wherein only bounded quantifiers are allowed. The target of the bounded arithmetic approach is to separate one class from another in bounded arithmetic, which may imply a separation of one class from another in PH (i.e., typically P≠NP). An idea to separating classes in bounded arithmetic is to employ an analogue of the second Gödel incompleteness theorem. That is, if a bounded arithmetic class can prove an analogue of the consistency of another class, then these classes can be separated by the incompleteness theorem.

The proof length of proposional logic can characterize the NP versus co-NP problem, since TAUT, the set of propositional tautologies, is co-NP complete. Therefore, the main target of this approach is to prove NP≠co-NP by showing a super-polynomial length lower bound of a formal propositional proof of TAUT. In this approach, the lower bounds of the proof lengths and limitation of provability of

some specific propositional proof systems (e.g., resolution, Frege system and extended Frege system) have been investigated.

The descriptive complexity characterizes NP by a class of problems definable by existential second order formulas and P by a class of problems definable in first order logic with an operator. The target of this approach is to separate P and NP using these logical characterizations.

This paper characterizes the concepts of P and P$\neq$NP etc., by formulas in Peano Arithmetic (PA). A novel viewpoint of our approach is to investigate the computational lower bound of a *prover* that produces a proof of a computational lower bound such as P$\neq$NP. To the best of our knowledge, no existing approach has studied computational lower bounds from such a viewpoint. [1] In our approach, an analogue (or resource bounded version) of Gödel incompleteness theorem plays a key role. Note that the bounded arithmetic approach also employs a (bounded arithmetic version of) incompleteness theorem, but its target is to show a computational lower bound (e.g., to prove P$\neq$NP), while the target of our approach, which employs a (resource bounded version of) incompleteness theorem, is to obtain a computational lower bound of a prover that proves a computational lower bound (e.g., to prove the resource bounded unprovability of P$\neq$NP).

3.4 Proof theory

This paper sheds light on a new concept of proof theory, *asymptotic proofs* and *polynomila-time proofs* where a computational complexity of (prover's) proving a set of statements asymptotically is bounded by a polynomial-time. In the conventional proof theory, the properties and capability of a proof system (e.g., consistency, completeness, incompleteness etc.) are of prime interest, but the required properties and capability of the prover are not considered (i.e., no explicit restriction nor condition is placed on the prover).

Note that the bounded arithmetic approach seems to follow this conventional paradigm and bounds the capability of the proof system (axioms and rule of inferences) to meet the capability of resource bounded computational classes. That is, the prover is still thought to exceed the scope of the approach.

Recently, an asymptotic and quantitative property of a proof, the length of a proof, has been studied [18], with motivated by the approaches of the proof length of proposional logic and of bounded arithmetic introduced in Section 3.3. The proof length is partially related to the computational complexity of a prover, since if a proof

[1] A prover is modeled as a Turing machine in the interactive proof system theory, and the required computational complexity of a prover has been investigated [11, 10]. However, no proof system that proves a computational lower bound and its prover's computational lower bound has been studied.

length is asymptotically much longer (e.g., a super-polynomial), the required computational complexity of a prover should be much greater (e.g., a super-polynomial-time). However, even if the proof length is bounded in short (e.g., a polynomial), it does not always imply that the required computational complexity of a prover should be bounded in a comparable amount (e.g., a polynomial-time). Actually, the P vs NP problem raises a related question, whether a short proof (witness) of a NP complete statement can be always efficiently produced or not. That is, the proof length is an important aspect of the *complexity* of proofs, but does not capture another important aspect of the *complexity* of proofs, the computational complexity of producing proofs.

Our formulation, an asymptotic (and polynomial-time) proof system, relaxes the concept of a conventional proof system. An asymptotic proof is a set of an infinite number of formal proofs, and a resource bounded (e.g., polynomial-time bounded or exponential-time bounded etc.) prover asymptotically produces an asymptotic proof of a set of infinitely many formal statements. We believe that our approach is more suitable for treating the computational lower bound problems than the conventional proof system.

It is worth noting that an asymptotic proof could be expressed as a *finite*-length *meta* proof, although an asymptotic proof consists of infinitely many formal proofs. Several successful examples of Natural Proofs [20] such as PARITY $\notin$ AC^0 and $NC^1 \neq AC^0$ [9, 23] are considered to be typical of this type of meta proofs. The self-defeating property of Natural Proofs implicitly implies the limitation of the conventional formal proof, since if there exists a (finite-length) formal proof of a super-AC^0 lower bound in a theory, then a finite-size AC^0 circuit with a large constant depth would be able to output such a formal proof, and no self-defeating property would occur. This fact means that several successful examples of Natural Proofs are considered to be (finite-length) meta proofs of asymptotic proofs.

4 Outline

We now show an outline of this paper.

Throughout this paper, we consider the impossibility/intractability of proving P$\neq$NP based on a formal proof system (i.e., theory) T, which is an extension of Peano Arithmetic (PA) and consistent. For the purpose, we need to formalize the statement of P$\neq$NP in T. This paper formalizes P$\neq$NP in two ways: one is formalized by a sentence $\overline{\text{P}\neq\text{NP}}_T$ in PA, which depends on T, and the other is by a sentence $\overline{\text{P}\neq\text{NP}}$ in PA, which does not depend on any theory T.

Therefore, the results in this paper are roughly divided into three parts: the first part icludes the results on $\overline{\mathrm{P}{\neq}\mathrm{NP}}_T$, the second part on $\overline{\mathrm{P}{\neq}\mathrm{NP}}$, and the last part on cryptography, which is an application of the former two parts.

In the first part (i.e., the results on $\overline{\mathrm{P}{\neq}\mathrm{NP}}_T$), the key idea is a polynomial-time proof version of incompleteness theorems. Informally speaking, this part considers a special sentence, $\rho_{e,T}$, (so-called Gödel sentence) like "this statement, $\rho_{e,T}$, cannot be proven by a polynomial-time Turing machine (PTM) e in theory T." If $\rho_{e,T}$ can be proven by PTM e in T, it contradicts the definition of $\rho_{e,T}$, assuming that T is consistent. It follows that $\rho_{e,T}$ cannot be proven by PTM e in T, although another PTM can prove it. Since computational complexity like polynomial-time is defined asymptotically, we consider a set of an infinite number of such sentences and show that the set of sentences, $\{\rho_{e,T}(\mathbf{x}) \mid x \in \mathbb{N}\}$, cannot be proven asymptotically by a PTM e in theory T. Based on this theorem, we show that, for any formula set $\{\psi(\mathbf{x}) \mid x \in \mathbb{N}\}$ (e.g., formula set on the satisfiability of 3CNF), for any PTM e, there exists another PTM e^* such that PTM e, on input $x \in \mathbb{N}$, cannot asymptotically prove that PTM e^* cannot prove $\psi(\mathbf{x})$. By using the Theorem, we can show that no PTM can prove $\overline{\mathrm{P}{\neq}\mathrm{NP}}_T$ asymptotically. We can also prove that $\overline{\mathrm{P}{\neq}\mathrm{NP}}_T$ cannot be proven formally in T by directly using the second Gödel incompleteness theorem.

In the second part (i.e., the results on $\overline{\mathrm{P}{\neq}\mathrm{NP}}$), we introduce a concept of polynomial-time *decision* systems. In a proof system, we usualy consider only one side, a proof of a true statement. In a decision system, however, we have to consider two sides, CA (correctly accept: accept of a true statement) and CR (correctly reject: reject of a false statement). CD (correctly decide) means CA or CR. The key idea in this part is a polynomial-time decision version of incompleteness theorems. Roughly speaking, this part considers a special sentence, $\rho_e^A(\mathbf{x})$, like "this statement, $\rho_e^A(\mathbf{x})$, cannot be correctly accepted by a polynomial-time Turing machine (PTM) e." If $\rho_e^A(\mathbf{x})$ can be correctly accepted by PTM e, it contradicts the definition of $\rho_e^A(\mathbf{x})$. It follows that $\rho_e^A(\mathbf{x})$ cannot be correctly accepted by PTM e. We also define another sentence, $\rho_e^R(\mathbf{x})$, which cannot be correctly rejected by PTM e. Based on these theorems, we show that, for any formula set $\{\psi(\mathbf{x}) \mid x \in \mathbb{N}\}$ (e.g., formula set on the satisfiability of 3CNF), for any PTM e, there exists another PTM e^* such that PTM e, on input $x \in \mathbb{N}$, cannot asymptotically prove that PTM e^* cannot correctly decide $\psi(\mathbf{x})$. By using the Theorem, we show that no PTM can prove $\overline{\mathrm{P}{\neq}\mathrm{NP}}$ asymptotically. This paper then introduces PTM-ω-consistency of T, which is a PTM version of ω-consistency Combining the Theorem and PTM-ω-consistency of T, We can show that $\overline{\mathrm{P}{\neq}\mathrm{NP}}$ cannot be proven formally in T under the assumption of PTM-ω-consistency of T.

In addition, using the results of the first part, we show that the PTM-ω-consistency of T, which is used for proving the Theorem, cannot be proven formally in T and cannot be proven (asymptotically) in polynomial-time over T.

We then introduce a concept of *proof complexity*, which is the required computational complexity to produce a formal proof or an asymptotic proof. Proof complexity of a statement characterizes the complexity/hardness of proving the statement. Based on the results of the first and second parts of this paper, we show that $\overline{\mathrm{P}\neq\mathrm{NP}}_T$, $\overline{\mathrm{P}\neq\mathrm{NP}}$ and a formalization of the PTM-ω-consistency of T for a formula, have *super-polynomial-time* proof complexity.

Finally, the unprovability of the security of the computational cryptogarphy in the standard setting of modern cryptography is presented.

5 Informal Observations

We consider that the complexity/hardness of proving a meta statement (by a meta proof) can be estimated by the proof complexity of an appropriate formal sentence of the meta statement.

The proof complexity of $\overline{\mathrm{P}\neq\mathrm{NP}}$ (and $\overline{\mathrm{P}\neq\mathrm{NP}}_T$) is super-polynomial-time with respect to T, under an assumption of T. This implies that the complexity/hardness of proving P$\neq$NP is estimated as super-polynomial-time, i.e., to prove P$\neq$NP (by a meta proof) requires a computational resource that is comparable to a super-polynomial-time computational power, or no machine whose power is comparable to those of polynomial-time Turing machines can produce a proof of P$\neq$NP.

Therefore, if the computational capability of our human being (along with our available/feasible computing facilities) is modeled as a polynomial-time Turing machine, which is widely accepted as a feasible computation model, our result implies that no human being can produce a (meta) proof of P$\neq$NP.

Gödel's incompleteness theorem taught us the *principal* or *unconditional* limitation of our capability of proving mathematical problems. This paper may demonstrate a *computational* or *resource bounded* limitation of our capability of proving mathematical natural problems like P$\neq$NP and the security of cryptography. (Even if P$\neq$NP is true and there exists a meta proof of this statement, such a proof might be too long or too complicated for our human being to create.) Note that our result does not deny the possibility of proving P=NP by a resource bounded (constant-time or polynomial-time) Turing machine, if P=NP is true.

Gödel's (second) incompleteness theorem has a positive significance in that it helps us to separate two distinct theories, T and S, because $T \vdash \mathrm{Con}(S)$ implies that $T \neq S$ (and $T \supset S$) since $S \not\vdash \mathrm{Con}(S)$ by Gödel's (second) incompleteness theorem. Here $\mathrm{Con}(S)$ denotes the consistency of S. (As already mentioned before,

a bounded arithmetic approach tries to use this technique to separate one class from another.)

Using this idea and our results in Parts 1 and 2 of this paper provides some hint of the computational capability of our human being. Let X be a machine whose computational capability is unknown. If C is a computational class, our result helps us to characterize the computational power of X relative to C, because $X \vdash_T \text{SuperLowerBound}(C)$ implies that the computational power of X should be beyond C, since the proof complexity of SuperLowerBound(C) is considered to have a super-C computational lower bound, due to our (resource bounded version of the second) incompleteness theorem etc.. Here SuperLowerBound(C) denotes a formula to represent the super-C computational lower bound. If we assume X to be a computational model of our human being, then our obtained computational lower bound result of $X \vdash_T \text{SuperLowerBound}(C)$ implies the upper bound of our computational power. For example, we have already obtained a proof of a super-AC^0 lower bound. This fact means that the computational power of our human being should be beyond AC^0.

This result may also give us some hint as to why all known results of computational lower bounds inside PSPACE are limited to very weak or restricted computational classes. If the computational capability of our human being is considered to be much beyond the target computational class for lower bound proof (e.g., the target class is AC^0), then it is likely that we may produce a (meta) proof of the lower bound statement. However, if our computational capability is comparable to (or is not much beyond) the target computational class for lower bound proof, then it will be very unlikely that we can provide its (meta) proof. In other words, the best result of computational lower bounds suggests the computational capability of our human being.

If we have only *very low level* (compared with P) computational lower bound results for many years in the future, it will imply that our computational capability might be much lower than P. Actually class P clearly includes many infeasible computation classes for us such as n^{10000} computational complexity in input size n. Thus, the known lower bound results so far might give us some hint on constructing a better feasible computation model in the future.

On the other hand, if we succeed in proving P$\neq$NP in the future, it may imply that the computational power of our human being should be beyond P, and we will have an implication that our feasible computation model with P may be wrong. However, even if we change our feasible computation model from P to class Q, we will still face the same phenomenon in that we will not able to prove the super-Q computational lower bound if model Q characterizes our computational

power correctly. Therefore, to prove a super-feasible-computation-class lower bound is considered to be essentially impossible for us to prove.

6 Concluding Remarks

This paper introduced a new direction of research in order to study computational complexity lower bounds; (computational) resource bounded (un)provability including (computational) proof complexity resource bounded (un)decidability and resource bounded models. This approach can be generalized to various systems by generalizing verification machines, $\mathrm{U}_{\mathrm{PTM}}(v_T, \cdot)$ in proof systems and $\mathrm{U}(v, \cdot)$ in decision systems.

In Part 2, we will extend these results to other computational classes and show that: for all $i \geq 1$, to prove a super-Σ_i^P lower bound and super-Π_i^P lower bound requires super-Σ_i^P computational power and super-Π_i^P computational power, respectively. For all $i \geq 1$, to prove a super-AC^{i-1} lower bound and super-NC^i lower bound requires super-AC^{i-1} computational power and super-NC^i computational power, respectively. In addition, Part 2 will present similar results on probabilistic and quantum computational classes, since a probabilistic TM and quantum TM can be simulated by a classical deterministic TM; they can be formulated in PA in a manner similar to that in Part 1. Thus, for example, we will show that to prove a super-BPP lower bound requires super-BPP computational power and to prove a super-BQP lower bound requires super-BQP computational power.

References

1. T.P. Baker, J. Gill and R. Solovay, Relativizations of the P=?NP Questions, SIAM J.Comput., Vol.4, No.4, pp.431-442, 1975.
2. J. Barwise, Mathematical Logic, (especially, Section D.1 "The Incompleteness Theorems," by C. Smorynski), North Holland, 1977.
3. P. Beame and T. Pitassi, Propositional Proof Complexity: Past, Present and Future, Tech. Rep. TR98-067, ECCC, 1998.
4. S. Ben-David and S. Halevi, On the Independence of P versus NP, Technion, TR 714, 1992.
5. S. Buss, Bounded Arithmetic, Bibliopolis, Napoli, 1986.
6. N.C.A. da Costa and F.A. Doria, Consequence of an Exotic Definition, Applied Mathematics and Computation, 145, pp.655-665, 2003.
7. H.B. Enderton, A Mathematical Introduction to Logic, Academic Press, 2001.
8. R. Fagin, Generalized First Order Spectra and Polynomial-time Recognizable Sets, Complexity of Computation, ed. R. Karp, SIAM-AMS Proc. 7, pp.27–41, 1974.
9. M. Furst, J.B. Saxe and M. Sipser, Parity, Circuits and the Polynomial-time Hierarchy. Math. Syst. Theory, 17, pp.13-27, 1984.
10. O. Goldreich, Foundations of Cryptography, Vol.1, Cambridge University Press, 2001.
11. O. Goldreich, Modern Cryptography, Probabilistic Proofs and Pseudorandomness, Springer-Verlag, 1999.

12. J. Hartmanis and J. Hopcroft, Independence Results in Computer Science, SIGACT News, 8, 4, pp.13-24, 1976.
13. J. Hartmanis, Feasible Computations and Provable Complexity Problems, SIAM, 1978.
14. R. Impagliazzo and S. Rudich, Limits on the Provable Consequences of One-Way Permutations, Proc. of STOC'89, 1989.
15. K. Iwama and H. Morizumi, An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits, Proc. of MFCF, pp.353-364, 2002.
16. S. Kurz, M.J. O'Donnell and S. Royer, How to Prove Representation-Independent Independence Results, Information Processing Letters, 24, pp.5-10, 1987.
17. J. Krajicek, Bounded Arithmetic, Propositional Logic, and Complexity Theory, Cambridge University Press, 1995.
18. P. Pudlák, The Lengths of Proofs, Chapter VIII, Handbook of Proof Theory (S. Buss Ed.), pp.547-637, Elsevier, 1998.
19. A.A. Razborov, Resolution Lower Bounds for Perfect Matching Principles, Proc. of Computational Complexity, IEEE, pp. 29-38, 2002.
20. A.A. Razborov and S. Rudich, Natural Proofs, JCSS, Vol.55, No.1, pp.24–35, 1997.
21. J.R. Shoenfield, Mathematical Logic, Association for Symbolic Logic, 1967.
22. M. Sipser, Introduction to the Theory of Computation, PWS Publishing Company, 1997.
23. R. Smolensky, Algebraic Methods in the Theory of Lower Bounds of Boolean Circuit Complexity, Proc. of STOC'87, pp.77-82, 1987.