Nondeterminism Within P

Jonathan F. Buss[*] Judy Goldsmith[†]
University of Waterloo Dartmouth College

Abstract

Classes of machines using very limited amounts of nondeterminism are studied. The P =? NP question is related to questions about classes lying within P. Complete sets for these classes are given.

AMS(MOS) Subject Classifications: 68Q15, 68Q05.
Key Words: nondeterminism, quasilinear time, computational complexity.

[*]Supported in part by a grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada. Address: Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

[†]Supported in part by National Science Foundation grant number RII-9003056. Current address: Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2.

1 Introduction

Traditional complexity theory gives a distinguished role to the class P of languages that are computable in polynomial time. Languages not in P are considered to be computationally intractable. However, the converse assertion, that languages in P have efficient solutions, is untenable. Historically, one reason for the choice of polynomial time bounds was that no smaller class appears to be as robust against change of model. We consider here subclasses of P, defined (roughly) according to the exponent of the polynomial bounding the running time and also according to the small amount of nondeterminism allowed. Of particular interest is the class of languages computable in deterministic "quasilinear" time, which is less robust than P with respect to change of model, but is much closer to being a practical notion of feasible computation. We develop a complexity theory based on quasilinear-time many-one reducibility that is analogous to the theory of polynomial-time reducibility.

We primarily consider the trade-off between time and nondeterminism. To replace nondeterminism by time in Turing machine computations is easy: one binary nondeterministic choice may be eliminated by doubling the time. Two fundamental questions are whether nondeterminism can be eliminated at less cost in time, and whether computation time can be decreased by adding nondeterminism. We define subclasses of P by limiting the time bound to fixed degree polynomials, and allowing nondeterminism over a fixed, polynomial-size search space. The inclusion relationships among the classes are particular cases of the two fundamental questions. We show that sufficient separation among these classes lying inside P implies that $P \neq NP$.

Let f and g be integer functions. We will write "$f \in Q(g)$" and say "f is of quasi-order g" if there is a positive constant k such that[1] $f \in O(g \log^k g)$. For any positive constant l, we denote by P_l the class of languages that can be accepted by a multitape Turing machine whose running time on inputs of length n is of quasi-order n^l. For integers $l \geq 1$ and $m \geq 0$, we denote by $N^m P_l$ the class of languages accepted by nondeterministic machines in time of quasi-order n^l making at most $m \log n$ binary nondeterministic choices (all logarithms are base 2). For any m and l, $N^m P_l$ is contained in P_{m+l}. We will also consider the classes NP_l having no bound on the amount of nondeterminism. The classes P_1 and NP_1 have been previously studied by Schnorr [16] under the names QL and NQL.

The $\log^k n$ factors in the allowed time bounds have a number of important con-

[1]We will write "$\log n$" to mean $\lfloor \log_2(n+1) \rfloor$ throughout the paper.

sequences. First, the classes are independent of the number of tapes of a multitape Turing machine. Second, quasilinear-time reducibility is transitive and preserves membership in $N^m P_l$, for any m and l. Third, quasilinear time is large enough to admit basic algorithmic techniques such as sorting and multiplication. For computation of practical importance, the input size is generally less than, say, 10^{12}; hence $\log n$ is bounded by 40, and ignoring arbitrary logarithmic factors is no worse than ignoring arbitrary constant factors.

The notion of classifying problems according to the amount of nondeterminism required has appeared previously; see, e.g., Kintala and Fischer [13] and Wolf [17]. Àlvarez, Díaz and Torán [4, 8] consider a hierarchy, similar to the present one, of classes lying between P and NP. Their hierarchy differs from the present one in two ways. First, any polynomial is allowed in the running times, and second, the bound on the number of nondeterministic choices that defines each class is a power of $\log n$ rather than a multiple. Thus each class contains all of P, and the hierarchy does not provide a classification among computations feasible in practice.

More similar in method to the present paper is the work of Geske [9], who considers relations among classes $TIME(n^l)$ and $NTIME(n^l)$, which are similar to P_l and NP_l, but more restrictive. Weaker variants of our Theorems 12 and 13 are obtained.

In a related work, Kaye [12] gives two logical characterizations of a family of classes $L^{(\alpha)}$, $\alpha \geq 0$, lying between linear time and P. Kaye's class L^2 is precisely quasi-linear time.

Gurevich and Shelah [10] consider quasilinear-time reductions among various models of random-access machines. They define *nearly linear time*, or NLT, and show that the models they consider, including Kolmogorov machines, Schönhage machines and random-access Turing machines, all compute the same NLT functions in quasilinear time. They also show that the languages accepted nondeterministically by any of these machines in quasilinear time are precisely those languages accepted nondeterministically in quasilinear time by a multitape Turing machine (denoted NQL in [16], herein NP_1). They conjecture that the deterministic classes NLT and QL (herein P_1) are different. We offer some candidate languages for this separation.

Quasilinear-time reductions have been studied in other contexts as well. Hunt and Stearns [11] define the notion of Turing-SAT-hard($n \operatorname{polylog} n, n$) in order to make precise the notion that sets in NP have deterministic time and space complexity no less than that of SAT. Their definition restricts the reductions considered to quasilinear time *and* linear space complexity. Because the sets they consider are in $NP - P$ unless $P = NP$, their results do not provide much insight into the classes considered here.

Abrahamson et al. [1, 2] study the complexity of families of languages parameterized by the size of an associated search space. The present paper concerns the complexity of individual problems, which may or may not be part of a larger family. The relationship between the two approaches is investigated in Section 4 below.

This paper is organized as follows. Section 2 gives some basic properties of the classes N^mP_l and P_l. Section 3 shows that the classes have complete sets, and some natural problems are considered as candidate complete sets. Section 4 contains a discussion of the relationship of the present work to that of Abrahamson, et al. Section 5 investigates relationships among the various classes N^mP_l. Section 6 concludes with some open problems and directions for future work.

2 Preliminaries

The classes N^mP_l can be characterized in terms of witnesses, analogously to NP.

Theorem 1 *For all m and l, a set A is in N^mP_l if and only if there is an R in P_l such that for all strings x, $x \in A \Leftrightarrow \exists y\,[|y| \leq m \log|x| \wedge R(x,y)]$.*

The proof is left to the reader.

Like most hierarchies of complexity classes, the limited-nondeterminism classes exhibit downward separation (upward collapse).

Theorem 2 *For all m and l,*

1. *if $N^1P_l \subseteq P_l$, then $N^1P_k \subseteq P_k$ for all $k \geq l$, and*
2. *if $N^{m+1}P_l \subseteq N^mP_l$, then $N^kP_l \subseteq N^mP_l$ for all $k \geq m$.*

Proof (sketch). Suppose that $N^1P_l \subseteq P_l$, and fix $k > l$ and $A \in N^1P_k$. Let R be as guaranteed in the previous result. Let $A' = \{\, x0^{\lfloor |x|^{k/l} \rfloor - |x|} \mid x \in A \,\}$. Then $x0^j \in A'$ if and only if $j = \lfloor |x|^{k/l} \rfloor - |x| \wedge \exists y\, |y| \leq \log|x| \wedge R(x,y)$ if and only if $\exists y\, |y| \leq \log|x0^j| \wedge R'(x0^j, y)$, where $R' = \{\, \langle x0^j, y\rangle \mid j = \lfloor |x|^{k/l} \rfloor - |x| \wedge y \leq \log|x| \wedge R(x,y) \,\}$. Hence $A' \in N^1P_l \subseteq P_l$, and thus $A \in P_k$.

Now suppose that $N^{m+1}P_l \subseteq N^mP_l$, and fix $k > m+1$ and $B \in N^kP_l$. There exists S such that $x \in B \Leftrightarrow \exists y\, |y| \leq k \log|x| \wedge \langle x, y\rangle \in S$. The latter condition is equivalent to $\exists y\, |y| \leq (k-m-1)\log|x| \wedge S'(x,y)$, where $S' = \{\, \langle x,y\rangle \mid \exists z\, z \leq (m+1)\log|x| \wedge \langle x, yz\rangle \in S \,\}$ is in $N^{m+1}P_l$ and hence in N^mP_l by assumption. Thus $B \in N^{k-1}P_l$, which is contained in N^mP_l by induction. ∎

The hierarchy also exhibits a kind of upward separation.

Theorem 3 *Suppose that for all l, there is an m such that N^mP_1 is not contained in P_l. Then $P \neq NP$.*

The result follows immediately from Theorem 4.

Theorem 4 *If $P = NP$, then there is an l such that $NP_k \subseteq P_{kl}$ for all k.*

PROOF. Let L be a complete set for NP_1 with respect to quasilinear-time reductions. (Schnorr has shown that *SAT*, the set of satisfiable Boolean formulas, is such a set [16].) Suppose $P = NP$; i.e., L is in P_l for some l. Let $A \in NP_k$. Membership questions of length n about A are reducible in time $Q(n^k)$ to membership questions of length $Q(n^k)$ about L (using a padding argument and Schnorr's result), which are answerable in time $Q(n^{kl})$ by assumption. Hence $NP_k \subseteq P_{kl}$. ∎

3 Complete sets

The sets N^mP_l all have complete sets under quasilinear-time reducibility. We consider first a problem concerning acceptance by Turing machines. Let $\langle \cdot \rangle$ be a coding function for Turing machines such that given $\langle M \rangle$, a single step of M can be simulated by a multitape Turing machine in time proportional to $\log |\langle M \rangle|$. A unary code will have the required property.

Define $G_l^m = \{\, \langle M \rangle \# x \# 1^j \mid M$ accepts input x within $j|x|^{l-1}$ steps using at most $m \log |x|$ nondeterministic choices $\}$.

Theorem 5 *For all m and l, G_l^m is complete for N^mP_l with respect to quasilinear-time reductions.*

PROOF (sketch). The coding convention ensures that $G_l^m \in N^mP_l$. For $A \in N^mP_l$, fix a machine M and constants r and s such that M halts in at most $rn^l \log^s n$ steps, uses $m \log n$ nondeterministic choices on inputs of length n, and accepts A. Let $f(x) = \langle M \rangle \# x \# 1^{r|x| \log^s |x|}$. Then $x \in A$ if and only if $f(x) \in G_l^m$, and f is computable in time in $Q(n)$. Hence f is the desired reduction from A to G_l^m. ∎

In some cases, complete sets for N^mP_l can be obtained from *NP*-complete problems by bounding the size of allowable witnesses for membership. We first consider topologically ordered Boolean circuits.[2] A circuit is *satisfiable* if some setting of the

[2]A circuit is topologically ordered if each gate is numbered, and the inputs to a gate all have lower numbers than the gate itself.

inputs results in an output of 1. Let $\mathrm{CSAT}(k)$ be the set of satisfiable, topologically ordered circuits that have fewer than $k \log n$ inputs, where n is the number of gates in the circuit. (Constants 0 and 1 are not counted as inputs.)

Theorem 6 *For all k, $\mathrm{CSAT}(k)$ is complete for N^kP_1 with respect to quasilinear-time reductions.*[3]

PROOF (sketch). The standard methods for proving the formula-satisfiability problem to be *NP*-complete can easily be modified to produce a circuit rather than a formula. Schnorr [16] and Cook [7] have shown that the circuit (or formula) corresponding to a given input for a fixed Turing machine may be constructed in quasilinear time. In particular, the circuit has quasilinear size. Also, it is topologically ordered.

Let $A \in N^kP_1$, and let $R \in P_1$ be such that $x \in A \Leftrightarrow \exists y\, |y| \leq k \log |x| \wedge \langle x, y\rangle \in R$, for all strings x. Let M accept R in quasilinear time. The circuit $C_M(x)$ corresponding to M with input $\langle x, y\rangle$, for fixed x and undetermined y, has size quasilinear in $|x|$ and only $k \log |x|$ inputs. Hence $x \in A$ if and only if $C_M(x) \in \mathrm{CSAT}(k)$, and $\mathrm{CSAT}(k)$ is hard for N^kP_1.

Evaluation of a circuit at given inputs can be accomplished in quasilinear time on a Turing machine using an algorithm of Pippenger [14]. Hence $\mathrm{CSAT}(k)$ is in N^kP_1. ∎

The satisfiability problem for formulas with a limited number of variables, $\mathrm{SAT}(k)$, is perhaps easier than $\mathrm{CSAT}(k)$. The above proof of hardness of $\mathrm{CSAT}(k)$ does not apply to $\mathrm{SAT}(k)$, because too many auxiliary variables are required in a formula to simulate a Turing machine.[4] In the absence of a hardness proof for the restricted formula-satisfiability problem, hardness proofs for restrictions of other standard *NP*-complete problems are problematic. In some cases, the natural restriction is essentially as hard as the general case. For example, graph k-colorability remains *NP*-complete when k is fixed at 3.

The restricted vertex-cover problem exhibits different behaviour. Let $\mathrm{VC}(k)$ be the language of undirected graphs that have a vertex cover of size k. It is easy to show that $\mathrm{VC}(k)$ is in $N^{k-1}P_1$: the first $k-1$ vertices of a cover can be chosen nondeterministically, and the last vertex found by linear search. In fact, the language

[3]Díaz and Torán [8] have shown that a variant of $\mathrm{CSAT}(k)$ is complete under logspace reductions for their class β_k, for each k.

[4]We return to the formula-satisfiability problem in section 4.

VC(k) can be accepted in linear time by the following algorithm due to S. Buss [6]. Given a graph $G = (V, E)$,

1. Let U be the set of vertices of degree more than k. If $|U| > k$, then reject; there is no cover of size k or less.

2. Let G' be the subgraph of G induced by $V - U$. Every k-cover of G consists of U together with a $(k - |U|)$-cover of G'. If G' has more than $k(k - |U|)$ edges, then reject; G' has no $(k - |U|)$-cover.

3. If G' has a cover of size $k - |U|$, then accept; otherwise reject.

Steps 1 and 2 can easily be implemented in quasilinear time. If step 3 is reached, then G' has a bounded number of edges; hence step 3 requires $Q(1)$ time. Therefore, VC(k) is in P_1, and a completeness proof for VC(k) would have strong consequences.

Theorem 7 *If* VC(k) *is hard for* N^jP_1 *for some* $j \geq 1$*, then* $N^mP_1 \subseteq P_1$ *for all* m*.*

The result follows immediately from Theorem 2.

The k-clique problem, although often regarded as a trivial variation on the k-vertex cover problem, cannot be substituted for VC(k) in the above proof, because the condition on the degrees does not hold. Clique remains as a candidate hard problem for $N^{k-1}P_1$.

Next, we consider the following problem concerning context-free grammars. Given a grammar $G = (N, \Sigma, P, S)$, does G generate any string comprising repetitions of a single terminal; i.e., is $\mathcal{L}(G) \cap \bigcup_{\sigma \in \Sigma} \sigma^* \neq \emptyset$? Let UnaryGen be the set of grammars for which the answer is yes.

Theorem 8 CSAT(1) $\leq_{ql}$ UnaryGen.

Proof. Let C be a circuit with n gates $g_1, \ldots, g_n$. Assume without loss of generality that gates g_i with $i \leq \log n$ are the inputs and g_n is the output. Each gate g_j is represented by two nonterminal symbols A_j and A'_j. There are also n terminal symbols $b_0, \ldots, b_{n-1}$, each corresponding to one setting of the values of the input variables. The start symbol is A_n.

The productions are determined as follows.

- If g_i is an and-gate with predecessors g_j and g_k, use productions $A_i \longmapsto A_j A_k$ and $A'_i \longmapsto A'_j \mid A'_k$.

- If g_i is an or-gate with predecessors g_j and g_k, use productions $A_i \longmapsto A_j \mid A_k$ and $A'_i \longmapsto A'_j A'_k$.
- If g_i is a not-gate with predecessor g_j, use productions $A_i \longmapsto A'_j$ and $A'_i \longmapsto A_j$.
- For $1 \leq i \leq \log n$, for all j such that bit i of the binary expansion of j is 1, use $A_i \longmapsto b_j$.
- For $1 \leq i \leq \log n$, for all j such that bit i of the binary expansion of j is 0, use $A'_i \longmapsto b_j$.

The variable A_j derives a unary word if and only if g_j evaluates to 1, and A'_j derives a unary word if and only if g_j evaluates to 0.

Lemma 8.1 *For all $1 \leq i \leq n$ and $0 \leq j \leq n-1$, $A_i \longmapsto^* b_j^k$ for some k if and only if g_i evaluates to 1 when the input vector is the binary expansion of j. Similarly, $A'_i \longmapsto^* b_j^k$ for some k if and only if g_i evaluates to 0 when the input vector is the binary expansion of j.*

Proof of lemma. Fix j. Define the *level* of a gate to be the length of the longest directed path from the gate to any input. We use induction on the level of gate g_i.

The base case is that g_i is an input. Then g_i evaluates to 1 iff bit i of j is 1 iff $A_i \longmapsto b_j$ iff $A_i \longmapsto^* b_j$. Also g_i evaluates to 0 iff bit i of j is 0 iff $A'_i \longmapsto b_j$ iff $A'_i \longmapsto^* b_j$.

Suppose that the lemma holds for all gates with level at most $l-1$, and let g_i have level l. If g_i is an and-gate with predecessors g_{i_1} and g_{i_2}, then g_i evaluates to 1 iff both g_{i_1} and g_{i_2} evaluate to 1 iff $A_{i_1} \longmapsto^* b_j^{k_1}$ and $A_{i_2} \longmapsto^* b_j^{k_2}$ for some k_1 and k_2 iff $A_i \longmapsto^* b_j^{k_1+k_2}$. Likewise, g_i evaluates to 0 iff either g_{i_1} or g_{i_2} evaluates to 0 iff $A'_{i_1} \longmapsto^* b_j^k$ or $A'_{i_2} \longmapsto^* b_j^k$ for some k iff $A'_i \longmapsto^* b_j^k$. The cases where g_i is a negation gate or an or-gate are similar. ∎

The theorem follows immediately from the lemma. ∎

The reduction given in the proof holds even if the circuit is not topologically ordered. If the circuit is ordered, then in the resulting grammar, every appearance of each non-terminal in the left-hand side of a production occurs after every appearance of the non-terminal on the right-hand side of a production. Call a grammar satisfying this condition an ordered grammar. Let UnaryGen$_{\mathrm{ord}}$ be the set of ordered grammars in UnaryGen.

Theorem 9 UNARYGEN$_{\text{ord}}$ *is complete for* N^1P_1.

PROOF. Because the reduction of the previous theorem produces an ordered grammar from an ordered circuit, UNARYGEN$_{\text{ord}}$ is hard for N^1P_1.

To show that UNARYGEN$_{\text{ord}}$ is in N^1P_1, we use the following lemma.

Lemma 9.1 *The emptiness problem for ordered context-free grammars is in deterministic quasilinear time.*

PROOF OF LEMMA. Call a nonterminal "true" if some terminal string can be derived from it, and "false" otherwise. The grammar is then a network in the sense of Pippenger [14], and thus can be evaluated in time $Q(n)$. ▮

Let $G = (N, \Sigma, P, S)$ be an ordered grammar. Nondeterministically guess a terminal $\sigma \in \Sigma$, using $\log |\Sigma| \leq \log |G|$ bits. Then eliminate all productions involving terminals other than σ. Accept if an only if the resulting grammar generates a nonempty language.

The above algorithm places UNARYGEN$_{\text{ord}}$ in N^1P_1, and hence it is complete. ▮

4 Families of constrained-search problems

Many *NP*-complete problems involve a parameter that defines a constraint on the associated search problem. Restriction of this parameter produces a problem whose complexity falls into one of at least three classes. (1) The restricted problem may be *NP*-complete; e.g., graph k-coloring, for any $k \geq 3$. (2) Every fixed value of the parameter may give the same polynomial complexity; e.g., vertex k-cover. (3) Every fixed value of the parameter may give polynomial-time complexity, but the exponent of the polynomial may depend on k. No instances of case 3 are proven, but CSAT(k), k-clique and many other problems are candidates. The distinction between cases 2 and 3 has been recently investigated by Abrahamson, et al. [1, 2], who considered families of associated search problems. We show here that limited nondeterminism provides an alternative to their methods.

Consider the case of CSAT(k), for $k \geq 1$. These languages form an infinite collection of complete sets, one for each class N^kP_1. It is not an arbitrary collection, however, since the complete set for N^kP_1 can easily be determined given k. We describe this situation using the concept of *language family*. A language family is a subset of $\Sigma \times \mathbf{N}$. The kth slice of a family B, denoted B_k, is the language $\{\, x \mid (x,k) \in B \,\}$.

The class N^*P_j (for all $j \geq 1$) consists of all families B with the following properties.

- The kth slice of B is a language in N^kP_j.
- There is a quasilinear-time-computable function f such that for all k, the value of $f(k)$ is the code of a Turing machine witnessing that the kth slice of B is in N^kP_j. (An object x *witnesses* an existential property $\exists y\, p(y)$ if and only if $p(x)$ is true.)

A language family C in N^*P_j is *$\leq_{ql}$-complete for N^*P_j* if the following hold.

- For all k, the slice C_k is complete for N^kP_j.
- There is a quasilinear-time-computable function f such that for all k, the value of $f(k)$ is the code of a Turing machine witnessing that C_k is complete for N^kP_j.

For example, the family $\mathrm{CSAT}(*) = \{\, (x,k) \mid x \in \mathrm{CSAT}(k) \,\}$ is complete for N^*P_1.

We draw a second example from Abrahamson, et al. [2]. Consider a Boolean formula. A partial assignment of truth values to variables in the formula may induce values on other variables if the formula is to be true. Consider a formula in conjunctive normal form with three or fewer variables per clause (3CNF). If $(a_1 \vee a_2 \vee a_3)$ is a clause, and all but one of the a_i's are false, then the remaining variable is induced to assume the value true; if one variable a_i is true, then the clause is true, and no additional assignment is induced. If this procedure can be applied iteratively until all clauses evaluate to true, then we say the original partial assignment caused the formula to *unravel*. Let $\textsc{ShortSAT}(k)$ be the set of 3CNF formulas B such that some assignment to the first $k \log |B|$ variables causes B to unravel.[5]

A formula $B = \bigwedge_{i=1}^{k} C_i$ in conjunctive normal form *unravels in order* if each initial subformula $\bigwedge_{i=1}^{j} C_i$ unravels. Let $\textsc{ShortSAT}_{\mathrm{ord}}(k)$ be the set of formulas B such that some assignment to the first $k \log |B|$ variables causes B to unravel in order.

Theorem 10 $\textsc{ShortSAT}_{\mathrm{ord}}(k)$ *is $\leq_{ql}$-complete for N^kP_1, and* $\textsc{ShortSAT}(k)$ *is $\leq_{ql}$-hard for N^kP_1, for all k.*

[5]The definition in [2] allowed arbitrary CNF formulas instead of 3CNF. The difference appears to be necessary for Theorem 10. It does not affect the results of [1].

PROOF (sketch). We first show that $\mathrm{CSAT}(k) \leq_{ql} \textsc{Short}\mathrm{SAT}_{\mathrm{ord}}(k)$. Let Γ be an instance of $\mathrm{CSAT}(k)$. The corresponding formula φ has one variable for each gate of the circuit, and two additional variables denoted z_1 and z_2. The first $k \log n$ clauses of φ represent the input. The connections of a gate g in Γ are represented in φ by a conjunction B of clauses. The conjunction B always unravels in φ, and the only unravelling induces the variable representing g to take on the value of g. The final clause of φ unravels if and only if the circuit evaluates to 1.

The clauses are as follows.

- For each input variable x_i of Γ, the corresponding clause is $(x_i \vee \neg x_i)$.
- If g is an and-gate with predecessors a and b, then the corresponding conjunction is $(\neg a \vee \neg b \vee g) \wedge (a \vee \neg g) \wedge (b \vee \neg g)$.
- If g is an or-gate with predecessors a and b, then the corresponding conjunction is $(a \vee b \vee \neg g) \wedge (\neg a \vee g) \wedge (\neg b \vee g)$.
- If g is a not-gate with predecessor a, then the corresponding conjunction is $(a \vee \neg g) \wedge (\neg a \vee g)$.
- If the output gate is g_{out}, the final clause is $(g_{\mathrm{out}} \vee z_1 \vee z_2)$.

The reader can easily check that the formula has the required properties.

To show that $\textsc{Short}\mathrm{SAT}_{\mathrm{ord}}(k) \leq_{ql} \mathrm{CSAT}(k)$, let φ be a formula in 3CNF, with clauses $\{\varphi_i\}_{i=1}^{c}$. Each clause φ_i will be represented by a circuit C_i, whose inputs will be outputs of the circuits representing earlier clauses. We assume without loss of generality that no variable appears twice in any one clause.

Consider $\varphi_i = v_{i1} \vee v_{i2} \vee v_{i3}$. The corresponding circuit C_i will have six inputs and six outputs. The circuit C_i will compute the status of each literal v_{ij} (either a variable or a negated variable) and produce two corresponding output values. Output $v_{ij}(1)$ is given value 1 iff the unravelling procedure assigns a value to v_{ij} at or before clause φ_i. If $v_{ij}(1) = 1$, then output $v_{ij}(2)$ is given the value assigned to v_{ij}.

The inputs to C_i are determined by the last occurrences of the variables of φ_i previous to φ_i itself. Denote by v'_{ij} the last literal containing the variable of v_{ij} appearing in φ before φ_i. The first input corresponding to v_{ij} is $v'_{ij}(1)$. The second input corresponding to v_{ij} is $v'_{ij}(2)$ if v_{ij} and v'_{ij} are both positive or both negative literals, and is $\neg v'_{ij}(2)$ otherwise. The construction of C_i is straightforward except for the identification of the input literals v'_{ij}.

If each v'_{ij} is found individually, the construction of the entire circuit may take quadratic time. Hence the successive occurrences of each variable must all be determined at once, as follows. For each v_{ij}, form a triple (x, i, j), where x is the index of the variable ocurring in v_{ij}. Sort the triples, using the order $(x, i, j) < (y, k, l)$ iff $x < y$ or $x = y$ and $i < k$. If (x, i, j) and (x, k, l) are adjacent in the sorted list, then $v'_{kl} = v_{ij}$; form a quadruple (i, j, k, l). Sorting these quadruples gives the required adjacency information, in the order needed to construct the circuit in topological order.

The final circuit comprises all subcircuits C_i and an additional output subcircuit. The output subcircuit computes $\bigwedge_{i=1}^{c} \bigvee_{j=1}^{3} \Big(v_{ij}(1) \wedge v_{ij}(2) \Big)$, which has value 1 if and only if the original formula φ unravels in order. ▮

The family $\textsc{ShortSAT}_{\text{ord}}(*) = \{\, (x, k) \mid x \in \textsc{ShortSAT}_{\text{ord}}(k) \,\}$ is thus complete for N^*P_1.

Theorem 11 *For all k,* $\textsc{ShortSAT}(k) \in N^kP_2$.

Proof (sketch). To determine whether a formula φ is in $\textsc{ShortSAT}(k)$, it suffices to nondeterministically guess the values of the initial $k \log |\varphi|$ variables, and then make at most $|\varphi|$ passes through φ, inducing variables and propagating their values. ▮

A language family C is *$\leq_{ql}$-hard for N^*P_j* if the following hold.

- For all k, there is a $k' \geq k$ such that $C_{k'}$ is $\leq_{ql}$-hard for N^kP_j.
- There is a quasilinear-time-computable function f such that for all k, the value of $f(k)$ is the code of a Turing machine witnessing that $C_{k'}$ is $\leq_{ql}$-hard for N^kP_j.

For example, $\textsc{ShortSAT}(*) = \{\, (x, k) \mid x \in \textsc{ShortSAT}(k) \,\}$ is $\leq_{ql}$-hard for N^*P_1. The family $G_* = \{\, (G_k^0, k) \mid k \geq 1 \,\}$ is $\leq_{ql}$-hard for N^*P_l, for every l.

Finally, a language family C is *weakly $\leq_{ql}$-complete for N^*P_j* if it is both in N^*P_j and $\leq_{ql}$-hard for N^*P_j. One example of a weakly complete set for N^*P_1 is $\{\, (x, 2k) \mid x \in \text{CSAT}(k)$.

The definition of N^*P_j is motivated by the work of Abrahamson, et al. [1, 2] on their class PGT. Their work focused on families of problems in P, usually generated by parameterized problems in NP, such as $\textsc{ShortSAT}()$ and $\text{CSAT}()$. The class PGT is roughly equivalent to the notion "in some N^*P_j and hard for some N^*P_k." The major differences are that PGT includes non-uniform families and that reductions may be non-uniform; in the terms of this paper, the functions f above are not required to be computable. All of the complete families considered by Abrahamson, et al., hovever, are uniform and are examples of families in some N^*P_j and hard for some N^*P_k. Thus, their work provided examples of sets in the classes N^iP_k, some of which were also hard for classes N^iP_1. Further work on limited nondeterminism is likely to yield additional complete sets for PGT.

5 Relationships among the classes

The classes N^mP_l bear a strong analogy to NP; however, the analogy should not be taken too far. For example, one might conclude from Theorem 3 that NP_l behaves as a limiting case of N^mP_l as m goes to infinity. This conclusion, however, can be false in the presence of an oracle. Denote by C^X the class C relative to oracle X. The results of Section 2 hold relative to any oracle X.

Theorem 12 *There is an oracle A such that $N^mP_k^A = P_k^A$ for all m and k, and yet $NP^A \not\subseteq P^A$.*

Proof. By Theorem 2, we need only show that $N^1P_1^A = P_1^A$ and $P^A \neq NP^A$.

Fix an enumeration $\{M_i\}_{i=1}^{\infty}$ of nondeterministic machines such that M_i runs in quasilinear time and makes at most $\log n$ nondeterministic choices on inputs of length n. Fix an enumeration $\{T_i\}_{i=1}^{\infty}$ of deterministic machines such that T_i runs for at most $p_i(n) = n^i + i$ steps on inputs of length n.

We construct A to have the following two properties for all i.

C$_i$: The coding condition: For all strings x and all positive integers s, the string $0^{2s}\#i\#x$ is in A if and only if $M_i^A(x)$ accepts within s steps. (Note that $M_i^A(x)$ cannot query whether $0^{2s}\#i\#x$ is in A within s steps.)

D$_i$: The diagonalization condition: The language $L_A = \{\, 1^n \mid \exists y\, |y| = n \wedge 1^n y \in A \,\}$ is not accepted by T_i.

The construction proceeds in stages. During stage k, all strings of length k are fixed to be in or out of A, in such a way that conditions C$_i$ hold up to length k. In addition, some strings of length longer than k may be fixed to satisfy some condition D$_i$. Because each string is fixed exactly once during the construction, the oracle A is well-defined. In fact, A is computable in space $n^{O(\log n)}$.

Initially, all strings are unfixed. Set $k = d = l = 1$; k is the stage number, d is the current diagonalization condition, and l is a lower bound on the input at which condition D_d can be satisfied.

Stage k: For all strings z of length k of the form $z = 0^{2s}\#i\#x$ for some i, x and s, fix z in A if M_i accepts x with oracle A in $\leq s$ steps, and otherwise fix z out of A. (Note that this condition is determined by the preceding stages of the construction.) Fix all strings of length k not of the above form out of A. If $k < l$ or $2^k \leq p_d(k)^{1+\log p_d(k)}$, then stage k is complete—continue to stage $k+1$.

If $k \geq l$ and $2^k > p_d(k)^{1+\log p_d(k)}$, then a diagonalization condition can be satisfied. Simulate T_d on input 1^k. When T_d queries a string z not yet fixed, there are two cases.

1. If $z = 0^{2s}\#i\#x$ for some i, x and s, then simulate M_i on input x for s steps, and fix z accordingly. Fix strings queried during the simulation by applying these cases recursively.

2. Otherwise, fix z out of A.

If T_d accepts 1^k, then fix all remaining strings of the form $1^k y$ with $|y| = k$ out of A. If T_d rejects 1^k, then fix all remaining strings of the form $1^k y$ with $|y| = k$ to be in A. In either case, set $l = p_d(k)$, increase d by one, and continue to stage $k+1$.

End of stage k.

We first show that the construction can be carried out.

Lemma 12.1 *If the construction simulates T_d on input 1^k, then some string $1^k y$ with $|y| = k$ remains unfixed at the end of the simulation.*

Proof of lemma. The parameter l is an upper bound on the length of strings that have been fixed due to any diagonalization attempt prior to stage k. Because $k \geq l$ at the start of the simulation of T_d, all of the 2^k strings $1^k y$ are unfixed at that time.

Let $S(n)$ be the maximum number of strings fixed during the simulation of some $M_i^A(x)$ for s steps, where $|0^{2s}\#i\#x| = n$. In s steps, $M_i^A(x)$ queries fewer than $s2^{\log|x|} < n^2$ strings over all of its nondeterministic computations, and each queried string has length less than $n/2$. Hence $S(n) < n^2 S(n/2)$, and $S(1) = 1$. Therefore, $S(n) \leq n^{\log n}$.

T_d runs for at most $p_d(k)$ steps; therefore, the total number of strings queried during the simulation is at most $p_d(k) \cdot S(p_d(k)) \leq p_d(k)^{1+\log p_d(k)} < 2^k$. Hence some string of the form $1^k y$ with $|y| = k$ remains unfixed. ▌

Lemma 12.2 *The set A constructed as above satisfies conditions C_i and D_i for all i.*

Proof of lemma. Each string of the form $0^{2s}\#i\#x$ is eventually fixed in or out of A. Such a string is fixed only after the acceptance or rejection of $M_i^A(x)$ within s steps is determined, and is fixed according to that acceptance or rejection. Hence condition C_i holds for all i.

Because $p_d(n)^{1+\log p_d(n)} = o(2^n)$, each value of d is eventually considered at some stage k. The previous lemma ensures that condition D_d is established following the completion of stage k. ▌

Conditions C_i together imply that $N^1P_1^A = P_1^A$, and hence $N^mP_k^A = P_k^A$ for all k and m by Theorem 2. Conditions D_i together imply that $L^A \in NP^A - P^A$. ▌

Informally, Theorem 12 shows that large amounts of nondeterminism can be useful even if small amounts are not. The reverse is also possible—a large amount of nondeterminism may be no more advantageous than a small amount.

Theorem 13 *There is an oracle B such that $P_m^B \neq N^1P_m^B$ and $N^1P_m^B = NP_m^B$ for all m.*

Proof. The proof depends in a small way on the definition of an oracle machine. There are two possibilities to be considered. If the query tape is erased following a query, define $\delta = 0$. If the query tape is not erased following a query, define $\delta = 1$. Although the relativized classes depend on the model used (see Buss [5]), the necessary properties for the present proof are captured by the value of δ.

Let $\{M_{l,i}\}_{l,i=1}^{\infty}$ be an enumeration of nondeterministic oracle machines, and let $\{T_{l,i}\}_{l,i=1}^{\infty}$ be an enumeration of deterministic oracle machines, such that both $M_{l,i}$ and $T_{l,i}$ are clocked to run for $p_{l,i}(n) = n^l \log^i n + i$ steps. We construct B to meet the following conditions.

1. Coding conditions: For all l and i, for all sufficiently large x, $M_{l,i}^B(x)$ accepts if and only if there is a string y of length $(1 + \delta l)\log|x|$ such that $0^{p_{l,i}(|x|)}\#l\#i\#x\#y$ is in B.

2. Diagonalization conditions: For all l, the language

$$L_l = \{\, x \mid \exists y\, |y| = (1+\delta l)\log|x| \wedge 1^{|x|^l}xy \in B \,\}$$

is not accepted by $T^B_{l,i}$ for any i.

Condition 1 implies that $NP^B_l \subseteq N^1P^B_l$ for all l, because $\log|x|$ bits of y can be guessed nondeterministically, and if $\delta = 1$, the remaining $l\log|x|$ bits found by exhaustive search in time $O(n^l)$. (If a string is erased upon being queried, each query of a coding string requires $Q(n^l)$ steps, and the search cannot be carried out deterministically.) Condition 2 implies that $L_l \in N^1P^B_l - P^B_l$.

Let $\langle\cdot,\cdot\rangle$ be a pairing function from positive integers to positive integers; e.g., let $\langle l,i\rangle = (l^2+i^2+2li-3l-i+2)/2$. To start the construction, set $k = d = b = 1$; k is the stage number, d the current diagonalization condition, and b the barrier so that diagonalizations do not overlap. Throughout, l and i will be determined by $d = \langle l,i\rangle$. Once again, at stage k, all strings of length k will be fixed in or out of B.

Stage k: If possible, pick an integer $\hat{k}$ such that $b < \hat{k} < k^{1/l}$, $p_{l,i}(\hat{k}) < \hat{k}^{l+(1/d)}$ and $\hat{k}^l + \hat{k} + (1+\delta l)\log\hat{k} \geq k$. If $\hat{k}$ exists, then a diagonalization condition can be satisfied. Otherwise, continue immediately to the coding part of the stage.

To satisfy the diagonalization condition, simulate T_d on input $1^{\hat{k}}$. If T_d queries a string not yet fixed, fix that string to be out of B. (This may fix a coding string. See below.) At the end of the simulation, if $T^B_d(1^{\hat{k}})$ accepts, fix all strings of the form $1^{\hat{k}^l+\hat{k}}y$ with $|y| = (1+\delta l)\log k$ to be out of B. If $T^B_d(1^{\hat{k}})$ rejects, fix all remaining strings $1^{\hat{k}^l+\hat{k}}y$ with $|y| = (1+\delta l)\log k$ to be in B. Set $b = p_{l,i}(\hat{k})+1$, and increase d by one.

After the diagonalization condition is considered, satisfy the possible coding conditions up to length k, as follows. For all unfixed strings z of length k having the form $0^{p_{m,j}(|x|)}\#m\#j\#x\#y$ for some m, j, x and y with $|y| = (1+\delta m)\log|x|$, fix z in B if $M^B_{m,j}(x)$ accepts and fix z out of B if $M^B_{m,j}(x)$ rejects.

End of stage k.

For each fixed l, i and d, the conditions on $\hat{k}$ are satisfied for infinitely many values of k. Therefore, each diagonalization condition will eventually be satisfied, and $L_l \in N^1P^B_l - P^B_l$ for all l.

Coding strings may be fixed during diagonalization stages. However, for each fixed m and j, there is a bound on the size of x such that a coding string for $M_{m,j}(x)$

may be fixed during a diagonalization. A given coding string for $M_{m,j}$ may be fixed out of B during the diagonalization against at most one $T_{l,i}$, at some stage k. Since all strings of length less than k are fixed before stage k, the stage number satisfies $k \leq |0^{p_{m,j}(|x|)}\#m\#j\#x\#y| = Q(|x|^m)$. Note that $k < c|x|^{m+1/3}$ for some c and all sufficiently long strings x. At most $p_{l,i}(\hat{k})/k^{1-\delta}$ strings of length k are fixed during the diagonalization. The conditions on $\hat{k}$ imply that $p_{l,i}(\hat{k})/k^{1-\delta}$ is bounded by $\hat{k}^{l+(1/d)}k^{\delta-1} < k^{\delta+(1/dl)}$, which is less than $|x|^{1+\delta m}$ when $d > 3m$ and x is sufficiently long. Since d is nondecreasing and unbounded, each $M_{m,j}$ will be correctly coded for all but finitely many x, and condition 1 will be satisfied.

Therefore, the oracle B meets the required conditions. ▮

We expect that neither oracle A nor B represents the unrelativized case. The following result seems more likely.

Theorem 14 *There is an oracle C such that $N^iP_j^C \neq N^kP_l^C$ for all distinct pairs (i,j) and (k,l). Also $P^C \neq NP^C$.*

Proof. Each class N^iP_j satisfies the following conditions.

1. Let X be any oracle, and let $A \in N^iP_j^X$. For each string x, to test if $x \in A$ requires only finite knowledge of X.

2. For all oracles X and Y that differ on only finitely many strings, $N^iP_j^X = N^iP_j^Y$.

3. For all oracles X, and for all sets A and B that differ on only finitely many strings, $A \in N^iP_j^X$ if and only if $B \in N^iP_j^X$.

4. Let X be any oracle, and let machine M accept A in $N^iP_j^X$. Then there are an oracle X^* and machine M^* running in N^iP_j such that

   - $N^iP_j^X = N^iP_j^{X^*}$,
   - M^* with oracle X^* accepts the language A, and
   - for all oracles Y and Z that differ on only finitely many strings, the language accepted by M^* with oracle Y differs from the language accepted by M with oracle Z on at most finitely many strings.

Conditions 1, 2, and 3 are immediate. The oracle X^* of condition 4 is simply $X \times \mathbf{N}$. Machine M^* acts as M except that a query s is replaced by $\langle s, n\rangle$, where n is the length of the input.

These properties permit the use of a result of Poizat [15].

Lemma 14.1 (Poizat) *Let $\mathcal{S}$ and $\mathcal{T}$ be two classes that satisfy conditions 1–4 above, with the same mapping $X \mapsto X^*$. Suppose that there is an oracle X such that $\mathcal{S}^X \neq \mathcal{T}^X$. Then $\mathcal{S}^G \neq \mathcal{T}^G$ for all generic sets G.*

Hence one may reverse the order of quantifiers in the statement of the theorem.

Lemma 14.2 *For all distinct pairs (i, j) and (k, l), there is an oracle C such that $N^iP_j^C \neq N^kP_l^C$.*

Proof (sketch). If $j \neq l$, the desired oracle is the set A of Theorem 12, because $P_j^A \neq P_l^A$. (An alternative proof may be obtained via a diagonalization using queries of length $n^{\max\{j,l\}}$ on inputs of length n.)

Suppose $j = l$ and $i < k$. For an oracle C, define $L_C = \{\, x \mid \exists y\, |y| = k \log x \wedge xy \in C \,\}$. A standard diagonalization constructs a C such that $L_C \notin P_{i+j}^C$. Because $L_C \in N^kP_l^C$, the desired result follows. ▮

The previous lemmas imply that $N^iP_j^C \neq N^kP_l^C$ for all generic oracles C. ▮

Theorems 12 and 13 indicate that mere equality or inequality among the classes N^mP_l is insufficient to determine whether $P = NP$. It is consistent either that small amounts of nondeterminism ($m \log n$ bits) are useful, but large amounts (polynomially many bits) are no more useful, or that small amounts of nondeterminism are useless, but large amounts are quite powerful. However, additional information about the complexity of these classes, such as in the hypothesis of Theorem 3, can suffice to decide $P =? NP$.

6 Conclusions and open problems

We have shown several problems to be complete for quasilinear time with restricted nondeterminism. Satisfiability of topologically ordered circuits with $k \log n$ inputs and ordered unravelling of Boolean formulas are each complete for N^kP_1, and a problem concerning context-free grammars is complete for N^1P_1. All of these complete problems require that the input be ordered appropriately. The corresponding unordered problems remain hard for N^kP_1, but do not appear to be in any class below N^kP_2. If random-access machines were used as the underlying model of computation instead of Turing machines, then the unordered problems would be in quasilinear time, but the hardness proofs would fail, due to our inability to simulate random access.

One motivation for this work, as yet unrealized, was to provide general lower bounds for problems of "nearly feasible" complexity. Existing lower bounds usually apply to restricted models of computation, such as one-tape Turing machines or constant-depth circuits, or prove only very large lower bounds, as in the proofs of hardness for exponential time. A proof that some problem S is hard for a particular P_k or N^jP_k $(k > 1)$ is a general lower bound on the time complexity of problem S. Although S may already be known to be in P, and thus be considered "feasible," such a hardness proof would give concrete information about the practical complexity of S. The only results of this kind known to the authors are for certain pebble games [3].

This paper leaves many questions unanswered. Although the most dramatic question is whether the hypothesis of Theorem 3 holds, there are other, more approachable questions. There are several problems that are easily shown to be in Gurevich and Shelah's class *NLT*, but are not known to be in *QL*. These include topological sorting, and the problem of generalized Boolean formulas (shown to be *NLT*-complete in [10]). Providing *QL* algorithms for these problems, or showing that none exist (i.e., that $NLT \neq QL$), would give us important information about the power of random access.

The fundamental question remains whether there are problems in P that can be computed more quickly with limited nondeterminism than without it. For instance, can one show a $Q(n^k)$ deterministic lower bound for $(k+1)$-clique, or for CSAT(k)?

Acknowledgements

The authors thank N. Immerman, M. Groszek, A. Klapper, A. Peterson, W. L. Ruzzo and A. Selman for useful discussions of this material.

References

[1] K. R. Abrahamson, J. A. Ellis, M. R. Fellows and M. E. Mata, "On the Complexity of Fixed Parameter Problems," in *30th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society Press, 1989, pp. 210–215.

[2] K. R. Abrahamson, J. A. Ellis, M. R. Fellows and M. E. Mata, "Completeness for Families of Fixed Parameter Problems," U. Victoria (Canada) report DCS-141-IR, 1990.

[3] A. Adachi, S. Iwata and T. Kasai, "Some Combinatorial Game Problems Require $\Omega(n^k)$ Time," *J. Assoc. Comput. Mach.* 31 (1984) 361–376.

[4] C. Àlvarez, J. Díaz and J. Torán, "Complexity Classes With Complete Problems Between *P* and *NP*-Complete," in *Foundations of Computation Theory,* Lecture Notes in Computer Science 380, Springer-Verlag, 1989, pp. 13-24.

[5] J. F. Buss, "Relativized Alternation and Space-Bounded Computation," *J. Comput. System Sci.* 36 (1988) 351–378.

[6] S. R. Buss, personal communication.

[7] S. A. Cook, "Short Propositional Formulas Represent Nondeterministic Computations," *Inform. Process. Lett.* 26 (1987/88) 269–270.

[8] J. Díaz and J. Torán, "Classes of Bounded Nondeterminism," *Math. Systems Theory* 23 (1990) 21–32.

[9] J. G. Geske, *On the Structure of Intractable Sets,* Ph.D. thesis, Iowa State University, 1987.

[10] Y. Gurevich and S. Shelah, "Nearly Linear Time," in A. R. Meyer and M. A. Taitslin, eds., *Logic at Botik '89,* Lecture Notes in Computer Science 363, Springer-Verlag, 1989, pp. 108–118.

[11] H. B. Hunt, III, and R. E. Stearns, "The Complexity of Very Simple Boolean Formulas with Applications," *SIAM J. Comput.* 19 (1990) 44–70.

[12] R. Kaye, "Characterizing some low complexity classes using theories of arithmetic," M.Sc. thesis, University of Manchester, 1985.

[13] C. M. R. Kintala and P. C. Fischer, "Refining Nondeterminism in Relativized Polynomial-Time Bounded Computations," *SIAM J. Comput.* 9 (1980) 46–53.

[14] N. Pippenger, "Fast Simulation of Combinational Logic Circuits by Machines without Random-Access Storage," in *Fifteenth Allerton Conference on Communication, Control, and Computing,* 1977, pp. 25–33.

[15] B. Poizat, "$\mathcal{Q} = \mathcal{NQ}$?," *J. Symbolic Logic* 51 (1986) pp. 22–32.

[16] C. P. Schnorr, “Satisfiability is Quasilinear Complete in *NQL*,” *J. Assoc. Comput. Mach.* 25 (1978) 136–145.

[17] M. J. Wolf, *Limited Nondeterminism in Parallel Models of Computation*, Ph.D. Dissertation, University of Wisconsin, 1990.