Many facets of complexity in logic

Antonina Kolokolova

Memorial University of Newfoundland
kol@cs.mun.ca

Abstract. There are many ways to define complexity in logic. In finite model theory, it is the complexity of describing properties, whereas in proof complexity it is the complexity of proving properties in a proof system. Here we consider several notions of complexity in logic, the connections among them, and their relationship with computational complexity. In particular, we show how the complexity of logics in the setting of finite model theory is used to obtain results in bounded arithmetic, stating which functions are provably total in certain weak systems of arithmetic. For example, the transitive closure function (testing reachability between two given points in a directed graph) is definable using only NL-concepts (where NL is non-deterministic log-space complexity class), and its totality is provable within NL-reasoning.

1 Introduction

How do we talk about the concept of hardness in the context of mathematical logic? Historically, there are several approaches using different notions of hardness, including the following:

1. The hardness of *describing* an object (by some formula)
2. The hardness of *proving* properties of an object in a formal system.
3. The hardness of *solving* a problem (e.g., when the problem is expressed in the language of logic.)

These notions of hardness loosely correspond to the following fields

1. Finite model theory (in particular, descriptive complexity).
2. Bounded arithmetic and proof complexity
3. Complexity theory and some areas of computational logic

Each of these fields has tight and well-studied connections to computational complexity theory. However, the direct relationships among different notions of hardness in logic are only now becoming a focus of attention. Here we will present, as main example of a such a direct relationship, results in bounded arithmetic we obtain using the notion of hardness from descriptive complexity.

We start with the canonical notion of hardness as defined in complexity theory.

2 The computational complexity setting

The computational complexity of a problem is measured in terms of resources used by an algorithm to solve the problem. The standard resources are space (memory) and time. Algorithms can be deterministic and non-deterministic. For example, the famous class NP consists of problems solvable by the non-deterministic polynomial-time algorithms. Similarly, the problems solvable by non-deterministic *log-space* algorithms comprise the class NL.

A complexity class that has very robust definitions in the context of logic is uniform $\mathtt{AC}^0$, the class of problems solvable by a uniform family of constant-depth polynomial-size boolean circuits. In bounded arithmetic, this class corresponds to the theory V^0 or, in the first-order setting, $I\Delta_0$ (where $I\Delta_0$ is Peano Arithmetic with induction restricted to bounded formulae). In the descriptive complexity setting $\mathtt{AC}^0$ corresponds to the first-order logic (data complexity of first-order logic is $\mathtt{AC}^0$ when the language contains arithmetic). We will discuss it in more detail later.

A classical example of a problem complete for NP is 3-colorability: given a graph as an input, determine if it can be colored with three colors so that no edge connects vertices of the same color. A problem complete for NL is graph reachability (determining if there is a path between two vertices). The class $\mathtt{AC}^0$ is one of the few for which there are non-trivial lower bounds: the Parity problem of determining if an input string has an even number of 1s is not in $\mathtt{AC}^0$. That is, there is no first-order formula which would be true on all and only structures with even number of elements (in fact, since this result holds in a non-uniform setting, adding to the languages any kind of numerical predicates, even undecidable ones, would still not help a first-order formula to express a parity of a string). Thus binary addition can be done in $\mathtt{AC}^0$, but multiplication cannot.

3 Finite model theory and descriptive complexity

Finite model theory developed as a subfield of model theory with emphasis on finite structures. In this new setting many of the standard techniques of model theory, most notably compactness, do not apply. Since testing if a first-order formula has a finite model is undecidable [Tra50], our focus will be on the complexity of model checking: given a finite structure and formula of some logic, decide if this structure is a model of this formula. Considering both the formula and the structure as inputs gives fairly high complexity: checking if a first-order formula holds on a Boolean (two-element) structure is complete for PSPACE (class of problems solvable in polynomial space). Here we consider *data complexity* of the model checking, where a formula is fixed and the only input is the structure. For example, a formula might encode the conditions for a graph to be 3-colourable, and structures are graphs.

This leads us to descriptive complexity, the area studying the direct correspondence between data complexity of logics of different power and complexity classes. In this setting a logic is said to *capture* a complexity class over a class of structures if, informally, the model checking problem for the logic is solvable in the complexity class and every problem in the class is representable in the logic.A classical reference on the subject is the book "Descriptive complexity" by Immerman [Imm99].

Definition 1 (Capture by a logic). *Let C be a complexity class, L a logic and K a class of finite structures. Then L captures C on K if*

1. *For every L-sentence ϕ and every $\mathcal{A} \in K$, testing if $\mathcal{A} \models \phi$ with ϕ fixed and an encoding of $\mathcal{A}$ as an input can be done in C.*
2. *For every collection K' of structures closed under isomorphism, if this collection is decidable in C then there is a sentence $\phi_{K'}$ of L such that $\mathcal{A} \models \phi_{K'}$ iff $\mathcal{A} \in K'$, for every $\mathcal{A} \in K$.*

In descriptive complexity the class of structures is often fixed to be arithmetic structures, that is, structures with $\min, \max, +, \times, \leq, =$ in the language which receive standard interpretations. In particular, the universe of a structure is always considered to be $\{0, \ldots, n-1\}$.

*Example 1 (*Parity*(X)).* This is a formula over successor structures (with $\min, \max, S \in \tau$), models of which have interpretations of X as sets with an odd number of 1's. It encodes a dynamic-programming algorithm for computing parity of X: $P_{odd}(i)$ is true (and $P_{even}(i)$ is false) iff the prefix of X of length i contains an odd number of 1's.

$$\begin{aligned}\exists P_{even} \exists P_{odd} \forall i P_{even}(\min) \wedge \neg P_{odd}(\min) \\ \wedge (P_{odd}(\max) \leftrightarrow \neg X(max)) \wedge (\neg P_{even}(i+1) \vee \neg P_{odd}(i+1)) \\ \wedge (P_{even}(i) \wedge X(i) \rightarrow P_{odd}(i+1)) \wedge (P_{odd}(i) \wedge X(i) \rightarrow P_{even}(i+1)) \\ \wedge (P_{even}(i) \wedge \neg X(i) \rightarrow P_{even}(i+1)) \wedge (P_{odd}(i) \wedge \neg X(i) \rightarrow P_{odd}(i+1))\end{aligned}$$

The first capture result that started this field is due to Fagin [Fag74], who showed that existential second-order logic captures precisely the complexity class NP. His proof has a similar structure to Cook's original proof of NP-completeness of propositional formula satisfiability, but instead of propositional variables Fagin used existentially quantified second-order variables. A notable feature of Fagin's result is that it holds over all structures, whereas $\mathtt{AC}^0$ vs. first-order logic only holds for arithmetic structures. A big open problem is whether there is a logic capturing on all (including unordered) structures the complexity class P of polynomial-time decidable languages.

3.1 Logics between first-order and existential second-order

In the data complexity setting, first-order logic captures $\mathtt{AC}^0$ and existential second-order logic already captures NP. However, many interesting complexity classes lie between these two, in particular classes P (polynomial time) and NL (non-deterministic logspace). The two ways to capture these classes (in the presense of order) are either by extending first-order logic with additional predicates, or restricting second-order logic. In the first approach, a logic for NL is obtained by adding a transitive closure operator to first-order logic, and P is captured by first-order logic together with a least fixed point operator [Imm83,Imm82,Var82]. Here we will concentrate on the second approach, due to Grädel [Grä91].

Definition 2. Restricted second-order *formulae are of the form* $\exists P_1 \dots P_k \forall x_1 \dots x_l \psi(\bar{P}, \bar{x}, \bar{a}, \bar{Y})$*, where* ψ *is quantifier-free.*

Two important types of restricted second-order formulae are $SO\exists$-Horn and $SO\exists$-Krom:

– $SO\exists$-Horn: ψ is a CNF with no more than one positive literal of the form $P_i(t)$ per clause.
– $SO\exists$-Krom: ψ is a CNF with no more than two P_i literals per clause.

In particular, the formula for Parity in the example above is both a second-order Horn and a second-order Krom formula.

Theorem 1 ([Grä92]). *Over structures with successor,* $SO\exists$*-Horn and* $SO\exists$*-Krom capture complexity classes* P *and* NL*, respectively.*

4 Bounded arithmetic

Just like in complexity classes P and NP the computation length is bounded by a polynomial, in bounded arithmetic quantified variables are bounded by a term in the language (e.g., a polynomial in free variables of a formula). Here, instead of describing functions by formulae the goal is to prove their totality within a system; we will say that a system of arithmetic captures a function class if it proves totality of all and only functions in this class. In this setting, arithmetic reasoning with formulas having an unbounded existential quantifier captures primitive recursive functions in the same sense that reasoning with an appropriate bounded version of these formulae captures polynomial-time functions.

4.1 The language and translation from the finite model theory setting

There are two notions of hardness appearing in the setting of theories of arithmetic. One of them is a "descriptive" notion of arithmetic formulae *representing* (that is, describing) a property; another is a more recursion-theoretic notion in terms of the *provable* totality of functions (note that term "defining" is used in both contexts in different literature). In this section we show how results in descriptive complexity, translated into the framework of bounded arithmetic to become representability results, give us provability of properties.

Definition 3. *Define Σ_0^B to be the class of bounded formulas with no second order quantifiers over $\mathcal{L}_2$, and Σ_1^B as a closure of Σ_0^B under bounded existential second-order quantification. Σ_i^B is defined inductively in a natural way.*

It seems that the correspondence of logics in finite model theory framework with representability in bounded arithmetic is folklore. This translation works most naturally in the setting of second-order systems of arithmetic such as systems V_i. Here, by "second-order" systems we mean two-sorted, with one sort for numbers, and another for strings (viewed as sets of numbers). For a thorough treatment of this framework, please see [Coo03]

Let ϕ be a (possibly second-order) formula with free relational variables $R_1, \ldots, R_k$ over a vocabulary that contains arithmetic. The corresponding Σ_i^B formula has R_i as parameters, where non-monadic relational variables are represented using a pairing function, as well as an additional parameter n denoting the size of the structure. It is easy to see that this formula holds on its free variables iff the corresponding structure is a model of the original formula in the finite model theory setting.

In particular, arithmetic versions of $SO\exists$-Horn and $SO\exists$-Krom are subsets of Σ_1^B, with no existential first-order quantifier and Horn (resp. Krom) restriction on the occurrences of quantified second-order variables. In this paper we will abuse the notation and say $SO\exists$-Horn and $SO\exists$-Krom meaning their translation into the language of arithmetic.

Example 2. The Parity formula defined in example 1 can be written as follows in the bounded arithmetic setting:

$$\begin{aligned}
\textsc{Parity}(X) \equiv\ & \exists P_{even} \exists P_{odd} \forall i < |X| \\
& P_{even}(0) \wedge \neg P_{odd}(0) \wedge P_{odd}(|X|) \wedge (\neg P_{even}(i+1) \vee \neg P_{odd}(i+1)) \\
& \wedge (P_{even}(i) \wedge X(i) \rightarrow P_{odd}(i+1)) \wedge (P_{odd}(i) \wedge X(i) \rightarrow P_{even}(i+1)) \\
& \wedge (P_{even}(i) \wedge \neg X(i) \rightarrow P_{even}(i+1)) \wedge (P_{odd}(i) \wedge \neg X(i) \rightarrow P_{odd}(i+1))
\end{aligned}$$

Here, second-order variables are implicitly bounded by the largest value of the indexing term ($i + 1 = |X|$ in this example). Note that here it is possible to reference $P_{odd}(|X|)$, and so we do to simplify the formula. Of course, a direct translation would not account for such details.

4.2 Systems of bounded arithmetic

Early study of weak systems of arithmetic concentrated on restricted fragments of Peano Arithmetic, e.g., $I\Delta_0$ in which induction is over bounded first-order formulae [Par71]. This system, $I\Delta_0$, was used by Ajtai [Ajt83] to obtain lower bounds for the Parity Principle, which implied lower bounds for the complexity class $\mathtt{AC}^0$. A different approach was used by Cook: in 1975 he presented a system PV for polynomial-time reasoning [Coo75]. PV is an equational system with a function for every polynomial-time computable function.

The major development in bounded arithmetic came in the 1985 PhD thesis of S. Buss [Bus86]. There, several (classes of) systems of bounded arithmetic were described, capturing major complexity classes such as `P` and `EXP` (viewed as classes of functions). The best known system is S_2^1, which is a first-order system capturing `P`. To capture higher complexity classes such as `PSPACE` and `EXP`, Buss extends his systems to second-order.

In second-order systems we consider here, the richer language of Buss's first-order systems is simulated using second-order objects. Second-order quantified variables are strings of bounded length; the notation $\exists Z \leq b$ corresponds to $\exists Z\ |Z| \leq b$. First-order objects or numbers are index variables: their values are bounded by a term in number variables and lengths of second-order variables. The translation between first and second-order system is given by the RSUV isomorphism due to [Raz93,Tak93]. Here, the isomorphism is between first-order systems (R and S) and second-order systems (U and V). In particular, it translates Buss's S_2^1 into V_1, a system that reasons with existential second-order definable predicates.

Let $\mathcal{L}_2$ be the language of Peano Arithmetic with added terms $|X|$ (length of X) and $X(t)$ (membership of t in X), where X is second-order. We look at the systems axiomatized by Peano axioms on the number variables, together with the axioms defining length of second order variables: L1: $X(y) \rightarrow y < |X|$ and L2: $y + 1 = |X| \rightarrow X(y)$. Additionally, there is a comprehension axiom

$$\exists Z \leq b \forall i < b (Z(i) \leftrightarrow \phi(i, \bar{a}, \bar{X})), \qquad \text{(Comprehension)}$$

where ϕ is a class of formulae such as Σ_1^B or $SO\exists$-Horn. Such systems of arithmetic reason with objects of complexity allowed in the comprehension axiom. For example, reasoning in V_0 is limited to reasoning with Σ_0^B-definable objects.

Definition 4. *For an integer $i \geq 0$, define V_i to be the system with comprehension over Σ_i^B formulas (e.g., V_1 has comprehension over $\exists SO$ formulae). For a general class Φ of formulas, V-Φ is the system with comprehension over Φ. In particular, V-Horn and V-Krom are the systems with comprehension over $SO\exists$-Horn and $SO\exists$-Krom, respectively.*

Note that even in the weakest of this class of systems, V_0 with its comprehension over formulae with no second-order quantifiers, it is possible to prove induction and minimization principles for the respective class of formulae from comprehension and length axioms.

An interested reader can find more information in [Kra95,Bus86,Coo03] and an upcoming book by Stephen Cook and Phuong Nguyen.

5 Defining functions in the bounded arithmetic setting

In the setting of bounded arithmetic, the "hardness" is usually taken to be the complexity of properties *provable* in this system. In particular, the correspondence with complexity-theoretic notion of hardness is via the provability of the function totality. That is, the strength of a system of arithmetic is associated with the computational complexity of functions that this system proves total. For example, a version of Peano Arithmetic with one unbounded existential quantifier allowed in induction formulae ($I\Sigma_1$) proves the totality of primitive recursive functions, and V_0 where all quantifiers are bounded does the same for $\mathtt{AC}^0$ functions (in the second-order setting).

Definition 5. *(Capture by a system of arithmetic) A system of arithmetic captures a function class if it proves totality of all and only functions in this class.*

Traditionally, functions are introduced by their recursion-theoretic characterization (see [Cob65] for the original such result or [Zam96]). For example, Cobham's characterization of $\mathtt{P}$ uses limited recursion on notation:

$$F(0,\bar{x},\bar{Y}) = G(\bar{x},\bar{Y}) \tag{1}$$
$$F(z+1,\bar{x},\bar{Y}) = cut(p(z,\bar{x},\bar{Y}), H(z,\bar{x},\bar{Y},F(z,\bar{x},\bar{Y}))). \tag{2}$$

Here, the function $cut(x,y)$ cuts out the rest of $H(..)$ beyond the bound $p(z,\bar{x},\bar{Y})$, where $p()$ is a polynomial (that is, a term in the language).

Since we are trying to relate the expressive power of the formulas in comprehension and complexity of functions, we introduce function symbols by setting their bitgraphs to be formulas from the comprehension scheme as follows.

Definition 6. *Let Φ be a logic capturing a complexity class C in the descriptive setting. We define a corresponding function class FC by defining functions f and F in FC as follows:*

$$z = f(\bar{x},\bar{Y}) \leftrightarrow \phi(z,\bar{x},\bar{Y}) \qquad F(\bar{x},\bar{Y})(i) \leftrightarrow i < t \wedge \phi(i,\bar{x},\bar{Y})$$

Here, f and F are number and string functions, respectively, and $\phi \in \Phi$. That is, define functions by formulae from Φ by stating that graphs of number functions and bitgraphs of string functions are representable by formulae from Φ.

In particular, $\mathtt{NL}$ functions are the ones definable by $SO\exists$-Krom formulae and bitgraphs of polynomial-time functions are described by $SO\exists$-Horn formulae.

With these definitions we obtain the following capture results.

- System V_0 captures complexity class $\mathtt{AC}^0$. [Zam96,Coo02]
- Systems V-Horn and V-Krom with comprehension over $SO\exists$-Horn and $SO\exists$-Krom, respectively, capture $\mathtt{P}$ and $\mathtt{NL}$. [CK01,CK03]
- System V_1 also captures $\mathtt{P}$. [Zam96]

Note that the system V_1, although likely stronger than V-Horn because it reasons with $\mathtt{NP}$-predicates, also captures $\mathtt{P}$. That is, Σ_1^B-theorems of V_1 and V-Horn are the same, but it is likely that V-Horn does not prove the comprehension axiom of V_1, which is a Σ_2^B statement. This leads to the following question:

6 When do systems based on formulas describing a complexity class capture the same class?

What makes systems such as V_0, V-Horn and V-Krom "minimal" among systems capturing the corresponding classes? They operate only with objects from the class, and yet prove totality of all functions they can define. The informal answer is *provable closure under first-order operations.* If a system can prove its own closure under $\mathtt{AC}^0$ reductions such as conjunction, disjunction and complementation, then, with some additional technicalities, this system can prove totality of all functions definable by objects in its comprehension axiom.

Let C be a complexity class. Suppose that Φ_C is a class of (existential second-order) formulas that captures C in the descriptive complexity setting. We define a theory of bounded arithmetic V-Φ_C to be V_0 together with comprehension over bounded Φ_C. The following is an informal statement of the general result:

Claim. [Kol04,Kol05] Let $\mathtt{AC}^0 \subseteq C \subseteq \mathtt{P}$. Suppose that Φ_C is closed under first-order operations provably in V-Φ_C. Also, suppose that for every $\phi(\bar{x}, \bar{Y}) \in \Phi_C$, if V-$\Phi_C \vdash \phi$ then there is a function F on free variables of ϕ which is computable in C and witnesses existential quantifiers of ϕ. Then the class of provably total functions of V-Φ_C is the class of functions computable in C.

Two examples of such systems are V-Horn and V-Krom. In particular, V-Horn formalizes and proves correctness of Horn formula satisfiability algorithm. Although V-Horn is provably in the system equivalent to limited recursion-based systems of polynomial-time reasoning, it has an additional nice feature: it is finitely axiomatizable. The system V-Krom formalizes the non-trivial inductive counting algorithm of Immerman [Imm88,Sze88], used to prove the closure of $\mathtt{NL}$ (and thus $SO\exists$-Krom) under complementation. However, we don't know whether $\mathtt{NP} = \mathtt{coNP}$ and thus whether the class of predicates in the comprehension axiom of V_1 is closed under complementation. Moreover, even if it is proven that $\mathtt{NP} = \mathtt{coNP}$, the proof has to be constructive enough to be formalizable within V_1 to allow us to apply the claim.

This brings up an interesting meta-question: when are the properties of a complexity class itself provable within this class? We were able to prove the basic properties of $\mathtt{P}$ and $\mathtt{NL}$ with reasoning no more complex then the class itself. However, for classes such as symmetric logspace, now proven to be equal to logspace [Rei04], it is not clear whether the proof of complementation (or the proof of equivalence with logspace) are formalizable using only reasoning within this class. This line of research is related to other meta-questions in complexity theory such as what proof techniques are needed to prove complexity separations. For example, natural proofs of Razborov and Rudich [RR97] address the question of $\mathtt{NP}$ vs. $\mathtt{P/poly}$, motivated by the $\mathtt{P}$ vs. $\mathtt{NP}$ question.

7 Conclusions

In this paper we touch upon one example of a connection between two different notions of hardness: the hardness of describing a property versus the hardness of proving a property. Another line of research that should be mentioned here is proof complexity. In proof complexity, the object of study is proof systems such as resolution system and Frege system; there, the lengths of proofs is the main complexity measure. For many systems of bounded arithmetic there are proof systems that are their non-uniform counterparts (that is, the system of arithmetic proves soundness of the proof system and the proof systems proves the axioms of the system of arithmetic). The line of research exploring connections between finite model theory and proof complexity is pursued by Atserias. He uses his results in finite model theory to obtain resolution proof system lower bounds [Ats02], and, in his later work with Kolaitis and Vardi, uses constraint satisfaction problems as a generic basis for a class of proof system [AKV04].

Complexity in logic is a broad area of research, with many problems still unsolved. Little is known about connections among different settings and notions of hardness. Here we have given one such example and have mentioned a couple more; other connections among various notions of complexity in logic are waiting to be discovered.

Acknowledgements. The results on connections between finite model theory and bounded arithmetic mentioned here come from my joint work with Stephen Cook (which resulted in my PhD thesis). I am very grateful to him for suggesting many of the ideas, as well as the framework in which these connections became natural.

References

[Ajt83] M. Ajtai. Σ^1_1-Formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.

[AKV04] A. Atserias, Ph. G. Kolaitis, and M. Y. Vardi. Constraint propagation as a proof system. In *10th International Conference on Principles and Practice of Constraint Programming*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer-Verlag, 2004.

[Ats02] Albert Atserias. *Fixed-point logics, descriptive complexity and random satisfiability.* PhD thesis, UCSC, 2002.

[Bus86] S. Buss. *Bounded Arithmetic.* Bibliopolis, Naples, 1986.

[CK01] S.A. Cook and A. Kolokolova. A second-order system for polynomial-time reasoning based on Grädel's theorem. In *Proceedings of the Sixteens annual IEEE symposium on Logic in Computer Science*, pages 177–186, 2001.

[CK03] S.A. Cook and A. Kolokolova. A second-order system for polytime reasoning based on Grädel's theorem. *Annals of Pure and Applied Logic*, 124:193–231, 2003.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel, editor, *Logic, Methodology and Philosophy of Science*, pages 24–30, Amsterdam, 1965. North-Holland.

[Coo75] S. A. Cook. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, pages 83 –97, 1975.

[Coo02] S. A. Cook. CSC 2429S: Proof Complexity and Bounded Arithmetic. Course notes, URL: "http://www.cs.toronto.edu/∼sacook/csc2429h", Spring 1998-2002.

[Coo03] S. Cook. Theories for complexity classes and their propositional translations. In Jan Krajicek, editor, *Complexity of computations and proofs*, pages 175–227. Quaderni di Matematica, 2003.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. *Complexity of computation, SIAM-AMC proceedings*, 7:43–73, 1974.

[Grä91] E. Grädel. The Expressive Power of Second Order Horn Logic. In *Proceedings of 8th Symposium on Theoretical Aspects of Computer Science STACS '91, Hamburg 1991*, volume 480 of *LNCS*, pages 466–477. Springer-Verlag, 1991.

[Grä92] E. Grädel. Capturing Complexity Classes by Fragments of Second Order Logic. *Theoretical Computer Science*, 101:35–57, 1992.

[Imm82] N. Immerman. Relational queries computable in polytime. In *14th ACM Symp.on Theory of Computing, Springer Verlag (Heidelberg, FRG and NewYork NY, USA)-Verlag*, pages 147 –152, 1982.

[Imm83] Neil Immerman. Languages that capture complexity classes. In *15th ACM STOC symposium*, pages 347–354, 1983.

[Imm88] Immerman. Nondeterministic space is closed under complementation. In *SCT: Annual Conference on Structure in Complexity Theory*, 1988.

[Imm99] N. Immerman. *Descriptive complexity.* Springer Verlag, New York, 1999.

[Kol04] A. Kolokolova. *Systems of bounded arithmetic from descriptive complexity.* PhD thesis, University of Toronto, October 2004.

[Kol05] A. Kolokolova. Closure properties of weak systems of bounded arithmetic. In *Computer Science Logic: 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005. Proceedings*, volume 3634 of *LNCS*, pages 369–383, 2005.

[Kra95] J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory.* Cambridge University Press, New York, USA, 1995.

[Par71] R. Parikh. Existence and feasibility of arithmetic. *Journal of Symbolic Logic*, 36:494 –508, 1971.

[Raz93] A. Razborov. An equivalence between second-order bounded domain bounded arithmetic and first-order bounded arithmetic. In P. Clote and J. Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 247–277. Clarendon Press, Oxford, 1993.

[Rei04] O. Reingold. Undirected ST-Connectivity in Log-Space. *Electronic Colloquium on Computational Complexity*, ECCC Report TR04-094, 2004.

[RR97] A.A. Razborov and S. Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55:24–35, 1997.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Tak93] G. Takeuti. RSUV isomorphism. In P. Clote and J. Krajíček, editors, *Arithmetic, proof theory and computational complexity*, pages 364–386. Clarendon Press, Oxford, 1993.

[Tra50] B. Trahtenbrot. The impossibility of an algorithm for the decision problem for finite domains. *Doklady Academii Nauk SSSR*, 70:569–572, 1950. In Russian.

[Var82] Moshe Y. Vardi. The complexity of relational query language. In *14th ACM Symp.on Theory of Computing, Springer Verlag (Heidelberg, FRG and NewYork NY, USA)-Verlag*, 1982.

[Zam96] D. Zambella. Notes on polynomially bounded arithmetic. *The Journal of Symbolic Logic*, 61(3):942–966, 1996.