On Complete Problems for NP∩CoNP[1]

Juris Hartmanis[2]
Neil Immerman[3]
TR 85-670

Department of Computer Science
Cornell University
Ithaca, New York 14853

---

[1] Presented at ICALP '85. July 1985, Natplion, Greece.
[2] Supported in part by National Science Foundation Grant DCR8301766.
[3] Reseach supported by an NSF Postdoctoral Fellowship.

On Complete Problems for $NP \cap CoNP$

Juris Hartmanis*
Department of Computer Science
Cornell University
Ithaca, NY 14853

Neil Immerman+
Department of Computer Science
Yale University
New Haven, CT 06520

Abstract

It is not known whether complete languages exist for $NP \cap CoNP$ and Sipser has shown that there are relativizations so that $NP \cap CoNP$ has no $\leq_m^P$-complete languages. In this paper we show that $NP \cap CoNP$ has $\leq_m^P$-complete languages if and only if it has $\leq_T^P$-complete languages. Furthermore, we show that if a complete language L_0 exists for $NP \cap CoNP$ and $NP \cap CoNP \neq NP$ then the reduction of $L(N_i) \in NP \cap CoNP$ cannot be effectively computed from N_i. We extend the relativization results by exhibiting an oracle E such that $P^E \neq NP^E \cap CoNP^E \neq NP^E$ and for which there exist complete languages in the intersection. For this oracle the reduction to a complete language can be effectively computed from complementary pairs of machines (N_i, N_j) such that $L(N_i) = \overline{L(N_j)}$. On the other hand, there also exist oracles F such that $P^F \neq NP^F \cap CoNP^F \neq NP^F$ for which the intersection has complete languages, but the reductions to the complete language cannot be effectively computable from the complementary pairs of machines. In this case, the reductions can be computed from

$$(N_i, N_j, \text{Proof that } L(N_i) = \overline{L(N_j)}) \quad .$$

Introduction

It is not known whether $P \neq NP$ or $P \neq NP \cap CoNP \neq NP$. Furthermore, it is not known whether complete languages exist in $NP \cap CoNP$. In this paper we investigate the possible existence of complete languages in $NP \cap CoNP$ and study their properties, should they exist.

*Supported in part by National Science Foundation Grant DCR8301766.
+ Research supported by an NSF Postdoctoral Fellowship.

Sipser [Si] has shown that there exists an oracle C such that $P^C \neq NP^C \cap CoNP^C \neq NP^C$ and that $NP^C \cap CoNP^C$ has no complete languages under many-one polynomial time reductions ($\leq_m^P$-complete). The existence of complete languages in the intersection was shown by collapsing P and NP by the well known BGS method [BGS].

In this paper we show that there exist $\leq_m^P$-complete languages for $NP \cap CoNP$ if and only if there exist polynomial-time Turing complete languages ($\leq_T^P$-complete). This extends Sipser's result to show that for his oracle C there are neither $\leq_m^P$-complete nor $\leq_T^P$-complete languages in the intersection. Furthermore, we show that there exists an oracle E such that $P^E \neq NP^E \cap CoNP^E \neq NP^E$ for which $NP^E \cap CoNP^E$ has complete languages. At the same time, we show that if $NP \cap CoNP \neq NP$ and $NP \cap CoNP$ has complete languages, then these languages have radically different behaviour than complete languages in NP.

More precisely, let $N_1, N_2, \ldots$ be a standard enumeration for polynomial time clocked nondeterministic Turing machines, let $P_1, P_2, \ldots$ be the corresponding enumeration of deterministic polynomial time clocked machines and let $M_1, M_2, \ldots$ be a standard enumeration of Turing machines. We first show that if $NP \cap CoNP \neq NP$ then there does not exist a recursive function which can bound for all $L(N_i)$ in $NP \cap CoNP$, in the size of N_i, $|N_i|$, the size of a complementary machine N_j such that $L(N_i) = \overline{L(N_j)}$. From this result it follows that if $NP \cap CoNP \neq NP$ and there is a complete language L_0 in the intersection then the reductions of $L(N_i) \in NP \cap CoNP$ to L_0 cannot be effectively computed from N_i. As a matter of fact, the size of the reducing machines is not recursively bounded in $|N_i|$.

We observe that for the above oracle E, the reductions to the complete language L_0 can be effectively computed from complementary pairs of machines (N_i, N_j) such that $L(N_i^E) = \overline{L(N_j^E)}$.

On the other hand, this is not always the case, and we exhibit another oracle F such that $P^F \neq NP^F \cap CoNP^F \neq NP$ and L_0 is a complete language for the intersection, but the reductions cannot be effectively computed from the complementary pairs (N_i, N_j) such that $L(N_i^F) = \overline{L(N_j^F)}$. The reductions can be computed from the pair of machines N_i, N_j and the proof that $L(N_i^F) = \overline{L(N_j^F)}$.

These results show that there is great logical latitude which permits different relativizations for $NP \cap CoNP$ and if $P \neq NP \cap CoNP \neq NP$ then $NP \cap CoNP$ behaves, in a strict technical sense, differently from NP with respect to complete languages: *either they do not exist or the reductions are not effectively computable.*

Complete Languages for $NP \cap CoNP$

We will now show that $NP \cap CoNP$ has a $\leq_T^P$-complete language if and only if it has a $\leq_m^P$-complete language.

It has been observed by Kowalczyk [Ko] that $NP \cap CoNP$ has a $\leq_m^P$-complete language if and only if there exists a recursively enumerable set of pairs of complementary machines $\{(N_{i_k}, N_{j_k}) \mid k \geq 1\}$ such that $L(N_{i_k}) = L(\overline{N_{j_k}})$ and $\{L(N_{i_k}) \mid k \geq 1\} = NP \cap CoNP$.

Theorem 1: There exists a $\leq_T^P$-complete language in $NP \cap CoNP$ if and only if there exists a $\leq_m^P$-complete language.

Proof: If there exists a $\leq_m^P$-complete language L_0 in $NP \cap CoNP$, then $NP \cap CoNP \subseteq P^{L_0}$ and we see that L_0 is also $\leq_T^P$-complete.

Conversely, if for some L_1 in $NP \cap CoNP$, $L(N_s) = L_1 = \overline{L(N_t)}$, we have $NP \cap CoNP \subseteq P^{L_1}$, then $NP \cap CoNP = P^{L_1}$. To see this, note that for each $L(P_i^{L_1})$ in P^{L_1} we can effectively construct an equivalent machine $N_{\sigma(i)}$. The machine $N_{\sigma(i)}$ simulates $P_i^{L_1}$ and for each oracle querry $N_{\sigma(i)}$ starts N_s and N_t on the querried string. Since $L(N_s) = \overline{L(N_t)} = L_1$, for each querry there is a computation path such that N_s or N_t accepts the querried string. Thus $L(N_{\sigma(i)}) = L(P_i^{L_1})$. As a matter of fact, for each $P_i^{L_1}$ we can effectively construct a pair of complementary machines $N_{\sigma(i)}$ and $N_{\zeta(i)}$ such that $L(P_i^{L_1}) = L(N_{\sigma(i)}) = \overline{L(N_{\zeta(i)})}$ and clearly, $\{L(N_{\sigma(i)}) \mid i \geq 1\} = NP \cap CoNP$. But then, by Kowalczyk's Theorem, or an explicit construction of a universal language of the type

$$\{N_{\sigma(i)} \# N_{\zeta(i)} \# x \# \text{ padding} \mid x \text{ is in } L(N_{\sigma(i)})\},$$

we obtain a $\leq_m^P$-complete language for $NP \cap CoNP$. □

Next we show that if $NP \cap CoNP \neq NP$ then there is no recursive succinctness bound in the size of N_i, for $L(N_i)$ in $NP \cap CoNP$, which bounds the size of a complementary machine N_j such that $L(N_i) = \overline{L(N_j)}$.

Succinctness Lemma: If $NP \cap CoNP \neq NP$ then there is no recursive function G such that for all N_i with $L(N_i)$ in $NP \cap CoNP$ there exists an N_j satisfying

$$L(N_i) = \overline{L(N_j)} \text{ and } G(|N_i|) \geq |N_j|.$$

Proof: First we show that

$$\Delta = \{N_i \mid L(N_i) \in NP - NP \cap CoNP\}$$

is a Π_2-complete set in the Kleene Hierarchy [Ro]. To see this, note that Δ can be expressed in a $\forall\exists$ form:

$$\Delta = \{N_i \mid (\forall N_s, N_t)(\exists x)[N_i(x) \neq N_s(x) \text{ or } N_s(x) = N_t(x)]\}.$$

Thus Δ is in Π_2. Its completeness in Π_2 is seen by reducing the Π_2-complete set $\{M_i \mid L(M_i) = 1^*\}$ to Δ as follows. Map M_i onto $N_{\sigma(i)}$, where $N_{\sigma(i)}$ has an n^3-clock and computes in stages. If the clock shuts $N_{\sigma(i)}$ off then the input is rejected. Let $\{(N_{t_k}, N_{s_k}) \mid k \geq 1\}$ be a recursive enumeration of all possible pairs of nondeterministic polynomial time machines.

For input x $N_{\sigma(i)}$ recomputes its previous computations to determine what stage it is in (if this computation is not completed in n^3 steps the input is rejected). In stage k, $k \geq 1$, $N_{\sigma(i)}$ simulates M_i on input 1^k, if it is found that M_i accepts 1^k, then $N_{\sigma(i)}$ checks deterministically on shorter inputs whether $L(N_{\sigma(i)}) \neq L(N_{t_k})$ or $L(N_{t_k}) \neq \overline{L(N_{s_k})}$ (if for all k one of these conditions is satisfied then $L(N_{\sigma(i)})$ is in $NP - NP \cap CoNP$). If neither condition is satisfied in n^2 steps of computation the input x is accepted if and only if x is a satisfiable Boolean formula in CNF, i.e. x is in SAT. Since we have assumed that $NP \cap CoNP \neq NP$, SAT is not in $NP \cap CoNP$ and eventually, for sufficiently long x, one of the above conditions will be found to be satisfied, when this is detected $N_{\sigma(i)}$ enters stage $k+1$.

From the above construction we see that if $L(M_i) = 1^*$ then for all k $L(N_{\sigma(i)}) \neq L(N_{t_k})$ or

$L(N_{t_k}) \neq \overline{L(N_{s_k})}$ and $L(N_{\sigma(i)}) \in NP - NP \cap CoNP$, otherwise $L(N_{\sigma(i)})$ is finite and therefore in $NP \cap CoNP$. Thus Δ is Π_2-complete.

If a recursive succinctness bound G would exist, bounding the size of a complementary machine N_j to the size of N_i for each $L(N_i)$ in $NP \cap CoNP$, then

$$\Delta = \{N_i \mid L(N_i) \neq \overline{L(N_j)} \text{ for all } N_j \text{ with } |N_j| \leq G(|N_i|)\}.$$

But then Δ would be recursively enumerable which is a contradiction to Π_2-completeness. Thus no recursive succinctness bound exists. □

Theorem 2: If $NP \cap CoNP \neq NP$ and L_0 is a $\leq_m^P$-complete language for $NP \cap CoNP$, then there is no recursive function G which bounds in the size of N_i, for $L(N_i)$ in $NP \cap CoNP$, the size of the minimal size P_j reducing $L(N_i)$ to L_0. Therefore the reduction of $L(N_i)$ in $NP \cap CoNP$ to L_0 cannot be effectively computable from N_i.

Proof: Let $L(N_{i_0}) = \overline{L(N_{j_0})} = L_0$. Then any P_t reducing $L(N_i)$ in $NP \cap CoNP$ to L_0 defines to machines $P_t \cdot N_{i_0}$ and $P_t \cdot N_{j_0}$ which accept $L(N_i)$ and $\overline{L(N_i)}$, respectively. Since the size of the smallest machine, N_j, accepting $\overline{L(N_i)}$ cannot be recursively bounded in the size of N_i and N_{j_0} is fixed, we see that the size of $P_t \cdot N_{j_0}$ cannot be recursively bounded in the size of N_i and therefore the size of the reducing machine, P_t, cannot be recursively bounded in $|N_i|$. But then P_t cannot be effectively computed from N_i. □

By similar reasoning we get the next result.

Theorem 3: Let L_1 be in $NP \cap CoNP$ and $P^{L_1} = NP \cap CoNP \neq NP$. Then there is no recursive function which bounds in the size of N_i, for $L(N_i)$ in $NP \cap CoNP$, the size of the smallest P_j such that $L(P_j^{L_1}) = L(N_i)$.

Proof: A careful inspection of the proof of Theorem 1 shows that a recursive succinctness bound in this result would imply a recursive succinctness bound for the previous result, contradicting Theorem 2. □

Relativized Computations

Let $h(n) = \log_* n$. We will use h in several places in our constructions; but, any sufficiently slowly growing unbounded function would do. We will assume for convenience in the next two theorems that deterministic polynomial time turing machine P_i and nondeterministic polynomial time turing machine N_i both have clocks constraining them to run in at most $n^{h(i)}$ steps on inputs of size n.

Theorem 4: There is an oracle E such that

$$P^E \neq NP^E \cap CoNP^E \neq NP^E$$

and such that there exists a many-one complete set for $NP^E \cap CoNP^E$.

Proof: We will construct E as the disjoint union of two sets:

$$E = 00S \cup 1T$$

The prefix for S is 00 in order to leave room for one more set in the next theorem. The set T will code a universal complete set for $NP^E \cap CoNP^E$; and the set S will hide strings in order to insure that $NP^E \neq CoNP^E$. For each string u, T will contain exactly one string uv where $|u| = |v|$. Thus the function f which takes each u to its corresponding v will be computable in $NP^E \cap CoNP^E$;

$$f(u) = v \text{ such that } |u| = |v| \text{ and } uv \in T \ .$$

We will say that the two turing machines N_i^E, N_j^E form a *complementary pair* if they accept complementary languages. We will construct T so that the following two conditions hold concerning f:

(A)The function f is not computable in P^E.

(B)Suppose that N_a^E, N_b^E is a complementary pair; and let $u = (a, b, w, \#^m)$ where $m = |w|^{h(\max(a, b))}$ bounds the number of steps taken by N_a or N_b on input w. Then for all but finitely many $w \in \Sigma^*$ first($f(u)$) - the first bit of $f(u)$ - will be 1 if and only if N_a^E accepts w.

Condition (B) will insure that the set

$$C(E) = \{(u, x) \mid (\exists v) 1uv \in E; \ \text{first}(v) = x\}$$

is many-one complete for $NP^E \cap CoNP^E$. Condition (A) will insure that $NP^E \cap CoNP^E \neq P^E$ because f is computable in $NP^E \cap CoNP^E$ but not in P^E. We break this condition into infinitely many pieces:

(A_i) The function f is not computed by P_i.

Condition (A_i) will be easy to meet because each P_i can only look at a few of the exponentially many candidates for each $f(u)$.

Finally in the set S we will hide strings to guarantee that $NP^E \neq CoNP^E$. Let

$$L(E) = \{0^m \mid m \in N, (\exists z)(\mid z \mid = m, 00z \in E)\} \quad .$$

The following conditions will insure that $L(E) \notin CoNP^E$ and thus that $NP^E \neq CoNP^E$.

(C_i) The language accepted by N_i is not $\overline{L(E)}$.

The Construction

We construct an oracle meeting the above conditions (A), (B), and (C). Assume inductively that E_s, an initial segment of E has been constructed, and that E_s only includes strings of length less than s.

Part (A): Let $i \leq s$ be minimal so that the condition (A_i) has not yet been satisfied. Simulate P_i on input 0^s. Let $w_1, \ldots, w_k$ be the strings of length at least s queried by P_i. Freeze these strings permanently out of E.

Part (B): Let $k \leq s$ be minimal so that the condition (C_k) has not yet been satisfied. For j going from s to $s^{k(k)}-1$ do the following: For each u with $\mid u \mid = j/2$ pick a unique v such that $\mid v \mid = \mid u \mid$ and add $1uv$ to E. The string v must meet the following two criteria:

- The string $1uv$ has not been previously frozen.
- If $u = (a, b, w, \#^m)$ where $m = \mid w \mid^{k(\max(a,b))}$ and N_a accepts w, (necessarily within m steps), then first(v) should be 1. Otherwise first(v) should be 0.

Note that the number of v's excluded by freezing in part (A) is less than $s^{k(s)-1}$ and therefore less than the number of possible v's, i.e. $2^{j/2}$. Thus the above two steps can always be carried out.

Part (C): Recall that k was chosen to be minimal so that the condition (C_k) has not yet been satisfied. Let $r = s^{k(k)}$ and assume that E_r has been constructed correctly so far. Run $N_k^{E_r}$ on input 0^s. If it rejects then freeze E_r. There are no strings of the form $00z \in E_r$ with $\mid z \mid = s$. Thus $N_k^{E_r}$ does not

accept the complementary language to $L(E)$. If $N_t^{E_r}$ accepts 0^s then pick one of its accepting computations and freeze the strings $y_1, \ldots, y_t$ queried in this computation. Note that

$$\sum_{j=1}^{t} |y_j| < s^{k(k)} = r$$

because N_k runs in at most $s^{k(k)}$ steps on inputs of size s. Note that some of the y_j's in E may be of the form $1u_jv_j$ where $u_j = (a, b, w, \#^m)$ with $N_a^{E_s}, N_b^{E_s}$ a valid $NP^{E_s} \cap CoNP^{E_s}$ pair on inputs of length less that s; and such that $m = |w|^{k(\max(a,b))}$. In this case if $N_a^{E_r}$ accepts w then pick one of its accepting computations and freeze it. Otherwise, if $N_b^{E_r}$ accepts w then pick one of its accepting computations and freeze it. In either case we may be forced to freeze some additional strings, $y_j^1, \ldots, y_{t_j}^j$, in E_r. Note that N_a runs in at most m steps, and $m < |y_j|/2$. If follows that

$$\sum_{l=1}^{t_j} |y_l^j| < |y_j|/2 \quad .$$

It easily follows that within $\log r$ interactions all strings to be frozen will have been identified; and that there are fewer than r of them. (In particular their total length is less than $2r$.) Finally, we now pick a string z of length of s such that $00z$ has not been frozen, and add $00z$ to E. Such a z exists because fewer than $2r$ of the 2^s possible z's have been excluded. It follows that we have met the condition that $L(N_t^E)$ is not the complementary language to $L(E)$.

Note that by adding $00z$ to E we may change many of the computations which have been recorded in T. Thus *we must repeat the loop of part* (B) to recompute E_r; but, of course we do not change those strings which have been frozen.

A problem may occur in that some frozen $y = uv$, with $u = (a, b, w, \#^m)$, may now be incorrect. To be incorrect means that $N_a^{E_r}$ and $N_b^{E_r}$ form a valid complementary pair so far; but that first(v) does not reflect the pair's answer on w. How could this have happened? Recall that if either N_a or N_b had accepted w then one of these accepting computations would have been frozen as well. It follows that for an error to have occurred, originally $N_a^{E_r}$ and $N_b^{E_r}$ must both have rejected w. To meet condition (B) we only have to insure that for each valid complementary pair, N_a, N_b, this sort of error happens only

finitely often. We insure this by modifying part (C) of the construction as follows:

(Proviso to Part C): **IF** E_r contains a string $y = 1uv$ with $u = (a, b, w, \#^m)$, such that $s \leq |y| < r$; $a + b < h(s)$; $N_a^{E_s}, N_b^{E_s}$ is a complementary pair up to s; and $N_a^{E_r}$ and $N_b^{E_r}$ both reject w **THEN** skip (C) this time.

The effect of the above proviso is that if a very small pair has the potential to mess up the construction then both parts of the pair must now reject some w. If we defer the (C) step and freeze the present initial piece of the oracle then the pair is no longer a complementary pair. It follows that the pair N_a, N_b will never be involved in an error after stage $h^{-1}(a + b)$.

Lemma 5: If the above construction is carried out using Proviso 1, then conditions (A), (B), and (C) are met.

Proof: We must only show that any complementary pair N_a, N_b can be put in error only finitely often; and that the conditions (C_k) are all eventually satisfied.

The first is clear because the pair N_a, N_b cannot be put in error on any input of length s, where $a + b < h(s)$.

It is clear that the second is true as long as the function $h(s)$ grows slowly enough; and a straight forward calculation shows that $h(s) = \log_* s$ does grow sufficiently slowly. This proves the lemma and thus the theorem. □

Corollary 6: There is an oracle E such that $P^E \neq NP^E \cap CoNP^E \neq NP^E$, and such that there is an $NP^E \cap CoNP^E$ complete set, L. Furthermore, the map taking a complementary pair N_a, N_b to its reduction to L is computable.

Proof: By the above construction we know that for N_a, N_b a complementary pair no errors are made after stage $s = h^{-1}(a + b)$. Let $u \in C(E), v \notin C(E)$. Suppose that N_a^E, N_b^E is a complementary pair. Let $e = \max(a, b)$. Put

$$r(w) = \begin{cases} (a,\ b,\ w,\ |w|^e,\ 1) & \text{if } h(w) > a + b \\ u & \text{if } h(w) \leq a + b \text{ and } N_a(w)\downarrow \\ v & \text{otherwise} \end{cases}$$

Then r is a valid reduction of $L(N_a)^E$ to $C(E)$. □

The next theorem shows that there is also an oracle for which there is still a complete set for $NP \cap CoNP$; but, such that the reductions cannot be computed from pairs of complementary machines.

Theorem 7: There is an oracle F such that

$$P^F \neq NP^F \cap CoNP^F \neq NP^F \quad ;$$

such that there exists a many-one complete set, C, for $NP^F \cap CoNP^F$; but, such that reductions are not computable. That is there is no F-computable map, φ^F, such that for each complementary pair, $N_a^F, N_b^F, \varphi^F(a, b)$ converges to an index of a polynomial time reduction of $L(N_a)$ to C.

Proof: We modify the above construction to include parts A, B, and C, as well as a new part D in order to satisfy the following new conditions:

(D_i): The ith partial recursive function, φ_i^F, does not compute reductions to $C(F)$.

In order to foil φ_i's chances of computing reductions, and thus satisfy (D_i) we use an idea from [Si]. For each i, and some $c \in N$ we will consider the following language:

$$L(i, c) = \{(i, c, w) \mid (\exists v.\ |v| = |w|)((01, i, c, v) \in F \text{ and first}(v) = 1)\} \ .$$

We will modify the construction so that for sufficiently many c we will put exactly one string z of the correct form into F. In these cases the language $L(i, c)$ is in $NP^F \cap CoNP^F$ and will be computed by an appropriate linear time complementary pair $N_{a(i,c)}, N_{b(i,c)}$. We will satisfy (P_i) by making sure that if $\varphi_i^F(a(i, c), b(i, c))$ converges to some polynomial time reduction P_j then P_j does not reduce $L(i, c)$ to $C(F)$.

We will say that D_i *is ready at stage* s if there exists an integer $c \leq s$ as well as integers a, b, j, m, x such that the following two conditions are met:

1. $\varphi_i^{F_s}(a(i, c), b(i, c))$ converges in at most s steps to some polynomial time transducer P_j.

2. $P_j^{F_s}((01, i, c, 0^s))$ converges in at most $s^{h(s)}$ steps to some string $(a, b, w, \#^m, x)$.

We now describe part (D) of the construction. As above we assume that F_s, an initial segment of F,

has been computed correctly. F_s includes only strings of length less than s. Let i be minimal so that D_i is ready at stage s; and let a, b, c, j, m, x be the associated values in the definition of being ready. Freeze those strings queried by $P_j^{F_s}((01, i, c, 0^s))$ permanently out of F. Now extend F_s to F_r where $r = s^{h(s)}$; but do this without putting any strings of the form $(01, i, c, v)$ into F where $|u| = s$. We will have foiled P_j as a reduction if the following equivalence does not hold:

$$(a, b, w, \#^m, x) \in C(F) \Leftrightarrow (01, i, c, 0^s) \in L(i, c) \qquad (1)$$

What we do is to freeze the string of the form $(1, a, b, w, \#^m, v) \in F_r$ where $|v| = |(1, a, b, w, \#^m)|$. Then we try to insure that this frozen answer is wrong. There are three cases of interest:

case 1: At least on of $N_a^{F_r}$, $N_b^{F_r}$ accepts w. In this case we freeze one of the accepting computations as in part (C).[1] Next we put a string of the form $(01, i, c, v)$ into F where $|v| = s$ and first(v) is chosen to make equation (1) false. Finally we recompute F_r.

case 2: Neither of $N_a^{F_r}$, $N_b^{F_r}$ accepts w; and $a + b \geq i$ or N_1, N_2 is not a correct pair up to s. In this case as above we put a string of the form $(01, i, c, v)$ into F where $|v| = s$ and first(v) is chosen to make equation (1) false. Again we recompute F_r. Note that in this case we may be putting the pair N_a, N_b in error, but we do this at most finitely many times for each pair.

case 3: Otherwise: neither element of the pair accepts w; and the pair is a complementary pair up to s with $a + b$ less than i. In this case we freeze F_r as it is. Thus N_a, N_b will not be a valid pair. Note that we are also causing $N_{a(i,c)}$, $N_{b(i,c)}$ to be an invalid pair. This is the cost of eliminating N_a, N_b but it can happen to φ_i at most i^2 times.

Claim: The construction described above successfully meets condition (D) as well as still satisfying conditions (A), (B) and (C).

Proof: To the reader who has made it through to this point it should be clear that part (D) can only put any given pair in error finitely often. Furthermore for any potentially correct φ_i there will be some

1 We are assuming that none of the strings in this frozen computation which are in F_r and of the form $(1, a', b', \ldots)$ are potentially in error, where furthermore $a' + b' < i$. Otherwise we freeze the error as in case 3.

stage

s at which (D_i) is the minimal ready condition, and all potentially complementary pairs N_a, N_b with $a + b < i$ which will ever be eliminated will already be gone. At this stage case 1 or case 2 will apply, and in either case we will have that P_j, the machine computed by φ_i, is not a valid reduction.

This proves the claim and thus the theorem. □

Acknowledgement

We would like to thank Jim Kadin for valuable discussions and in particular for the formulation of Theorem 1.

References

[BGS]Baker, T., J. Gill and R. Solovay. "Relativization of the $P = ?NP$ Question". *SIAM Journal of Computing* (1975), 431-442.

[GJ]Garey, M.R. and D.S. Johnson. "Computers and Intractability: A guide to the Theory of *NP*-Completeness". W.H. Freeman and Co., San Francisco, 1979.

[Ko]Kowalczyk, W. "Some Connections Between Representability of Complexity Classes and the Power of Formal Systems of Reasoning". Mathematical Foundations of Computer Science, *Lecture Notes in Computer Science 176.* Springer-Verlag 1984, 364-369.

[Ro]Rogers, H., Jr. "Theory of Recursive Functions and Effective Computability". McGraw-Hill, New York, 1967.

[Si]Sipser, M. "On Relativization and the Existence of Complete Sets". Automata Languages and Programming *Lecture Notes in Computer Science 140.* Springer-Verlag 1982, 523-531.