Alternating Hierarchies for Time-Space Tradeoffs

Chris Pollett
Department of Computer Science
San Jose State University
1 Washington Square
San Jose, CA 95192
pollett@cs.sjsu.edu

Eric Miles
Department of Computer Science
San Jose State University
1 Washington Square
San Jose, CA 95192
enmiles@gmail.com

August 24, 2007– Draft

Abstract

Nepomnjaščiĭ's Theorem states that for all $0 \leq \epsilon < 1$ and $k > 0$ the class of languages recognized in nondeterministic time n^k and space n^ϵ, $\mathsf{NTISP}[n^k, n^\epsilon]$, is contained in the linear time hierarchy. By considering restrictions on the size of the universal quantifiers in the linear time hierarchy, this paper refines Nepomnjaščiĭ's result to give a sub-hierarchy, $\mathsf{Eu\text{-}LinH}$, of the linear time hierarchy that is contained in NP and which contains $\mathsf{NTISP}[n^k, n^\epsilon]$. Hence, $\mathsf{Eu\text{-}LinH}$ contains NL and SC. This paper investigates basic structural properties of $\mathsf{Eu\text{-}LinH}$. Then the relationships between $\mathsf{Eu\text{-}LinH}$ and the classes NL, SC, and NP are considered to see if they can shed light on the $\mathsf{NL} = \mathsf{NP}$ or $\mathsf{SC} = \mathsf{NP}$ questions. Finally, a new hierarchy, ξ-LinH, is defined to reduce the space requirements needed for the upper bound on $\mathsf{Eu\text{-}LinH}$.
Mathematics Subject Classification: 03F30, 68Q15
Keywords: structural complexity, linear time hierarchy, Nepomnjaščiĭ's Theorem

1 Introduction

Fortnow [3] began the study of time-space trade-offs as an approach to showing $\mathsf{L} \neq \mathsf{NP}$. His paper gives the first non-trivial time-space trade-offs for satisfiability. These results were subsequently improved upon and the interested reader can consult Fortnow et al [4] and Williams [16] for an introduction to the literature as well as current results. One tool used in these time-space results is Nepomnjaščiĭ's Theorem. The form of this theorem we are interested in shows that $\mathsf{NTISP}(n^k, n^\epsilon)$, the class of languages decidable simultaneously in nondeterministic time n^k and space n^ϵ for $0 < \epsilon < 1$ and $k > 0$, is contained in the linear time hierarchy, LinH. If one examines the proof of this result, one is struck that although a language L in $\mathsf{NTISP}(n^k, n^\epsilon)$ is seemingly mapped into the $\Sigma^{\mathsf{lin}}_{2k+1}$ level of the linear-time hierarchy, the universal quantifiers that appear are all of logarithmic size. It is thus natural to ask what is the complexity of the sub-class of Σ^{lin}_k where we restrict the universal quantifiers to be logarithmic size. It is this hierarchy that we investigate in the present paper.

Given a class of languages $\mathcal{C}$, let $\mathsf{E}\mathcal{C}$ (resp. $\mathsf{U}\mathcal{C}$) denote the class of languages of the form $\{x \mid \exists y \in \{0,1\}^{k\cdot|x|}\ \langle x, y\rangle \in L\}$ (resp. $\{x \mid \forall y \in \{0,1\}^{k\cdot|x|},\ \langle x, y\rangle \in L\}$) for some $L \in \mathcal{C}$ and

fixed integer k. (Note that we say $\{0,1\}^n$ to refer to the set of all binary strings with length no greater than n.) We use lower case, e or u to denote the case where the bounding term $k \cdot |x|$ is replaced with $k \log |x|$ for some fixed integer k. We will call these kind of quantifiers *sharply bounded*. Using this notation, we write $(\mathsf{Eu})_k^{\mathsf{lin}}$ for the class languages given by k quantifier blocks whose values are finally fed into a deterministic linear time languages, where each quantifier block is of the form an E quantifier followed by an u quantifier. By examining the proof of Nepomnjaščiĭ's theorem, we show every language in $\mathsf{NTISP}[n^k, n^\epsilon]$ can be represented as a language in $\mathsf{Eu\text{-}LinH} := \cup_k (\mathsf{Eu})_k^{\mathsf{lin}}$. We in fact show a level-wise result with an almost matching upper bound: $\mathsf{NTISP}(n^{k(1-\epsilon)}, n^\epsilon) \subseteq (\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTISP}[n^{k+1}, n]$. So our hierarchy is closely associated with the class of languages that can be solved using nondeterministic polynomial time and linear space. As it is simultaneously contained in the linear time hierarchy and in NP, it also contains NL and $\mathsf{SC} := \cup_k \cup_m \mathsf{DTISP}[n^k, \log^m n]$. As such our hierarchy might be a useful tool in trying to separate these classes from NP.

Quantifiers such as e and u have been considered before in the context of linear time and quasi-linear in Bloch, Buss, Goldsmith [1]. In particular, they look at a so-called sharply bounded hierarchy over quasi-linear time, using the notation $\tilde{\exists}$ and $\tilde{\forall}$ for sharply bounded quantifiers. Their class is contained in P and has interesting closure properties, but it is unclear if it contains a well-known class like NL.

For this paper, we are interested in the structural and closure properties of $\mathsf{Eu\text{-}LinH}$ and the connections between $\mathsf{Eu\text{-}LinH}$ and the classes NL, SC, and NP. As indicated above, one has the containments $\mathsf{NL}, \mathsf{SC} \subseteq \mathsf{Eu\text{-}LinH} \subseteq \mathsf{NP}$. Since NL and SC are closed under complement one also has $\mathsf{NL}, \mathsf{SC} \subseteq \cup_k (\mathsf{Ue})_k^{\mathsf{lin}} \subseteq \mathsf{co\text{-}NP}$. We first show some basic structural results such as if $(\mathsf{Eu})_k^{\mathsf{lin}} = (\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ then $\mathsf{NL} \neq \mathsf{NP}$ and $\mathsf{SC} \neq \mathsf{NP}$. We further show if $(\mathsf{Eu})_1^{\mathsf{lin}}$ is contained in $(\mathsf{Ue})_k^{\mathsf{lin}}$ for some fixed k, then $\mathsf{NP} = \mathsf{co\text{-}NP}$. We then consider possible inclusions between our class and the classes NL, SC, and NP. We show if NL or $\mathsf{SC} = \mathsf{Eu\text{-}LinH}$ then $\mathsf{NP} = \mathsf{co\text{-}NP}$, the polynomial time hierarchy is equal to the linear time hierarchy, and the linear time hierarchy is infinite. We then turn our attention to whether it is possible for $\mathsf{Eu\text{-}LinH}$ to equal NP. We show that if $\mathsf{Eu\text{-}LinH} = \mathsf{NP}$, then the $\mathsf{Eu\text{-}LinH}$ hierarchy is infinite, LinH contains the boolean hierarchy over NP, and either the linear time hierarchy is infinite or $\mathsf{NP} \neq \mathsf{co\text{-}NP}$. Then we show that if the total sum of the length bounds on the universal quantifiers in $\mathsf{Eu\text{-}LinH}$ hierarchy is bounded to $k \log n$, then the restricted hierarchy, $\mathsf{Eu\text{-}LinH}_k$, is strictly contained in NP. So in particular, if $\mathsf{Eu\text{-}LinH}_k$ contains NL, which would happen if $\mathsf{Eu\text{-}LinH}_k = \mathsf{Eu\text{-}LinH}$, then $\mathsf{NL} \neq \mathsf{NP}$. It might be possible to further restrict the sizes of the universal quantifiers if the space requirements in the containment $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTISP}[n^{k+1}, n]$ could be reduced. To this end we define a new quantifier ξ and obtain partial results in this direction.

This paper is organized as follows: The next section contains the notations and main definitions used in this paper. This is followed by a section with some basic containments and closure properties of $\mathsf{Eu\text{-}LinH}$. Then Nepomnjaščiĭ's Theorem is proven so as to connect it with $\mathsf{Eu\text{-}LinH}$. This is followed by a short section reminding the reader of some useful padding and diagonalization techniques. The next two sections contain our main results, the first of these gives conditional results concerning what would happen if the $\mathsf{Eu\text{-}LinH}$ collapses, the second of these proves our results concerning $\mathsf{Eu\text{-}LinH}$ and the classes NL, SC, and NP. The section which follows defines a new quantifier ξ and a new hierarchy $\xi\text{-}\mathsf{LinH}$, and uses these to show a containment for $\mathsf{Eu\text{-}LinH}$ with a slightly lower space bound. Finally,

there is a conclusion with some suggestions for further investigation.

2 Preliminaries

A good survey of basic machine models and complexity classes can be found in Johnson [10]. We briefly summarize here the models, classes and notation which will be used in this paper. We will use deterministic, nondeterministic, or alternating multi-tape Turing Machines as our basic machine models. These machines are assumed to have a read only input tape, a write only output tape, and some finite number of work tapes. Space is measured in terms of the total number of tape squares used on the work tapes. For the sake of simplicity, we assume without loss of generality that all languages are over the binary alphabet. We will often need at our disposal the ability to encode and decode pairs or finite sequences of strings. To do this, the sequence of values $\langle x_1, \ldots, x_n \rangle$ — when $n = 2$, one has a pairing operation — is defined to be the string obtained by replacing 0s and 1s in the x_is by 00 and 10 respectively and by inserting a 01 in between numbers.

We write $\mathsf{DTIME}[t(n)]$ and $\mathsf{NTIME}[t(n)]$ for the languages decided by machines which are respectively deterministic or nondeterministic, and which run for at most $t(n)$ steps on inputs of length n. We write $\Sigma_k\text{-}\mathsf{TIME}[t(n)]$ and $\Pi_k\text{-}\mathsf{TIME}[t(n)]$ for the languages decided in $t(n)$ steps by machines which alternate at most k times, in the former case the outermost alternation being existential, and in the latter case universal. We define $\mathsf{DSPACE}[s(n)]$ and $\mathsf{NSPACE}[s(n)]$ similarly except now the bound is on space rather than time. The classes $\mathsf{DTISP}[t(n), s(n)]$ and $\mathsf{NTISP}[t(n), s(n)]$ are the languages decided by machines in $t(n)$ time steps using at most $s(n)$ work tape squares in respectively the deterministic or nondeterministic machine model. For a language L, $\bar{L}$ is the complement of L, consisting of those strings x over the alphabet that are not in L. Given a class of languages $\mathcal{C}$, we write $\mathsf{co}\text{-}\mathcal{C}$ for the languages $\{\bar{L} \mid L \in \mathcal{C}\}$.

Given these basic definitions, the following well-studied complexity classes can now be defined: $\mathsf{DLIN} := \mathsf{DTIME}[n]$, $\mathsf{P} := \cup_k \mathsf{DTIME}[n^k]$, $\mathsf{NLIN} := \mathsf{NTIME}[n]$, $\mathsf{NP} := \cup_k \mathsf{NTIME}[n^k]$, $\mathsf{L} := \mathsf{DSPACE}[\log n]$, $\mathsf{NL} := \mathsf{NSPACE}[\log n]$, and $\mathsf{SC} := \cup_k \cup_m \mathsf{DTISP}[n^k, \log^m n]$. It is known that $\mathsf{L} \subseteq \mathsf{NL} = \mathsf{co}\text{-}\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP}$ and that $\mathsf{L} \subseteq \mathsf{SC} \subseteq \mathsf{P}$. As SC, which stands for Steve's Class in honor of Steve Cook, is less-known we mention that SC, in addition to the containments above, contains all the deterministic context free languages. Finally, we will be in interested in the linear and polynomial time hierarchies: $\mathsf{LinH} := \cup_j \Sigma_j\text{-}\mathsf{TIME}[n]$ and $\mathsf{PH} := \cup_j \Sigma_j^{\mathsf{p}}$, where $\Sigma_j^{\mathsf{p}} := \cup_k \Sigma_j\text{-}\mathsf{TIME}[n^k]$ and $\Pi_j^{\mathsf{p}} := \cup_k \Pi_j\text{-}\mathsf{TIME}[n^k]$. At the bottom levels of these hierarchies we have $\mathsf{NLIN} = \Sigma_1\text{-}\mathsf{TIME}[n]$, $\mathsf{co}\text{-}\mathsf{NLIN} = \Pi_1\text{-}\mathsf{TIME}[n]$ and $\Sigma_1^{\mathsf{p}} = \mathsf{NP}$, $\Pi_1^{\mathsf{p}} = \mathsf{co}\text{-}\mathsf{NP}$. We also have trivially that $\Sigma_j\text{-}\mathsf{TIME}[n] \subseteq \Pi_{j+1}\text{-}\mathsf{TIME}[n] \subseteq \Sigma_{j+2}\text{-}\mathsf{TIME}[n]$ and $\Sigma_j^{\mathsf{p}} \subseteq \Pi_{j+1}^{\mathsf{p}} \subseteq \Sigma_{j+2}^{\mathsf{p}}$, so given our definition for each j, $\Pi_j\text{-}\mathsf{TIME}[n] \subseteq \mathsf{LinH}$ and $\Pi_j^{\mathsf{p}} \subseteq \mathsf{PH}$.

We consider strings w built out of the quantifier alphabet E, U, e, and u. Let τ be a collection of nondecreasing functions on the nonnegative integers. We will use τ to interpret quantifiers. The quantifier E (resp. U) is supposed to represent a quantifier of the form $\exists y \in \{0,1\}^{t(|x|)}$ (resp. $\forall y \in \{0,1\}^{t(|x|)}$) for some $t \in \tau$; the quantifier e (resp. u) is supposed to represent a quantifier of the form $\exists y \in \{0,1\}^{\log t(|x|)}$ (resp. $\forall y \in \{0,1\}^{\log t(|x|)}$). Given a class $\mathcal{C}$ of languages, a string $w = w_k w_{k-1} \cdots w_0$ over our quantifier alphabet, and a set of functions τ as above, we define $(w\mathcal{C})^\tau$ by induction on k. If $k = 0$ then $(w\mathcal{C})^\tau = \mathcal{C}$. For $k > 0$, let $w' = w_{k-1} \cdots w_0$. Then one has four cases depending on the quantifier w_k: For

instance, if w_k is E, then $(w\mathcal{C})^\tau$ is the class of languages of the form $L = \{x \mid \exists y \in \{0,1\}^{t(|x|)} \langle x, y\rangle \in L'\}$ for some $t \in \tau$ and $L' \in (w'\mathcal{C})^\tau$. The other three cases are similarly defined using our way of interpreting U, e, and u explained above. The main choices for τ we will be interested in are lin which consists of functions of the form $\ell(n) := k \cdot n$ for some $k > 0$, and poly which consists of the polynomials $p(n) := n^k$ for some $k > 0$. The principal starting choices for the complexity class $\mathcal{C}$ we will be interested in are DLIN and P. We will use two further ways to abbreviate our notation: (1) we will write $(w)^{\mathsf{lin}}$ for $(w\mathsf{DLIN})^{\mathsf{lin}}$ and write $(w)^{\mathsf{poly}}$ for $(w\mathsf{P})^{\mathsf{poly}}$; (2) we will sometimes use subscripted notations such as $(w)_m$ as abbreviation for the string $\overbrace{w \cdots\cdots\cdots\cdots w}^{m \text{ concatentations}}$. As an example of our notation scheme, the class $\mathsf{NP} = \Sigma_1^{\mathsf{p}}$ could be written as $(\mathsf{E})^{\mathsf{poly}}$ and the class $\mathsf{co\text{-}NP} = \Pi_1^{\mathsf{p}}$ could be written as $(\mathsf{U})^{\mathsf{poly}}$. The levels of the polynomial hierarchy for $k > 1$ could be defined by saying $\Sigma_k^{\mathsf{p}} = (\mathsf{E}\Pi_{k-1}^{\mathsf{p}})^{\mathsf{poly}}$ and $\Pi_k^{\mathsf{p}} = (\mathsf{U}\Sigma_{k-1}^{\mathsf{p}})^{\mathsf{poly}}$. Finally, the new classes we will be interested in for this paper are $(\mathsf{Eu})_k^{\mathsf{lin}} := ((\mathsf{Eu})_k)^{\mathsf{lin}}$ and $(\mathsf{Ue})_k^{\mathsf{lin}} := ((\mathsf{Ue}_k))^{\mathsf{lin}}$ for some integer $k \geq 0$. We will write $\mathsf{Eu\text{-}LinH}$ for the whole hierarchy $\cup_k (\mathsf{Eu})_k^{\mathsf{lin}}$ and $\mathsf{Ue\text{-}LinH}$ for the whole hierarchy $\cup_k (\mathsf{Ue})_k^{\mathsf{lin}}$.

3 Basic Closure Properties and Containments

This section explores some of the basic relationships between, and closure properties of, our newly defined complexity classes $(\mathsf{Eu})_k^{\mathsf{lin}}$, $(\mathsf{Ue})_k^{\mathsf{lin}}$, $\mathsf{Eu\text{-}LinH}$, $\mathsf{Ue\text{-}LinH}$. To begin we observe $(\mathsf{Eu})_k^{\mathsf{lin}}$ and $(\mathsf{Ue})_k^{\mathsf{lin}}$ are complements of each other. We also connect these classes to the levels of the linear time hierarchy.

Proposition 1 *(a) For $k \geq 0$, $(\mathsf{Eu})_k^{\mathsf{lin}} = \mathsf{co\text{-}}(\mathsf{Ue})_k^{\mathsf{lin}}$. (b) For $k \geq 0$, $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \Sigma_{2k}\text{-}\mathsf{TIME}[n]$ and $(\mathsf{Ue})_k^{\mathsf{lin}} \subseteq \Pi_{2k}\text{-}\mathsf{TIME}[n]$. So both $\mathsf{Eu\text{-}LinH}$ and $\mathsf{Ue\text{-}LinH}$ are contained in LinH.*

Proof. For (a), in the $k = 0$ case, both classes are DLIN. Further DLIN is closed under complement since given a machine M for $L \in \mathsf{DLIN}$ we can swap its accept and reject states to obtain a machine for $\bar{L}$. Now assume the proposition is true up to some $k \geq 0$, and let $t \in \mathsf{lin}$. Remember that $x \in (\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ iff $\exists y_1 \in \{0,1\}^{t(|x|)}$ such that $\forall y_2 \in \{0,1\}^{\log t(|x|)}$, $\langle x, y_1, y_2\rangle \in (\mathsf{Eu})_k^{\mathsf{lin}}$. Then it is clear that $x \in \mathsf{co\text{-}}(\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ iff $\neg\exists y_1 \in \{0,1\}^{t(|x|)}$ such that $\forall y_2 \in \{0,1\}^{\log t(|x|)}$, $\langle x, y_1, y_2\rangle \in (\mathsf{Eu})_k^{\mathsf{lin}}$. By the induction hypothesis we have $\langle x, y_1, y_2\rangle \notin (\mathsf{Eu})_k^{\mathsf{lin}}$ iff $\langle x, y_1, y_2\rangle \in (\mathsf{Ue})_k^{\mathsf{lin}}$; using that, and the fact that in first order logic $\neg\exists\forall$ is equivalent to $\forall\exists\neg$, we can say $x \in \mathsf{co\text{-}}(\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ iff $\forall y_1 \in \{0,1\}^{t(|x|)}$, $\exists y_2 \in \{0,1\}^{\log t(|x|)}$ such that $\langle x, y_1, y_2\rangle \in (\mathsf{Ue})_k^{\mathsf{lin}}$. This shows that $x \in \mathsf{co\text{-}}(\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ iff $x \in (\mathsf{Ue})_{k+1}^{\mathsf{lin}}$, so the induction step is proved.

To prove (b), we first note by increasing the size of the tape alphabet we can increase the number of computations that be done in n steps by a constant factor (linear speed-up), so $\Sigma_{2k}\text{-}\mathsf{TIME}[O(n)] = \Sigma_{2k}\text{-}\mathsf{TIME}[n]$. Further for a language L in $(\mathsf{Eu})_k^{\mathsf{lin}}$ an alternating machine operating in time $O(n)$ can guess first the outermost quantifier, then universally guess the small universal quantifier, and so on, with at most $2k$ alternations. Finally, it could simulate the innermost deterministic linear time machine used in deciding L. Thus, $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \Sigma_{2k}\text{-}\mathsf{TIME}[n]$. □

Proposition 2 *(a)* LinH *is closed under union, intersection, and complement. (b)* $(\mathsf{Eu})_k^{\mathsf{lin}}$, $(\mathsf{Ue})_k^{\mathsf{lin}}$, $\mathsf{Eu\text{-}LinH}$, *and* $\mathsf{Ue\text{-}LinH}$ *are closed under union and intersection.*

Proof. LinH is closed under complement since if L is in LinH then it is in $\Sigma_k\text{-}\mathsf{TIME}[n]$ for some k, so $\bar{L}$ will be in $\Pi_k\text{-}\mathsf{TIME}[n]$. As we already observed in the preliminaries $\Pi_k\text{-}\mathsf{TIME}[n] \subseteq \Sigma_{k+1}\text{-}\mathsf{TIME}[n] \subseteq \mathsf{LinH}$. For both parts (a) and (b) of the proposition, the arguments for closure under union and intersection are essentially the same, so we indicate the proof for $(\mathsf{Eu})_k^{\mathsf{lin}}$. Let L_1 and L_2 be two $(\mathsf{Eu})_k^{\mathsf{lin}}$ languages. On input x we can use pairing to guess pairs of strings for each E or u quantifier, one of the two values to be used to check if x is in L_1 and the other to check if x is in L_2. This at most increases the size of the bound on each quantifier by a constant multiplicative factor, so this sequence of existential and universal guesses can still be done in $(\mathsf{Eu})_k^{\mathsf{lin}}$. The deterministic machine for the union, first unpacks each pair to produce a string to start simulating the DLIN machine for L_1. If this simulation accepts then the union machine accepts; otherwise, the other components of each pair are extracted and the machine simulates the DLIN machine for L_2. If it accepts, then the union machine accepts; otherwise it reject. The total runtime will be at most the sum of the two runtimes so still linear. For intersections the same sort of algorithm is done except that the machine always performs the simulation of the DLIN machine for L_1 followed by the DLIN machine for L_2 and only accepts if both accepted. □

Proposition 3 *(a)* $\mathsf{NLIN} \subseteq (\mathsf{Eu})_1^{\mathsf{lin}}$ *and* $\mathsf{co\text{-}NLIN} \subseteq (\mathsf{Ue})_1^{\mathsf{lin}}$. *(b) For* $k \geq 0$, $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTISP}[n^{k+1}, n] \subseteq \mathsf{NTIME}[n^{k+1}]$, *so* $\mathsf{Eu\text{-}LinH}$ *is contained in* NP *and* $\mathsf{Ue\text{-}LinH}$ *is contained in* $\mathsf{co\text{-}NP}$. *(c) For* $k \geq 1$, $(\mathsf{Eu})_k^{\mathsf{poly}} = \mathsf{NP}$.

Proof. For (a) notice that using an E quantifier to guess a string w of nondeterministic moves of an NLIN machine M on input x, a deterministic linear time machine could then use these choices together with x to decide M's language. This show $\mathsf{NLIN} \subseteq (\mathsf{E})^{\mathsf{lin}} \subseteq (\mathsf{Eu})_1^{\mathsf{lin}}$. By taking complements we have the $\mathsf{co\text{-}NLIN}$ result.

To prove (b) we show by induction on k that $(\mathsf{Eu})_k^{\mathsf{lin}}$ is contained in $\mathsf{NTISP}[n^{k+1}, n]$. As $\mathsf{Eu\text{-}LinH} = \cup_k (\mathsf{Eu})_k^{\mathsf{lin}}$ and $\mathsf{NP} = \cup_k \mathsf{NTIME}[n^{k+1}]$, this implies $\mathsf{Eu\text{-}LinH}$ is contained in NP. The fact that $\mathsf{Ue\text{-}LinH} \subseteq \mathsf{co\text{-}NP}$ then follows from Proposition 1 (a). For the $k = 0$ case, we obviously have $\mathsf{DLIN} \subseteq \mathsf{NLIN} = \mathsf{NTISP}[n, n]$. So assume the statement is true up to some $k \geq 0$. Let L be a language in $(\mathsf{u}(\mathsf{Eu})_k)^{\mathsf{lin}}$. We will argue L is contained in $\mathsf{NTISP}[n^{k+2}, n]$. By the definition of L there must be some $(\mathsf{Eu})_k^{\mathsf{lin}}$ languages L' such that $x \in L$ if and only if $\forall y \in \{0,1\}^{\log t(|x|)}, \langle x, y\rangle \in L'$. Here $t(n)$ is some growth rate from lin so is of the form $m \cdot |x|$. By our induction hypothesis there is a $\mathsf{NTISP}[n^{k+1}, n]$ machine M' which decides L'. Let M be the machine which cycles through the $m \cdot |x|$ strings y of length $\log(m \cdot |x|)$, reusing the space. For each y, M then simulates M' on $\langle x, y\rangle$. If M' ever rejects, then M rejects. Otherwise, if M' accepts all such strings, then M accepts. This involves less than a linear amount of additional space. Further, M's runtime is $O(m \cdot |x|)$ times longer than the runtime of M', so using linear speed-up, M will operate in time n^{k+2}. Further, if $x \in L$ then for every string y with $|y| \leq \log(m \cdot |x|)$, $\langle x, y\rangle \in L'$, so there must be some path making M' accept. These accepting paths of M' for each y, taken together, can be used to make an accepting path for M. On the other hand, if $x \notin L$, then for some y, there is no path which makes M' accepts $\langle x, y\rangle$, and so when this y is checked by M, it too will reject. Thus, we have shown $\mathsf{u}(\mathsf{Eu})_k^{\mathsf{lin}}$ is contained in $\mathsf{NTISP}[n^{k+2}, n]$. But then $(\mathsf{Eu}(\mathsf{Eu})_k)^{\mathsf{lin}} = (\mathsf{Eu})_{k+1}^{\mathsf{lin}}$ will

also be in $\mathsf{NTISP}[n^{k+2}, n]$ since given a language in $(\mathsf{Eu})^{\mathsf{lin}}_{k+1}$, an $\mathsf{NTISP}[n^{k+2}, n]$ machine could in nondeterministic linear time and space guess the outermost existential quantifier and then run the $\mathsf{NTISP}[n^{k+2}, n]$ machine for the corresponding language in $(\mathsf{u}(\mathsf{Eu})_k)^{\mathsf{lin}}$. Hence, the induction step is proved and the result follows.

For (c), we first note by the same argument as in (a), we have $\mathsf{NP} \subseteq (\mathsf{E})^{\mathsf{poly}} \subseteq (\mathsf{Eu})^{\mathsf{poly}}_k$ for $k \geq 1$. To see $(\mathsf{Eu})^{\mathsf{poly}}_k \subseteq \mathsf{NP}$, we can use the same algorithm as in (b). Guessing an E quantifier will now involve guessing a string of polynomial length, but this can be handle in NP; simulating a u quantifier will involve cycling over the at $|x|^m$ string of length $\log t(|x|)$ where t is a polynomial of the form $|x|^m$ for some m, so can be done in polynomial time. □

Given the result (b) above it is reasonable to ask if $(\mathsf{Eu})^{\mathsf{lin}}_k = \mathsf{NTISP}[n^{k+1}, n]$ or if $(\mathsf{Eu})^{\mathsf{lin}}_k = \mathsf{NTIME}[n^{k+1}]$ for $k > 0$? We conjecture the latter is not the case and we will give some evidence for this in Corollary 4. Nevertheless, the results of the next section can be viewed as telling us that $(\mathsf{Eu})^{\mathsf{lin}}_k$ might be close to $\mathsf{NTISP}[n^{k+1}, n]$.

4 Nepomnjaščiĭ's Theorem

Theorem 1 *For $k \geq 1$ and $0 \leq \epsilon < 1$, $\mathsf{NTISP}(n^{k(1-\epsilon)}, n^{\epsilon})$ is contained in $(\mathsf{Eu})^{\mathsf{lin}}_k$.*

Proof. Our proof of Nepomnjaščiĭ's Theorem is essentially the same as found in say Hájek and Pudlák [8] or Fortnow et al. [4]. Here, though, we emphasize the details that imply our containment. Consider a k-tape Turing Machine M for a language L in $\mathsf{NSPACE}(n^{\epsilon})$. To make our argument easier, we will assume that when M halts, it erases its tapes, moves to the left-most square on each tape, and then either halts accepting or rejecting. A configuration C of M is $k+3$ tuple $\langle q, i, c, t_1, \ldots, t_k \rangle$ representing the state of M, the index of the square being read on the input tape, the character being written to the output tape, and the strings representing the visited contents of each work tape. The position of work tape heads are indicated in each t_i by using an alphabet symbol with an underscore beneath it for that tape square.

Consider the language L_k consisting of 4-tuples of the form $(x, C_s, C_t, 1^{|x|^{k(1-\epsilon)}})$ such that C_s and C_t represent configurations of M on input x, and there is some $|x|^{k(1-\epsilon)}$-step computation of M beginning in configuration C_s and ending in configuration C_t. We proceed by induction on k and argue that L_k is in $(\mathsf{Eu})^{\mathsf{lin}}_k$. When $k = 1$, we need to argue that L_1 is contained in $(\mathsf{Eu})^{\mathsf{lin}}_1$. Verifying that C_s and C_t are indeed configurations of M can be done in linear time in C_s and C_t. Using the E quantifier we can guess a sequence $C_1, \ldots, C_v$ of configurations of M where $C_s = C_1$, $C_t = C_v$, $v = |x|^{1-\epsilon}$. This is possible since each configuration has size $|x|^{\epsilon}$ and $v = |x|^{1-\epsilon}$, so the string we are guessing has length $O(|x|)$. Using a u quantifier to guess every $i \in \{1, \ldots, v-1\}$, one can then verify in deterministic linear time that (for each i) C_{i+1} follows from C_i according to a transition in M, or, if C_i is a halted configuration, that $C_i = C_{i+1}$. This shows the result holds for $k = 1$.

Assume $L_k \in (\mathsf{Eu})^{\mathsf{lin}}_k$ holds for $k \geq 1$ and consider an instance $(x, C_s, C_t, 1^{|x|^{(k+1)(1-\epsilon)}})$ of L_{k+1}. Again, with an E quantifier we can guess a sequence $C_1, \ldots, C_v$ of configurations of M where $C_s = C_1$, $C_t = C_v$, $v = |x|^{1-\epsilon}$. Then using a u quantifier we can check if for each $i = 1, \ldots v-1$, that we can guess the 4-tuple $(x, C_i, C_{i+1}, 1^{|x|^{k(1-\epsilon)}})$ and verify that it is in L_k. This shows L_{k+1} is in $(\mathsf{EuE}(\mathsf{Eu})_k)^{\mathsf{lin}}$, but $(\mathsf{E}(\mathsf{Eu})_k)^{\mathsf{lin}} = (\mathsf{Eu})^{\mathsf{lin}}_k$ because we can always

combine outer two existential quantifiers of linear size into one existential quantifier still of linear size. Hence, we have L_{k+1} is in $(\mathsf{Eu})^{\mathsf{lin}}_{k+1}$ and the induction step is proved.

Now if M was for a language L that was not only in $\mathsf{NSPACE}[n^\epsilon]$ but in $\mathsf{NTISP}[n^{k(1-\epsilon)}, n^\epsilon]$, then on input x we could guess the 4-tuple $(x, C_s, C_h, |x|^{k(1-\epsilon)})$, where C_s is the starting configuration of M on x and C_h is the unique possible accepting, halting configuration; and check if this 4-tuple is in L_k. This would show $L \in \mathsf{E}(\mathsf{Eu})_k)^{\mathsf{lin}}$, but as we have just argued, $(\mathsf{E}(\mathsf{Eu})_k)^{\mathsf{lin}} = (\mathsf{Eu})^{\mathsf{lin}}_k$. Hence, the theorem follows. □

Combining Proposition 3 and Theorem 1 above, we have an almost optimal containment for the levels of our hierarchy:

Corollary 1 *For $k \geq 1$ and $0 \leq \epsilon < 1$, $\mathsf{NTISP}(n^{k(1-\epsilon)}, n^\epsilon) \subseteq (\mathsf{Eu})^{\mathsf{lin}}_k \subseteq \mathsf{NTISP}[n^{k+1}, n]$.*

The next corollary also follows from Theorem 1 since $\mathsf{NL} = \cup_k \mathsf{NTISP}(n^k, \log n)$ and $\mathsf{SC} := \cup_k \cup_m \mathsf{DTISP}[n^k, \log^m n]$.

Corollary 2 *(a)* NL *is contained in* $\mathsf{Eu\text{-}LinH}$. *(b)* SC *is contained in* $\mathsf{Eu\text{-}LinH}$.

The following graph shows the inclusions we have just seen.

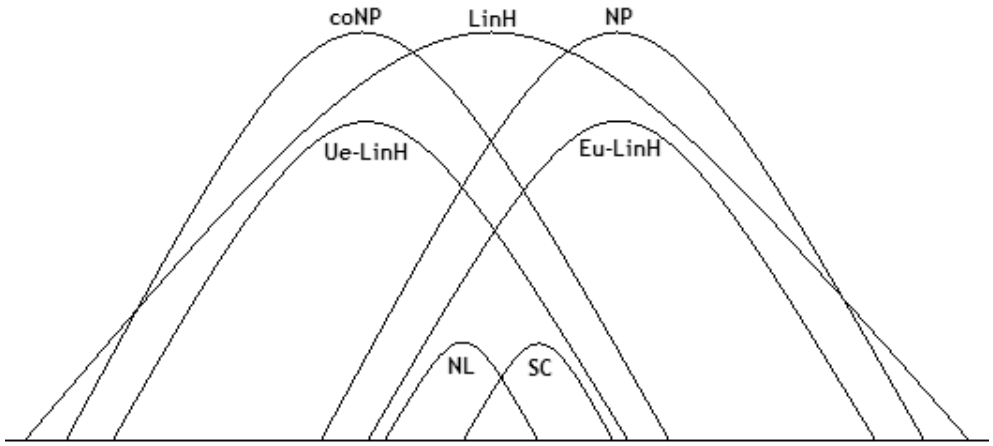

Figure 1: Inclusions related to $\mathsf{Eu\text{-}LinH}$

5 Common Arguments

In this section, we briefly present well-known results concerning padding and time hierarchies which we will need for our later results. To begin we say a function $t(n)$ is *time constructible* if there is a $t(n)$ time bounded Turing Machine M such that for each n, M runs for exactly $t(n)$ steps on an input of length n. Given a complexity class $\mathcal{C}$, we write $R_{\mathsf{pad}}(\mathcal{C})$ for the class of languages L for which there is a logspace computable, time constructible function $t(n) \geq n$, such that $x \in L$ if and only if $x0^i \in L'$ where $L' \in \mathcal{C}$ and $|x0^i| = t(n)$. The string $x0^i$ here is the string x padded by a string of 0's of length i. The next lemma collects together the main results we need about the padded versions of the complexity classes of this paper.

Lemma 1 *(a) Let $\mathcal{C}_1 \subseteq \mathcal{C}_2$ be classes of languages. Then $R_{\mathsf{pad}}(\mathcal{C}_1) \subseteq R_{\mathsf{pad}}(\mathcal{C}_2)$. (b) $R_{\mathsf{pad}}(\mathsf{NLIN}) = R_{\mathsf{pad}}(\mathsf{Eu\text{-}LinH}) = R_{\mathsf{pad}}(\mathsf{NP}) = \mathsf{NP}$. (c) $R_{\mathsf{pad}}(\mathsf{co\text{-}NLIN}) = R_{\mathsf{pad}}(\mathsf{Ue\text{-}LinH}) = R_{\mathsf{pad}}(\mathsf{co\text{-}NP}) = \mathsf{co\text{-}NP}$.*

Proof. (a) If L is a language in $R_{\mathsf{pad}}(\mathcal{C}_1)$. Then there must be some language L' in $\mathcal{C}_1$ and a time constructible $t(n)$ such that $x \in L$ if and only if $x0^i \in L'$ where $|x0^i| = t(n)$. Since $\mathcal{C}_1 \subseteq \mathcal{C}_2$, this same L' and t show L is in $R_{\mathsf{pad}}(\mathcal{C}_2)$.

(b) As $\mathsf{NLIN} \subseteq \mathsf{Eu\text{-}LinH} \subseteq \mathsf{NP}$ by (a) it suffices to show $R_{\mathsf{pad}}(\mathsf{NP}) \subseteq \mathsf{NP}$ and $\mathsf{NP} \subseteq R_{\mathsf{pad}}(\mathsf{NLIN})$. The first inclusion follows since given a language $L \in R_{\mathsf{pad}}(\mathsf{NP})$ and an input x a nondeterministic Turing machine can compute the logspace computable, time constructible function $t(n)$ for L to get $x0^i$. It can then simulate the NP machine for this language on $x0^i$. The total runtime will be bounded by a composition of polynomials, showing L is in NP. For the second inclusion, let $L \in \mathsf{NP}$ be decidable in time bounded by a polynomial $p(n)$. Then the language $\{x0^{p(|x|)-|x|} \mid x \in L\}$ will be in NLIN and show $L \in R_{\mathsf{pad}}(\mathsf{NLIN})$.

The argument for (c) is essentially the same as for (b), except that the inclusions shown are $R_{\mathsf{pad}}(\mathsf{co\text{-}NP}) \subseteq \mathsf{co\text{-}NP}$ and $\mathsf{co\text{-}NP} \subseteq R_{\mathsf{pad}}(\mathsf{co\text{-}NLIN})$. □

We next turn to two diagonalization results for time complexity classes which we will use but not prove. The first of these is due to Chandra and Stockmeyer [2], which Williams [16] gives the catchy name: The No Complementary Speedup Theorem. The second of these is the Nondeterministic Time Hierarchy Theorem [14, 17].

Theorem 2 *Let k be a positive integer and t a time constructible function. Then Π_k-$\mathsf{TIME}[t] \not\subseteq \Sigma_k$-$\mathsf{TIME}[o(t)]$.*

Theorem 3 *Let $t_1(n)$ and $t_2(n)$ be functions with $t_2(n)$ time constructible. If $t_1(n+1) \in o(t_2(n))$ then $\mathsf{NTIME}[t_1] \subsetneq \mathsf{NTIME}[t_2]$.*

6 Levels of Eu-LinH

In this section we present consequences of Eu-LinH being contained in Ue-LinH and we consider the question of whether or not the Eu-LinH hierarchy is infinite.

Theorem 4 *If $(\mathsf{Eu})^{\mathsf{lin}}_1$ is contained in $(\mathsf{Ue})^{\mathsf{lin}}_k$ then $\mathsf{NP} = \mathsf{co\text{-}NP}$.*

Proof. As $\mathsf{NLIN} \subseteq (\mathsf{Eu})^{\mathsf{lin}}_1$, this implies $\mathsf{NLIN} \subseteq (\mathsf{Ue})^{\mathsf{lin}}_k$, and thus $\mathsf{NLIN} \subseteq \mathsf{Ue\text{-}LinH}$. So by Lemma 1 we have $\mathsf{NP} = R_{\mathsf{pad}}(\mathsf{NLIN}) \subseteq R_{\mathsf{pad}}(\mathsf{Ue\text{-}LinH}) = \mathsf{co\text{-}NP}$, and hence $\mathsf{NP} = \mathsf{co\text{-}NP}$ □

Theorem 5 *Let $k > 0$. If $(\mathsf{Eu})^{\mathsf{lin}}_k = (\mathsf{Eu})^{\mathsf{lin}}_{k+1}$ then $\mathsf{NL} \neq \mathsf{NP}$ and $\mathsf{SC} \neq \mathsf{NP}$.*

Proof. First, if $(\mathsf{Eu})^{\mathsf{lin}}_k = (\mathsf{Eu})^{\mathsf{lin}}_{k+1}$, then

$$(\mathsf{Eu})^{\mathsf{lin}}_{k+2} = (\mathsf{Eu}(\mathsf{Eu})_{k+1})^{\mathsf{lin}} = (\mathsf{Eu}(\mathsf{Eu})_k)^{\mathsf{lin}} = (\mathsf{Eu})^{\mathsf{lin}}_k.$$

Continuing up the hierarchy in this fashion one thus has $(\mathsf{Eu})^{\mathsf{lin}}_k = \mathsf{Eu\text{-}LinH}$. By Proposition 3 (b), this implies $\mathsf{Eu\text{-}LinH} \subseteq \mathsf{NTIME}[n^{k+1}] \subsetneq \mathsf{NP}$. The result then follows, as by Theorem 1, we know $\mathsf{SC} \subseteq \mathsf{Eu\text{-}LinH}$ and $\mathsf{NL} \subseteq \mathsf{Eu\text{-}LinH}$. □

7 Other complexity classes and Eu-LinH

In this section we investigate various consequences of certain inclusions holding among Eu-LinH, LinH, SC, NL, and NP.

Proposition 4 *If* $\mathsf{LinH} = \mathsf{PH}$ *then for all* k, $\mathsf{LinH} \neq \Sigma_k\text{-}\mathsf{TIME}[n]$. *That is, the linear time hierarchy is infinite.*

Proof. By Theorem 2 we know that $\Pi_k\text{-}\mathsf{TIME}[n^2] \not\subseteq \Sigma_k\text{-}\mathsf{TIME}[n]$. We also know $\Pi_k\text{-}\mathsf{TIME}[n^2] \subseteq \Pi_k^{\mathsf{p}} \subseteq \Sigma_{k+1}^{\mathsf{p}}$. Therefore, if $\mathsf{LinH} = \Sigma_k\text{-}\mathsf{TIME}[n]$ for some k, then it is strictly contained in PH. □

Corollary 3 *If* $\mathsf{co\text{-}NP} \subseteq \mathsf{Eu\text{-}LinH}$ *then for all* k, $\mathsf{LinH} \neq \Sigma_k\text{-}\mathsf{TIME}[n]$.

Proof. Proposition 3 (b) and $\mathsf{co\text{-}NP} \subseteq \mathsf{Eu\text{-}LinH}$ would imply $\mathsf{co\text{-}NP} \subseteq \mathsf{Eu\text{-}LinH} \subseteq \mathsf{NP}$. Hence, $\mathsf{NP} = \mathsf{co\text{-}NP} = \mathsf{Eu\text{-}LinH}$. Further as $\mathsf{NP} = \mathsf{co\text{-}NP}$ implies $\mathsf{NP} = \mathsf{PH}$ and $\mathsf{Eu\text{-}LinH} \subseteq \mathsf{LinH} \subseteq \mathsf{PH}$, we would have $\mathsf{LinH} = \mathsf{PH}$ and so the result follows from Proposition 4. □

For the next result recall that the boolean hierarchy above NP is defined as $\mathsf{BH} := \cup_i \mathsf{BH}_i$, where $\mathsf{BH}_1 = \mathsf{NP}$, where BH_{2i} is the class of languages consisting of the intersection of a language in BH_{2i-1} with a language in co-NP, and where BH_{2i+1} is the class of languages consisting of the intersection of a language in BH_{2i} with a language in NP. Kadin [6] has shown if this hierarchy collapses so does the polynomial hierarchy. We next consider some consequences of Eu-LinH being equal to NP.

Theorem 6 *If* $\mathsf{Eu\text{-}LinH} = \mathsf{NP}$ *then: (a) For all* $k \geq 0$, $\mathsf{Eu\text{-}LinH} \neq (\mathsf{Eu})_k^{\mathsf{lin}}$. *(b)* LinH *contains* BH. *(c) Either for all* k, $\mathsf{LinH} \neq \Sigma_k\text{-}\mathsf{TIME}[n]$ *or* $\mathsf{NP} \neq \mathsf{co\text{-}NP}$.

Proof. To see (a) notice Proposition 3 (b) implies $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTIME}[n^{k+1}] \subsetneq \mathsf{NP}$, where $\mathsf{NTIME}[n^{k+1}] \subsetneq \mathsf{NP}$ follows by the nondeterministic time hierarchy theorem (Theorem 3).

For (b) observe that if $\mathsf{Eu\text{-}LinH} = \mathsf{NP}$ then by Proposition 3 (b) we have $\mathsf{NP} = \mathsf{Eu\text{-}LinH} \subseteq \mathsf{LinH}$. The result then follows as LinH is closed under unions, intersections, and complements.

Lastly, for (c) suppose $\mathsf{Eu\text{-}LinH} = \mathsf{NP}$, $\mathsf{NP} = \mathsf{co\text{-}NP}$, and $\mathsf{LinH} = \Sigma_k\text{-}\mathsf{TIME}[n]$ for some k. Then $\mathsf{Eu\text{-}LinH} = \mathsf{NP} = \mathsf{co\text{-}NP} = \mathsf{PH}$, as $\mathsf{Eu\text{-}LinH} \subseteq \mathsf{LinH} \subseteq \mathsf{PH}$ we have $\mathsf{LinH} = \mathsf{PH}$. Thus, $\mathsf{LinH} = \Sigma_k\text{-}\mathsf{TIME}[n]$ contradicts Proposition 4. □

At the end of Section 3, we said it was unlikely that $(\mathsf{Eu})_m^{\mathsf{lin}} = \mathsf{NTIME}[n^{m+1}]$. The next corollary gives some consequences of this equality.

Corollary 4 *If for all* $m' > 0$, *there is a* $m > m'$ *such that* $(\mathsf{Eu})_m^{\mathsf{lin}} = \mathsf{NTIME}[n^{m+1}]$, *then: (a) For all* $k \geq 0$, $\mathsf{Eu\text{-}LinH} \neq (\mathsf{Eu})_k^{\mathsf{lin}}$. *(b)* LinH *contains* BH. *(c) Either for all* k, $\mathsf{LinH} \neq \Sigma_k\text{-}\mathsf{TIME}[n]$ *or* $\mathsf{NP} \neq \mathsf{co\text{-}NP}$.

Proof. The hypothesis above implies $\mathsf{Eu\text{-}LinH} = \mathsf{NP}$ as there are unboundedly large m satisfying $(\mathsf{Eu})_m^{\mathsf{lin}} = \mathsf{NTIME}[n^{m+1}]$, so the union $\cup_m (\mathsf{Eu})_m^{\mathsf{lin}}$ for these m's is Eu-LinH and the union $\cup_m \mathsf{NTIME}[n^{m+1}]$ is NP. The result then follows from Theorem 6. □

Theorem 7 *(a) If* $\mathsf{Eu\text{-}LinH} \subseteq \mathsf{co\text{-}NP}$ *then* $\mathsf{NP} = \mathsf{PH}$. *(b) If* $\mathsf{Eu\text{-}LinH} = \mathsf{LinH}$ *then* $\mathsf{NP} = \mathsf{PH}$. *(c) If* $\mathsf{SC} = \mathsf{Eu\text{-}LinH}$ *or* $\mathsf{NL} = \mathsf{Eu\text{-}LinH}$ *then* $\mathsf{Eu\text{-}LinH} = \mathsf{LinH} = \mathsf{NP} = \mathsf{PH}$.

Proof. For (a), if Eu-LinH $\subseteq$ co-NP, then in particular NLIN $\subseteq$ co-NP and so by padding NP $\subseteq$ co-NP. Thus, NP = co-NP = PH.

To see part (b) notice that LinH contains co-NLIN, so if Eu-LinH = LinH then co-NLIN $\subseteq$ Eu-LinH. Thus, by Lemma 1, we have co-NP $\subseteq$ NP. Hence NP = co-NP = PH.

For (c) let $\mathcal{C}$ be either SC or NL. In either case, $\mathcal{C}$ is closed under complement and closed under log space (and hence, polynomial length padding) reductions [9, 15]. So if $\mathcal{C}$ = Eu-LinH then in particular NLIN $\subseteq \mathcal{C}$ and by Lemma 1 we get NP $\subseteq \mathcal{C}$. Since in both cases, we know $\mathcal{C} \subseteq$ NP, we get $\mathcal{C}$ = NP. So by closure under complement of $\mathcal{C}$ = NP = co-NP = PH. As we are assuming $\mathcal{C}$ = Eu-LinH $\subseteq$ LinH $\subseteq$ PH, we also have $\mathcal{C}$ = Eu-LinH = LinH = PH. □

The argument for part (c) thus shows:

Corollary 5 *If* SC = NP *or* NL = NP *then* Eu-LinH = LinH = NP = PH.

If the conclusion were to happen then by Proposition 4 and Theorem 6, both LinH and Eu-LinH must not collapse.

Corollary 6 *For $k \geq 1$, if* Eu-LinH $\subseteq \Sigma_k$-TIME$[n]$ *then* NL $\neq$ NP *and* SC $\neq$ NP.

Proof. By the previous corollary, SC = NP or NL = NP implies Eu-LinH = LinH = PH. By Proposition 4 we then have LinH $\neq \Sigma_k$-TIME$[n]$ $\forall k \geq 1$. Since Σ_k-TIME$[n] \subseteq$ LinH by definition, we have LinH $\not\subseteq \Sigma_k$-TIME$[n]$, and thus Eu-LinH $\not\subseteq \Sigma_k$-TIME$[n]$. □

Intuitively, Eu-LinH = LinH comes very close to showing that LinH collapses. We already know by Theorem 7 (b) that in itself this condition implies NP = PH. So if Eu-LinH = LinH does imply LinH collapses, then together with the consequence NP = PH, this would force us to conclude the premise of Corollary 5 is false. i.e., SC $\neq$ NP and NL $\neq$ NP. In an attempt to give at least some justification for this intuition that Eu-LinH is a small subclass of LinH, one might consider a slightly weaker version of Eu-LinH with smaller sharply bounded universal quantifiers and see if it can equal NP or not. Let $s \in O(\log n)$. Extending our notation slightly we will write u^s in a quantifier string to mean the same as u except where the bound on the quantifier is $s(n)$ rather $k \log n$. Consider the class $\mathcal{C}_{m,k} := (\mathsf{Eu}^{s_m} \cdots \mathsf{Eu}^{s_1})^{\mathsf{lin}}$ where $(\sum_{j=1}^{m} s_j(n)) \leq k \log n$. Let Eu-LinH$_k := \cup_m \mathcal{C}_{m,k}$. So Eu-LinH$_k$ is analogous to Eu-LinH, except that we have forced the sharply bounded quantifiers to be slightly smaller. As an example, if each $s_1, \ldots s_m$ were $\log \log n$, then for each $m \geq 1$, $(\mathsf{Eu}^{s_m} \cdots \mathsf{Eu}^{s_1})^{\mathsf{lin}}$ would be contained in Eu-LinH$_1$. The next result shows if we could tighten Nepomnjaščiĭ's result to show SC and NL are in Eu-LinH$_k$ for any $k \geq 1$ then SC $\neq$ NP and NL $\neq$ NP.

Theorem 8 *For $k \geq 1$,* Eu-LinH$_k \subseteq$ NTIME$[n^{k+1}] \subsetneq$ NP.

Proof. Suppose L is in Eu-LinH$_k$ and hence in $\mathcal{C}_{m,k}$ for some m. Let $s_1(n), \ldots s_m(n)$ be the size bounds on the sharply bounded quantifiers. By the definition of Eu-LinH$_k$, we have $(\sum_{j=1}^{m} s_j(n)) \leq k \log n$. If one examines the proof of Proposition 1, the run time of the NP machine given there to simulate L would be

$$O(n + 2^{s_m}(n + 2^{s_{m-1}}(\cdots(n + 2^{s_1} \cdot n)))),$$

the n's coming from guessing the existential quantifiers, the 2^{s_i}'s coming from looping over all the choices of strings of length s_i. This run time can be rewritten as

$$O(n(1 + 2^{s_m} + 2^{s_m+s_{m-1}} + \cdots + 2^{(\sum_{j=1}^{m} s_j(n))})).$$

which can be bounded by $O(n \cdot m2^{k \log n}) = O(n^{k+1})$, the latter equality following as m is constant. Thus, the result follows. □

8 Reducing the Space Requirements for Eu-LinH

With the goal of reducing the space requirements needed for our upper bound on Eu-LinH, we define a new quantifier, and thus a new hierarchy. Given a class $\mathcal{C}$, define $\xi\mathcal{C}$ to be the class of languages of the form $\{x \mid \exists w \in \{0,1\}^{|x|^{d_1}} \forall i < |x|^{d_2}, \langle x, i, (w)_{f_1(i)}, (w)_{f_2(i)}\rangle \in L\}$ for some $L \in \mathcal{C}$, some constants d_1 and d_2 (where $d_1 \geq d_2$), and some $f_1, f_2 : [0, |x|^{d_2} - 1] \to [0, |x|^{d_2} - 1]$. Note that $(w)_i$ denotes the ith block of w when w is split into $|x|^{d_2}$ blocks, each of size $|x|^{d_1-d_2}$, and the functions f_1 and f_2 are used to select blocks of w. With this new quantifier ξ, we can define $(\xi)_k^{\mathsf{lin}}$ and ξ-LinH in exactly the same way that we defined $(\mathsf{Eu})_k^{\mathsf{lin}}$ and Eu-LinH. Notice that, like in $(\mathsf{Eu})_k^{\mathsf{lin}}$, the universal quantifiers of $(\xi)_k^{\mathsf{lin}}$ are of logarithmic size, though the existential quantifiers are of polynomial rather than linear size.

We first show the relationship between the two hierarchies.

Theorem 9 *For $k \geq 0$, $(\mathsf{Eu})_k^{\mathsf{lin}} \subseteq (\xi)_k^{\mathsf{lin}}$.*

Proof. The proof is by induction. When $k = 0$, we have $(\mathsf{Eu})_k^{\mathsf{lin}} = \mathsf{DLIN} = (\xi)_k^{\mathsf{lin}}$, so the theorem holds. Now assume the theorem holds up to some $k - 1$. Let L be a language in $(\mathsf{Eu})_k^{\mathsf{lin}}$. Then there is an $L' \in (\mathsf{Eu})_{k-1}^{\mathsf{lin}} \subseteq (\xi)_{k-1}^{\mathsf{lin}}$ such that $L = \{x \mid \exists y_1 \in \{0,1\}^{c_1|x|} \forall y_2 \in \{0,1\}^{\log(c_2|x|)}, \langle x, y_1, y_2\rangle \in L'\}$ for some constants c_1, c_2. We shall define a $(\xi)_k^{\mathsf{lin}}$-machine M that decides L.

On input x, for M's outermost ξ quantifier, the guessed w has length $c_1c_2|x|^2$ (which is within the polynomial bound), i ranges from 0 to $c_2|x| - 1$ (also within the polynomial bound), and we have $f_1(i) = i$, $f_2(i) = \min(i + 1, c_2|x| - 1)$. ($f_1$ and f_2 were chose in this way to pick consecutive blocks from w, but to pick the same block when i is at its maximum value.) Note that the set of all is, when written in binary, is exactly $\{0,1\}^{\log(c_2|x|)}$. Each block of w is of length $c_1|x|$, and in fact w is meant to represent $c_2|x|$ concatenations of the string guessed for y_1 in the definition of L.

Given the above definitions, M operates on strings of the form $\langle x, i, (w)_i, (w)_{i+1}\rangle$, where we understand $(w)_{i+1}$ to mean $(w)_i$ when i is at its maximum value. M's first step is to verify that $(w)_i = (w)_{i+1}$; this is done to ensure that w is indeed $c_2|x|$ concatenations of the same string. This check can be done in linear time. Assuming that that check passes, M then "renames" its inputs by saying $y_1 = (w)_i$ and $y_2 = i$ (in binary), and checks if $\langle x, y_1, y_2\rangle \in L'$ by using the $(\xi)_{k-1}^{\mathsf{lin}}$-machine for L'.

To verify M's correctness, we need only consider the branches where the guessed w is $c_2|x|$ concatenations of the same string, since we know M will reject on all existential branches where this is not the case. If $x \in L$, then there is some fixed y_1 such that

$\langle x, y_1, y_2\rangle \in L'$ for each $y_2 \in \{0,1\}^{\log(c_2|x|)}$, and so M will accept x when the outer ξ chooses the w which consists of repeated y_1s. Conversely, if $x \notin L$, then for all w which consist of concatenations of the same y_1, there is some $y_2 \in \{0,1\}^{\log(c_2|x|)}$ such that $\langle x, y_1, y_2\rangle \notin L'$, and so M will reject x. $\square$

Corollary 7 $\mathsf{Eu\text{-}LinH} \subseteq \xi\text{-}\mathsf{LinH}$.

Theorem 9 also gives us an analogous result to Nepomnjaščiĭ's Theorem for ξ-LinH.

Corollary 8 *For $k \geq 1$ and $0 \leq \epsilon < 1$, $\mathsf{NTISP}(n^{k(1-\epsilon)}, n^{\epsilon})$ is contained in $(\xi)_k^{\mathsf{lin}}$.*

In order to prove an upper bound on this new hierarchy, we now show that the ξ quantifiers are collapsable.

Theorem 10 *For any class $\mathcal{C}$, $\xi\xi\mathcal{C} \subseteq \xi\mathcal{C}$.*

Proof. First, let us be precise about the definition of $\xi\xi\mathcal{C}$. A machine M which decides a language $L \in \xi\xi\mathcal{C}$ computes the following on input x:

$$\exists w \in \{0,1\}^{|x|^{d_1}}\, \forall i < |x|^{d_2}\, \exists w' \in \{0,1\}^{|x|^{d_1'}}\, \forall i' < |x|^{d_2'},$$

$$R(x, i, (w)_{f_1(i)}, (w)_{f_2(i)}, i', (w')_{f_1'(i')}, (w')_{f_2'(i')})$$

where R is a predicate computable within the bounds of the class $\mathcal{C}$. In order to compress the two ξs into one, we define the string v to be $\langle w_0', w_1', \ldots, w'_{|x|^{d_2}-1}\rangle$, where w_n' is the string guessed by the inner ξ when the i (from the outer ξ) is equal to n. Then, we can rewrite the quantifier string $\exists w \forall i \exists w' \forall i'$ as $\exists w \exists v \forall i \forall i'$. Now, we use the following notation for our new "compressed" ξ:

$$\exists y \in \{0,1\}^{|x|^{e_1}}\, \forall j < |x|^{e_2}, S(x, j, (y)_{g_1(j)}, (y)_{g_2(j)})$$

To show that this is the same, we will define the new existential string y to contain all of w and v, though y will not be a simple concatenation of these two strings. As we will see shortly, the size of y is $2(|x|^{d_2+d_2'})(|x|^{d_1-d_2} + |x|^{d_1'-d_2'})$, and so we choose e_1 so that the bound on y, $|x|^{e_1}$, is bigger than this quantity. Each block of y contains one block of w and one block of *one* of the w_k's, so each block has size $|x|^{d_1-d_2} + |x|^{d_1'-d_2'}$. And thus, we choose e_2 such that $|x|^{e_2} \geq 2(|x|^{d_2+d_2'})$.

For the string y, we define it as follows. The first $2|x|^{d_2'}$ blocks contain the necessary information when $i = 0$ and i' ranges from 0 to $|x|^{d_2'} - 1$. The next $2|x|^{d_2'}$ blocks contain the necessary information when $i = 1$ and i' again ranges from 0 to $|x|^{d_2'} - 1$. (These groups contain $2|x|^{d_2'}$ blocks rather than just $|x|^{d_2'}$ blocks because we need two blocks from w and two from one of the w_k's, but each block of y contains only half this information.) More generally, if we let α range from 0 to $|x|^{d_2} - 1$ (the range of i) and we let β be the even numbers in the range 0 to $|x|^{d_2'} - 1$ (the range of i'), then the $(2\alpha|x|^{d_2'} + \beta)$th block of y is $\langle (w)_{f_1(\alpha)}, (w_\alpha')_{f_1'(\beta)}\rangle$, and the $(2\alpha|x|^{d_2'} + \beta + 1)$th block of y is $\langle (w)_{f_2(\alpha)}, (w_\alpha')_{f_2'(\beta)}\rangle$.

Due to our selection of e_2 above, we can treat each value of the index j as being of the form $\langle i, i'\rangle$, ignoring any extra bits that result from the fact that $\log(\max(j)) \geq$

$\log(\max(ii'))$. Then we define $g_1(j) = g_1(\langle i, i' \rangle) = 2i|x|^{d'_2} + 2i'$, and we define $g_2(j) = g_1(j) + 1$. And finally, by viewing the index j and the blocks of y as we have defined them above, the predicate $S(x, j, (y)_{g_1(j)}, (y)_{g_2(j)})$ is equal to $R(x, i, (w)_{f_1(i)}, (w)_{f_2(i)}, i', (w'_i)_{f'_1(i')}, (w'_i)_{f'_2(i')})$. □

Corollary 9 *For all $k \geq 0$, $(\xi)_k^{\mathsf{lin}} = (\xi)_1^{\mathsf{lin}} = \xi\mathsf{DLIN} \subseteq \mathsf{NP}$.*

Proof. The first equality follows from Theorem 10, and the second equality from the definition of $(\xi)_1^{\mathsf{lin}}$. The inclusion $\xi\mathsf{DLIN} \subseteq \mathsf{NP}$ follows from the fact that an NP machine could existentially guess the string w, and then deterministically try each of the polynomially-many values of i to decide a language in $\xi\mathsf{DLIN}$. □

We now use the following result from [5], which was largely based on work originally published in [13].

Theorem 11 $\mathsf{DLIN} \subseteq \Sigma_2\text{-}\mathsf{TISP}[n, \frac{n}{\log^* n}]$.

The result actually proved was that $\mathsf{DTIME}[t(n)] \subseteq \Sigma_2[\frac{t(n)}{\log^* n}]$ for all time-constructible $t(n) \geq n \log^* n$, but as the lower bound on $t(n)$ is not relevant to the space bound on the simulating machine in the proof, the same proof can be used to show Theorem 11.

Corollary 10 $\mathsf{Eu\text{-}LinH} \subseteq \xi\Sigma_2\text{-}\mathsf{TISP}[n, \frac{n}{\log^* n}]$.

Proof. From Theorem 11, we have that $(\xi)_1^{\mathsf{lin}} \subseteq \xi\Sigma_2\text{-}\mathsf{TISP}[n, \frac{n}{\log^* n}]$. From Corollary 9, we then have that $(\xi)_k^{\mathsf{lin}} \subseteq \xi\Sigma_2\text{-}\mathsf{TISP}[n, \frac{n}{\log^* n}]$ for all $k \geq 0$. Finally, since every language $L \in \mathsf{Eu\text{-}LinH}$ must be in some precise level of Eu-LinH, and thus in some precise level of ξ-LinH, the result follows. □

9 Conclusion

We believe we have shown that the hierarchy Eu-LinH is an interesting hierarchy closely connected with the NL = NP question. This is evidenced both by the containments

$$\mathsf{NTISP}[n^{k(1-\epsilon)}, n^{\epsilon})] \subseteq (\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTISP}[n^{k+1}, n]$$

and by the fact that if our hierarchy collapses then NL ≠ NP. We would like briefly to conclude with some future directions for research. One immediate question is whether the containments $\mathsf{NTISP}[n^{k(1-\epsilon)}, n^{\epsilon}] \subseteq (\mathsf{Eu})_k^{\mathsf{lin}} \subseteq \mathsf{NTISP}[n^{k+1}, n]$ can be made tighter? We have given some evidence for this in the preceding section, but is an exact relationship between our hierarchy and a nondeterministic time space class possible? It would also be interesting to study what happens when one considers the hierarchies like Eu-LinH where one has nonconstantly many alternations. Given that the total size of the string that would be fed into the machine in this case could be nonlinear in the input size, it would probably be worthwhile to consider quasi-linear variants of our class as well.

References

[1] S. Bloch, J. Buss and J. Goldsmith. Sharply bounded alternation and quasilinear time. Theory of Computing Systems, 31(2):187-214, March 1998.

[2] A. K. Chandra and L. J. Stockmeyer Alternation. IEEE Symposium on Foundations of Computer Science. pp98–106. 1976.

[3] L. Fortnow. Time-space tradeoffs for satisfiability. Journal of Computer and System Sciences, 60(2):337-353, April 2000.

[4] L. Fortnow and R. Lipton and D. van Melkebeek and A. Viglas. Time-space lower bounds for satisfiability. Journal of the ACM, 52(6):835-865, November 2005.

[5] S. Gupta. Alternating Time Versus Deterministic Time: A Separation. *Mathematical Systems Theory*. pp. 661-672. 1996.

[6] Jim Kadin. The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses. SIAM Journal on Computing, 17(6): 1263-1282 (1988).

[7] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. Transactions of the American Mathematical Society, 117:285–306, 1965.

[8] P. Hájek and P. Pudlák. *Metamathematics of First-Order Arithmetics*. Springer-Verlag, 1993.

[9] N. Immerman. Nondeterministic Space is Closed Under Complement. SIAM Journal on Computing, 17, pp. 935–938,1988.

[10] D. S. Johnson. A Catalog of Complexity Classes. In *The Handbook of Theoretical Computer Science*, Volume A. J. Van Leeuwen, Ed. pages 68–161. MIT Press. 1990.

[11] R. Lipton and A. Viglas On the Complexity of SAT. IEEE Symposium on Foundations of Computer Science. pp459–464. 1999.

[12] V.A. Nepomnjaščiĭ. Rudimentary predicates and Turing computations. *Dokl. Acad. Nauk*, Vol. 195, pages 282–284, 1970, transl. Vol. 11 pp1462–1465, 1970.

[13] W. J. Paul, N. Pippenger, E. Szemeredi and W. T. Trotter. On determinism versus nondeterminism and related problems. In *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science*. pp. 429-438. 1983.

[14] J. Seiferas and M. Fischer and A. Meyer. Separating nondeterministic time complexity classes. Journal of the ACM, 25:146167, 1978.

[15] R. Szelepcsenyi. The method of forcing for nondeterministic automata. Bulletin EATCS 33, pp. 96–100, 1987.

[16] Ryan Williams: Better Time-Space Lower Bounds for SAT and Related Problems. IEEE Conference on Computational Complexity (CCC 2005). pp.40–49. 2005.

[17] S. Žàk. A Turing machine time hierarchy. Theoretical Computer Science, 26:327333, 1983.