$P \neq NP \cap \text{co-}NP$ for Infinite Time Turing Machines

VINAY DEOLALIKAR, *Hewlett-Packard Research, 1501 Page Mill Road, M/S 3U-4, Palo Alto, CA 94304, USA.*
Email: vinay.deolalikar@hp.com

JOEL DAVID HAMKINS, *The College of Staten Island of CUNY & The CUNY Graduate Center, Mathematics Program, 365 Fifth Avenue, New York, NY 10016, USA.*
Email: jdh@hamkins.org

RALF SCHINDLER, *Institut für mathematische Logik und Grundlagenforschung, Universität Münster, Einsteinstr. 62, 48149 Münster, Germany.*
Email: rds@math.uni-muenster.de

Abstract

Extending results of Schindler, Hamkins and Welch, we establish in the context of infinite time Turing machines that P is properly contained in $NP \cap \text{co-}NP$. For higher analogues of these classes, we exhibit positive and negative results.

Keywords: Infinite time Turing machines, complexity theory, descriptive set theory.

1 Introduction

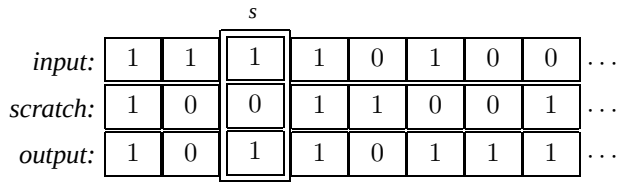
In this article, we take up the question of whether $P = NP \cap \text{co-}NP$ for infinite time Turing machines. The related $P = NP$ problem was first considered in connection with infinite time Turing machines by Schindler in [7], where he proved that $P \neq NP$ and introduced the other natural complexity classes P^+, NP^+, P^{++}, NP^{++}, P_α, NP_α, P^f and NP^f. He then showed that $P^+ \neq NP^+$ and posed the corresponding questions for P^{++} and P^f when f is a suitable function from $\mathbb{R}$ to the ordinals. Hamkins and Welch answered these questions in [3] by showing that $P^{++} \neq NP^{++}$ and, more generally, that $P^f \neq NP^f$ for almost every function f. Here, we extend the analysis of all these complexity classes to the analogues of the question of whether $P = NP \cap \text{co-}NP$ for infinite time Turing machines. Unfortunately, there is no uniform answer, as some of the complexity classes satisfy the equation and some do not, though the general tendency is towards inequality.

We show, in particular, that P is properly contained in $NP \cap \text{co-}NP$. Furthermore, $NP \cap \text{co-}NP$ is exactly the class of hyperarithmetic sets. At the next level, we establish $P^+ = NP^+ \cap \text{co-}NP^+ = NP \cap \text{co-}NP$. At a still higher level, once again P^{++} is properly contained in $NP^{++} \cap \text{co-}NP^{++}$. For α within any contiguous block of infinite clockable ordinals, we establish $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$, but if β begins a gap in the clockable ordinals, then $P_\beta = NP_\beta \cap \text{co-}NP_\beta$. Finally, for almost all functions $f : \mathbb{R} \to \text{ORD}$, the class P^f is properly contained in $NP^f \cap \text{co-}NP^f$, though there are functions for which $P^f = NP^f \cap \text{co-}NP^f$, even with $P^f \neq NP^f$.

Infinite time Turing machines were introduced by Hamkins and Lewis in [2], and we refer the

Vol. 15 No. 5,
doi:10.1093/logcom/exi022

reader to that article for reference and background. Let us quickly describe here for convenience how the machines operate. The hardware of an infinite time Turing machine is the same as that of



a classical three-tape Turing machine: a head moves left and right on a semi-infinite paper tape, reading and writing according to the rigid instructions of a finite program with finitely many states in exactly the classical manner. The operation of the machines is extended into transfinite ordinal time by defining the configuration of the machine at the limit ordinal stages. At such a stage, the head is returned to the leftmost cell, the machine is placed into the special *limit* state, and the tape is updated by placing into each cell the $\limsup$ of the values appearing in that cell before the limit stage. Thus, if the cell values have stabilized before a limit, then at the limit the cell displays this stabilized value, and otherwise, when the cell has changed from 0 to 1 and back again unboundedly often before the limit, then at the limit the cell displays a 1. Having specified the operation of the machines, one obtains for any program p the corresponding infinite time computable function φ_p, namely, $\varphi_p(x) = y$ when program p on input x halts with output y. The natural context for input and output is infinite binary sequences, that is, Cantor space ${}^{\omega}2$, which we refer to as the set of reals and denote by $\mathbb{R}$. A set $A \subseteq \mathbb{R}$ is infinite time decidable if its characteristic function is decidable. In the context of certain time-critical complexity classes, we adopt the formalism for deciding sets with two distinct halt states, *accept* and *reject*, so that the machines can announce their decisions as quickly as possible, without needing to position the head for writing on the output tape. For many of the complexity classes, however, including P, P^+, P^{++} and P_α for limit ordinals α and their successors, the additional steps required for writing on the tape pose no difficulty, and one can dispense with this formalism in favour of the usual characteristic function notion of decidability.

Many of our arguments will rely on elementary knowledge of descriptive set theory, and we refer readers to [6], [4] and [5] for excellent introductions. For background material on admissible set theory, we refer readers to [1]. We denote the first infinite ordinal by ω and the first uncountable ordinal by ω_1. Throughout the paper, we use ordinal as opposed to cardinal arithmetic in such expressions as ω^2 and ω^ω. The well-known ordinal ω_1^{CK}, named for Church and Kleene, is the supremum of the recursive ordinals (those that are the order type of a recursive relation on ω). The ordinal ω_1^{CK} is also the least *admissible* ordinal, meaning that the ω_1^{CK} level of Gödel's constructible universe $L_{\omega_1^{\text{CK}}}$ satisfies the Kripke–Platek (KP) axioms of set theory. We denote by ω_1^x the supremum of the x-recursive ordinals, and this is the same as the least x-admissible ordinal, meaning that $L_{\omega_1^x}[x] \models \text{KP}$.

An ordinal α is *clockable* if there is a computation of the form $\varphi_p(0)$ taking exactly α many steps to halt (meaning that the αth step moves into the *halt* state, so that the *halt* state first appears at stage $\alpha + 1$). Hamkins and Lewis [2, Theorem 3.4] established that the set of clockable ordinals is not an initial segment of the ordinals. An interval $[\alpha, \beta)$ forms a *gap* in the clockable ordinals if it contains no clockable ordinals and is maximal with this property. Thus, β is clockable and the clockable ordinals are unbounded in α. When $[\alpha, \beta)$ is a gap, [2, Lemma 3.3] shows that α and β must be limit ordinals, and we say that α is a *gap-starting* ordinal and β is a *gap-ending* ordinal. Hamkins and Lewis [2, Theorem 8.8] show that no admissible ordinal is clockable. A *writable* real is one

that is the output of a computation $\varphi_p(0)$. An ordinal is writable when it is coded by a writable real. The supremum of the writable ordinals is denoted λ, and by [10] this is equal to the supremum of the clockable ordinals. A real is *accidentally writable* when it appears on one of the tapes at same stage during a computation of the form $\varphi_p(0)$. The supremum of the accidentally writable ordinals, those that are coded by an accidentally writable real, is denoted Σ. A real is *eventually writable* if there is a computation of the form $\varphi_p(0)$ such that beyond some ordinal stage the real is written on the output tape (the computation need not halt). Ordinals coded by such reals are also said to be eventually writable, and we denote the supremum of the eventually writable ordinals by ζ. Hamkins and Lewis [2, 6.15, 8.2, 8.6] established that $\lambda < \zeta < \Sigma$ and that λ and ζ are admissible. Welch [10, 2.1] established that every computation $\varphi_p(0)$ either halts before λ or is trapped in a transfinite computational loop, exactly repeating the entire stage ζ machine configuration again at stage Σ and endlessly repeating this cycle thereafter. Furthermore, these ordinals are optimal in the sense that the universal computation that simulates all $\varphi_p(0)$ simultaneously first enters its repeating loop at ζ, first repeating it at Σ. It follows that Σ is not admissible.

2 Defining the complexity classes

Let us quickly recall the definitions of the complexity classes. The details depend on having a precise notion of the running time of an infinite time Turing machine computation. Specifically, during the computation of $\varphi_p(x)$, with program p operating on input x, let $\langle q_\alpha^{p,x} \rangle$ be the resulting sequence of states exhibited by the machine. Thus, $q_0^{p,x}$ will be the *start* state and $q_\xi^{p,x}$ will be the *limit* state when ξ is a limit ordinal. If the computation halts, then the *length* of the computation is the least β such that $q_{\beta+1}^{p,x}$ is the *halt* state. Thus, a computation has length β if it is the βth step that makes the transition to the *halt* state. Let us say that a program p is an α-*machine* if for every input x the computation $\varphi_p(x)$ has length less than α.

In the infinite time context, the complexity class P was introduced by Schindler [7], who generalized the classical polynomial time class to the infinitary context with the simple idea that all inputs $x \in \mathbb{R}$ have length ω and polynomial functions of ω are bounded by those of the form ω^n. Specifically, P contains exactly those $A \subseteq \mathbb{R}$ for which there is a natural number n and an ω^n-machine T such that $x \in A$ if and only if T accepts x. More generally, we define P_α to include all $A \subseteq \mathbb{R}$ for which there is $\beta < \alpha$ and a β-machine T that decides A. In this notation, P is simply P_{ω^ω}, midway up the hierarchy continuing up to P_{ω_1}, the class of sets that are decidable uniformly by some countable stage, and P_{ω_1+1}, the class of all decidable sets. The symbol P in P_α is meant to suggest the polynomial time context of classical complexity theory, because we have placed limitations on the lengths of the computations. But since all these classes concern infinite computations, one should not regard them as feasible in any practical sense.

The corresponding nondeterministic classes are defined by $A \in NP_\alpha$ if there is $\beta < \alpha$ and an β-machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ for which T accepts (x, y). Sets in NP_α are therefore simply the projections of sets in P_α. In accordance with $P = P_{\omega^\omega}$, we write NP for NP_{ω^ω}. It is clear that if $\alpha < \beta$, then $P_\alpha \subseteq P_\beta$ and $NP_\alpha \subseteq NP_\beta$, and for any limit ordinal ξ one can easily verify that $P_\xi = \bigcup_{\alpha<\xi} P_\xi$ and $NP_\xi = \bigcup_{\alpha<\xi} NP_\xi$.

For the nondeterministic classes we assume that the witness y is provided on a separate input tape, rather than coded together with x on one input tape. It follows that $P_\alpha \subseteq NP_\alpha \cap$ co-NP_α, since one can simply ignore the witness y without needing extra steps of computation. This convention is unnecessary when α is a limit ordinal or the successor of a limit ordinal.

So far, these complexity classes impose uniform bounds on the lengths of computation, independently of the input. But it may be more natural to allow a more complicated input to have a longer

computation. Taking ω_1^x as a natural measure of the complexity of x, Schindler defined $A \in P^+$ when there is an infinite time Turing machine deciding A and halting on input x in fewer than ω_1^x steps. The corresponding nondeterministic class is defined by $A \in NP^+$ when there is an infinite time Turing machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on input (x, y) in fewer than ω_1^x steps. With slightly longer computations, Schindler defined that $A \in P^{++}$ when there is an infinite time Turing machine deciding A and halting on input x in at most $\omega_1^x + \omega$ steps. Similarly, $A \in NP^{++}$ when there is an infinite time Turing machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on any input (x, y) in at most $\omega_1^x + \omega$ steps.

Finally, for any function f from $\mathbb{R}$ to the ordinals, Schindler defined $A \in P^f$ if there is an infinite time Turing machine deciding whether $x \in A$ in fewer than $f(x)$ steps.[1] And $A \in NP^f$ when there is an infinite time Turing machine T such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y), and T halts on any input (x, y) in fewer than $f(x)$ steps. In this notation, P^+ is the class P^{f_0}, where $f_0(x) = \omega_1^x + 1$, and $P^{++} = P^{f_1}$, where $f_1(x) = \omega_1^x + \omega + 1$. Note that if $g(x) = \alpha$ is constant, then $P^g = P_{\alpha+1}$.

3 Proving $P \neq NP \cap$ co-*NP*

We begin with our first main result, separating P from $NP \cap$ co-*NP*. Later, with more difficult methods, we will improve on this.

THEOREM 3.1
$P \neq NP \cap$ co-*NP* for infinite time Turing machines.

PROOF. Clearly P is contained in *NP* and closed under complements, so $P \subseteq NP \cap$ co-*NP*. We now show that the inclusion is proper. Consider the halting problem for computations halting before ω^ω given by

$$h_{\omega^\omega} = \{\, p \mid \varphi_p(p) \text{ halts in fewer than } \omega^\omega \text{ steps} \,\}.$$

We claim that $h_{\omega^\omega} \notin P$. This follows from [2, Theorem 4.4] and from Lemma 5.2 in this article, but let us quickly give the argument. If we could decide h_{ω^ω} in time before ω^ω, then we could compute the function $f(p) = 1$, if $p \notin h_{\omega^\omega}$, diverge otherwise, and we could do so in time before ω^ω for input $p \notin h_{\omega^\omega}$. If this algorithm for computing f is carried out by program q, then $q \notin h_{\omega^\omega}$ if and only if $f(q) \downarrow = 1$, which holds if and only if $\varphi_q(q)$ halts in fewer than ω^ω steps, which holds if and only if $q \in h_{\omega^\omega}$, a contradiction.

Let us now show that $h_{\omega^\omega} \in NP$. The idea is to verify whether $p \in h_{\omega^\omega}$ by inspecting (a code for) the computation sequence of $\varphi_p(p)$ up to ω^ω. To set this up, fix a recursive relation $\lhd$ on ω having order type ω^ω and a computable method of coding infinite sequences of reals as reals, so that we may interpret any real z as an infinite sequence of reals $\langle z_n \mid n \in \omega \rangle$. Viewing n as representing the ordinal α of its order type in $\lhd$, we therefore have a way to view any real z as an ω^ω-sequence of reals $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$. This coding is computable in the sense that given any $n \in \omega$ representing α with respect to $\lhd$, we can uniformly compute any digit of $(z)_\alpha$.

Consider the algorithm accepting (p, z) if in the sense above the real z codes a halting sequence of snapshots $\langle (z)_\alpha \mid \alpha < \omega^\omega \rangle$ of the computation $\varphi_p(p)$. That is, first, each $(z)_\alpha$ codes the complete configuration of an infinite time Turing machine, including the tape contents, the head position, the state and the program; second, the snapshot $(z)_{\alpha+1}$ is computed correctly from the previous

[1] This definition differs from that in [3], which allows $\leq f(x)$ steps in order to avoid the $+1$ that occurs when defining such classes as P^+ and P^{++}. Here we use the original definition of [7], which is capable of describing more classes.

snapshot $(z)_\alpha$, taking the convention that the snapshots should simply repeat after a halt; third, the limit snapshots $(z)_\xi$ for limit ordinals ξ are updated correctly from the previous snapshots $(z)_\alpha$ for $\alpha < \xi$; and finally, fourth, one of the snapshots shows the computation to have halted. Since all of these requirements form ultimately merely an arithmetic condition on the code z, they can be checked by an infinite time Turing machine in time uniformly before ω^2. And since $p \in h_{\omega^\omega}$ if and only if the computation sequence for $\varphi_p(p)$ halts before ω^ω, we conclude that $p \in h_{\omega^\omega}$ exactly if there is a real z such that (p, z) is accepted by this algorithm. Thus, $h_{\omega^\omega} \in NP$.

To see that $h_{\omega^\omega} \in$ co-*NP*, we simply change the fourth requirement to check that none of the snapshots show the computation to have halted. This change means that the input (p, z) will be accepted exactly when z codes a sequence of snapshots of the computation $\varphi_p(p)$, exhibiting it not to have halted in ω^ω steps. Since there is a real z like this if and only if $p \notin h_{\omega^\omega}$, it follows that the complement of h_{ω^ω} is in *NP*, and so $h_{\omega^\omega} \in$ co-*NP*. ∎

THEOREM 3.2
The classes NP_α for $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$ are all identical to the class Σ^1_1 of lightface analytic sets. In particular, $NP = NP_{\omega+2}$, and so membership in any *NP* set can be verified in only ω steps. Consequently, co-$NP_\alpha = \Pi^1_1$ and $NP_\alpha \cap$ co-$NP_\alpha = \Delta^1_1$.

PROOF. The idea is to ask more of our witnesses; they should code not merely a single computation, but all arithmetic truths.

LEMMA 3.3
Any $\Pi^0_2(x)$ statement can be decided on input x in a computation of length ω.

PROOF. To decide the truth of $\forall n \exists m \psi(n, m, x)$, where ψ has only bounded integer quantifiers, one considers each n in turn, searching for a witness m to work with it. Each time this succeeds, move to the next n and flash a master flag on and then off again. If the flag is on at a limit, it means that infinitely many n were considered, so the statement is true. If the flag is off, it means that for some n no witness m was found, so the statement is false. ∎

LEMMA 3.4
There is an infinite time Turing machine algorithm deciding in ω steps on input (a, A) whether A is the set of arithmetic truths in a.

PROOF. A simple induction on formulas shows that $A \subseteq \omega$ is the set of Gödel codes for true arithmetic statements in a (that is, using $a \subseteq \omega$ as a predicate in the language) if and only if the following conditions hold: (i) If ψ is atomic, then $\ulcorner \psi \urcorner \in A$ if and only if ψ is true; (ii) $\ulcorner \neg \psi \urcorner \in A$ if and only if $\ulcorner \psi \urcorner \notin A$; (iii) $\ulcorner \psi \wedge \phi \urcorner \in A$ if and only if $\ulcorner \psi \urcorner \in A$ and $\ulcorner \phi \urcorner \in A$; and (iv) $\ulcorner \exists u \psi(u) \urcorner \in A$ if and only if there is a natural number n such that $\ulcorner \psi(n) \urcorner \in A$. Considering each Gödel code in turn, an algorithm can systematically check these conditions. For a fixed formula, the first three conditions can be checked in finite time. For (iv), given a code for $\psi(n)$ in A, the algorithm can check whether the code for $\exists u \psi(u)$ is in A; conversely, given that $\exists u \psi(u)$ is in A, the algorithm can search for an n such that $\psi(n)$ is in A. If this search fails, then at the limit one can reject the input without more ado, since it has failed Condition (iv). Otherwise, a witness n is found in finitely many steps, and the next formula is considered. An alternative proof is provided via Lemma 3.3 by the observation that these four conditions have complexity $\Pi^0_2(a, A)$. ∎

Returning to the proof of Theorem 3.2, we now prove that when $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$, the classes NP_α are all identical. Since this is a nondecreasing sequence of classes, it suffices to show $NP_{\omega_1^{\text{CK}}} \subseteq NP_{\omega+2}$. For this, consider any set $B \in NP_{\omega_1^{\text{CK}}}$. By definition, this means that there is a program p and a recursive ordinal β such that $\varphi_p(x, y)$ halts in time β for all inputs and $x \in B$ if and only if there

is a y such that φ_p accepts (x, y). Fix a recursive relation $\lhd$ on ω having order type β. Consider the algorithm that accepts input (x, y, z, A) exactly when A codes the set of arithmetic truths in (x, y, z) and z codes the computation sequence of $\varphi_p(x, y)$ of length β (using the fixed recursive relation $\lhd$ as the underlying order of the snapshots), and this computation sequence shows the computation to have accepted the input. We claim that this algorithm can be carried out in just ω steps. To see this, observe first that the latter part of the condition, about z coding the accepting computation sequence for $\varphi_p(x, y)$, is arithmetic in (x, y, z). Consider the algorithm that first checks in finitely many steps whether the Gödel code of that arithmetic condition is in A. If not, it rejects the input (x, y, z, A). Otherwise, it checks whether A really is the set of arithmetical truths in (x, y, z) using the algorithm of Lemma 3.4. The algorithm accepts the input if it is and rejects if it is not. Observe that this algorithm accepts (x, y, z, A) exactly when A codes the set of arithmetic truths in (x, y, z) and z codes the computation sequence of $\varphi_p(x, y)$ of length β, showing it to have accepted its input. Since $x \in B$ if and only if there is y such that φ_p accepts (x, y), it follows that $x \in B$ if and only if there is (y, z, A) such that our algorithm accepts (x, y, z, A). Since our algorithm takes at most ω steps on any input, we conclude that $B \in NP_{\omega+2}$. We have therefore proved $NP_{\omega_1^{\text{CK}}} \subseteq NP_{\omega+2}$, and so the classes NP_α are identical for $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$.

Since $NP = NP_{\omega^\omega}$, it follows that $NP = NP_{\omega+2}$, and so membership in any *NP* set can be verified in ω steps. By [2, Theorem 2.7] we know that $P_{\omega_1^{\text{CK}}} = \Delta^1_1$. It follows immediately that $NP_{\omega_1^{\text{CK}}} = \Sigma^1_1$, as these sets are the projections of sets in $P_{\omega_1^{\text{CK}}}$. So $NP_\alpha = \Sigma^1_1$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$, as these classes are all identical. And finally, by taking complements, we conclude as well that co-$NP_\alpha = \Pi^1_1$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$. ∎

Since the classes Σ^1_1 and Π^1_1 are not identical, we conclude:

COROLLARY 3.5
NP $\neq$ co-*NP* for infinite time Turing machines.

4 Proving $P^+ = NP^+ \cap$ co-NP^+

The class P^+ appears at first more generous than the earlier classes, because computations on input x are now allowed up to ω_1^x steps, which can be considerably larger than ω_1^{CK}. Theorem 4.2 shows, however, that sets in P^+ can actually be decided by algorithms in uniformly fewer than ω_1^{CK} steps, so the apparent extra power actually provides no advantage. We will use the following fact from descriptive set theory, a special case of [6, Theorem 4D.3], due to Kleene.

LEMMA 4.1
Π^1_1 absorbs existential quantification over Δ^1_1. That is, if B is Π^1_1 and $x \in A \iff \exists y \in \Delta^1_1(x)\, B(x, y)$, then A is Π^1_1 as well.

THEOREM 4.2
(i) $NP^+ = \Sigma^1_1 = NP = NP_\alpha$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}} + 1$.
(ii) $P^+ = \Delta^1_1 = P_{\omega_1^{\text{CK}}} = P_{\omega_1^{\text{CK}}+1}$.
(iii) $P^+ = NP^+ \cap$ co-NP^+.

PROOF. For (*i*), we have already proved in Theorem 3.2 that $\Sigma^1_1 = NP$, and since clearly $NP \subseteq NP^+$, it follows that $\Sigma^1_1 \subseteq NP^+$. Conversely, suppose that $A \in NP^+$. Thus, there is a program p such that $\varphi_p(x, y)$ halts on all input (x, y) in fewer than ω_1^x steps, and $x \in A$ if and only if there is a real y such that φ_p accepts (x, y). The set A is therefore the projection of the set

$$B = \{\, (x, y) \mid \varphi_p \text{ accepts } (x, y) \,\}.$$

In order to see that A is in Σ^1_1, it suffices to show $B \in \Sigma^1_1$. Elements of the complement $\neg B = \{\, (x,y) \mid \varphi_p \text{ rejects } (x,y) \,\}$ are witnessed to be in $\neg B$ by computations of length less than ω_1^x. It follows that the computation sequence for $\varphi_p(x,y)$ exists in the model $L_{\omega_1^x}[x,y]$, and so $(x,y) \in \neg B$ if and only if $L_{\omega_1^x}[x,y] \models \theta(x,y)$, where $\theta(x,y)$ asserts that the computation $\varphi_p(x,y)$ rejects the input. Since this is a Σ_1 assertion, it follows that $(x,y) \in \neg B$ if and only if there is an ordinal $\beta < \omega_1^x$ such that $L_\beta[x,y] \models \theta(x,y)$. Since there is a code for the model $L_\beta[x,y]$ that is hyperarithmetic in (x,y), and any well-founded model showing the computation to reject the input will do, we see that $(x,y) \in \neg B$ if and only if there is a real $z \in \Delta^1_1(x,y)$ coding a well-founded model of $V = L[x,y]$ that satisfies $\theta(x,y)$. Thus, by Lemma 4.1, $\neg B$ is Π^1_1 and hence B and A are Σ^1_1, so we conclude $NP^+ = \Sigma^1_1$. It follows from Theorem 3.2 that $NP^+ = NP_\alpha$ whenever $\omega + 2 \leq \alpha \leq \omega_1^{\text{CK}}$. The remaining case of $\alpha = \omega_1^{\text{CK}} + 1$ follows from (*ii*) and the observation that $NP_{\omega_1^{\text{CK}}} = NP_{\omega_1^{\text{CK}}+1}$, as these are the projections of $P_{\omega_1^{\text{CK}}} = P_{\omega_1^{\text{CK}}+1}$.

For (*ii*) and (*iii*), observe that since $NP^+ = \Sigma^1_1$, it follows that co-$NP^+ = \Pi^1_1$, and so $P^+ \subseteq NP^+ \cap \text{co-}NP^+ = \Sigma^1_1 \cap \Pi^1_1 = \Delta^1_1$, which by [2, Theorem 2.7] is equal to $P_{\omega_1^{\text{CK}}}$, which is a subset of $P_{\omega_1^{\text{CK}}+1}$, which is clearly a subset of P^+. So all of them are equal, as we claimed. ∎

The fact that $P^+ = \Delta^1_1$ was Theorem 2.13 of [7], and one can view our argument here as a detailed expansion of that argument. In fact, however, once one knows $P^+ = P_{\omega_1^{\text{CK}}} = \Delta^1_1$, it follows immediately that sets in NP^+ are projections of sets in $P_{\omega_1^{\text{CK}}} = \Delta^1_1$, since all the computations halt uniformly before ω_1^{CK}, which is certainly not larger than ω_1^x, and consequently $NP^+ = \Sigma^1_1$. By these means, Theorem 4.2 follows from [7, Theorem 2.13].

The equality $P^+ = P_{\omega_1^{\text{CK}}}$ should be surprising, because it means that although the computations deciding $x \in A$ for $A \in P^+$ are allowed to compute up to ω_1^x, in fact there is an algorithm needing uniformly fewer than ω_1^{CK} steps. An affirmative answer to the following question would explain this phenomenon completely.

QUESTION 4.3
Suppose an algorithm halts on each input x in fewer than ω_1^x steps. Then does it halt uniformly before ω_1^{CK}? [2]

Secondly, the fact that $P_{\omega_1^{\text{CK}}} = P_{\omega_1^{\text{CK}}+1}$ is itself surprising, because the difference in the definitions of these two classes is exactly the difference between requiring the computations to halt before ω_1^{CK} and requiring them to halt *uniformly* before ω_1^{CK}, that is, before some fixed $\beta < \omega_1^{\text{CK}}$ on all input. Since $P_{\omega_1^{\text{CK}}} = P_{\omega_1^{\text{CK}}+1}$, this difference makes no difference.

5 The question whether $P_\alpha = NP_\alpha \cap$ co-NP_α

We turn now to the relation between P_α and $NP_\alpha \cap$ co-NP_α for various ordinals α. We begin by showing that the P_α hierarchy increases with every clockable limit ordinal α.

DEFINITION 5.1
The lightface *halting problem* is the set $h = \{\, p \mid \varphi_p(p) \text{ halts}\}$. Approximating this, for any ordinal α the halting problem for α is the set $h_\alpha = \{\, p \mid \varphi_p(p) \text{ halts in fewer than } \alpha \text{ steps}\}$.

LEMMA 5.2
If α is any ordinal, then $h_\alpha \notin P_\alpha$. Indeed, $h_\alpha \notin P_{\alpha+1}$. However, if α is a clockable limit ordinal, then $h_\alpha \in P_{\alpha+2}$.

[2] Philip Welch has now solved this question, providing an affirmative answer in [9].

PROOF. Suppose to the contrary that $h_\alpha \in P_{\alpha+1}$ for some ordinal α. It follows that there is an algorithm q deciding h_α in fewer than α steps. That is, q halts in fewer than α steps on all inputs and $x \in h_\alpha$ if and only if q accepts x. Consider the modified algorithm q_0 that runs $\varphi_q(x)$, but when the algorithm is just about to move into the *accept* state, it instead jumps into a non-halting transfinite repeating loop. This algorithm computes a function $\varphi_{q_0}(x)$ which halts in fewer than α steps if $x \notin h_\alpha$ and diverges otherwise. Therefore, $q_0 \in h_\alpha$ if and only if $\varphi_{q_0}(q_0)$ halts, which holds if and only if $q_0 \notin h_\alpha$, a contradiction. So we conclude $h_\alpha \notin P_{\alpha+1}$ for any ordinal α.

Finally, if α is a clockable limit ordinal, consider the algorithm that on input p simulates both the computation $\varphi_p(p)$ and the α clock (simulating ω steps of each in every ω actual steps). If the computation stops before the clock, the algorithm accepts the input, but if the clock runs out, it rejects the input. By placing the first column of the clock's computation in the actual first column, the algorithm can detect that the clock has stopped at exactly stage α, thereby halting in α steps. So $h_\alpha \in P_{\alpha+2}$. ∎

COROLLARY 5.3
If α is a clockable limit ordinal, then $P_{\alpha+1} \subsetneq P_{\alpha+2}$.

THEOREM 5.4
The inequality $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$ holds whenever $\omega + 2 \leq \alpha < \omega_1^{\text{CK}}$. Equality is attained at ω_1^{CK} and its successor with

$$P_{\omega_1^{\text{CK}}} = NP_{\omega_1^{\text{CK}}} \cap \text{co-}NP_{\omega_1^{\text{CK}}} = \Delta^1_1 = P_{\omega_1^{\text{CK}}+1} = NP_{\omega_1^{\text{CK}}+1} \cap \text{co-}NP_{\omega_1^{\text{CK}}+1}\,.$$

PROOF. For $\alpha < \omega_1^{\text{CK}}$ we know by Corollary 5.3 that P_α is a proper subset of $P_{\omega_1^{\text{CK}}}$, which by Theorem 3.2 is equal to $NP_{\omega_1^{\text{CK}}} \cap \text{co-}NP_{\omega_1^{\text{CK}}} = NP_\alpha \cap \text{co-}NP_\alpha$. So none of the earlier classes P_α are equal to $NP_\alpha \cap \text{co-}NP_\alpha$; but at the top we have $P_{\omega_1^{\text{CK}}} = NP_{\omega_1^{\text{CK}}} \cap \text{co-}NP_{\omega_1^{\text{CK}}}$ and $P_{\omega_1^{\text{CK}}+1} = NP_{\omega_1^{\text{CK}}+1} \cap \text{co-}NP_{\omega_1^{\text{CK}}+1}$ because Theorem 4.2 shows that these are instances of the identity $\Delta^1_1 = \Sigma^1_1 \cap \Pi^1_1$. ∎

Corollary 5.3 and Theorem 5.4 show that the class Δ^1_1 of hyperarithmetic sets is ramified by the increasing hierarchy $\bigcup_{\alpha<\omega_1^{\text{CK}}} P_\alpha$, just as it is by the traditional hyperarithmetic hierarchy $\Delta^1_1 = \bigcup_{\alpha<\omega_1^{\text{CK}}} \Delta^0_\alpha$, and one can probably give a tight analysis of the interaction of these two hierarchies.

We now prove that the pattern of Theorems 3.2 and 5.4—where the classes NP_α are all identical for α in the range from $\omega + 2$ up to $\omega_1^{\text{CK}} + 1$—is mirrored higher up within any contiguous block of clockable ordinals. It will follow that P_α is properly contained in $NP_\alpha \cap \text{co-}NP_\alpha$ within any such block of clockable ordinals. The pattern continues with $P_\beta = NP_\beta \cap \text{co-}NP_\beta$ at the top of the block, where β begins the next gap in the clockable ordinals.

THEOREM 5.5
If $[\nu, \beta)$ is a contiguous block of infinite clockable ordinals, then the classes NP_α for $\nu + 2 \leq \alpha \leq \beta + 1$ are all identical to each other. Consequently, the corresponding classes co-NP_α for such α are also all identical to each other as well.

PROOF. Since the sequence of classes NP_α is nondecreasing, it suffices to show $NP_{\beta+1} \subseteq NP_{\nu+2}$. Suppose $B \in NP_{\beta+1}$, so that there is an algorithm e such that $\varphi_e(x, y)$ halts on every input in in time less than β, and $x \in B$ if and only if there is y such that φ_e accepts (x, y). Since ν is clockable, there is a program q_0 such that $\varphi_{q_0}(0)$ takes exactly ν steps to halt. Consider the algorithm which on input (x, z) checks, first, whether z codes a model M_z of KP containing x in which the computation $\varphi_{q_0}(0)$ halts and there is a $y \in M_z$ such that φ_e accepts (x, y); and second, verifies that M_z is well-founded up to ν^{M_z}, the length of the clock computation $\varphi_{q_0}(0)$ in M_z. If both of these requirements are satisfied, then the algorithm accepts the input and otherwise rejects it.

If $x \in B$, then there is a y such that φ_e accepts (x, y), and so we may choose z coding a fully well-founded model M_z that is tall enough to see this computation and $\varphi_{q_0}(0)$. It follows that (x, z) will be accepted by our algorithm. Conversely, if (x, z) is accepted by our algorithm, then the corresponding model M_z is well-founded up to ν. Next, we make the key step of the proof, using the fact that the well-founded part of any model of KP must be an admissible ordinal. Since no clockable ordinal is admissible, it follows that M_z must be well-founded beyond the length of the computation $\varphi_e(x, y)$ (which is less than β), as all the ordinals in $[\nu, \beta)$ are clockable. Therefore, M_z will have the correct (accepting) computation for $\varphi_e(x, y)$, and so $x \in B$. Thus, our algorithm nondeterministically decides B. And as before, since ν is inadmissible, this algorithm will either discover ill-foundedness below ν, halting in time at most ν, or else halt at ν with well-foundedness up to $\nu^{M_z} = \nu$. So $B \in NP_{\nu+2}$, as desired. The corresponding fact for co-NP_α follows by taking complements. ∎

COROLLARY 5.6
The inequality $P_\alpha \neq NP_\alpha \cap$ co-NP_α holds for every clockable ordinal $\alpha \geq \omega + 2$, except possibly when α ends a gap in the clockable ordinals or is the successor of such a gap-ending ordinal.

PROOF. If $\alpha \geq \omega + 2$ is clockable but is neither a gap-ending ordinal nor the successor of a gap-ending ordinal, then there is an infinite ordinal $\nu < \nu + 2 \leq \alpha$ such that $[\nu, \alpha]$ is a contiguous block of clockable ordinals. By Theorem 5.5, the classes NP_ξ are identical for $\nu + 2 \leq \xi \leq \beta + 1$, where β is the next non-clockable ordinal beyond α. Since by Corollary 5.3 the corresponding classes P_ξ increase at every clockable limit ordinal in this range and are subsets of $NP_\xi \cap$ co-NP_ξ, it follows that $P_\alpha \subsetneq NP_\alpha \cap$ co-NP_α. ∎

Because of the possible exceptions in Corollary 5.6 at the gap-ending ordinals, we do not have a complete answer to the following question.

QUESTION 5.7
Is $P_\alpha \neq NP_\alpha \cap$ co-NP_α for any clockable ordinal $\alpha \geq \omega + 2$?

The first unknown instances of this occur at the first gap-ending ordinal $\omega_1^{\text{CK}} + \omega$ and its successor $\omega_1^{\text{CK}} + \omega + 1$. Thus, we don't know whether

$$P_{\omega_1^{\text{CK}}+\omega} = NP_{\omega_1^{\text{CK}}+\omega} \cap \text{co-}NP_{\omega_1^{\text{CK}}+\omega},$$

nor do we know whether

$$P_{\omega_1^{\text{CK}}+\omega+1} = NP_{\omega_1^{\text{CK}}+\omega+1} \cap \text{co-}NP_{\omega_1^{\text{CK}}+\omega+1}\,.$$

A related question concerns the gap-starting ordinals and their successors, such as ω_1^{CK} and $\omega_1^{\text{CK}} + 1$, where Theorem 5.4 shows the equalities

$$P_{\omega_1^{\text{CK}}} = NP_{\omega_1^{\text{CK}}} \cap \text{co-}NP_{\omega_1^{\text{CK}}} \qquad \text{and} \qquad P_{\omega_1^{\text{CK}}+1} = NP_{\omega_1^{\text{CK}}+1} \cap \text{co-}NP_{\omega_1^{\text{CK}}+1}\,.$$

We will now show that this phenomenon is completely general, appealing to the following unpublished results of Philip Welch.

LEMMA 5.8 (Welch [8])
If α is a clockable ordinal, then every ordinal up to the next admissible beyond α is writable in time $\alpha + \omega$.

THEOREM 5.9 (Welch [11, Theorem 10], [8])
Every ordinal beginning a gap in the clockable ordinals is admissible.

This latter result is a converse of sorts to [2, Theorem 8.8], which establishes that no admissible ordinal is clockable. It is not the case, however, that the gap-starting ordinals are exactly the admissible ordinals below λ, because admissible ordinals can appear in the middle of a gap. To see that this phenomenon occurs, observe that the suprema of the writable and eventually writable ordinals are both admissible, with no clockable ordinals in between, and this situation reflects downwards into an actual gap, because an algorithm can search for accidentally writable admissible ordinals having no clockable ordinals in between, and halt when they are found.

THEOREM 5.10
Suppose that β begins a gap in the clockable ordinals. Then $P_\beta = NP_\beta \cap \text{co-}NP_\beta$. If, in addition, β is not a limit of non-clockable ordinals, then $P_\beta = P_{\beta+1} = NP_{\beta+1} \cap \text{co-}NP_{\beta+1}$.

PROOF. Let us suppose first that β begins a gap in the clockable ordinals, but is not a limit of non-clockable ordinals, so that there is some $\nu < \beta$ such that $[\nu, \beta)$ is a contiguous block of clockable ordinals. Since ν is clockable, it follows by Lemma 5.8 that there is a real u coding ν that is writable in time $\nu + \omega$, which is of course still less than β. We claim that $\beta = \omega_1^u$. To see this, observe first that β is admissible by Theorem 5.9. Next, L_β has the computation producing u and so $u \in L_\beta$. Consequently, β is u-admissible and so $\omega_1^u \leq \beta$. For the converse inequality, we use the fact established by [2, Theorem 8.8] that no admissible ordinal is clockable. Since all ordinals in $[\nu, \beta)$ are clockable, none are admissible. Thus, since u codes ν and ω_1^u is admissible, it follows that $\beta \leq \omega_1^u$, and so $\beta = \omega_1^u$.

Next, we relativize Theorem 5.4 with respect to an oracle for u, with the conclusion that $P^u_{\omega_1^u} = NP^u_{\omega_1^u} \cap \text{co-}NP^u_{\omega_1^u} = \Delta^1_1(u) = P^u_{\omega_1^u+1} = NP^u_{\omega_1^u+1} \cap \text{co-}NP^u_{\omega_1^u+1}$, where the superscript indicates the presence of an oracle for u. (That is, a set A is in P^u_α if there is an oracle Turing machine T with oracle u and $\beta < \alpha$ such that on input x the machine T decides $x \in A$ and halts on all inputs in time less than β; similarly, A is in NP^u_α if there is an oracle Turing machine T with oracle u and $\beta < \alpha$ such that $x \in A$ if and only if there is $y \in \mathbb{R}$ such that T accepts (x, y) and T halts on all inputs in time less than β. The relativization of Theorem 5.4 to the oracle context is straightforward.) But since u is writable in time $\nu + \omega < \beta$ by Lemma 5.8, we can simulate such an oracle by simply taking the time first to write it out. By admissibility, $\nu + \omega + \beta = \beta$, and so this preparatory step will not cause any ultimate delay in our calculations. Therefore, $P_\beta = P^u_\beta$, $NP_\beta = NP^u_\beta$ and $\text{co-}NP_\beta = \text{co-}NP^u_\beta$, and the same for $\beta + 1$. We conclude that $P_\beta = NP_\beta \cap \text{co-}NP_\beta = \Delta^1_1(u) = P_{\beta+1} = NP_{\beta+1} \cap \text{co-}NP_{\beta+1}$, as desired.

It remains to consider the case of a gap-starting ordinal β that is a limit of gaps. Any such β is a limit of gap-starting ordinals. By considering the successor gaps below β (the next gap after any clockable ordinal), we observe that β is a limit of gap-starting ordinals ξ that are not limits of non-clockable ordinals. Let I be the set of such ξ below β. By the previous paragraph, such ordinals satisfy $P_\xi = NP_\xi \cap \text{co-}NP_\xi = P_{\xi+1} = NP_{\xi+1} \cap \text{co-}NP_{\xi+1}$. Because I is unbounded in β, it follows that P_β is the union of the nondecreasing sequence of classes P_ξ for $\xi \in I$, and the same for NP_β and $\text{co-}NP_\beta$. Therefore,

$$P_\beta = \bigcup_{\xi \in I} P_\xi = \bigcup_{\xi \in I} NP_\xi \cap \text{co-}NP_\xi = (\bigcup_{\xi \in I} NP_\xi) \cap (\bigcup_{\xi \in I} \text{co-}NP_\xi) = NP_\beta \cap \text{co-}NP_\beta,$$

and so the proof is complete. ∎

COROLLARY 5.11
In particular, $P_\lambda = NP_\lambda \cap \text{co-}NP_\lambda$, where λ is the supremum of the clockable ordinals.

More generally, we ask for a characterization of these exceptional ordinals.

QUESTION 5.12
Exactly which ordinals α satisfy $P_\alpha = NP_\alpha \cap \text{co-}NP_\alpha$?

Just for the record, let us settle the question for the classes P_ω and $P_{\omega+1}$, as well as P_n for finite n, which are all trivial in the sense that they involve only finite computations. The class P_ω concerns the uniformly finite computations, while $P_{\omega+1}$ allows arbitrarily long but finite computations. The class P_n for finite n concerns computations having at most $n-2$ steps. Observe that $P_0 = P_1 = \emptyset$ because computations have nonnegative length, and $P_2 = \{\mathbb{R}, \emptyset\}$ because a computation halts in 0 steps only when the *start* state is identical with either the *accept* or *reject* states. Infinite computations first appear with the class $P_{\omega+2}$. This result has nothing to do with the classical finite time P vs. *NP* question.

THEOREM 5.13
For the classes corresponding to finite computations:

(i) $P_n = NP_n = \text{co-}NP_n$ for any finite n. Consequently, $P_n = NP_n \cap \text{co-}NP_n$.
(ii) $P_\omega = NP_\omega = \text{co-}NP_\omega$. Consequently, $P_\omega = NP_\omega \cap \text{co-}NP_\omega$.
(iii) $P_{\omega+1} = \Delta^0_1$, $NP_{\omega+1} = \Sigma^0_1$ and co-$NP_{\omega+1} = \Pi^0_1$. Consequently, $P_{\omega+1} = NP_{\omega+1} \cap \text{co-}NP_{\omega+1}$.

PROOF. For (*i*), the computations placing a set in P_n, NP_n or co-NP_n are allowed at most $n-2$ steps, and so the sets they decide depend on at most the first $n-2$ digits of the input. Any such set is in P_n, because a program could simply read those digits in $n-2$ steps, remembering them with states, and move to the *accept* or *reject* states accordingly. So $P_n = NP_n = \text{co-}NP_n$. Claim (*ii*) follows, because $P_\omega = \bigcup_n P_n = \bigcup_n NP_n = NP_\omega$.

For (*iii*), observe that a set B is in $P_{\omega+1}$ if $x \in B$ can be decided by a Turing machine program that halts in finitely many steps. Since this is precisely the classical notion of (finite time) computability, it follows that $P_{\omega+1} = \Delta^0_1$, the recursive sets of reals. If $B \in NP_{\omega+1}$, there is an algorithm p such that $\varphi_p(x, y)$ halts in finitely many steps on all input and $x \in B$ if and only if there is a y such that φ_p accepts (x, y). Thus, $x \in B$ if and only if there is a finite piece $y \restriction n$ such that φ_p accepts $(x, y \restriction n)$, where the piece is long enough that the algorithm never inspects y beyond n bits. Since this has now become an existential quantifier over the integers, we conclude that $B \in \Sigma^0_1$. Conversely, every set in Σ^0_1 is clearly the projection of a set in Δ^0_1, so we conclude $NP_{\omega+1} = \Sigma^0_1$. By taking complements, co-$NP_{\omega+1} = \Pi^0_1$. ∎

Returning our focus to the infinite computations, let us now consider the case of ordinals that are not necessarily clockable. Our first observation is that the key idea of the proof of Theorem 3.1—the fact that one could easily recognize codes for ω^ω or any other recursive ordinal—generalizes to the situation where one has only nondeterministic algorithms for recognizing the ordinals in question.

DEFINITION 5.14
An ordinal α is *recognizable* (in time at most ξ) when there is a nonempty set of reals coding α that is decidable (in time at most ξ). The ordinal α is *nondeterministically recognizable* (in time at most ξ) if there is a nonempty set of codes for α that is nondeterministically decidable (in time at most ξ).

If α is nondeterministically recognizable in time at most ξ, then the set WO_α of *all* reals coding α is nondeterministically decidable in time at most ξ, because a real is in WO_α if and only if there is an isomorphism from the relation it codes to the relation coded by any other real coding α. That is, if A is a nonempty set of reals coding α that is nondeterministically decidable, then $w \in \text{WO}_\alpha$ if and only if there is (x, y, z) such that (x, y) is accepted by A and z codes an order isomorphism of the relation coded by w to that coded by x. By checking this latter condition simultaneously in the first ω steps, one will not increase the time required to carry out the algorithm.

LEMMA 5.15
If an ordinal α is nondeterministically recognizable in time at most ξ, then $h_\alpha \in NP_{\xi+2} \cap$ co-$NP_{\xi+2}$.

PROOF. Suppose that α is nondeterministically recognizable in time ξ, so there is a nonempty set D of codes for α that is in $NP_{\xi+2}$. Lemma 5.2 and Theorem 3.2 already handle the case when $\alpha < \omega_1^{\text{CK}}$. So we may assume α and hence also ξ are at least ω_1^{CK}. Consider the algorithm that on input (p, u, v, w) checks, first, that u codes a linearly ordered relation on ω with respect to which v codes the snapshot sequence of $\varphi_p(p)$, showing it to halt, and second, that (u, w) is accepted by the nondeterministic algorithm deciding D, verifying $u \in D$. (We don't need explicitly to check that u codes a well-order, since this will be true provided $u \in D$, but we need to know that u codes a linear order to make sense of the requirement that v codes a computational sequence along that order.) If $p \in h_\alpha$, then the computation $\varphi_p(p)$ halts in fewer than α steps, and so we may choose a real $u \in D$ coding α, along with a real w witnessing that $u \in D$, and a real v coding the halting snapshot sequence of $\varphi_p(p)$, so that (p, u, v, w) is accepted by our algorithm. Conversely, if (p, u, v, w) is accepted by our algorithm, then because (u, w) was accepted by the algorithm for D, we know u really codes α, and so the snapshot sequence must be correct in showing $\varphi_p(p)$ to halt before α, so $p \in h_\alpha$. Finally, the algorithm takes ξ steps, because the initial check takes fewer than ω^2 steps, being an arithmetic requirement, and so the computation takes $\omega^2 + \xi = \xi$ steps altogether. Thus, $h_\alpha \in NP_\xi$. To see that $h_\alpha \in$ co-NP_ξ, simply modify the algorithm to check that v codes a snapshot sequence with respect to the relation coded by u, but v shows the computation *not* to halt. ∎

One can use the same idea to show that if NP_α contains a set of codes for ordinals unbounded in α, then $P_\alpha \neq NP_\alpha$. We will now apply this result to show that $P_\alpha \neq NP_\alpha \cap$ co-NP_α for all sufficiently large countable ordinals α. Recall from the introduction that $\lambda < \zeta < \Sigma$ refer to the suprema of the writable, eventually writable and accidentally writable ordinals, respectively. The first two of these are admissible, while the latter is not, and every computation either halts before λ or repeats the ζ configuration of the machine (that is, the state, the head position and the contents of the tape) again at Σ. And furthermore, Σ is characterized by being the first repeat point of the universal computation simulating all $\varphi_p(0)$ simultaneously.

THEOREM 5.16
If $\Sigma + 2 \leq \alpha$, then $P_\alpha \neq NP_\alpha \cap$ co-NP_α. In fact, the class $NP_{\Sigma+2} \cap$ co-$NP_{\Sigma+2}$ contains a nondecidable set, the halting problem h.

PROOF. The proof relies on the following.

LEMMA 5.17
The ordinal Σ is nondeterministically recognizable in time Σ.

PROOF. The model-checking algorithm of [3, Theorem 1.7] essentially shows this, but let us sketch the details here. By results in [10], the ordinal Σ is the first stage at which the universal computation (simulating $\varphi_p(0)$ for all programs p) repeats itself. Consider the algorithm which on input (x, y) checks whether y codes a model $M_y \models$ 'KP + Σ exists' containing x and satisfying the assertion that the order type of the relation coded by x is Σ^{M_y}. If y passes this test, then the algorithm counts-through the relation coded by x to verify that it is well-founded. If all these tests are passed, then the algorithm accepts the input, and otherwise rejects it. If the well-founded part of M_y exceeds the true Σ, then M_y will have the correct value for Σ, and the algorithm will take exactly Σ steps. If the well-founded part of M_y lies below Σ, then this will be discovered before Σ and the algorithm will halt before Σ. Finally, because Σ is not admissible, the well-founded part of M_y cannot be exactly

Σ, and so in every case our algorithm will halt in at most Σ steps. And since the acceptable x have order type Σ, this shows that WO_Σ, the set of reals coding Σ, is nondeterministically decidable in Σ steps, as desired. ∎

By Lemma 5.15, it follows that $h_\Sigma \in NP_{\Sigma+2} \cap \text{co-}NP_{\Sigma+2}$. But since Σ is larger than every clockable ordinal, it follows that $h_\Sigma = h$, the full lightface halting problem. So we have established that if $\Sigma + 2 \leq \alpha$, then the halting problem h is in $NP_\alpha \cap \text{co-}NP_\alpha$. Since h is not decidable, it cannot be in P_α. So $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$. ∎

This establishes $P_\alpha \neq NP_\alpha \cap \text{co-}NP_\alpha$ for all but countably many α. We close this section with another definition and an application.

DEFINITION 5.18
An ordinal α is *nondeterministically clockable* if there is an algorithm p which halts in time at most α for all input and in time exactly α for some input. More generally, α is *nondeterministically clockable before* β if there is an algorithm that halts before β on all input and in time exactly α for some input.

Such an algorithm can be used as a clock for α in nondeterministic computations, since there are verifying witnesses making the clock run for exactly the right amount of time, with a guarantee that no other witnesses will make the clock run on too long.

THEOREM 5.19
If α is an infinite nondeterministically clockable limit ordinal, then $P_{\alpha+2} \neq NP_{\alpha+2}$.

PROOF. By Lemma 5.2, it follows that $h_{\alpha+\omega} \notin P_{\alpha+2}$. But we claim that $h_{\alpha+\omega} \in NP_{\alpha+2}$. If α is recursive, then $h_{\alpha+\omega}$ is hyperarithmetic and hence in $NP_{\omega+2} = NP_{\alpha+\omega}$ by Theorem 3.2. So we may assume $\alpha \geq \omega_1^{\text{CK}}$. Fix a nondeterministic clock for α, a program e such that $\varphi_e(z)$ halts in exactly α steps for some z and in at most α steps on all other input. We will now nondeterministically decide $h_{\alpha+\omega}$ by the following algorithm. On input (p, y, z), first determine whether p is finite (with respect to the usual representation of a natural number n by a block of n 1s followed by 0s). If not, then reject the input, otherwise, check whether y codes a model M_y of KP containing z and satisfying the assertion that $\varphi_e(z)$ halts, with $\varphi_p(p)$ halting at most finitely many steps later. Since this is an arithmetic condition on y, it can be checked in fewer than ω^2 steps. Next, assuming that these tests have been passed successfully, we verify that the model M_y is well-founded up to what it thinks is the halting time of $\varphi_e(z)$, which we denote α^{M_y}. If ill-foundedness is discovered, we reject the input. By flashing a master flag every time we delete what is the current smallest (in the natural ordering of ω) element still in the field, we can tell at a limit stage that we have finished counting, and when this occurs, we accept the input.

Let's argue that this algorithm accomplishes what we want. First of all, if $p \in h_{\alpha+\omega}$, then $\varphi_p(p)$ halts before $\alpha + \omega$ and there is a real z such that $\varphi_e(z)$ halts in α steps and a real y coding a fully well-founded model $M_y \models \text{KP}$ in which these computations exist. So the previous algorithm will accept the input (p, y, z). Conversely, if the algorithm accepts (p, y, z) for some y and z, then the corresponding model M_y is well-founded up to the length of the computation $\varphi_e(z)$, which is at most α because the computation $\varphi_e(z)$ in M_y agrees with the actual computation as long as the model remains well-founded. Since the model satisfies that $\varphi_p(p)$ halts finitely many steps later, it will therefore be correct that $\varphi_p(p)$ halts before $\alpha + \omega$. So the algorithm does nondeterministically decide $h_{\alpha+\omega}$.

It remains to see that the algorithm halts in at most α steps on all inputs. Since $\omega_1^{\text{CK}} \leq \alpha$, it follows that $\omega^2 + \alpha = \alpha$, and so the initial checks of those arithmetic properties do not ultimately cause any

delay. The only question is how many steps it takes to check the well-foundedness of M_y up to α^{M_y}. If M_y is well-founded up to α^{M_y}, then this takes exactly α^{M_y} steps (as the count-through algorithm is designed precisely to take β steps to count through a relation of limit order type β), and this is at most α. If M_y is ill-founded below α^{M_y}, then this will be discovered exactly ω steps beyond the well-founded part of M_y, and so the algorithm will halt in at most α steps. Lastly, the well-founded part of M_y cannot be exactly α, because α is not z-admissible. So in any case, on any input the algorithm halts in at most α steps. ∎

This argument does not establish $P_{\alpha+2} \neq NP_{\alpha+2} \cap$ co-$NP_{\alpha+2}$, however, because one cannot seem to use a nondeterministic clock to verify that a computation $\varphi_p(p)$ has *not* halted. A prematurely halting nondeterministic clock might erroneously indicate that $\varphi_p(p)$ does not halt in time $\alpha + \omega$ even when it does, leading to false acceptances for the complement of $h_{\alpha+\omega}$.

6 The cases of P^f and P^{++}

Let us turn now to the question of whether $P^f = NP^f \cap$ co-NP^f, where $f : \mathbb{R} \to$ ORD. A special case of this is the question of whether $P^{++} = NP^{++} \cap$ co-NP^{++}, because $P^{++} = P^{f_1}$, where $f_1(x) = \omega_1^x + \omega + 1$. We consider only functions f that are *suitable*, meaning that $f(x) \leq f(y)$ whenever $x \leq_T y$ and $f(x) \geq \omega + 1$.

Many of the instances of the question whether $P^f = NP^f \cap$ co-NP^f are actually solved by a close inspection of the arguments of [3], though the results there were stated only as $P^f \neq NP^f$. The point is that the model-checking technique of verification used in those arguments is able to verify both positive and negative answers. But more than this, the next theorem shows that the analysis of whether $P^f = NP^f \cap$ co-NP^f, at least for sets of natural numbers, reduces to the question of whether $P_\alpha = NP_\alpha \cap$ co-NP_α, where $\alpha = f(0) + 1$. And since the previous section provides answers to this latter question for many values of α, we will be able to provide answers to the former question as well, in Corollaries 6.2 and 6.3. We regard the natural numbers $\mathbb{N}$ as naturally coded in the reals $\mathbb{R}$ by coding any natural number n with the sequence consisting of n ones, followed by zeros.

THEOREM 6.1
For any suitable function f and any set A of natural numbers,

(i) $A \in P^f$ if and only if $A \in P_{f(0)+1}$;

(ii) $A \in NP^f$ if and only if $A \in NP_{f(0)+1}$;

(iii) $A \in$ co-NP^f if and only if $A \in$ co-$NP_{f(0)+1}$.

PROOF. By suitability, $f(0) \leq f(x)$ for all x, and $f(0) = f(n)$ for all natural numbers n. Since any set in $P_{f(0)+1}$ is decided by an algorithm that takes fewer than $f(0)$ steps, it follows that $P_{f(0)+1} \subseteq P^f$. Conversely, suppose that $A \subseteq \omega$ and $A \in P^f$. So there is an algorithm that decides whether $x \in A$ in fewer than $f(x)$ steps. Although this algorithm might be allowed to take many steps on a complicated input x for which $f(x)$ may be large, we know since $A \subseteq \omega$ that the ultimate answer will be negative unless $x \in \omega$. Thus, we design a more efficient algorithm by rejecting any input x that does not code a natural number. It is easy to see that the binary sequences not coding natural numbers are precisely the sequence of all ones, plus those containing the substring 01. While continuing with the algorithm to decide A, our modified algorithm searches for the substring 01 in the input, and also turns on a flag if 0 is encountered in the input. This algorithm decides $n \in A$ in fewer than $f(n) = f(0)$ steps, and rejects all other input either in finitely many steps, if the input contains 01, or in ω steps, if the input has no zeros. It therefore places A in $P_{f(0)+1}$, as desired.

A similar argument establishes the result for NP^f and $NP_{f(0)+1}$. Specifically, if $A \in NP^f$, then there is a nondeterministic algorithm such that $x \in A$ if and only if the algorithm accepts (x, y) for some y. Once again, we can modify this algorithm to reject any input (x, y) in finitely many steps unless x codes some finite n, in which case the algorithm is carried out as before. The result is that $x \in A$ is decided in finite time unless $x = n \in \omega$, in which case it is decided in fewer than $f(n) = f(0)$ steps, placing A in $NP_{f(0)+1}$. The result for co-NP^f and co-$NP_{f(0)+1}$ follows by taking complements. ∎

The following result is essentially proved by [3, Theorem 3.1], but we derive it here as a corollary to the previous theorem and Theorem 5.16. Note that if $f : \mathbb{R} \to \text{ORD}$ is suitable, then $f(q) = f(0)$ for any finite q.

COROLLARY 6.2
If $f : \mathbb{R} \to \text{ORD}$ is suitable and $f(0) > \Sigma$, then P^f is properly contained in $NP^f \cap$ co-NP^f.

PROOF. This follows immediately from Theorems 5.16 and 6.1, because the halting problem h is a set of natural numbers in $NP_{\Sigma+2} \cap$ co-$NP_{\Sigma+2}$ and therefore in $NP^f \cap$ co-NP^f. But it is not decidable and consequently not in P^f. ∎

COROLLARY 6.3
If $f : \mathbb{R} \to \text{ORD}$ is suitable and $f(0)$ is clockable, but does not end a gap in the clockable ordinals, then P^f is properly contained in $NP^f \cap$ co-NP^f.

PROOF. By Theorem 6.1, the sets of natural numbers in P^f and $NP^f \cap$ co-NP^f are exactly those in $P_{\alpha+1}$ and $NP_{\alpha+1} \cap$ co-$NP_{\alpha+1}$, respectively, where $\alpha = f(0)$. Since $f(0)$ does not end a gap in the clockable ordinals, it follows that $\alpha + 1$ is neither a gap-ending ordinal nor the successor of a gap-ending ordinal. Therefore, by the proof of Corollary 5.6 there are sets of natural numbers in $NP_{\alpha+1} \cap$ co-$NP_{\alpha+1}$ that are not in $P_{\alpha+1}$. Consequently, there are sets of natural numbers in $NP^f \cap$ co-NP^f that are not in P^f. ∎

An instance of this settles the question for P^{++}.

COROLLARY 6.4
$P^{++} \neq NP^{++} \cap$ co-NP^{++}.

PROOF. This follows from Corollary 6.3 and the fact that $P^{++} = P^{f_1}$, where $f_1(x) = \omega_1^x + \omega + 1$. By [2, Theorem 3.2], the ordinal $\omega_1^{\text{CK}} + \omega$ is clockable, and consequently so is $\omega_1^{\text{CK}} + \omega + 1$. ∎

So the previous corollaries establish that $P^f \neq NP^f \cap$ co-NP^f for many or most functions f. But of course, we have examples of ordinals α for which $P_\alpha = NP_\alpha \cap$ co-NP_α, such as $\alpha = \omega_1^{\text{CK}}$ or $\alpha = \omega_1^{\text{CK}} + 1$. If f is the constant function $f(x) = \omega_1^{\text{CK}}$, then $P^f = P_{\omega_1^{\text{CK}}+1}$ and $NP^f = NP_{\omega_1^{\text{CK}}+1}$, so this is an example where $P^f = NP^f \cap$ co-NP^f, even when $P^f \neq NP^f$. The equation $P^+ = NP^+ \cap$ co-NP^+ provides another such example.

Acknowledgements

The research of Joel David Hamkins has been supported in part by grants from Georgia State University, the Research Foundation of CUNY and the National Science Foundation.

References

[1] J. Barwise. *Admissible Sets and Structures*. Perspectives in Mathematical Logic. Springer Verlag, 1975.

[2] J. D. Hamkins and A. Lewis. Infinite time Turing machines. *J. Symbolic Logic*, **65**, 567–604, 2000.

[3] J. D. Hamkins and P. Welch. $P^f \neq NP^f$ for almost all f. *Mathematical Logic Quarterly*, **49**, 536–540, 2003.

[4] A. S. Kechris. *Classical Descriptive Set Theory*. Graduate Texts in Mathematics. Spring-Verlag, New York, 1995.

[5] R. Mansfield and G. Weitkamp. *Recursive aspects of descriptive set theory*, Volume 11 of *Oxford Logic Guides*. Oxford University Press, New York, 1985.

[6] Y. N. Moschovakis. *Descriptive Set Theory*, Volume 100 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, 1980.

[7] R.-D. Schindler. P $\neq$ NP for infinite time Turing machines. *Monatshefte für Mathematik*, **139**, 335–340, 2003.

[8] P. Welch. Arithmetical quasi-inductive defintions and the transfinite action of one tape Turing machines. in preparation.

[9] P. Welch. On a question of Deolalikar, Hamkins and Schindler. available on the author's web page at http://www2.maths.bris.ac.uk/∼mapdw/dhs.ps.

[10] P. Welch. The lengths of infinite time Turing machine computations. *Bulletin of the London Mathematical Society*, **32**, 129–136, 2000.

[11] P. Welch. The transfinite action of 1 tape Turing machines. *To appear in CiE LNCS volume, New Computational Paradigms*, 2005.

Received 24 November 2003