On Sets Bounded Truth-Table Reducible to P-selective Sets*

Thomas Thierauf
Fakultät für Informatik
Universtät Ulm

Seinosuke Toda
Dep. of Computer Science
Univ. Electro-Communications

Osamu Watanabe
Dep. of Computer Science
Tokyo Institute of Technology

Abstract

We show that if every NP set is $\leq^P_{btt}$-reducible to some P-selective set, then NP is included in DTIME($2^{n^{O(1/\sqrt{\log n})}}$). The result is extended for some unbounded reducibilities such as $\leq^P_{(\log n)^{O(1)}\text{-}tt}$-reducibility.

1 Introduction

One of the important questions in structural complexity theory is whether every NP problem is solvable by polynomial-size circuits, i.e., NP $\subseteq$? P/poly. Furthermore, it has been asked what is the deterministic time complexity of NP problems (e.g., NP $\subseteq$ P) if NP $\subseteq$ P/poly. That is, if NP is easy in the nonuniform complexity measure, how easy is NP in the uniform complexity measure? We study such type of questions in this paper.

Let $\mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$ be the class of languages that are polynomial-time Turing reducible to some sparse set. (In general, let $\mathrm{R}_r(\mathrm{SPARSE})$ be the class of languages that are polynomial-time r reducible to some sparse set. Since we consider only polynomial-time reducibilities, we often omit "polynomial-time" in the following.) In other words, L is in $\mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$ if and only if L is recognizable by some polynomial-time deterministic Turing machine using some sparse set as an oracle. Since $\mathrm{P/poly} = \mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$, the above question is equivalent to the following one: For which uniform complexity class $\mathcal{C}$ do we have $\mathrm{NP} \subseteq \mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE}) \Longrightarrow \mathrm{NP} \subseteq \mathcal{C}$? While no nontrivial answer is known to this question[1], we have obtained several interesting results under stronger assumptions that NP is contained in certain subclasses of $\mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$.

Since $\mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$ is the class of languages that are Turing reducible to some sparse set, one way of obtaining subclasses of $\mathrm{R}_{\mathrm{T}}^{\mathrm{P}}(\mathrm{SPARSE})$ is to consider some restriction on the reducibility. For example, Mahaney [Mah82] showed that if all NP sets are many-one reducible to some sparse set, then P = NP. That is,

$$\mathrm{NP} \subseteq \mathrm{R}_{\mathrm{m}}^{\mathrm{P}}(\mathrm{SPARSE}) \Longrightarrow \mathrm{NP} \subseteq \mathrm{P}.$$

Recently, Ogiwara and Watanabe [OW91] proved a similar result for bounded truth-table reducibility, one restriction of the Turing reducibility which is more general than the many-one reducibility. They extended Mahaney's result by showing that

$$\mathrm{NP} \subseteq \mathrm{R}_{\mathrm{btt}}^{\mathrm{P}}(\mathrm{SPARSE}) \Longrightarrow \mathrm{NP} \subseteq \mathrm{P}.$$

This result has been improved further more recently; see [AHH+93]. However, it is open whether the result can be improved for $b(n)$-bounded truth-table reducibility for some nonconstant func-

[1] We should note here that we have some evidence indicating that NP $\subseteq$ P/poly is unlikely. That is, it has been proved that NP $\subseteq$ P/poly $\Longrightarrow$ PH $\subseteq \Sigma_2^{\mathrm{P}}$ [KL82]. However, although PH $\subseteq \Sigma_2^{\mathrm{P}}$ indirectly shows that NP is not as difficult as we expect, it does not give any specific upper deterministic time bound for NP.

tion $b(n)$. Indeed, Saluja [Sal92] showed that such an improvement is impossible with any relativizable technique.

Other subclasses of $R_T^P(SPARSE)$ are obtained by considering P-selective sets introduced by Selman [Sel79]. A set A is P-selective, if there exists a polynomial-time computable function that selects one of two given input strings such that if any of the two strings is in A, then also the selected one. Let SELECT denote the class of P-selective sets, and for any polynomial-time reducibility r, let $R_r(SELECT)$ denote the class of languages that are r reducible to some P-selective set. It follows from Theorem 12 of [Sel82b] and Theorem 3 of [Ko83] that $R_T^P(SELECT) = R_T^P(SPARSE)$. Thus, by using a restricted reducibility r, we define some subclass $R_r(SELECT)$ of $R_T^P(SPARSE)$ that can be different from $R_r(SPARSE)$. (For example, $R_{tt}^P(SELECT) \neq R_{tt}^P(SPARSE)$ because $R_{tt}^P(SPARSE) = R_T^P(SPARSE)$ but $R_{tt}^P(SELECT) \subsetneq R_T^P(SELECT)$ [Wat90].)

For some of those $R_r(SELECT)$ subclasses, we have been able to solve our question. Selman [Sel79] proved that no NP set is many-one reducible to a P-selective set unless P = NP. Furthermore, by extending this argument slightly, one can also prove that no NP set is 1-truth table reducible to a P-selective set unless P = NP [HHO+93]. That is,

$$NP \subseteq R_{1\text{-}tt}^P(SELECT) \implies NP \subseteq P.$$

For a more general type of reducibility, Toda [Tod91] and Beigel [Bei88] proved the following.

$$NP \subseteq R_{tt}^P(SELECT) \implies NP \subseteq R. \tag{1.1}$$

Note that the last upper bound of NP (namely, the class R) is a randomized complexity class. Thus, one interesting question is to show some deterministic upper bound from more general assumption than "$NP \subseteq R_{1\text{-}tt}^P(SELECT)$". In this paper, we study this question and obtain the following result:

$$NP \subseteq R_{btt}^P(SELECT) \implies NP \subseteq DTIME(2^{n^{O(1/\sqrt{\log n})}}). \tag{1.2}$$

The result is extended to $R_{b(n)\text{-}tt}^P(SELECT)$ for some polynomial-time computable function $b(n)$, e.g., $b(n) = (\log n)^{O(1)}$.

Recently, Jenner and Torán [JT93] obtained a weaker subexponential upper bound for NP (from a weaker assumption) by a very different technique. However, it seems impossible [Tor93] to obtain the same upper bound by their technique.

In the following, we briefly explain the outline of our proof, but let us first review the proof of (1.1). First note that the following fact [Tod91, Bei88]:

$$UP \subseteq R_{tt}^P(SELECT) \implies UP \subseteq P,$$

where UP is the class of languages recognized by polynomial-time nondeterministic Turing machines that have at most one accepting path on each input. Indeed, from $NP \subseteq R_{tt}^P(SELECT)$,

one can deduce a slightly stronger consequence: (*) for any polynomial-time nondeterministic computation, if it has *exactly one* accepting path, then the path is computable in deterministic polynomial-time. This is used to show NP $\subseteq$ R. For any polynomial-time nondeterministic machine and any input, consider its nondeterministic computation on the input. This computation can be regarded as a tree T. Roughly speaking, by using Valiant-Vazirani's randomized hashing technique [VV86], we can define subtrees $T_1, ..., T_m$ of T so that if T has some accepting path, then, say, 1/4 of $T_1, ..., T_m$ has exactly one accepting path. (These subtrees share the same root with T, and hence, each subtree's accepting path is also T's accepting path.) Then from (*), for such T_k with exactly one accepting path, one can compute the accepting path. Thus, by choosing T_k randomly for several times, one can compute some accepting path of T with high probability if T indeed has an accepting path. This is the idea of showing NP $\subseteq$ R.

We also use (*) for proving (1.2). Consider again any nondeterministic computation tree T. From our assumption, namely, NP $\subseteq \mathrm{R}^{\mathrm{P}}_{\mathrm{btt}}(\mathrm{SELECT})$, we can define subtrees $T'_1, ..., T'_n$ of T such that if T has some accepting path, then some T_k has exactly one accepting path. Again from (*), if T_k has exactly one accepting path, then it is verified by some polynomial-time deterministic machine M. Thus, the NP question "Does T have an accepting path?" is reduced to another NP question "Is there any k such that M verifies that T'_k has an accepting path?". The important point here is that n can be taken fairly small compared with the number of paths in T. (*Cf.* The randomized hashing technique needs to define large number of subtrees.) Hence, for solving the reduced NP question, one need smaller number of nondeterministic guesses. Therefore, iterating this process, we can finally solve the original problem deterministically. This is the outline of obtaining our *deterministic* upper bound.

2 Preliminaries

In this abstract, we follow the standard definitions and notations in computational complexity theory (see, e.g., [BDG88, BDG91]).

Throughout this abstract, we fix our alphabet to $\Sigma = \{0, 1\}$. For any set X, we denote the complement of X as $\overline{X}$. Natural numbers are encoded in Σ^* in an ordinary way. For any string x, let $|x|$ denote the length of x, and for any set X, let $||X||$ denote the cardinality of X. The standard lexicographical ordering of Σ^* is used; that is, for strings $x, y \in \Sigma^*$, x is *lexicographically smaller* than y (denoted by $x < y$) if either (i) $|x| < |y|$, or (ii) $|x| = |y|$ and there exists $z \in \Sigma^*$ such that $x = z0u$ and $y = z1v$. We consider a standard one-to-one pairing function from $\Sigma^* \times \Sigma^*$ to Σ^* that is computable and invertible in polynomial time. For inputs x and y, we denote the output of the pairing function by (x, y); this notation is extended to denote any n tuple. We may assume that $|(x, y)| \leq 2(|x| + |y| + 1)$. For a function f, we simply write $f(x, y)$ instead of $f((x, y))$. A set S of strings is called *sparse* if for some polynomial p and

for all n, $\| S^{\leq n} \| \leq p(n)$, where $S^{\leq n}$ is the set of strings $x \in S$ of length $\leq n$.

We use the standard Turing machine as our computation model. P (resp., NP) denotes the class of languages that can be recognized by some polynomial-time DTM (resp., NDTM). (DTM (resp., NDTM) is an abbreviation of "deterministic (resp., nondeterministic) Turing machine".) We assume that for any nondeterministic computation, every nondeterministic configuration of a Turing machine has exactly two succeeding ones, and hence, each nondeterministic computation corresponds naturally to a binary string.

For any sets A and B, we say that A is *many-one reducible to* B (and write $A \leq_{\mathrm{m}}^{\mathrm{P}} B$) if there is some polynomial-time computable function f, the *reduction*, such that for any $x \in \Sigma^*$, we have $x \in A \iff f(x) \in B$. A set C is called *NP-complete* if (i) every NP set is many-one reducible to C, and (ii) C itself is in NP. The reducibility notions that we are interested in are generalization of this "many-one reducibility". We say that A is *truth-table reducible to* B (and write $A \leq_{\mathrm{tt}}^{\mathrm{P}} B$) if there are two polynomial-time computable functions, *generator* g that, for a given $x \in \Sigma^*$, produces a set of strings, and *evaluator* e that, when knowing which of the strings produced by g are in B, decides membership of x in A. That is, for any $x \in \Sigma^*$,

$$x \in A \iff e(x, g(x), g(x) \cap B) = 1,$$

where we assume that $g(x)$ (resp., $g(x) \cap B$) is encoded as a string by some appropriate coding method. For any $b(n) \geq 0$, we say that A is *$b(n)$-truth-table reducible to* B (and write $A \leq_{b(n)\text{-tt}}^{\mathrm{P}} B$) if the generator g produces at most $b(n)$ strings for each input of length n. If A is $\leq_{k\text{-tt}}^{\mathrm{P}}$-reducible to B for some constant $k \geq 0$, we say that A is *bounded-truth-table reducible to* B (and write $A \leq_{\mathrm{btt}}^{\mathrm{P}} B$). A set H is *$\leq_{b(n)\text{-tt}}^{\mathrm{P}}$-hard* (resp., *$\leq_{\mathrm{btt}}^{\mathrm{P}}$-hard*) for NP if every NP set is $\leq_{b(n)\text{-tt}}^{\mathrm{P}}$- (resp., $\leq_{\mathrm{btt}}^{\mathrm{P}}$-) reducible to H.

P-selective sets were introduced by Selman [Sel79] as the polynomial-time analog of semi-recursive sets [Joc68]. A set A is *P-selective*, if there exists a polynomial-time computable function f, called a *P-selector for* A, such that for all $x, y \in \Sigma^*$,

(1) $f(x, y) \in \{x, y\}$, and
(2) if $x \in A$ or $y \in A$, then $f(x, y) \in A$.

Intuitively, f selects the one of two given strings that is "more likely" to be in A. More formally, if $f(x, y) = x$ and $y \in A$, then $x \in A$.

Ko [Ko83] showed that from the P-selector function f of a P-selective set A, one can define a linear ordering on a quotient of Σ^* such that A is the union of an initial segment of this ordering. Toda [Tod91] modified this to an ordering on a given finite set Q (instead of Σ^*). Here, we use this ordering. That is, the relation $\preceq_{f,Q}$ on Q is defined as follows: For all $x, y \in Q$,

$$x \preceq_{f,Q} y \iff \exists\, z_1, \ldots, z_n \in Q : \begin{array}{l} f(z_i, z_{i+1}) = z_i \text{ for } i = 1, \ldots, n-1, \\ f(x, z_1) = x, \text{ and } f(z_n, y) = z_n. \end{array}$$

Define $x \cong_{f,Q} y \iff x \preceq_{f,Q} y \wedge y \preceq_{f,Q} x$. Then $\cong_{f,Q}$ is an equivalence relation on Q, and $\preceq_{f,Q}$ induces a linear ordering on the quotient $Q/\cong_{f,Q}$. This is reflected by the following partial ordering $\prec_{f,Q}$ on Q:

$$x \prec_{f,Q} y \iff x \preceq_{f,Q} y \wedge x \ncong_{f,Q} y.$$

For simplicity, we omit the subscripts f and Q when both are clear from the context. For technical reasons, we introduce a minimum and a maximum element, denoted as $\bot$ and $\top$ respectively, such that $\bot \prec x \prec \top$, for all $x \in Q$.

It is easy to see that the relations $\prec$ and $\cong$ are decidable in polynomial-time in $\sum_{x \in Q} |x|$. The crucial point is that $A \cap Q$ is an initial segment of Q with respect to $\preceq$. That is, we have

$$(*)\ \exists z \in Q \cup \{\bot\}:\ Q \cap A = \{y \in Q \mid y \preceq z\} \text{ and } Q \cap \overline{A} = \{y \in Q \mid y \succ z\}.$$

We call a string z witnessing $(*)$ a *cutpoint* of A in Q (with respect to $\preceq$). A consequence of this property is that $\forall x, y \in Q:\ x \preceq y \wedge y \in A \Longrightarrow x \in A$.

3 Main Result

Here, we state the proof of our main result. We begin by recalling some notion and result that will be used in our proof.

Definition 3.1. [ESY84] A *promise problem* is a pair of sets (Q, R). A set L is called a *solution* of the promise problem (Q, R), if for all $x \in Q$ we have $x \in R \iff x \in L$.

Toda [Tod91] showed that if all NP sets are $\leq^{\mathrm{P}}_{\mathrm{tt}}$-reducible to some P-selective set, then the promise problem $(1SAT, SAT)$ has a solution in P, where $1SAT$ is the set of Boolean formulas that have at most one satisfying assignment. We restate his theorem in a slightly more general form.

Theorem 3.2. [Tod91] If NP $\subseteq \mathrm{R}^{\mathrm{P}}_{\mathrm{tt}}$(SELECT), then, for any NP machine N the promise problem $(1L(N), L(N))$ has a solution in P, where $1L(N)$ is the set of strings x such that N has at most one accepting path on input x. Furthermore, if N is $p(n)$-time bounded, then the solution is in DTIME$(q_{\mathrm{T}}(p(n)))$, for some fixed polynomial q_{T}.

Now, we prove our main theorem.

Theorem 3.3. If there exists a P-selective set that is $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard for NP, then NP $\subseteq$ DTIME$(2^{n^{O(1/\sqrt{\log n})}})$.

Remark The following proof also works for some "unbounded" reducibility. For example, the theorem is provable for $\leq^{\mathrm{P}}_{(\log n)^{O(1)}\text{-tt}}$-reducibility.

Proof. Let us first define two NP sets. The first one is similar to the canonical universal NP-complete set except that the number of nondeterministic steps is stated explicitly.

$$\begin{aligned} UNIV \;=\; \{\;\; & (M, x, 0^d, 0^t) \mid \text{there exists } w \in \Sigma^d \text{ such that} \\ & \text{DTM } M \text{ accepts input } (x, w) \text{ in at most } t \text{ steps } \}. \end{aligned}$$

A string $w \in \Sigma^d$ is called a *nondeterministic path* of $(M, x, 0^d, 0^t)$, and if w witnesses $(M, x, 0^d, 0^t) \in UNIV$, it is called an *accepting path* of $(M, x, 0^d, 0^t)$. Obviously, $UNIV$ is NP-complete.

Our second set is defined similarly except that it has, as an additional component, the prefix of an accepting path for the considered machine.

$$\begin{aligned} PrefixPATH \;=\; \{\;\; & (M, x, 0^d, 0^t, u) \mid \text{there exists } v \in \Sigma^{d-|u|} \text{ such that} \\ & \text{DTM } M \text{ accepts input } (x, uv) \text{ in at most } t \text{ steps } \}. \end{aligned}$$

Consider any instance $\tau = (M, x, 0^d, 0^t)$ for $UNIV$, and let it be fixed for a while. We can regard τ's nondeterministic paths as paths in some binary tree T. That is, T is a binary tree whose nodes are of the form (τ, u), for $u \in \Sigma^{\leq d}$. T's root is (τ, λ) (where λ is empty string), and T's leaves are nodes (τ, u) such that $|u| = d$. A binary string $u \in \Sigma^{\leq d}$ is regarded as a path from the root to (τ, u). A string $w \in \Sigma^d$ is called an *accepting path* of T if M accepts input (x, w). Clearly, $\tau \in UNIV$ if and only if there exists an accepting path in T.

Let c and e be some integers that will be specified later. Below, we define $c^{\lceil d/e \rceil}$ subtrees T_k of T in such a way that if there is an accepting path in T, then there exist a subtree T_k that has *exactly one* accepting path. That is,

$$\begin{aligned} \tau \in UNIV \;&\Longleftrightarrow\; \exists w \in \Sigma^d :\; w \text{ is an accepting path in } T \\ &\Longleftrightarrow\; \exists k \leq c^{\lceil d/e \rceil} :\; T_k \text{ has exactly one accepting path.} \end{aligned} \tag{3.3}$$

At this point, we can explain our proof idea; that is, the strategy of deciding whether $\tau \in UNIV$ in deterministic subexponential-time. Consider a promise problem $(1SubTREE, SubTREE)$, where $1SubTREE$ is the set of T_k with at most one accepting path, and $SubTREE$ is the set of T_k with an accepting path. Then since we are assuming that NP $\subseteq \mathrm{R}^{\mathrm{P}}_{\mathrm{tt}}$(SELECT), by Theorem 3.2, this promise problem has a solution in P. Thus if T_k has exactly one accepting path, we can verify it in polynomial-time. Hence, the second condition of (3.1) is an NP-type predicate. Thus, we now have two NP-type predicates in (3.1) for deciding whether $\tau \in UNIV$. While there are 2^d possibilities for w, we can reduce the scope of k by choosing e large; in other words, while d (binary) nondeterministic guesses are necessary for the first NP-type predicate, $\frac{d \log c}{e}$ guesses are enough for the second. On the other hand, enlarging e will increase the time to decide the promise problem. We will see below that by appropriately choosing e (i.e., $e \approx \log c \log n$), we can reduce the number of nondeterministic guesses by about $1/\log n$ factor,

without increasing the time to decide the promise problem so much. That is, the original NP-type predicate is reduced to a simpler one. By iterating this process, we can finally solve the problem without using guesses, i.e., deterministically, and we will see that the whole process can be done in subexponential-time.

Let us define the subtrees more precisely. For this, we divide T into "blocks" of depth e. In other words, for each h $(0 \leq h \leq \lceil d/e \rceil - 1)$ and $u \in \Sigma^{h \cdot e}$, we consider[2] a set $X(\tau, u) = \{ (\tau, uv) \mid v \in \Sigma^e \}$ of nodes in T, which is regarded as a block of depth e. Notice that if $(\tau, u) \in \mathit{PrefixPATH}$, then some elements of $X(\tau, u)$ also belong to $\mathit{PrefixPATH}$. Here, for the decomposition of T satisfying (3.1), we would like to divide $X(\tau, u)$ into $X_1(\tau, u), \ldots, X_r(\tau, u)$ so that (if $(\tau, u) \in \mathit{PrefixPATH}$ then) some $X_i(\tau, u)$ has exactly one element in $\mathit{PrefixPATH}$. Key point of our proof is that this is possible by using the assumption that $\mathit{PrefixPATH}$ $(\in \mathrm{NP})$ is $\leq^{\mathrm{P}}_{btt}$-reducible to some P-selective set. That is, we have the following lemma.

Key Lemma. Let L be any set that is $\leq^{\mathrm{P}}_{b\text{-}tt}$-reducible to some P-selective set. Then for any set X of strings of length n, there exist r disjoint subsets $X_1, \ldots, X_r$ of X (where $r \leq 6(\lfloor b/2 \rfloor + 1) - 1$), with following property.

$$X \cap L \neq \emptyset \iff \exists i \in \{1, \ldots, r\} : \; \| X_i \cap L \| = 1. \tag{3.4}$$

Furthermore, we can compute $X_1, \ldots, X_r$ in polynomial-time w.r.t. n and $\| X \|$.

Since $\mathit{PrefixPATH}$ is in NP, it is $\leq^{\mathrm{P}}_{b\text{-}tt}$-reducible to some P-selective set by assumption. Thus, from this lemma (with $L = \mathit{PrefixPATH}$ and $X = X(\tau, u)$) we can divide each $X(\tau, u)$ into c disjoint subsets $X_1(\tau, u), \ldots, X_c(\tau, u)$ of $X(\tau, u)$ such that

$$\begin{aligned} &(\tau, u) \in \mathit{PrefixPATH} \\ &\iff \exists j \in \{1, \ldots, c\} : \; X_j(\tau, u) \text{ has exactly one element in } \mathit{PrefixPATH}, \end{aligned} \tag{3.5}$$

where $c = 6(\lfloor b/2 \rfloor + 1) - 1$. (For simplifying our discussion, we assume that $X(\tau, u)$ is always divided into exactly c subsets; we may assume this by defining $X_j(\tau, u) = \emptyset$, for all j, $r < j \leq c$.) An important point to note here is that c does not depend on e.

In order to define subtrees T_k of T, we assign an integer label k to each node (τ, u), where $u = v_1 v_2 \cdots v_h$ for some $1 \leq h \leq \lceil d/e \rceil$ and $v_1, \ldots, v_h \in \Sigma^e$. The label of (τ, u) is determined by the *history* $(j_1, \ldots, j_h)$ of indices, where each j_i $(1 \leq i \leq h)$ is the index such that $(\tau, v_1 \cdots v_i) \in X_{j_i}(\tau, v_1 \cdots v_{i-1})$. Since the sets $X_1(\tau, v_1 \cdots v_{i-1})$, $\ldots$, $X_c(\tau, v_1 \cdots v_{i-1})$ are pair-wise disjoint, each node has an unique history and label. Note that each history is expressed as a path (from the root to some node) of a c-ary tree. We give numbers to nodes of the c-ary tree as in Figure 3.1 and regard them as history labels. Then each node of T is given the label associated with its history.

---

[2] Precisely speaking, when $|u| = (\lceil d/e \rceil - 1)e$ (i.e., $h = \lceil d/e \rceil - 1$), $X(\tau, u)$ should be $\{ (\tau, uv) \mid v \in \Sigma^{d-|u|} \}$. In the following, we omit explaining such exceptional cases.

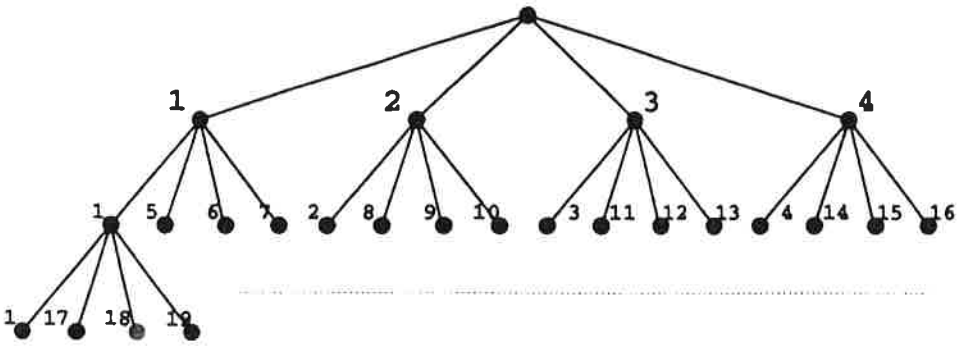

Figure 3.1: 4-ary tree and its labeling

Define $history(\tau, u)$ to be the history $(j_1, \ldots, j_h)$ of (τ, u), and define $label(j_1, \ldots, j_h)$ to be the label of the history $(j_1, \ldots, j_h)$ in the c-ary tree. Then $label(\tau, u)$ is defined as $label(history(\tau, u))$. It is easy to see that the label of (τ, u) is bounded by c^{h+1}; that is, no node in T is assigned a label $> c^{\lceil d/e \rceil}$. Now, for each k, $1 \leq k \leq c^{\lceil d/e \rceil}$, define T_k is the subtree of T consisting of all nodes with label k and their father nodes. Then from (3.3), we have the following.

Claim 1. There exists an accepting path in T if and only if for some k, $1 \leq k \leq c^{\lceil d/e \rceil}$, T_k has exactly one accepting path.

Next, consider the following set.

$$SubTREE_e = \{ \ (M, x, 0^d, 0^t, k) \mid 1 \leq k \leq c^{\lceil d/e \rceil} \text{ and } T_k \text{ has an accepting path,} \\ \text{where } T_k \text{ is the subtree of } T \text{ defined by } (M, x, 0^d, 0^t) \ \}.$$

For each e, we could now solve the promise problem $(1SubTREE_e, SubTREE_e)$ deterministically in polynomial-time (Theorem 3.2). But we should be careful about the polynomial-time bound, which depends on the choice of e. Precisely speaking, $(1SubTREE_e, SubTREE_e)$ has the following upper bound.

Claim 2. For some polynomial q_S, and for all $e \geq 1$, there exists a DTM M_e such that (i) M_e is $q_S(n + 2^e)$-time bounded, and (ii) $L(M_e)$ is a solution of $(1SubTREE_e, SubTREE_e)$. (That is, for every input $\eta = (M, x, 0^d, 0^t, k)$, (i) M_e halts in $q_S(|\eta| + 2^e)$ steps, and (ii) if $\eta \in 1SubTREE_e$, then $\eta \in SubTREE_e \iff M_e$ accepts η.)

Thus, we reached our goal to reduce the scope of the existential quantifier; that is, $(\tau \in UNIV \iff) \ \exists w \in \Sigma^d : w$ is an accepting path in $T \iff \exists k \leq c^{\lceil d/e \rceil} : (\tau, k) \in L(M_e)$. Here notice that we can easily translate our reduced problem to a new instance for $UNIV$. Then we can apply the above construction recursively!

Claim 3. For any e, there exists some $\tilde{M}_e$ such that for every input $\tau = (M, x, 0^d, 0^t)$,

$$\tau \in UNIV \iff (\tilde{M}_e, \tau, 0^{d'}, 0^{t'}) \in UNIV,$$

where $d' = \lceil \log c \rceil \cdot \lceil d/e \rceil$, and $t' = q_U(|\tau| + 2^e)$ for some fixed polynomial q_U.

Note that although the time bound t increases to t', the crucial point is that the number of nondeterministic steps d' decreases about a factor $\frac{\log c}{e}$.

Finally, to show that every NP set L belongs to DTIME($2^{n^{O(1/\sqrt{\log n})}}$), let M_L be a DTM and p_L be a polynomial such that for every $x \in \Sigma^*$, $x \in L \iff (M_L, x, 0^{p_L(|x|)}, 0^{p_L(|x|)}) \in UNIV$.

Let x be any string for which we want to decide membership in L. Let $n = |x|$, and we define $e = \lceil 3\delta(n) \log c \rceil$, where function δ will be chosen appropriately at the end of the proof. (We assume that n is large enough so that $e \geq 3$.) First define $x_0 = x$, $d_0 = p_L(n)$, $t_0 = p_L(n)$, and $\tau_0 = (M_L, x_0, 0^{d_0}, 0^{t_0})$. For each $i \geq 1$, define inductively $x_i = \tau_{i-1}$, $d_i = \lceil \log c \rceil \cdot \lceil d_{i-1}/e \rceil$, $t_i = q_U(|\tau_{i-1}| + 2^e)$, and $\tau_i = (\tilde{M}_e, x_i, 0^{d_i}, 0^{t_i})$, until $d_i < e$ ($= \lceil 3\delta(n) \log c \rceil$). Let m be the first integer such that $d_m < e$. Then from Claim 3, we have $\tau_0 \in UNIV \iff \tau_1 \in UNIV \iff \cdots \iff \tau_m \in UNIV$. On the other hand, $x \in L \iff \tau_0 \in UNIV$. Hence, $x \in L \iff \tau_m \in UNIV$. That is, the problem of deciding $x \in L$ is reduced to that of deciding $\tau_m \in UNIV$.

Let us evaluate the deterministic computation time for deciding $\tau_m \in UNIV$. First, we give an upper bound for t_m. Note that for some polynomial p_1, we have $|\tau_i| \leq p_1(t_i)$. Thus, $t_i = q_U(|\tau_{i-1}| + 2^e) \leq q_U(p_1(t_{i-1}) + 2^e) \leq q_U(p_1 \circ q_U(\cdots(p_1 \circ q_U(p_L(n) + 2^e))\cdots) + 2^e)$. Hence, for some constant c_1 and c_2, we have $t_m \leq n^{(c_1)^m} 2^{(c_1)^m e} = 2^{(c_1)^m (e + \log n)} \leq 2^{(c_1)^m (c_2 \delta(n) \log c + \log n)}$. On the other hand, note that for any $d \geq e \geq 3$, $d' = \lceil \log c \rceil \cdot \lceil d/e \rceil \leq (3d \log c)/e \leq d/\delta(n)$. Thus, $m \leq \log_{\delta(n)} d_0 \leq c_3 \log n / \log \delta(n)$, for some constant c_3. Therefore, for some c_4,

$$t_m \leq 2^{((c_1)^{\frac{c_3 \log n}{\log \delta(n)}})(c_2 \delta(n) \log c + \log n)} \leq 2^{(n^{\frac{c_4}{\log \delta(n)}})(c_2 \delta(n) \log c + \log n)},$$

which takes the smallest order when $\delta(n) = n^{1/\sqrt{\log n}}$. Thus, we define $\delta(n) = n^{1/\sqrt{\log n}}$; then for some constant c_5, we have $t_m \leq 2^{n^{c_5/\sqrt{\log n}}}$.

Clearly, "$\tau_m \in UNIV$?" is deterministically decidable in polynomial-time w.r.t. $|\tau_m|$. Also τ_m is deterministically computable in polynomial-time w.r.t. $|\tau_m|$. Recall that $|\tau_m| \leq p_1(t_m)$. Thus, the deterministic computation time for computing τ_m and deciding $\tau_m \in UNIV$ is polynomially bounded by t_m. Therefore, with some constant c_0, it is bounded by $2^{n^{c_0/\sqrt{\log n}}}$. That is, $x \in L$ is deterministically decidable in $2^{n^{c_0/\sqrt{\log n}}}$ steps. □

It remains to prove the Key Lemma.

Proof of Key Lemma. Let g and e be the generator and the evaluator of a $\leq^P_{b\text{-}tt}$-reduction from L to a P-selective set A, and let f be a P-selector for A. Define Q to be the set of queries

to A for all $x \in X$; that is, $Q = \bigcup_{x \in X} g(x)$. Let $\preceq$ denote $\preceq_{f,Q}$. Notice that $\preceq$ is polynomial-time decidable w.r.t. n and $||X||$.

For any $u, v \in Q \cup \{\bot, \top\}$, an *interval* $[u, v)$ is a set $\{ w \in Q \mid u \preceq w \prec v \}$. For any set $\mathcal{I}$ of intervals, we simply write $\bigcup \mathcal{I}$ for $\bigcup_{I \in \mathcal{I}} I$.

For each $x \in X$, we can define an associated set of intervals in Q that characterizes the membership of x in L according to a cutpoint of A in Q. More formally, letting $g(x) = \{ y_1 \preceq \cdots \preceq y_h \}$ (where $h \leq b$), $y_0 = \bot$, and $y_{r+1} = \top$, we define

$$\mathcal{I}_x = \{ [y_i, y_{i+1}) \mid e(x, g(x), \{y_1, \ldots, y_i\}) = 1, \text{ where } i \in \{0, \ldots, h\} \}.$$

If two adjacent intervals, i.e., $[y_i, y_{i+1})$ and $[y_{i+1}, y_{i+2})$, belong to $\mathcal{I}_x$, we regard them as one interval $[y_i, y_{i+2})$. Note that each $\mathcal{I}_x$ has at most $\lfloor b/2 \rfloor + 1$ intervals.

Let $J_x = \bigcup \mathcal{I}_x$, $J = \bigcup_{x \in X} J_x$, and let z_* be a cutpoint of A in Q. Then, for all $x \in X$, we have $x \in L \iff z_* \in J_x$, and hence, $X \cap L \neq \emptyset \iff z_* \in J$.

By Combinatorial Lemma stated below, we can select r subsets $X_1, \ldots, X_r$ of X such that

$$\forall z \in J,\ \exists i \in \{1, \ldots, r\},\ \exists x! \in X_i :\ z \in J_x,$$

where $r \leq 6(\lfloor b/2 \rfloor + 1) - 1$.

Now, we show that $X_1, \ldots, X_r$ have property (3.2). Clearly, the only-if direction (i.e., $\Longleftarrow$) holds. Thus, it suffices to consider the if direction. Suppose that $X \cap L \neq \emptyset$. Hence, $z_* \in J$ Then, from the above property of $X_1, \ldots, X_r$, there exists some X_i that has exactly one x such that $z_* \in J_x$. This means that X_i has exactly one element (namely, x) in L. (Recall that $x \in L \iff z_* \in J_x$.) Therefore, $||X_i \cap L|| = 1$. □

Combinatorial Lemma. Let $\{\mathcal{I}_x\}_{x \in X}$ be any family of sets of intervals in Q, where the index set X is finite, and each $\mathcal{I}_x$ consists of at most ℓ intervals. Let $\mathcal{I}$ be the set of intervals appearing in $\mathcal{I}_x$ for some $x \in X$; i.e., $\mathcal{I} = \{ I \mid I \in \mathcal{I}_x \text{ for some } x \in X \}$. Let $J = \bigcup \mathcal{I}$ and $J_x = \bigcup \mathcal{I}_x$. Then there exist $r \leq 6\ell - 1$ disjoint subsets $X_1, \ldots, X_r$ of X such that

$$\forall z \in J,\ \exists i \in \{1, \ldots, r\},\ \exists! x \in X_i :\ z \in J_x.$$

Furthermore, if $\preceq$ is polynomial-time computable w.r.t. $\sum_{u \in Q} |u|$, then the selection of $X_1, \ldots, X_r$ can be done in polynomial-time w.r.t. ℓ, $||X||$, and $\sum_{u \in Q} |u|$.

(The proof of this lemma is given in Appendix.)

References

[AHH+93] V. Arvind, Y. Han, L. Hemachandra, J. Köbler, A. Lozano, M. Mundhenk, M. Ogiwara, U. Schöning, R. Silvestri, and T. Thierauf. Reductions to sets of low infor-

mation content. *Recent Developments in Complexity Theory.* Cambridge University Press, to appear 1993. (Available as Technical Report TR-417, University of Rochester, Department of Computer Science, Rochester, NY, April 1992.)

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings 1st Structure in Complexity Theory Conference,* 1–11, IEEE Computer Society, 1986.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1988).

[BDG91] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II.* EATCS Monographs on Theoretical Computer Science, Springer-Verlag (1991).

[Bei88] R. Beigel. NP-hard sets are P-superterse unless R = NP. Technical Report 88-04, Department of Computer Science, The John Hopkins University, 1988.

[CLR90] T. Corman, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, McGraw-Hill Book Company, 1990.

[ESY84] S. Evan, A. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61:114–133, 1984.

[HHO+93] L. Hemachandra, A. Hoene, M. Ogiwara, A. Selman, T. Thierauf, and J. Wang. Selectivity. To appear in *Proceedings of the 5th International Conference on Computing and Information,* 1993.

[HOW92] L. Hemachandra, M. Ogiwara, and O. Watanabe. How hard are sparse sets? In *Proc. 7th Structure in Complexity Theory Conference*, IEEE 222–238, 1992.

[JT93] B. Jenner and J. Torán. Computing functions with parallel queries to NP. In *Proc. 8th Structure in Complexity Theory Conference*, IEEE 280–291, 1993.

[Joc68] C. Jockusch. Semirecursive sets and positive reducibility. *Transactions of the AMS*, 131(2):420–436, 1968.

[KL82] R. Karp, R. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191-209, 1982.

[Ko83] K. Ko. On self-reducibility and weak P-selectivity. *Journal of Computer and System Sciences*, 26:209–221, 1983.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[Mah82] S. Mahaney. Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences* 25:130-143, 1982.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal on Computing*, 20(3):471–483, 1991.

[Sal92] S. Saluja. Relativized limitations of the left set technique and closure classes of sparse sets. Manuscript.

[Sch90] U. Schöning. The power of counting. In *Complexity Theory Retrospective* (A. Selman Ed.), Springer-Verlag (1990), 204–223.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel82a] A. Selman. Analogues of semirecursive sets and effective reducibilities to the study of NP complexity. *Information and Control*, 52:36–51, 1982.

[Sel82b] A. Selman. Reductions on NP and P-selective sets. *Theoretical Computer Science*, 19:287–304. 1982.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science* 3:1–22, 1977.

[Tod91] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Mathematical Systems Theory*, 24:69–82, 1991.

[Tor93] j. Torán. Personal Communication.

[Val76] L. Valiant. Relative complexity of checking and evaluating. *Information Processing Letters*, 5(1):20-23, 1976.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science* 47:85–93, 1986.

[Wat90] O. Watanabe. Unpublished note.

Appendix

Here, we give the proof of the Combinatorial Lemma, which itself is of some interest.

Combinatorial Lemma. Let $\{\mathcal{I}_x\}_{x\in X}$ be any family of sets of intervals in Q, where the index set X is finite, and each $\mathcal{I}_x$ consists of at most ℓ intervals. Let $\mathcal{I}$ be the set of intervals appearing in $\mathcal{I}_x$ for some $x \in X$; i.e., $\mathcal{I} = \{ I \mid I \in \mathcal{I}_x \text{ for some } x \in X \}$. Let $J = \bigcup \mathcal{I}$ and $J_x = \bigcup \mathcal{I}_x$. Then there exist $r \leq 6\ell - 1$ disjoint subsets $X_1, \ldots, X_r$ of X such that

$$\forall z \in J,\ \exists i \in \{1, \ldots, r\},\ \exists! x \in X_i :\ z \in J_x. \tag{A.1}$$

Furthermore, if $\preceq$ is polynomial-time computable w.r.t. $\sum_{u\in Q} |u|$, then the selection of $X_1, \ldots, X_r$ can be done in polynomial-time w.r.t. ℓ, $||\, X \,||$, and $\sum_{u\in Q} |u|$.

Proof. First we construct a minimum size cover of $\mathcal{I}$. We say that $\hat{\mathcal{I}}$ is a *minimum size cover* of $\mathcal{I}$ if (i) $\hat{\mathcal{I}} \subseteq \mathcal{I}$, (ii) $\bigcup \hat{\mathcal{I}} = J$, and (iii) no $\mathcal{I}'$ such that $||\mathcal{I}'|| < ||\hat{\mathcal{I}}||$ satisfy both (i) and (ii). From $\mathcal{I}$, we can construct such a set in polynomial-time.

Claim 1. There is a polynomial-time algorithm that takes Q, $\preceq$ (by a table), and $\{\mathcal{I}_x\}_{x\in X}$ as input, and computes a minimum size cover of $\mathcal{I}$.

A related problem is the *activity-selection problem* [CLR90] (which is a maximization problem) that can be solved by a greedy algorithm. One can easily modify this algorithm to compute minimum size covers.

Next, we consider some graphs defined from $\hat{\mathcal{I}}$. For each $I \in \hat{\mathcal{I}}$, define $support(I)$ to be an index x such that $I \in \mathcal{I}_x$, and let $support(\hat{\mathcal{I}}) = \{\, support(I) \mid I \in \hat{\mathcal{I}} \,\}$. (If there are many indices x such that $I \in \mathcal{I}_x$, choose one for $support(I)$.) First consider the following directed (simple) graph $G' = (V, E')$.

$$\begin{aligned} V &= support(\hat{\mathcal{I}}), \text{ and} \\ E' &= \{\, (x', x) \mid \exists I \in \hat{\mathcal{I}} :\ support(I) = x, \text{ and } J_{x'} \cap I \neq \emptyset \,\}. \end{aligned}$$

Then G' has the following property.

Claim 2. Every vertex of G' has an outdegree of at most $3\ell - 1$.

Proof. Notice first that every interval in $\mathcal{I}$ intersects with at most three intervals in $\hat{\mathcal{I}}$, since otherwise, one can define a cover of J that has less elements than $\hat{\mathcal{I}}$, contradicting Claim 1. Similarly, every interval in $\hat{\mathcal{I}}$ intersects with at most two intervals in $\hat{\mathcal{I}}$. On the other hand, each $x \in V$ has at least one interval in $\hat{\mathcal{I}}$ and thus at most $\ell - 1$ intervals not in $\hat{\mathcal{I}}$. Therefore, J_x intersects with at most $3(\ell - 1) + 2\ (= 3\ell - 1)$ intervals in $\hat{\mathcal{I}}$. □ Claim 2

Next consider the underlying undirected graph G of G'; that is, G is specified as (V, E), where $E = \{\, \{x, x'\} \mid \text{either } (x, x') \in E \text{ or } (x', x) \in E \,\}$. The crucial property of G is that it is $6\ell - 1$-colorable.

Claim 3. G is $6\ell-1$-colorable. That is, there exists a partition $V_1, \ldots, V_r$ of V, where $r \leq 6\ell-1$, such that every V_i forms an independent set in G. Furthermore, some polynomial-time algorithm computes the partition from a given G.

Proof. First we show that G has the following property.

(*) Every subgraph of G has a vertex with degree at most $6\ell - 2$.

Consider any subgraph $\hat{G} = (\hat{V}, \hat{E})$ of G. From the definition of G', it is clear that $\hat{G}$ has at most $(3\ell-1)||\hat{V}||$ edges; that is, the sum of the degrees of all vertices is at most $2(3\ell-1)||\hat{V}||$. Hence, there is a vertex with degree at most $2(3\ell-1) = 6\ell-2$.

Then the claim is immediate from the fact that any graph G satisfying (*) is $6\ell-1$-colorable. Indeed, G is colorable by a simple greedy algorithm that colors vertices in order of their degree (from the largest ones). This fact is provable by induction on the size of G. Let x be the vertex of G that is colored last by the algorithm, and let $\hat{G}$ be the subgraph of G obtained by deleting x from G. Then by induction, the algorithm colors $\hat{G}$ correctly; thus, the algorithm works correctly before coloring x. Now, since the degree of x is at most $6\ell - 2$, the algorithm will find a color for x. □ Claim 3

Now for each i, $1 \leq i \leq r$, define $X_i = V_i$. Let us see why $X_1, \ldots, X_r$ satisfy (A.1). Consider any $z \in J$. Since $\hat{\mathcal{I}}$ is a cover of J, there is some $I \in \hat{\mathcal{I}}$ containing z. Let $x = \mathit{support}(I)$, and let X_i be the subset containing x. Then from Claim 3, no other x' in X_i is adjacent to x. That is, $J_{x'} \cap I = \emptyset$. Therefore, there exists exactly one index (namely x) in X_i such that $z \in J_x$.

Finally, we note that $X_1, \ldots, X_r$ are polynomial-time computable (if $\preceq$ is polynomial-time decidable), which is clear from the above discussion. □