Expressing vs. proving: relating forms of complexity in logic

Antonina Kolokolova

Memorial University of Newfoundland
kol@cs.mun.ca

Abstract. Complexity in logic comes in many forms. 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 the non-deterministic log-space complexity class), and its totality (and, thus, the closure of NL under complementation) is provable within NL-reasoning. Lastly, we will touch upon the topic of formalizing complexity theory using logic, and the meta-question of complexity of logical reasoning about complexity-theoretic statements.
This is intended to be a high-level overview, suitable for readers who are not familiar with complexity theory and complexity in logic.

1 Introduction

What do we mean when we say that a certain object is "complex" or a problem is "hard"? It may be that the object is hard to describe, or its properties are hard to prove. It may be that the object in question is a problem for which it is not easy to find a solution. Often, there is a relationship among various forms of hardness: problems that cannot be described in a certain language would not be solvable with a limited amount of resources. A system can be complex if its behaviour is hard to predict (i.e., describe and prove its properties).

In the context of mathematical logic it is possible to define many forms of complexity. Historically, the notions of hardness that received much attention were 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.)

It is natural that different areas of of mathematic (and mathematical logic) emphasize distinct notions of hardness, appropriate for their framework. In particular,

1. Finite model theory (and its subfield descriptive complexity) focuses more on the expressive power of logics, that is, complexity of describing objects.
2. Bounded arithmetic and proof complexity, as the name suggests, focus on provability and the complexity of proving properties of objects in theories and proof systems of varying power.
3. Computational complexity theory and some areas of computational logic are concerned with complexity of solving a problem (such as determining a membership in a set), where the problem is often presented in the language of logic.

Complexity theory developed as an area devoted to study of the concepts of hardness in their multitude, so it is not surprising that both finite model theory and bounded arithmetic with proof complexity have strong ties with complexity theory, making connections between the concepts, sharing the terminology and proof techniques. However, at this point it might be beneficial to look at the direct relationships among different notions of hardness in logic without going through a computational model as an intermediate step.

An example of such a connection is the relationship between systems of bounded arithmetic and proof systems. In some cases, there is a two-way relationship between the system of arithmetic which proves the soundness of its counterpart proof system, and, for the other direction, has all of its proofs translatable into the proofs in the proof system. For example, the system of arithmetic V^0 has such a connection with the proof system Bounded-Depth Frege, and the system of arithmetic V^1 with Extended Frege.

Here we will present a different example of a direct relationship between logical characterizations, a connection between descriptive complexity and proving power of systems of bounded arithmetic. We show how such connection helps us obtain results in bounded arithmetic using the notion of hardness from descriptive complexity. We are interested in studying the power of proof systems that are allowed to reason only about objects of predefined complexity. .We show that for a certain range of complexities, if the class of objects is "sufficiently robust" (we will define this more precisely later), then it is possible to infer directly what is the power of a system of arithmetic which is allowed to reason with the objects of this class.

We start with the canonical notion of hardness from computational complexity theory.

2 The computational complexity setting

The computational complexity of a problem is measured in terms of amount of resources necessary for an algorithm to solve the problem. The conventional resources are space (memory) and time. Algorithms (programs) can be deterministic and non-deterministic; in the latter case we sometimes talk about complexity

of verifying a (guessed) solution. For example, the famous class NP consists of problems solvable by the non-deterministic polynomial-time algorithms (that is, problems which have solutions verifiable by polynomial-time algorithms). Similarly, the problems solvable by non-deterministic algorithms using amount of space logarithmic in the length of the input comprise the class NL.

An important feature which supports the concept of a complexity class as a useful framework is that most of the complexity classes are robust, in a sense that definitions in different frameworks, often with different computational models often give the same class of algorithms. In particular, a composition or a Boolean combination of problems in the class is usually still within the same class (with the notable exception of complementation: for non-deterministic time classes the closure under complementation is a major open problem). One of the most robust in this respect is the class polynomial time, which is closed under Boolean operators, composition of functions and more. For a contrast, the class of problems solvable in the time linear in their input does not have such nice closure properties. This is one of the reasons that the class P of polynomial-time solvable problems is has been a more popular object of research than $LinTIME$.

A complexity class that is also very robust, and has several natural definitions in the context of logic is uniform $\mathtt{AC}^0$, the class of problems solvable by a uniform (that is, computable by a very simple algorithm) 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 proof complexity, it corresponds to the Bounded Frege proof system. In the descriptive complexity setting $\mathtt{AC}^0$ can be defined as the class of properties expressible in first-order logic over structures that contain arithmetic. We will discuss it in more detail later.

Yet another attraction of complexity classes is that they often have a set of "maximal" problems, where maximality is with respect to reductions between problems. That is, a problem belonging to a complexity class is complete for this class if any other problem in this class can be solved by reducing it to the complete one. A classical example of a problem complete for NP is 3-colourability: given a graph as an input, determine if it can be coloured with three colours so that no edge connects vertices of the same colour. That is, any problem in NP can be "disguised" (with little effort) as 3-colourability. This makes it possible to concentrate on just one problem when studying properties of a complexity class: if such a problem is proven to be solvable in another class, then the whole class of problems for which it is complete can be solved within that class. A recent result of this kind is due to Reingold [Rei05], who have shown how to solve graph reachability in undirected graphs using only logarithmic amount of space (in L), which implies that all problems in the corresponding class symmetric logspace are solvable in L. The general graph reachability problem (determining if there is a path between two vertices) is complete for the non-deterministic logspace class NL; it is still open whether L is a proper subset of NL.

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 odd 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 odd number of elements (in fact, since this result holds in a non-uniform setting: adding to the vocabulary any kind of numerical predicates, even undecidable ones, would still not help a first-order formula to express a parity of a string). 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[1]. 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 to an algorithm deciding this 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 using amount of memory polynomial in the length of the input), which is believed to be larger than P or NP. Therefore, 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 correspond to graphs: the elements of their universes are treated as vertices, and a binary relation E is viewed as an edge relation of a graph. This E is the only free relational variable occurring in the formula for 3-colourability.

This leads us to descriptive complexity, the area studying the direct correspondence between complexity classes and data complexity of logics of different power. 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 by a formula of this logic. A classical reference on the subject is the book "Descriptive complexity" by Immerman [Imm99]. Here, we follow the presentation from [Lib04].

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 $\mathcal{A} \in K$.*

[1] Please see [Fag93] for an excellent survey of this area

In descriptive complexity the class of structures is often restricted to arithmetic structures, that is, structures with $\min, \max, +, \times, \leq, =$ in the language which receive standard interpretations. In that case, the universe of a structure is considered to be $\{0, \ldots, n-1\}$. This comes from the need for arithmetic operations to arithmetize Turing machines in the logic context. Over general structures, most such logics (in particular, first-order logic) are strictly weaker, since it becomes impossible to express computation.

Example 1 ($\textsc{Parity}(X)$*).* This is a formula over successor structures (that is, structures with min, max, and successor function s in the vocabulary which always receive standard interpretations). The structures also contain a unary relational symbol X. Models of the formula below are those of such structures that interpret X as a unary relation (that is, a set) with an odd number of elements. 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. Here, si is the successor of i, that is $i+1$.

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

We usually think of the parity problem in the complexity-theoretic setting as a problem of determining if an input string contains an odd number of 1s. This example illustrates how a translation between these contexts is done: a binary string of length n becomes a structure over the universe of size n with a unary relation, X, such that for any i in the universe of the structure $X(i)$ holds iff the i^{th} bit of the string in the complexity-theoretic setting is 1. Thus, any language in complexity-theoretic setting can be translated into the finite model theory framework using this method for representing binary strings (provided some order is relation either already in the vocabulary or is definable, to allow us to say 'the 'i^{th} element"). However, often it is more convenient to consider other types of structures: for example, although graphs can be represented as strings (and they are when considered as inputs to Turing machines), in the finite model theory setting it is much more natural to view them as structures with a binary relation representing the edge relation of the graph.

The first capture result, a result relating a complexity class to a logic, 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 [Coo71], but instead of propositional variables Fagin uses existentially quantified second-order variables. These variables encode the computation (Cook's computational tableau of the NP-machine), and the formula states that the computation is correct and the final line states that the input string (encoded as a unary relation in a structure in Fagin's setting) is in the language. 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: existential second-order logic is powerful enough to express all relations needed to encode computation (which, for example, first-order logic provably cannot express).

Another feature of Fagin's theorem is that it shows how local the verification of a solution is for an NP-language. Although this is already present in the proof of Cook's theorem, the fact that a finite formula is enough to describe the computation of a NP Turing machine on input of any size makes this fact very explicit. This is an very interesting property of NP languages, especially intriguing in that most of the relativized versions of NP, that is $\mathtt{NP}^O$ for some oracle O, do not possess it. Here, $\mathtt{NP}^O$ is a class of languages computable by NP algorithm with an additional ability to ask questions "is string s in O?" where O is some fixed language. It is a famous result by Baker, Gill and Solovay [BGS75] that none of the proof techniques that is insensitive to the presence of oracles (such as diagonalization) can resolve P vs NP question. The logic characterization of relativization and locality has been studied in [AIV92] in the setting of systems of arithmetic. In particular, they show that the locality condition only holds for a small limited subset of possible oracles.

A major open problem in finite model theory is whether there is a logic capturing on all (including unordered) structures the complexity class P of polynomial-time decidable languages. It is conjectured [Gur88] that there is no such logic: if so, then $\mathtt{P} \neq \mathtt{NP}$.

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 presence 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. SNP *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 SNP 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. The notion of SNP was introduced by Papadimitriou and Yanakakis in [PY88]; Grädel does not use this terminology when defining $SO\exists$-Horn and $SO\exists$-Krom formulas.

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

The proof is based on the fact that propositional Horn (resp. 2CNF) formula satisfiability is complete for P (resp. NL). To convert these formulas in the propositional setting, it is enough to take a conjunction of copies of the formulas for every possible value of the tuple of variables under the universal quantifier: the relational variables on a fixed tuple of values become essentially propositional variables. Since there are only constantly many quantified first-order variables, the resulting formula has length polynomial in the size of the structure. For the other direction of the proof in the $SO\exists$-Horn case note that the formula from Fagin's proof encoding the computation of an NP-machine is $SO\exists$-Horn when the machine is deterministic.

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 as 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 theories of arithmetic setting. 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 (the term "defining" is used in either context in literature). In this section we show how results in descriptive complexity, translated into the framework of bounded arithmetic to become representability results, help us obtain results about provability of function totality.

Definition 3. *Call a (first-order) quantifier bounded if the domain of quantified variables is restricted to values less than the value of a bound, where the bound is a term in the language. For second-order quantifiers, the bounding term bounds the length of the second-order object (i.e., the length of a string). A formula is bounded if all of its quantifiers are bounded.*

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 Cook's survey [Coo03] or a book by Cook and Nguyen [CN08], scheduled to be published in March 2009.

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_j 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. So here in the arithmetic setting a binary string X corresponding to an input string of an algorithm would be a free second-order variable in a formula.

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 both Grädel's logics and 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, X is the only free variable, and all 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. Here we did not need to refer to the size of the universe n since $|X| = n$; if there were no relational symbols in the vocabulary a corresponding formula would have to reference n explicitly.

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 defined using Cobham's [Cob65] recursion-theoretic characterization of polynomial-time functions. Yet another way of characterizing polynomial-time functions was presented by Leivant in [Lei91,Lei94]; his characterization is somewhat similar in nature to the one we use here.

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`, `EXP` and the levels of polynomial-time hierarchy `PH` (viewed as classes of functions). The best known system is S^1_2, 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; in Buss's second-order systems for `PSPACE` and `EXP` the vocabulary contains powerful functions missing in our setting (function $x\#y = 2^{|x|\cdot|y|}$). 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, S) and second-order systems (U and V). In particular, it translates Buss's S^1_2 into V^1, a system that reasons with Σ^B_1, that is, 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 viewed as a set or a binary string. 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 scheme: for every formula ϕ from a given class of formulae Φ such as $\Phi = \Sigma^B_1$ or $\Phi = SO\exists$-Horn,

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

Such systems of arithmetic reason with objects of complexity allowed in the comprehension axiom scheme. For example, reasoning in V^0 is limited to reasoning with Σ^B_0-definable objects.

Although the general definition uses an axiom scheme (and thus an infinite set of axioms) in many cases such as $SO\exists$-Horn it is possible to obtain a finitely axiomatized system by encoding by a Φ-formula an evaluation algorithm for the formulas from Φ. In particular, the base system V^0 is finitely axiomatized. Please see [CK03] for details.

Definition 4. *For an integer $i \geq 0$, define V^i to be the system with comprehension over Σ^B_i 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 V^0, the weakest of this class of systems 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. That is, an induction axiom of the form $(Z(0) \wedge \forall i < n(Z(i) \rightarrow Z(i+1))) \rightarrow Z(n)$ is provable directly from the (Σ_0^B) comprehension scheme and length axioms. Thus there is no need for an explicit induction axiom as in Buss's systems or Peano Arithmetic.

An interested reader can find much more information about bounded arithmetic and related areas in [Kra95,Bus86,HP93,Coo03] and an upcoming book by Stephen Cook and Phuong Nguyen [CN08].

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 P uses limited recursion on notation:

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

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 occurring in the in comprehension axiom scheme 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, 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 P and NL. [CK01,CK03]
- System V^1 also captures P. [Zam96]

Recently, Kuroda [Kur07] characterized the complexity class LogCFL using the same framework. He used comprehension over essentially the descriptions of uniform semi-unbounded fan-in circuits (which form a class SAC^1 equivalent to LogCFL) to capture this class, and showed that he can prove all the main properties of the class inside the resulting system. In particular, he formalized within his system V-Q^{SAC} the inductive counting algorithm for closure of SAC^1 under complementation due to [BCD$^+$89].

Note that the system V^1, although likely stronger than (that is, not conservative over) V-Horn because it reasons with NP-predicates, also captures P by our definition of capture. That is, Σ^B_1-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^B_2 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.

Examples of such systems are V-Horn, V-Krom and Kuroda's V-Q^{SAC}; V^0 is another example of a system like this, although in its case the conditions are vacuously true. 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 NL (and thus $SO\exists$-Krom) under complementation. A version of inductive counting is also used to prove closure of LogCFL under complementation [BCD$^+$89], which Kuroda showed to be formalizable using LogCFL reasoning.

However, we don't know whether NP = 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 NP = 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 P and NL with reasoning no more complex then the class itself. Somewhat surprisingly, Kuroda proved that this holds for the class LogCFL which is not even known to be equal to any of Schaefer's classes [Sch78]. However, for classes such as symmetric logspace, now proven to be equal to L [Rei05], it is not clear whether the proof of complementation (or the proof of equivalence with L) are formalizable using only reasoning within this class.

7 Formalizing complexity theory

This line of research leads to a more general meta-question in complexity theory: what proof techniques could be used to prove separations and collapses of complexity classes? Here, we briefly mention some of the directions of research in this area.

In complexity theory there were several meta-results showing that a certain class of proof techniques cannot resolve long-standing open problems such as P vs. NP: relativization of [BGS75], natural proofs of [RR97] (under some assumptions) and, recently, algebrization of [AW08]. All of them have a flavour of independence results – the answers to complexity-theoretic questions are somehow independent of the assumptions implicit in the current techniques.

In case of relativization, this notion of independence was made precise in the work of Arora, Impagliazzo and Vazirani [AIV92]. A technique is said to *relativize* if it is oblivious to the presence of oracles (in arithmetic setting, to undefined predicate symbols): that is, to whether an algorithm can ask queries "is a given string in O" for some fixed language O, and get a reply in constant time. Baker, Gill and Solovay showed that there are oracles A and B such that $\mathtt{P}^A = \mathtt{NP}^A$ and $\mathtt{P}^B \neq \mathtt{NP}^B$. Thus, no technique oblivious to the presence of oracles can resolve P vs. NP question.

Although formalization of computation with oracles goes back to Buss [Bus86], authors of [AIV92] chose a different route by presenting a theory of arithmetic $\mathcal{RCT}$) all standard models of which correspond exactly to classes of polynomial-time functions with an oracle, the difference between the models corresponding to differences between oracles. That is, any standard model of their system can be viewed as $\mathtt{P}^O$ for some language O, and for any O there would be a corresponding model. So a complexity-theoretic concept of a relativized complexity class becomes a model-theoretic notion of a standard model of some theory. In this context the notion of a non-relativizing technique became very clear: a technique is non-relativizing iff it is independent of $\mathcal{RCT}$. This theory, based on Cobham's axiom without the minimality condition on the class of functions, is similar to PV_1, but it has an unrestricted induction scheme and some additional axioms.

Another line of research studies which complexity-theoretic questions might be solvable already with the limited power of systems of bounded arithmetic. Such are the formalizations of closures under complementation in V-Horn, V-Krom and related systems. Rephrasing classical result from Buss's thesis [Bus86] in our framework, if the system V^1 proves that a certain function is in $\mathtt{NP} \cap \mathtt{coNP}$, then this function is in polynomial-time. It is consistent with our knowledge that $\mathtt{NP} \cap \mathtt{coNP}$ might not collapse to $\mathtt{P}$, but if this is the case, then V^1 does not prove totality of functions that are both in $\mathtt{NP}$ and in $\mathtt{coNP}$ and yet are not polynomial-time computable. So provability of complexity-theoretic properties in a weak system of arithmetic is an indicator that this property is "relatively not too complex" in a very precise sense.

It is natural that the stronger the theory, the stronger are the complexity-theoretic statements it can prove. A study of this with the focus on consequences of $\mathtt{coNP} \subseteq P/poly$ was done by Cook and Krajicek [CK07].

There are many more results and formalizations relating complexity-theoretic concepts and the concepts in bounded arithmetic and proof complexity. Books and papers by Cook, Krajicek, Buss, Pudlak, Jerabek, Nguyen and others would be a good source for an interested reader.

8 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. Although there are well-established relationships of both of these areas with complexity theory, studying the direct relationship helped us obtain new results.

For the other examples of such connections one can look at the relationship between proof complexity and areas of bounded arithmetic, as one example, and the finite model theory and constraint satisfaction on as another. 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 systems [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.

Acknowledgments. 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

[AIV92] S. Arora, R. Impagliazzo, and U. Vazirani. Relativizing versus nonrelativizing techniques: The role of local checkability. Manuscript, 1992.

[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.

[AW08] S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 731–740, 2008.

[BCD+89] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SIAM J. Comput.*, 18(3):559–578, 1989.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[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.

[CK07] S. Cook and J. Krajicek. Consequences of the provability of $np \subseteq p/poly$. *Journal of Symbolic Logic*, 72(4):1353–1371, 2007.

[CN08] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Perspectives in Logic of the Association for Symbolic Logic. Cambridge University Press, 2008.

[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.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[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.

[Fag93] R. Fagin. Finite-model theory – a personal perspective. *Theoretical Computer Science*, 116:3–31, 1993.

[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.

[Gur88] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current trends in theoretical computer science*, pages 1–57. Computer Science Press, Rockville, MD, 1988.

[HP93] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*. Springer Verlag, 1993.

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

[Imm88] N. 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. Krajiček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*. Cambridge University Press, New York, USA, 1995.

[Kur07] Satoru Kuroda. Generalized quantifier and a bounded arithmetic theory for LOGCFL. *Arch. Math. Log.*, 46(5-6):489–516, 2007.

[Lei91] Daniel Leivant. A foundational delineation of computational feasibility. In *Proceedings of the Sixth IEEE conference on Logic in Computer Science*, pages 2–11, 1991.

[Lei94] Daniel Leivant. A foundational delineation of poly-time. *Information and Computation*, 110:391–420, 1994.

[Lib04] L. Libkin. *Elements of Finite Model Theory*. Springer Verlag, 2004.

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

[PY88] Christos Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 229–234, 1988.

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

[Rei05] Omer Reingold. Undirected st-connectivity in log-space. In *STOC*, pages 376–385, 2005.

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

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.

[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. Krajiček, 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.