Electronic Colloquium on Computational Complexity, Report No. 45 (2007)

The Complexity of Deciding if a Boolean Function Can Be Computed by Circuits over a Restricted Basis*

Heribert Vollmer

Theoretische Informatik, Universität Hannover, Appelstr. 4, 30167 Hannover, Germany. vollmer@thi.uni-hannover.de

Abstract. We study the complexity of the following algorithmic problem: Given a Boolean function f and a finite set of Boolean functions B, decide if there is a circuit with basis B that computes f. We show that if both f and all functions in B are given by their truth-table, the problem is in quasipolynomial-size AC^0, and thus cannot be hard for $\mathrm{AC}^0(2)$ or any superclass like NC^1, L, or NL. This answers an open question by Bergman and Slutzki (SIAM J. Comput., 2000). Furthermore we show that, if the input functions are not given by their truth-table but in a succinct way, i.e., by circuits (over any complete basis), the above problem becomes complete for the class coNP.

1 Introduction

Emil Post in his dissertation [Pos21] studied the completeness of the propositional part of Russel and Whitehead's *Principia Mathematica* and, more generally, he became interested in the question which logical functions can be expressed by means of other basic functions. For a set of Boolean functions B he defined the closure of B, denoted by $[B]$, as the set of functions that can be obtained from functions in B by general composition (his term was "superposition"). Remarkably he was able to identify all classes of functions that are closed in this sense [Pos20, Pos41]—these classes are nowadays known as *clones*.

In modern terms, borrowing from Boolean circuit theory, the closure of a set B consists exactly of those functions computable by Boolean circuits whose gates evaluate functions from B; here one speaks of B as the *basis* of the circuit. Hence, Post asked, given a basis (i.e., a set of Boolean functions) B and a Boolean function f, if f can be computed by circuits with basis B (that is: if $f \in [B]$).

Though this is certainly a very fundamental question which, from a mathematical standpoint, is completely solved by the contributions of

* Supported in part by DFG Grant Vo 630/5-2 and EPSRC Grant 531174.

ISSN 1433-8092

Emil Post, it has not received much attention from an algorithmic and computational complexity point of view until recently. It is clearly decidable, but given a function f of arity n and a basis B, if we simply generate all functions in $[B]$ of arity n and check if f is one of them (see [Pip97]), we may have to generate up to 2^{2^n} functions; hence if f is given by its truth-table (of size 2^n), this naive approach yields only an exponential-time algorithm. Bergman and Slutzki [BS00], making use of results from universal algebra and Post's list of clones [Pos41] improved this considerably and presented an NL-algorithm. They asked if the problem is actually complete for NL.

In this short note, we give an algorithm which is only a slight modification of the algorithm by Bergman and Slutzki [BS00] and show that the problem, given a function f and a basis B by their truth-tables, to decide if $f \in [B]$, is in quasipolynomial-size AC^0. This answers the mentioned open question in [BS00] negatively: The problem cannot be hard for NL, in fact not even hard for L or NC^1.

From a more practical point of view, one might object that usually Boolean functions are not given by their truth-table but in some high-level description language, and the complexity of the problem above for truth-tables is not very relevant. One possibility for a more succinct description of Boolean functions is of course to use general circuits (i.e., over any complete basis) as representation of the functions they compute. This approach was already followed in a paper by Böhler and Schnoor [BS07]. They looked at the list of clones from [Pos41], and studied for each of them the membership problem, i.e., for each fixed clone B we have a membership problem MEM(B): Given f, decide if $f \in B$. Since this is a different computational problem for each individual clone, we will speak of the *non-uniform membership problem*, in contrast to the *uniform membership problem* studied by Bergman and Slutzki [BS00] where both f and B are part of the input. When using the input representation of [BS00] by truth-tables we will see that for all except 8 clones the non-uniform membership problem is in AC^0, while for the remaining 8 it is in qAC^0. Böhler and Schnoor [BS07] showed that, if the input function is represented by a Boolen circuit, the non-uniform clone membership problem is tractable only for a very small number of degenerated clones, and coNP-complete in all other cases. For our study of the uniform membership problem, the coNP lower bound of course remains valid. A contribution of this paper is a matching upper bound: The uniform clone membership for circuit representation is coNP-complete.

When moving from the Boolean to a larger (finite) universe, the number of clones becomes uncountable [Pip97]. As a consequence, an approach as above, relying on a the inclusion structure among clones as identified by Post for the Boolean case, no longer works. Bergman, Juedes, and Slutzki [BJS99] showed that the uniform membership problem is complete for EXPTIME, if the input functions are given by their function table. Here we note that, if the input functions are given in a succinct representation, the uniform membership problem over arbitrary finite domains is complete for double exponential time.

In Sect. 2 we recall some notions and results from universal algebra that lead us in principle to an algorithm to check if a function f can be computed by circuits over basis B. The algorithm will mainly consist of a number (linear in the arity of f) of calls to a subroutine to check if a Boolean relation is a so called *invariant* of a Boolean function. In Sect. 3 we turn to a complexity examination of this algorithm, and we will see that the complexity is determined by the complexity of the mentioned check of invariants of Boolean functions.

2 Some Basics and Algorithms from Universal Algebra

Let B be a set of Boolean functions. The *closure of B under superposition* (or short: the *closure of B*), denoted by $[B]$, is the smallest set of Boolean functions that satisfies:

- $B \subseteq [B]$.
- Every function I_k^n is in $[B]$, where $\mathrm{I}_k^n(x_1, \ldots, x_n) = x_k$, the n-ary projection to the k-th coordinate, $1 \leq k \leq n$.
- If $g_1, \ldots, g_m \colon \{0,1\}^n \to \{0,1\}$, $f \colon \{0,1\}^m \to \{0,1\}$ are in $[B]$, then the function $h \colon \{0,1\}^n \to \{0,1\}$ is in $[B]$, where $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$.

In terms of Boolean circuits, the closure of B is exactly the class of Boolean functions that can be computed with circuits over basis B [PK79, Pip97, Vol99].

A set B of Boolean functions is *closed*, if $B = [B]$. Emil Post identified all closed classes of Boolean functions, gave for each of them a finite basis, and characterized their inclusion structure [Pos41], see Fig. 1. Closed classes of functions are nowadays mostly referred to as *clones* [Coh65]. Clones denoted with a bold circle in the figure denote the famous 5 Post's classes, i. e., the dual atoms (maximal pre-complete classes) of the lattice. To show that a basis is complete in the sense that it can compute all

Boolean functions, it suffices to show that it is not included in one of these classes.

We will not give a full description of all classes of *Post's lattice*, as the inclusion diagram became known; important here is only that the lattice consists only of a finite number of clones plus 8 countably infinite chains of clones, the S-classes. A gentle introduction of the lattice and some central results can be found in [Pip97]; for a concise introduction the reader may consult [PK79, Sze86]. Relations to complexity questions are surveyed in [BCRV03, BCRV04]. In the sequel we will restrict ourselves to the description of a Galois connection between this lattice and a lattice of Boolean relations, since this will be of central importance for our results.

For this, let R be a Boolean relation of arity n and let f be a Boolean function of arity m. We say that R is *closed* under f (or f *preserves* R, or f is a *polymorphism* of R, or R is an *invariant* of f), in symbols: $f \approx R$, if for all $x_1, \ldots, x_m \in R$, where $x_i = (x_i[1], x_i[2], \ldots, x_i[n])$, we have

$$\Big(f\big(x_1[1], \cdots, x_m[1]\big), f\big(x_1[2], \cdots, x_m[2]\big), \ldots, f\big(x_1[n], \cdots, x_m[n]\big)\Big) \in R.$$

Let us denote the set of all polymorphisms of R by $\mathrm{Pol}(R)$, and for a set Γ of Boolean relations, we define $\mathrm{Pol}(\Gamma)$ to be the set of Boolean functions that are polymorphisms of every relation in Γ. Conversely, for a set B of Boolean functions, let $\mathrm{Inv}(B)$, denote the set of all Boolean relations that are invariants of every function in B.

Also on the side of Boolean relations a closure operator has been studied. Let Γ be a set of Boolean relations. The closure of Γ, denoted by $\langle\Gamma\rangle$, is the smallest set of Boolean relations satisfying:

- $\Gamma \subseteq \langle\Gamma\rangle$.
- The binary equality relation, $\{(0,0),(1,1)\}$ is in $\langle\Gamma\rangle$.
- If R is defined by $R(x_1, \ldots, x_n) \iff \exists x_{n+1} \ldots \exists x_m F$, where F is a conjunction $F = C_1 \wedge \cdots \wedge C_p$ with clauses that are obtained by applying a relation in $\langle\Gamma\rangle$ to a vector of variables taken from $\{x_1, \ldots, x_m\}$, then $R \in \langle\Gamma\rangle$. We say that R is obtained from relations in Γ by *positive primitive definition.*

Sets of relations closed in this sense are called *relational clones* or *co-clones*. It is known that the pair Pol-Inv defines a Galois connection between the lattice of clones and the lattice of co-clones. In particular the following holds (see [Pip97, PK79, Sze86]):

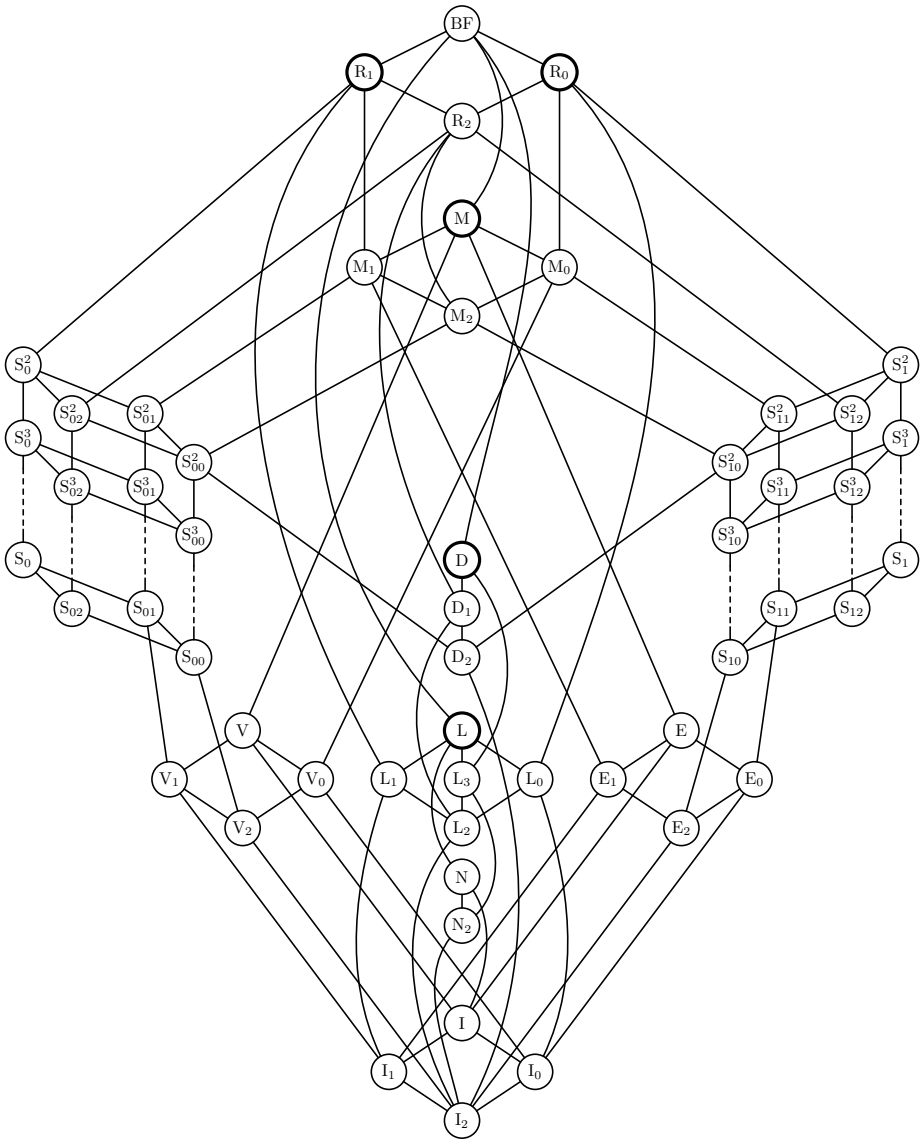

Fig. 1. Graph of all Boolean clones.

Proposition 2.1. *For every set B of Boolean functions and every set Γ of Boolean relations,*

$$\mathrm{Pol}(\mathrm{Inv}(B)) = [B],$$
$$\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \langle\Gamma\rangle.$$

In other words, the clones and relational clones are exactly the fixed points of the operators $\mathrm{Pol} \circ \mathrm{Inv}$ *and* $\mathrm{Inv} \circ \mathrm{Pol}$*, resp.*

In particular, this implies that every clone can be characterized by a set of invariants. For this, let B be a clone and Γ_0 be a basis of $\mathrm{Inv}(B)$. Then, $f \in B$ iff $f \approx R$ for every $R \in \Gamma_0$. Thus, given f and B,

$$f \in [B] \text{ iff } f \approx R \text{ whenever } B \approx R \tag{1}$$

(in the sense that $g \approx R$ for every $g \in B$), and in fact, it is sufficient to restrict ourselves to relations R that appear in some basis of a co-clone, say from the list of basises given in [BRSV05]. In the sequel of this paper we fix the list of basises from that paper, and if we refer to a basis of some co-clone we mean precisely the basis given in that paper. Our results, however, are independent of which basis we use.

This does not immediately lead to an algorithm to decide if $f \in [B]$. The reason is that, first there are infinitely many clones (and hence co-clones), and second some co-clones have only infinite basises. We show how to deal with these problems.

Looking at the lattice of clones, we see that it is finite except for eight infinite chains. Let us say that the classes S_x^m and S_{xy}^m are of type m, the classes S_x and S_{xy} are of type ∞ ($x \in \{0,1\}$, $y \in \{0,1,2\}$), and all remaining classes are of type 0. It is known [BRSV05] that for every clone B of finite type, $\mathrm{Inv}(B)$ has a finite basis. A basis for $\mathrm{Inv}(\mathrm{S}_1^m)$ is $\mathrm{NAND}^m = \{0,1\}^m \setminus \{(1,1,\ldots,1)\}$, a basis for $\mathrm{Inv}(\mathrm{S}_0^m)$ is $\mathrm{OR}^m = \{0,1\}^m \setminus \{(0,0,\ldots,0)\}$, a basis for $\mathrm{Inv}(\mathrm{S}_1)$ is $\left\{ \mathrm{NAND}^m \;\middle|\; m \geq 2 \right\}$, a basis for $\mathrm{Inv}(\mathrm{S}_0)$ is $\left\{ \mathrm{OR}^m \;\middle|\; m \geq 2 \right\}$. For the remaining classes S_{xy}^m and S_{xy} of positive type, a basis is obtained by adding to the basises just given a set of at most 3 fixed (i.e., independent of m, but depending on x, y) functions.

Looking at the inclusions among clones of positive types and their invariants, it is clear that if a function preserves NAND^n then it preserves NAND^{n-1}. More important, but again easy to check, is that for every function $f\colon \{0,1\}^n \to \{0,1\}$,

$$\text{if } f \approx \mathrm{NAND}^n \text{ then } f \approx \mathrm{NAND}^m \text{ for all } m \geq n. \tag{2}$$

Thus we observe that in order to check clone membership according to equation (1) above, only finitely many invariants have to be checked, namely those that appear in the basis of a co-clone related to a clone of type at most the arity of the given function.

The following theorem thus yields an algorithm for membership in Boolean clones.

Theorem 2.2. *Let $f\colon \{0,1\}^n \to \{0,1\}$ and B be a set of Boolean functions. Then $f \in [B]$ if for every clone $\widehat{B}$ of type at most n, $f \in \mathrm{Pol}(\Gamma)$ whenever $B \subseteq \mathrm{Pol}(\Gamma)$, where Γ is the basis of $\widehat{B}$.*

3 Complexity Results

We want to study the computational complexity of the following decision problem, the *uniform clone membership problem*:

Problem:	GEN
Input:	a Boolean function f, a finite set B of Boolean functions
Question:	Is $f \in [B]$?

By our remarks in the preceding section, $[B]$ consists of exactly those functions that can be computed by Boolean circuits with gates in which functions from B are evaluated. The problem GEN, thus, is exactly the computability problem for circuits discussed in the introduction.

Since we want to make use of the algorithm given in Theorem 2.2, we have to determine the complexity of checking invariants. Thus, for every fixed relation R, we have to study the problem

Problem:	$\mathrm{Pol}(R)$
Input:	a Boolean function f
Question:	Is $f \approx R$?

When studying the problems above, an important point to clarify is in what way the input functions f and the elements of B are given. We first consider the case that all Boolean functions are given by their-truth table, i.e., an n-ary function is given as a string of length $N = 2^n$.

Lemma 3.1. *If the input is given in truth-table representation,* $\mathrm{Pol}(R) \in \mathrm{AC}^0$ *for every fixed relation R.*

Proof. Let f be n-ary and R be k-ary. By definition, $f \approx R$ iff

$$\bigwedge_{a_1,\ldots,a_n \in \{0,1\}^k} \Big(\big(\bigwedge_{1 \le i \le n} a_i \in R \big) \implies f^{(k)}(a_1, \ldots, a_n) \in R \Big). \qquad (3)$$

Here, $f^{(k)}$ denotes the component-wise application of f to vectors of length k, i. e., if $a_i = (a_i[1], a_i[2], \ldots, a_i[k])$ for $1 \le i \le n$, then $f^{(k)}(a_1, \ldots, a_n) = \big(f(a_1[1], \ldots, a_n[1]), \ldots, f(a_1[k], \ldots, a_n[k])\big)$. Observe that the unbounded fan-in AND's have a number of input-wires that is polynomial in the input length N, hence we conclude that $\mathrm{Pol}(R) \in \mathrm{AC}^0$. □

The above lemma handles the invariant tests for all clones of type 0. For clones of positive type, the following problem is the cornerstone:

Problem:	Pol-NAND
Input:	a Boolean function f, a number m
Question:	Is m at most the arity of f and $f \approx \mathrm{NAND}^m$?

Lemma 3.2. *If the input is given in truth-table representation, then* Pol-NAND $\in \mathrm{qAC}^0$.

Proof. We use the circuit given in (3) above, with $m = k$ and $R = \mathrm{NAND}^m$. Observe that this time, the top AND-gate ranges over $2^{n \cdot k} = 2^{(\log N)^2}$ values, which is quasipolynomial in the input length. Hence, Pol-NAND is in quasipolynomial-size AC^0. □

Theorem 3.3. *If the input is given in truth-table representation,* GEN $\in \mathrm{qAC}^0$.

Proof. We follow the algorithm given in Theorem 2.2. Thus, our circuit consists essentially of a constant number of subcircuits for $\mathrm{Pol}(R)$ for some relations R plus a linear number of subcircuits for Pol-NAND. By the just given lemmas, this yields a qAC^0-circuit. □

Our algorithm is only a simple modification of an algorithm given by Bergman and Slutzki in [BS00].Their algorithm showed that for truth-table representation, GEN $\in$ NL, and they asked if the problem is actually complete for NL. Theorem 3.3 answers their question negatively: Since the parity function cannot be computed by quasipolynomial-size AC^0 circuits [Yao85] (cf. also [Hås88, Weg87]), parity does not reduce to GEN under any reduction under which qAC^0 is closed such as *plt*-reductions

[Vol98] or qAC^0-Turing-reductions (or any stricter reducibility, e. g., FO-reductions [Imm99], uniform projections [CMTV98] or quantifier-free projections [Imm99]). Hence, GEN cannot be hard under any of these reductions for a class that contains parity such as $AC^0(2)$ or any of its superclasses like NC^1, L, NL, ...

Coming back to the *non-uniform membership problems* studied by Böhler and Schnoor [BS07], i.e., problems of the form

Problem:	MEM(B)
Input:	a Boolean function f
Question:	Is $f \in B$?

for each fixed clone B, we conclude from the above:

Corollary 3.4. *Let B be a Boolean clone, and consider truth-table representations of Boolean functions.*

1. *If B is of finite type, then* $\mathrm{MEM}(B) \in \mathrm{AC}^0$.
2. *If B is of infinite type, then* $\mathrm{MEM}(B) \in \mathrm{qAC}^0$.

Next, we consider the case that the input functions are not given by their truth-table but in a succinct way, i. e., by a Boolean circuit over basis $\{\wedge, \vee, \neg\}$ or any other complete basis. As mentioned in the introduction, Böhler and Schnoor [BS07], studying this kind of input representation, identified a few tractable cases of the non-uniform memberhip problems and showed that all other cases are coNP-complete. The lower bound for the non-uniform problem of course immediately translates to the uniform problem; however, since the algorithms given in [BS07] do not rely on the Galois connection described in Sect. 2 above but make use of particular properties of the individual clones, no upper bound for the uniform membership problem GEN can be obtained from that paper. Translating the methods we used above for truth-table representation to the context of succinct input description, we obtain an upper bound that matches the coNP lower bound.

Theorem 3.5. *If the input is given in circuit representation,* GEN *is* coNP*-complete.*

Proof. As in Theorem 3.3, the algorithm follows Theorem 2.2. For circuit representation, both Pol(R) and Pol-NAND are in coNP, since the AND's in (3) range over a number of values exponential in the input size. Thus we immediately obtain that GEN $\in$ coNP. Hardness follows from [BS07], where it is shown that the membership problem for, say, N_2 is already coNP-hard. □

Finally let us very briefly turn to the case of functions over an arbitrary finite (not necessarily Boolean) domain. The whole machinery of Post's lattice breaks down in this case—in fact, the lattice of clones already over a 3-element universe is uncountable [Pip97]. Bergman et al. [BJS99] show that for arbitrary finite domains, the problem GEN is complete for the class EXPTIME, if the input is given in truth-table representation. Also here in the case of functions over non-Boolean domains, succinct representation by Boolean circuits makes a lot of sense; we assume that elements of a universe of size s are encoded in the usual way by sequences of $\lceil \log s \rceil$ bits.

Theorem 3.6. *If the input is given in circuit representation, the problem* GEN *for functions over arbitrary finite domain is complete for the class* $\mathrm{EXP}^2\mathrm{TIME} = \mathrm{DTIME}(2^{2^{n^{O(1)}}})$.

Proof. A careful reading of the completeness proof given in [BJS99] shows that, if the input is given by function tables, GEN over arbitrary finite domains is complete for the class EXPTIME under polylogarithmic-time bit-reductions $\leq_{plt}$. A general translational result by Veith [Vei98] states that, if a problem A is complete under $\leq_{plt}$ for a complexity class $\mathcal{C}$, the succinct version of A is complete for the class $\mathrm{BLeaf}^{\mathrm{P}}(\mathcal{C})$ of all languages that can be accepted by nondeterministic polynomial-time machines with leaf languages from $\mathcal{C}$. Our claim follows since $\mathrm{BLeaf}^{\mathrm{P}}(\mathrm{EXPTIME}) = \mathrm{EXP}^2\mathrm{TIME}$ [HLS$^+$93]. □

Acknowledgement

This research was performed while the author visited the Isaac Newton Institute in Cambridge for the programme on "Logic and Algorithms". He wishes to thank Anuj Dawar–Cambridge and Moshe Vardi–Houston for inviting him to this event, and in particular Moshe Vardi for pointing him to the paper by Bergman and Slutzki [BS00] and for helpful discussions on the subject of this paper. The author also thanks Henning Schnoor–Hannover for explaining the results of [BS07].

References

[BCRV03] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part I: Post's lattice with applications to complexity theory. *ACM-SIGACT Newsletter*, 34(4):38–52, 2003.

[BCRV04] E. Böhler, N. Creignou, S. Reith, and H. Vollmer. Playing with Boolean blocks, part II: Constraint satisfaction problems. *ACM-SIGACT Newsletter*, 35(1):22–35, 2004.

[BJS99] C. Bergmann, D. Juedes, and G. Slutzki. Computational complexity of term equivalence. *International Journal of Algebra and Computation*, 9:113–128, 1999.

[BRSV05] E. Böhler, S. Reith, H. Schnoor, and H. Vollmer. Bases for Boolean co-clones. *Information Processing Letters*, 96:59–66, 2005.

[BS00] C. Bergman and G. Slutzki. Complexity of some problems concerning varieties and quasi-varieties of algebras. *SIAM Journal on Computing*, 30(2):359–382, 2000.

[BS07] E. Böhler and H. Schnoor. The complexity of the descriptiveness of boolean circuits over different sets of gates. *Theory of Computing Systems*, 2007. To appear.

[CMTV98] H. Caussinus, P. McKenzie, D. Thérien, and H. Vollmer. Nondeterministic NC^1 computation. *Journal of Computer and System Sciences*, 57:200–212, 1998.

[Coh65] P. M. Cohn. *Universal Algebra*. Harper's Series in Modern Mathematics. Harper & Row Publishers, New York, Evanston, and London, 1965.

[Hås88] J. Håstad. *Computational Limitations of Small Depth Circuits*. MIT Press, Cambridge, 1988.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings 8th Structure in Complexity Theory*, pages 200–207, 1993.

[Imm99] N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science. Springer Verlag, New York, 1999.

[Pip97] N. Pippenger. *Theories of Computability*. Cambridge University Press, Cambridge, 1997.

[PK79] R. Pöschel and L. A. Kalužnin. *Funktionen- und Relationenalgebren*. Deutscher Verlag der Wissenschaften, Berlin, 1979.

[Pos20] E. L. Post. Determination of all closed systems of truth tables. *Bulletin of the AMS*, 26:437, 1920.

[Pos21] E. L. Post. Introduction to a general theory of elementary propositions. *American Journal of Mathematics*, 43:163–185, 1921.

[Pos41] E. L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Sze86] Á. Szendrei. *Clones in Universal Algebra*. Séminaire de mathématiques supérieures, NATO Advanced Study Institute. Les Presses de l'Université de Montréal, Montréal, 1986.

[Vei98] H. Veith. Succinct representation, leaf languages, and projection reductions. *Information & Computation*, 142:207–236, 1998.

[Vol98] H. Vollmer. Relating polynomial time to constant depth. *Theoretical Computer Science*, 207:159–170, 1998.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.

[Weg87] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner series in computer science. B. G. Teubner & John Wiley, Stuttgart, 1987.

[Yao85] A. C. C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings 26th Foundations of Computer Science*, pages 1–10. IEEE Computer Society Press, 1985.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/