Rudimentary reductions revisited

Eric Allender*
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Vivek Gore†
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

April 23, 1997

Abstract: We show that log-bounded rudimentary reductions (defined and studied by Jones in 1975) characterize Dlogtime-uniform AC^0.

1 Introduction

In the first part of the oft-cited paper [Jon75], Jones introduced logspace reductions as a tool for studying the relative complexity of problems in P. In the second part of [Jon75], Jones introduced a restricted version of logspace reducibility, called log-bounded rudimentary reductions. The motivation for introducing this restriction came from (1) the desire to have a tool for talking about the structure of very small complexity classes such as DSPACE($\log n$), and (2) interest in generalizing the notion of "rudimentary relations," which at that time was the object of a considerable amount of attention [Smu61, Sal73, Wra78, Wil79, PD80]. In [Jon75], Jones went on to show that a number of problems were complete for various complexity classes under log-bounded rudimentary reductions. Although logspace reducibility has proved to be a very useful notion in complexity theory in the intervening years, log-bounded rudimentary reducibility has been mentioned explicitly only seldom in subsequent work – although, as we demonstrate, it has been considered implicitly many times, under different names.

A great deal of insight about the complexity of various problems has been gained by the study of uniform circuit complexity; many complexity classes have been characterized in terms of the computational power of families of circuits $\{C_n : n \in \mathbf{N}\}$ (where C_n takes inputs of length n), by varying the types of gates and allowable fan-in on the circuits, and imposing a "uniformity condition" which specifies that the definition of C_n in terms of n be "easy" in some sense. Unfortunately, the question of

*Supported in part by NSF grant CCR-9000045.
†Supported in part by a DIMACS research assistantship.

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC-88-09648; and also receives support from the New Jersey Commission on Science and Technology.

which uniformity condition to use has remained somewhat controversial. For example, although much work on uniform circuit complexity uses logspace uniformity, it is not clear that this is appropriate when defining subclasses of DSPACE($\log n$), such as AC^0, the class of languages accepted by polynomial-size, constant-depth circuits of unbounded fan-in AND and OR gates.

Competing definitions of "uniform AC^0" were proposed by Immerman [Imm87, Imm89] and Buss [Bus87]. Then it was shown by Barrington, Immerman, and Straubing, that these definitions in fact coincide [BIS90]; they show that Dlogtime-uniform AC^0 may be defined alternatively in terms of first-order logic, $O(1)$ time on a CRAM, inductive definitions with $O(1)$ inductive depth, and Sipser's log-time hierarchy [Sip83]. Other characterizations of Dlogtime-uniform AC^0 are found in [Clo90]. Since some of these notions were defined entirely independently of each other and independently of considerations of uniform circuit complexity, the fact that these notions coincide is taken as evidence for the "correctness" of the Dlogtime-uniformity condition when choosing a definition of uniform AC^0.

In this paper, we add to this body of evidence by showing that Jones' logspace rudimentary reductions provide yet another characterization of Dlogtime-uniform AC^0. **(Throughout the rest of this paper, all references to AC^0 will denote Dlogtime-uniform AC^0.)**

2 Preliminaries

Definition 1 The following definitions are due to Jones [Jon75].

- Let Σ be an alphabet. The ternary predicate $\sigma(x, i, a)$ is true iff a is the i^{th} symbol of x where $x \in \Sigma^*$.
- For a positive integer c and a space bound S, define the predicates $P_{c,S}$, $\overline{P}_{c,S}$ as follows:

$$P_{c,S}(x, u, v, w) \Leftrightarrow uv = w \wedge |uvw| \leq c \cdot S(|x|), \overline{P}_{c,S}(x, u, v, w) \Leftrightarrow uv \neq w \wedge |uvw| \leq c \cdot S(|x|).$$

- The class of $S(\cdot)$-bounded rudimentary predicates, RUD_S is the class of predicates built from $\sigma, \neg\sigma, P_{c,S}$, and $\overline{P}_{c,S}$ for every c via logical connectives, explicit transformations, and $S(\cdot)$-bounded existential and universal quantification. See [Jon75] for a complete definition.

Definition 2 A function $f : \Sigma^* \to \Sigma^*$ is $S(\cdot)$-bounded rudimentary iff the predicate R defined by $R(x, i, a) \Leftrightarrow \sigma(f(x), i, a)$ is in RUD_S.

That is, f is $S(\cdot)$-bounded rudimentary iff the predicate "a is the i^{th} symbol of $f(x)$" is in RUD_S. The fact that $|i| \leq c \cdot S(|x|)$ for some c and all x implies that $|f(x)| \leq 2^{c \cdot S(|x|)}$.

Note that, as defined above, RUD_S is a set of *predicates*; however we can just as easily view RUD_S as a set of *languages*. A language L will be said to be in RUD_S iff the characteristic function of L is $S(\cdot)$-bounded rudimentary. Alternatively, each predicate $Q \in \text{RUD}_S$ over alphabet Σ defines a language L

over alphabet $\Sigma \cup \{,\}$ (where "," is any symbol not in Σ) defined by $L = \{x, y_1, \ldots, y_m : Q(x, y_1, \ldots, y_m)\}$. These alternative definitions are easily seen to be equivalent. For more formal arguments along these lines, see [Wra78].

In this paper (as in [Jon75]) we are interested primarily in $\mathrm{RUD}_{\log}$, the class obtained when the space bound $S(\cdot)$ is logarithmic. In Section 3, we show that $\mathrm{RUD}_{\log} = \mathrm{AC}^0$.

The proofs in this paper will not make use of the circuit formalism; thus detailed definitions of the Dlogtime-uniformity conditions are omitted here. (They may be found in [BIS90, BCGR90].) Instead, we will use two characterizations of AC^0 that were shown to be equivalent in [BIS90] (see also [BCGR90]), one in terms of alternating Turing machines, and one in terms of first-order logic.

We use the model of alternating Turing machine used by Ruzzo [Ruz81] in which access to the input is provided via a special tape which we call the *input address tape* onto which an address i may be written in binary, following which in unit time the i^{th} input symbol is available. This convention is necessary to allow machines with sublinear running times to have access to all of the input. Consult [BIS90, BCGR90] for examples illustrating how these machines compute.

Now let $\Sigma_k\mathrm{time}(\log n)$ denote the class of languages accepted by alternating Turing machines running in $O(\log n)$ time, beginning in an existential configuration, and making at most $k \Leftrightarrow 1$ alternations between existential and universal configurations on any computation path. Let LH (the logtime hierarchy of [Sip83]) denote $\bigcup_k \Sigma_k\mathrm{time}(\log n)$.

Immerman [Imm87, Imm89] has studied the expressive power of first-order logic in defining languages; he calls the resulting class FO. Due to space limitations, we refer the reader to [Imm87, Imm89] for the relevant definitions.

Theorem 3 [BIS90] $\mathrm{AC}^0 = \mathrm{LH} = FO$.

Two minor complications arise from the statement of Theorem 3:

1. Alternating Turing machines can accept input over any alphabet Σ, whereas FO and AC^0 are classes of languages over the alphabet $\{0, 1\}$. Thus we simply assume a binary encoding of any other alphabet. (Alternatively, the FO and circuit frameworks can be modified to allow arbitrary alphabets.)

2. It is clear what it means for a function to be computed by an AC^0 circuit, but it is less clear how a function can be specified in the LH or FO settings. To resolve this difficulty, we use a standard technique that has been used since at least [Jon75]. For a function f with polynomial growth rate[1], let $A_f(c, i, z) \stackrel{\text{def}}{=}$ the i^{th} symbol of $f(z)$ is c. Then we say that f is in LH (or FO) if A_f is in LH

[1] That is, there is some polynomial p such that for all x, $|f(x)| \leq p(|x|)$.

(or FO). It has been noticed by many authors (e.g., [BIS90]) that f is computed by AC^0 circuits iff $A_f \in \mathrm{AC}^0$.[2]

Note that the composition of two functions in AC^0 is also a function in AC^0.

3 Equivalence of $\mathbf{AC}^0$ and $\mathrm{RUD}_{\log}$

Theorem 4 $\mathrm{AC}^0 = \mathrm{RUD}_{\log}$.

Proof.

($\supseteq$) This containment is mentioned in Proposition 5 of [Vol84] (where the converse containment is left as an open question). We present a complete proof here.

The proof proceeds by induction on the definition of $\mathrm{RUD}_{\log}$, showing that each language in $\mathrm{RUD}_{\log}$ is in LH. The input to an ATM is assumed to have both ends marked by the special symbol \$.

Consider the inductive definition of $\mathrm{RUD}_{\log}$:

1. $\sigma(x,i,a)$ and $\neg\sigma(x,i,a)$ are in $\mathrm{RUD}_{\log}$.

   Let $L = \{x,i,a : \sigma(x,i,a)\}$. Consider an alternating TM M for L:

   On input $\$x,i,a\$$,

   - M existentially guesses the positions of all the commas (there are only a constant number of them).
   - M universally verifies that that the guessed positions contain the appropriate markers and the other positions contain 0's or 1's.
   - Using the marker positions, M calculates $|x|$ and $|i|$. It was observed in [Bus87] that this can be done in logarithmic time. M calculates $\log(|x|)$ from $|x|$ and verifies that $|i| \leq \log(|x|)$.
   - M existentially guesses the bits of i and writes them on a worktape, and then universally checks that for each $j \leq i$, the guessed value for position $|x|+1+j$ is correct.
   - M again uses marker positions to read a, and check that the i^{th} bit of x is a.

2. If c is a positive integer then both $P_{c,\log}(x,u,v,w)$ and $\overline{P}_{c,\log}(x,u,v,w)$ are in $\mathrm{RUD}_{\log}$ where

$$P_{c,\log}(x,u,v,w) \Leftrightarrow uv = w \wedge |uvw| \leq c \cdot \log(|x|),$$
$$\overline{P}_{c,\log}(x,u,v,w) \Leftrightarrow uv \neq w \wedge |uvw| \leq c \cdot \log(|x|).$$

[2] Again, for equivalence to hold, the output of the circuit must be interpreted in such a way as to allow an "endmarker," or else all functions f in AC^0 must have $|f(x)| = |f(y)|$ if $|x| = |y|$, which is clearly not the case for some functions in LH and FO.

Consider an ATM M that behaves as follows:

On input $\$x, u, v, w\$$,

- M first guesses and verifies the positions of all the markers as in the previous case, and uses this to compute $c \cdot \log |x|$.
- Using the marker positions, determine whether $|uvw| \leq c \cdot \log(|x|)$, and if so, read each of u, v, w from the input, as in the previous case, and verify whether $uv = w$ (or $uv \neq w$ for $\overline{P}_{c,\log}$).

3. If $Q(x, y_1, \ldots, y_m)$ and $R(x, y_1, \ldots, y_m)$ are in $\mathrm{RUD}_{\log}$, then so are
$Q(x, y_1, \ldots, y_m) \wedge R(x, y_1, \ldots, y_m)$ and $Q(x, y_1, \ldots, y_m) \vee R(x, y_1, \ldots, y_m)$.

   This case is trivial, using universal and existential branching, respectively.

4. If $Q(x, z_1, \ldots, z_m)$ is in $\mathrm{RUD}_{\log}$ and each of $\xi_1, \ldots, \xi_m$ is either a string in Σ^* or one of the variables $y_1, \ldots, y_r$, then the predicate R defined by
$R(x, y_1, \ldots, y_r) \Leftrightarrow Q(x, \xi_1, \ldots, \xi_m)$ is in $\mathrm{RUD}_{\log}$.

   Inductively, Q can be accepted by an alternating TM M in $\bigcup_k \Sigma_k \mathrm{Time}(\log n)$. Note that r and m are constants here and $\xi_1, \ldots, \xi_m$ depend only on $y_1, \ldots, y_r$ and not on x. Therefore, the conversion of $y_1, \ldots, y_r$ into $\xi_1, \ldots, \xi_m$ only requires a finite amount of information that can be supplied to an ATM in its finite control.

   Consider a function f that converts strings of the form $x, y_1, \ldots, y_r$ into $x, \xi_1, \ldots, \xi_m$. If we can show that f can be computed in LH, then we can use the fact that LH functions are closed under composition to conclude the proof for this case, using the inductive hypothesis. That is, we need to show that the language $A_f = \{c\#i\#z : \text{the } i^{\text{th}} \text{ symbol of } f(z) \text{ is } c\}$ is in LH. (Note that, for well-formed inputs, z will be of the form $x, y_1, \ldots, y_r$.)

   Consider an ATM M that behaves as follows:

   On input $\$c\#i\#z\$$

   - M guesses all the marker positions in $\$c\#i\#z\$$ as well as in z itself (there are a constant number of these) and verifies them as in the previous cases.
   - M uses the marker positions to compute $|x|$ and $|i|$, and rejects if $|i|$ is too long. Otherwise, as in the previous cases, M guesses the contents of the c, i, and $y_1, \ldots, y_r$ fields of the input, and checks that the guesses are correct.

- If $i \leq |x|$, accept iff the i^{th} bit of x is c. Otherwise, compute $\xi_1, \ldots, \xi_m$ and accept iff c is in position $i \Leftrightarrow |x|$ of $\xi_1, \ldots, \xi_m$. (It is easy to verify that there is ample time to perform these operations.)

5. If c is a positive integer, and $Q(x, y_1, \ldots, y_{m+1})$ is in $\text{RUD}_{\log}$, then the predicates R and R' defined as follows are also in $\text{RUD}_{\log}$

$$R(x, y_1, \ldots, y_m) \Leftrightarrow (\exists z)[|z| \leq c \cdot \log(|x|) \wedge Q(x, y_1, \ldots, y_m, z)]$$
$$R'(x, y_1, \ldots, y_m) \Leftrightarrow (\forall z)[|z| \leq c \cdot \log(|x|) \Rightarrow Q(x, y_1, \ldots, y_m, z)]$$

Inductively, Q is accepted by some LH machine M. We construct a machine M' accepting R'. On input $\$x, y_1, \ldots, y_m\$$, M' starts off by getting the marker positions as in the previous cases. Then it universally verifies that for all z with $|z| \leq c \cdot \log(|x|))$, M accepts $\$x, y_1, \ldots, y_m, z\$$. To see that this is possible, note that M consults its input only at the end of a computation branch. As soon as M goes into input mode and has j written on its index tape, M' either consults the j^{th} symbol of its input, or reads the symbol in position $j \Leftrightarrow |x, y_1, \ldots, y_m|$ of z.

A similar argument, using existential branching, shows that R is in LH.

($\subseteq$) To prove this inclusion, let $L \in FO$. Therefore, there exists a sentence in Prenex Normal Form

$$\eta = (Q_1 j_1)(Q_2 j_2) \ldots (Q_m j_m)\, \phi(j_1, \ldots, j_m)$$

such that $\forall x$

$$x \in L \Leftrightarrow x \models (Q_1 j_1)(Q_2 j_2) \ldots (Q_m j_m)\, \phi(j_1, \ldots, j_m)$$

where each Q_i is a quantifier ($\exists$ or $\forall$).

We will show that for each such η (and its associated ϕ) there is a $\text{RUD}_{\log}$ predicate R_ϕ such that for each string x,

$$x \models (Q_1 j_1)(Q_2 j_2) \ldots (Q_m j_m)\, \phi(j_1, \ldots, j_m)$$
$$\text{if and only if}$$
$$(Q'_1 y_1)(Q'_2 y_2) \ldots (Q'_m y_m)\, R_\phi(x, y_1, \ldots, y_m)$$

where each Q'_i ranges over all binary strings of length at most $\lceil \log |x| \rceil + 1$ and is the same type (existential or universal) as Q_i. By the closure properties of $\text{RUD}_{\log}$, and by the fact that the length of each y_i is bounded, it follows that since R_ϕ is in $\text{RUD}_{\log}$, L is also in $\text{RUD}_{\log}$, and the proof is complete. It remains only to construct R_ϕ.

The formula ϕ is built up from the primitive predicates $=, <$, X, and BIT, and from the constants 1 and n. Clearly, it will suffice to show that each of these primitive predicates is "equivalent" in the

same sense to a relation in $\mathrm{RUD}_{\log}$. For example, we will show that there is a relation S in $\mathrm{RUD}_{\log}$ such that, for any numbers j_1 and j_2, the primitive predicate $j_1 < j_2$ is true iff the relation $S(x, y_1, y_2)$ holds, where y_1 and y_2 are binary strings of length at most $\lceil \log |x| \rceil + 1$ representing the numbers j_1 and j_2, respectively.

The constant 1 is explicitly definable in $\mathrm{RUD}_{\log}$, and Jones [Jon75] has already observed that the relations $u + v = w, u \cdot v = w, u^v = w$ and $|u| = w$ are all in $\mathrm{RUD}_{\log}$, so long as u, v, w are constrained to have length $\leq c \cdot \log(|x|)$ for some c. Thus the constant n is definable, since the relation $|x| = j$ is equivalent to $(\sigma(x, j, 0) \vee \sigma(x, j, 1)) \wedge \neg(\sigma(x, j+1, 0) \vee \sigma(x, j+1, 1))$. It follows easily that the following set is in $\mathrm{RUD}_{\log}$: $\{x, y_1, \ldots, y_m$: for each i, $|y_i| = \lceil \log |x| \rceil + 1\}$. Therefore for the rest of this proof, we will assume that all variables y_i refer to strings of exactly this length. (This will result in a few simplifications.)

Now we consider each of the primitive predicates in turn.

- $(j_1 = j_2) \leftrightarrow (\forall i)[\sigma(y_1, i, 0) \Leftrightarrow \sigma(y_2, i, 0)]$
- $(j_1 < j_2) \leftrightarrow (\exists u)(\exists v)(\exists w)[P_{1,\log}(x, u, v, y_1) \wedge P_{1,\log}(x, u, w, y_2) \wedge \sigma(v, 1, 0) \wedge \sigma(w, 1, 1)]$
  This says that the binary representation of j_1 is uv and the binary representation j_2 is uw such that the most significant bits of v and w are 0 and 1 respectively. Note that the values taken by each of the variables u, v, w have lengths $\leq \log(|x|)$.
- $X(j_1) \leftrightarrow \sigma(x, y_1, 1)$.
- $BIT(j_1, j_2) \leftrightarrow \sigma(y_2, y_1, 1)$.

This completes the proof. ∎

We remark that standard translational techniques may now be used to prove the following generalization of Theorem 4. Note that for the case of $S(n) = n$, this is the well-known theorem of Wrathall [Wra78].

Corollary 5 For functions S such that the binary representation of $S(n)$ can be computed from n in time $O(S(n))$, $\mathrm{RUD}_S = \bigcup_k \Sigma_k \mathrm{time}(S(n))$.

4 Reducibilities

Jones defined $\mathrm{RUD}_{\log}$ as a first step towards defining a very restrictive notion of reducibility. Jones took the familiar notion of a many-one reduction (namely, f reduces A to B if $x \in A \Leftrightarrow f(x) \in B$), and imposed the restriction that f be log-bounded rudimentary. There was one additional restriction that Jones imposed in defining his logspace-rudimentary reducibility (denoted $\leq_{\log}^{\mathrm{rud}}$); he required that there

be some constant c such that for all x, $\log|f(x)| = c \cdot \log(|x|)$. That is, the length of $f(x)$ should be *equal* to a polynomial in $|x|$, and not just bounded by one. In [Jon75], Jones comments that this final restriction is stronger than the more natural restriction of simply having f have polynomial growth rate, but explains that it "seems to be necessary" in order to have the $\leq^{\text{rud}}_{\log}$ relation be transitive (i.e., in order for the composition of two log-bounded rudimentary functions to be log-bounded rudimentary). However by Theorem 4 we see that for functions f with polynomial growth rate, f is log-bounded rudimentary iff f is in LH. Since the composition of two functions in LH is easily seen to be in LH, it follows that Jones' rationale for imposing the stronger length requirement was unfounded.

5 Conclusion

We have considered the log-bounded rudimentary predicates defined by Jones in [Jon75] and we have shown that they correspond exactly to the class of sets accepted by Dlogtime-uniform AC^0 circuits. Thus Jones was probably the first to study this complexity class, which has loomed large in importance in recent years. We believe that this augments the (already compelling) arguments of [BIS90] in favor of using the Dlogtime-uniformity condition when studying small circuit complexity classes.

It is instructive to note some of the open questions posed by Jones in [Jon75]. He mentions on more than one occasion the question of whether or not $\text{RUD}_S = \text{DSPACE}(S(n))$, and he explicitly considers the possibility that $\text{RUD}_{\log} = \text{NSPACE}(\log n)$ ([Jon75], Theorem 23). However, the results of [Ajt83] and [FSS84] refute this possibility by showing that PARITY is in $\text{NSPACE}(\log n) \Leftrightarrow \text{AC}^0$. The strongest results along this line are those of [Hås87] and [Yao85], showing that for any function S such that for all ϵ, $S(n) = o(n^\epsilon)$, the PARITY language is in $\text{DSPACE}(1) \Leftrightarrow \text{RUD}_S$. (Here, DSPACE(1) denotes the class of regular sets.) However, any generalization of this result for larger functions S will require entirely different techniques and will have to take uniformity into account in some way, since the techniques of [Nep70] (see also [Vol84]) show that for each $\epsilon > 0$, $\text{NSPACE}(\log n) \subseteq \text{RUD}_{n^\epsilon}$, and it follows from [Wra78] that RUD_{n^ϵ} contains complete sets for each level of the polynomial hierarchy. Thus the question of whether or not $\text{RUD}_{n^\epsilon} = \text{DSPACE}(n^\epsilon)$ remains an interesting open problem. (Note, however, that an affirmative answer would imply that the polynomial hierarchy is equal to PSPACE.)

References

[Ajt83] M. Ajtai. $\sum^1_1$-formulae on finite structures. *Annals of Pure and Applied Logic* 24 (1983) 1–48.

[BIS90] D. A. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences* 41 (1990) 274–306.

[Bus87] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th ACM Symposium on Theory of Computing*, 1987, pp. 123–131.

[BCGR90] S. Buss, S. Cook, A. Gupta, and V. Ramachandran. An optimal parallel algorithm for formula evaluation. Submitted, 1990.

[Clo90] P. Clote. Bounded arithmetic and computational complexity. In *Proc. 5th Structure in Complexity Theory Conference*, 1990, pp. 186–199.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory* 17 (1984) 13–27.

[Hås87] J. Håstad. Computational limitations for small depth circuits. MIT Press, Cambridge, 1987.

[Imm87] N. Immerman. Expressibility as a complexity measure: Results and directions. In *Proc. 2nd Structure in Complexity Theory Conference*, 1987, pp. 194–202.

[Imm89] N. Immerman. Expressibility and parallel complexity. *SIAM Journal on Computing* 18 (1989) 625–638.

[Jon75] N. D. Jones. Space-bounded reducibility among combinatorial problems. *Journal of Computer and System Sciences* 11 (1975) 68–85. Corrigendum: *Journal of Computer and System Sciences* 15 (1977) 241.

[Nep70] V.A. Nepomnjaščiĭ. Rudimentary predicates and Turing calculations. *Soviet Math. Dokl.* 11 (1970) 1462–1465.

[PD80] J.B. Paris and C. Dimitracopoulos. Truth definitions for Δ_0 formulae. In *Logic and Algorithmic: an International Symposium Held in Honour of Ernst Specker, Monographie no. 30 de l'Enseignement Mathématique*, 1980, pp. 317–329.

[Ruz81] W.L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences* 21 (1981) 365–383.

[Sal73] A. Salomaa. Formal Languages. Academic Press, 1973.

[Sip83] M. Sipser. Borel sets and circuit complexity. In *Proc. 15th ACM Symposium on Theory of Computing*, 1983, pp. 61–69.

[Smu61] R. Smullyan. Theory of formal systems. In *Annals of Math. Studies* 47. Princeton University Press, 1961.

[Vol84] H. Volger. The role of rudimentary relations in complexity theory. *Bulletin de la Societé Mathématique de France*, series 2, number 16 (1984) pp. 41–51.

[Wil79] A. Wilkie. Applications of complexity theory to Σ_0-definability problems in arithmetic. Lecture Notes in Mathematics vol. 834, 1979, pp. 363–369.

[Wra78] C. Wrathall. *Rudimentary predicates and relative computation. SIAM Journal on Computing* 7 (1978) 194–209.

[Yao85] A. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th IEEE Symposium on Foundations of Computer Science*, 1985, pp. 1–10.