Electronic Colloquium on Computational Complexity, Report No. 27 (2008)

Generalizations of the Hartmanis–Immerman–Sewelson Theorem and Applications to Infinite Subsets of P-Selective Sets

Till Tantau
Institut für Theoretische Informatik
Universität zu Lübeck
D-23538 Lübeck, Germany

December 4, 2007

Abstract

The Hartmanis–Immerman–Sewelson theorem is the classical link between the exponential and the polynomial time realm. It states that $\mathrm{NE} = \mathrm{E}$ if, and only if, every sparse set in NP lies in P. We establish similar links for classes other than sparse sets:

1. $\mathrm{E} = \mathrm{UE} \iff$ all functions $f\colon \{1\}^* \to \Sigma^*$ in $\mathrm{NPSV_g}$ lie in FP.
2. $\mathrm{E} = \mathrm{NE} \iff$ all functions $f\colon \{1\}^* \to \Sigma^*$ in NPFewV lie in FP.
3. $\mathrm{E} = \mathrm{E}^{\mathrm{NP}} \iff$ all functions $f\colon \{1\}^* \to \Sigma^*$ in OptP lie in FP.
4. $\mathrm{E} = \mathrm{E}^{\mathrm{NP}} \iff$ all standard left cuts in NP lie in P.
5. $\mathrm{E} = \mathrm{EH} \iff \mathrm{PH} \cap \mathrm{P/poly} = \mathrm{P}$.

We apply these results to the immunity of P-selective sets. It is known that they can be bi-immune, but not $\Pi_2^{\mathrm{p}}/1$-immune. Their immunity is closely related to top-Toda languages, whose complexity we link to the exponential realm, and also to king languages. We introduce the new notion of *superkings*, which are characterized in terms of $\exists\forall$-predicates rather than $\forall\exists$-predicates, and show that king languages cannot be Σ_2^{p}-immune. As a consequence, P-selective sets cannot be $\Sigma_2^{\mathrm{p}}/1$-immune and, if $\mathrm{E}^{\Sigma_2^{\mathrm{p}}} = \mathrm{E}$, not even P/1-immune.

1 Introduction

The exponential time realm and the polynomial time realm are related by a line of implications: $\mathrm{P} = \mathrm{NP} \implies \mathrm{E} = \mathrm{EH} \implies \mathrm{E} = \mathrm{NE}$, where E and NE denote the deterministic and nondeterministic classes of languages decidable in time $2^{O(n)}$, respectively, and $\mathrm{EH} = \mathrm{E}^{\mathrm{PH}}$ denotes the exponential time hierarchy. It is is not known whether any of the reverse implications hold. In particular, it is not known whether $\mathrm{E} = \mathrm{NE}$ forces all languages in NP to lie in P. The Hartmanis–Immerman–Sewelson theorem [8] tells us that the link between exponential time and polynomial time seems to be of a different nature: $\mathrm{E} = \mathrm{NE}$ holds if, and only if, all sparse sets in NP lie in P.

The two implications directions of the Hartmanis–Immerman–Sewelson theorem are known as an *upward collapse* (a collapse down in the polynomial realm forcing a collapse up in the exponential realm) and a *downward collapse* (a collapse up in the exponential time realm forcing a collapse down in the polynomial realm). Upward collapses, like the implication $\mathrm{P} = \mathrm{NP} \implies \mathrm{E} = \mathrm{NE}$, are typically proved by padding arguments. Downward collapses are more rare and generally harder to prove, celebrated results include the Hartmanis–Immerman–Sewelson theorem and the implication $\mathrm{E} = \mathrm{EH} \implies \mathrm{P} = \mathrm{BPP}$ proved in [2]. There are also downward collapses *inside* the polynomial time realm, see [9] for a starting point to the literature on this. In the present paper we explore two kinds of generalizations of the Hartmanis–Immerman–Sewelson theorem.

ISSN 1433-8092

Generalizations of the Hartmanis–Immerman–Sewelson Theorem. The first kind of results concern links between collapses in the exponential realm and polynomial-time *function* classes rather then *language* classes. The function classes that will be used are Krentel's class OptP [23] and the nondeterministic function classes NPSV of Book, Long, and Selman [4] and NPFewV, thus named by Hemaspaandra and Ogihara [16] and also known as FewPF [31]. When we say that a multi-valued function $f \in \mathrm{NPFewV}$ lies in FP, we mean that there is a polynomial-time machine that on input of a word x will output exactly all values of $f(x)$. A *tally function* is a function of the form $f \colon \{1\}^* \to \Sigma^*$. We prove the following results, each of which contains a downward and an upward collapse:

1. $\mathrm{E} = \mathrm{UE} \iff$ all tally functions in $\mathrm{NPSV_g}$ lie in FP.
2. $\mathrm{E} = \mathrm{NE} \iff$ all tally functions in NPFewV lie in FP.
3. $\mathrm{E} = \mathrm{E}^{\mathrm{NP}} \iff$ all tally functions in OptP lie in FP.

The classical Hartmanis–Immerman–Sewelson theorem is a special case of the second claim: Given a sparse set in NP, an NPFewV-machine can for each input 1^n guess a word of length n, verify that the word is in the language, and, if so, output the word. Then the set of outputs of this machine on input 1^n is exactly the set of words in the language of length n. By the above characterization, $\mathrm{E} = \mathrm{NE}$ implies that this mapping is in FP and, thus, the language is in P (and even P-printable).

The second kind of results concern links between collapses in the exponential realm and languages classes other than the sparse sets. It is well known that we can restrict ourselves to tally sets: $\mathrm{E} = \mathrm{NE}$ also holds if, and only if, all tally sets in NP lie in P. Our first new result concerns *standard left cuts*, which have the property that together with any word they also contain all lexicographically smaller words. We can now ask, what consequences does it have that all standard left cuts in NP lie in P? We show the following:

4. $\mathrm{E}^{\mathrm{NP}} = \mathrm{E} \iff$ every standard left cut in NP lies in P.

An interesting aspect of this result is that it is "harder" to store information in standard left cuts than in sparse sets: a standard left cut contains just one relevant bit of information for each word length relative to the previous word length.

Next, we study the class P-sel of P-*selective sets*. They were introduced by Selman [30] as the polynomial-time analogues of the semirecursive sets [18, 19] and they generalize polynomial-time decidable sets. For their definition *selectors* are used, which – for the purposes of this paper – are commutative, polynomial-time computable functions f taking two arguments x and y such that $f(x, y) \in \{x, y\}$ holds for all x and y (the selector function selects one of its inputs). A set A is called P-*selective* if there exists a selector f for A, which means that $x \in A \vee y \in A$ implies $f(x, y) \in A$. This is sometimes informally described as "f chooses the more likely element," but note that no randomness is involved. An example of P-selective sets are standard left cuts since a selector can always choose the lexicographically smaller word. A closely related concept are the slightly more general *lengthwise* P-selective sets. For them, the selector only needs to choose the more likely element when x and y are of the same length.

The P-selective sets are well-studied and many of their properties are well understood, see for instance the text book by Hemaspaandra and Torenvliet [17] and the survey [26]. On the one hand, it is unlikely that NP-complete sets are P-selective, as was already noted by Selman [30], since this would imply $\mathrm{P} = \mathrm{NP}$. This was later extend by three research groups [1, 3, 28] who showed independently that NP-complete sets cannot be sublinear truth-table reducible to a P-selective set, unless $\mathrm{P} = \mathrm{NP}$. On the other hand, every standard left cut is P-selective and, thus, there are uncountably many P-selective sets – which in turn implies that there are non-recursive P-selective sets.

We prove a downward collapse for P-selective sets, but no matching upward collapse:

5. $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E} \implies \mathrm{NP} \cap \text{P-sel} = \mathrm{P}$.

Since every standard left cut is in P-sel, we know at least that $\mathrm{NP} \cap \text{P-sel} = \mathrm{P}$ implies $\mathrm{E}^{\mathrm{NP}} = \mathrm{E}$.

Finally, we study the class P/poly. It has many characterization: by definition, it is the class of languages decidable in polynomial time with polynomial advice; it is the class of languages decidable by non-uniform polynomially-sized circuits; and it is the Turing-reduction closure of each of the classes of tally sets, sparse sets, standard left cuts, and of P-selective sets. We prove the following downward collapse:

6. $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E} \implies \mathrm{NP} \cap \mathrm{coNP} \cap \mathrm{P/poly} = \mathrm{P}$.

This result relativizes in a nontrivial way: Given any two oracles X and Y, we show that $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}^X}} \subseteq \mathrm{E}^Y \implies \mathrm{NP}^X \cap \mathrm{coNP}^X \cap \mathrm{P/poly} \subseteq \mathrm{P}^Y$. Since $\mathrm{ZPP} \subseteq \mathrm{NP} \cap \mathrm{coNP} \cap \mathrm{P/poly}$ and $\mathrm{BPP} \subseteq \Sigma_2^{\mathrm{p}} \cap \Pi_2^{\mathrm{p}} \cap \mathrm{P/poly}$, this relativized version has the following consequences:

7. $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E} \implies \mathrm{ZPP} = \mathrm{P}$.
8. $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}} = \mathrm{E} \implies \mathrm{BPP} = \mathrm{P}$.
9. $\mathrm{EH} = \mathrm{E} \iff \mathrm{PH} \cap \mathrm{P/poly} = \mathrm{P}$

Previously it was already known that $\mathrm{EH} = \mathrm{E} \implies \mathrm{BPP} = \mathrm{P}$, as shown by Babai et al. [2].

Infinite Subsets of P-Selective Sets. As we just saw, the equation $\mathrm{NP} \cap \text{P-sel} = \mathrm{P}$ is intimately linked to collapses in the exponential time hierarchy. This is not only true for the P-selective sets themselves, but also for important infinite subsets of P-selective sets. In the second part of the present paper we explore these links.

One approach to better understanding a difficult problem is to study whether it has at least an efficient *special case algorithm*. Consider for instance the computationally difficult satisfiability problem. Even though we do not know how to decide the satisfiability of an arbitrary formula efficiently, for an infinite number of special cases we can do so quite easily. This can be formalized by observing that the language SAT has an infinite subset in P (for instance, the set of all formulas consisting of a single variable). Many other difficult problems, including even the halting problem, exhibit this property of having an efficiently decidable infinite subset. Languages that do not have this property are called *immune*. More generally, a language is C-immune if it is infinite and does not have an infinite subset that is an element of C. An even stronger form of immunity is bi-immunity, which means that both the language and its complement are immune.

The question of whether there are immune P-selective sets and, if so, "how" immune these sets can become, has been studied independently by different authors. Recently, Hemaspaandra and Torenvliet [13] have shown that there exist EXP-immune P-selective sets, and they in fact show that exponential time can be replaced by any recursive time bound. It turns out that a much stronger result holds: In his PhD thesis, Nickelsen [25] mentions in passing that there are REC-bi-immune (also known just as bi-immune) P-selective sets – a considerably stronger result than the more recent one presented in [13]. Nickelsen's main observation (and it takes a keen eye to see this) was that the work of Goldsmith, Joseph, and Young [7], who study so-called P-*cheatable sets*, implicitly also applies to P-selective sets.

These results bash all hopes of finding efficiently decidable infinite subsets of arbitrary P-selective sets. However, there are loop-holes: Hemaspaandra and Torenvliet [13] have shown that P-sel is not immune against $\Pi_2^{\mathrm{p}}/1$, that is, it is not immune against a nonuniform class with very limited non-uniformity. In a different paper [12] it was shown that P-sel is also not immune against weakly $\mathrm{P}^{\Sigma_2^{\mathrm{p}}}$-rankable sets.

In Section 4.3 of the present paper we prove the following new nonimmunity results for P-selective sets:

1. P-sel is not immune against $\Sigma_2^{\mathrm{p}}/1$.
2. If $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E}$, then P-sel is not immune against P/1.

We can also ask whether the reverse implication holds, that is, if P-sel is not P/1-immune, does this imply at least, say, NE = E? While we do not know whether this is the case, we can at least show a link between the exponential hierarchy and the question of whether P-sel has *finder functions* in FP. A *finder* for a set A is a function $g\colon \{1\}^* \to \Sigma^*$ such that for all $n \geq 0$ we have $A \cap \Sigma^n \neq \emptyset \implies g(1^n) \in A \cap \Sigma^n$; in other words, a finder gets a word length coded in unary as input and outputs a word in the language having this particular length, if possible.

3. If $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E}$, then every lengthwise P-selective set has a finder in FP.
4. If every lengthwise P-selective set has a finder in FP, then NE = E.

The proofs of all known non-immunity results of P-selective sets are based on the study of the properties of certain natural infinite subsets of P-selective sets, namely of the *top-Toda equivalence classes* [12] and of the *king languages* [12, 11] induced by the selector.

For the class TOP-TODA of all top-Toda languages we establish nearly matching upward and downward collapses:

5. $\mathrm{UE}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E} \implies \text{TOP-TODA} \subseteq \mathrm{P}$
6. $\text{TOP-TODA} \subseteq \mathrm{P} \implies \mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E}$

Concerning the class KINGS of king languages, we prove the following new results:

7. KINGS is not immune against Σ_2^{p}.
8. Every set in KINGS has a finder in $\mathrm{FP}^{\mathrm{NP}^{\mathrm{NP}}}$.
9. If $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}}} = \mathrm{E}$, then every set in KINGS has a finder in FP.
10. If every set in KINGS has a finder in FP, then NE = E.

Organization of this paper. In following preliminaries section we fix the terminology and notation and provide detailed definitions of the key concepts. In Section 3 we present the generalizations of the Hartmanis–Immerman–Sewelson theorem. In Section 4 we study top-Toda languages and king languages and apply our findings to P-selectivity. In the conclusion we list open problems.

2 Preliminaries

We use the alphabet $\Sigma = \{0,1\}$. A *word* is, thus, the same as a bit string. We write $|x|$ for the length of the word x. The *lexicographical ordering* $<_{\mathrm{lex}}$ on words is the same as the telephone book ordering, so $0 <_{\mathrm{lex}} 00 <_{\mathrm{lex}} 01 <_{\mathrm{lex}} 1 <_{\mathrm{lex}} 10$. For a set $A \subseteq \Sigma^*$, let $\chi_A\colon \Sigma^* \to \{0,1\}$ denote the characteristic function. We extend χ_A to tuples by setting $\chi_A(x_1,\dots,x_n) = \chi_A(x_1)\dots\chi_A(x_n)$.

Given two words $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$, where x_i and y_j are the individual bits of x and y, we define the pairing $\langle x, y\rangle \in \Sigma^*$ to be the bitstring $0x_1 0x_2 \dots 0x_n 1y_1 1y_2 \dots 1y_m$. Then $|\langle x,y\rangle| = |\langle y,x\rangle| = 2(|x|+|y|)$. We extend this to tuples by $\langle x_1,\dots,x_n\rangle = \langle x_1, \langle x_2, \dots\rangle\rangle$. For a finite set $S \subseteq \Sigma^*$, we define an encoding $\langle S\rangle \in \Sigma^*$ as follows: We encode the empty set as the empty word. If S contains at least one element, let x be the lexicographically smallest one. Then $\langle S\rangle = \langle x, \langle S - \{x\}\rangle\rangle$.

A language $L \subseteq \Sigma^*$ is *sparse* if there exists a polynomial p such that for each word length n there are at most $p(n)$ words of length n in L. A language L is a *tally set* (also known as a *unary language*) if $L \subseteq \{1\}^*$. We call a language *populated* if for each $n \geq 0$ it contains at least one word of length n. Top-Toda and king languages are examples of populated languages.

For a number $n \geq 0$ let $\text{bin}(n) \in \Sigma^*$ denote its binary representation and for $n \geq 1$ let $\text{bin}'(n)$ denote the binary representation of n without the leading 1. Clearly, bin' is a bijection between the positive integers and the set $\{0,1\}^*$ and $|\text{bin}'(n)| = \lceil \log_2 n \rceil$.

A *directed graph* is a pair (V, E) with $E \subseteq V \times V$. A *path of length ℓ* in a graph is a sequence $(v_0, \dots, v_\ell)$ of ℓ distinct vertices such that $(v_{i-1}, v_i) \in E$ holds for all $i \in \{1, \dots, \ell\}$. A *tournament* is a directed graph such that for every pair $(u, v) \in V^2$ the set $\{(u, v), (v, u)\} \cap V$ has size 1, that is, there is exactly one edge between u and v. This definition enforces that there are self-loops at every vertex of a tournament (in the literature it is often required that there are *no* self-loops in a tournament, but this just a matter of taste).

Given a graph $G = (V, E)$ and a subset $U \subseteq V$, let $\text{dom}(U) = U \cup \{v \in V \mid \exists u \in U\colon (u, v) \in E\}$ be the set of all vertices that are *dominated* by U. A *dominating set* of graph G is a set U with $\text{dom}(U) = V$. The opposite of domination is anti-domination. Let $\text{anti-dom}(U) = U \cup \{v \in V \mid \exists u \in U\colon (v, u) \in E\}$ and an *anti-dominating set* is a set U with $\text{anti-dom}(U) = V$. It is well known that every tournament has a dominating set of size $\log_2(n+1)$, but see Lemma 4.15 for an even stronger claim.

Complexity Classes. We use the standard definitions of the complexity classes P, NP, coNP, and UP. We write C^X to denote the class C relativized in the standard manner to oracles from X. The classes of the polynomial hierarchy are defined in the standard way as $\Sigma_i^{\text{p}} = \text{NP}^{\Sigma_{i-1}^{\text{p}}}$ and $\Pi_i^{\text{p}} = \text{coNP}^{\Sigma_{i-1}^{\text{p}}}$ and $\Delta_i^{\text{p}} = \text{P}^{\Sigma_{i-1}^{\text{p}}}$. The *polynomial hierarchy* is the union $\text{PH} = \bigcup_i \Sigma_i^{\text{p}} = \bigcup_i \Pi_i^{\text{p}} = \bigcup_i \Delta_i^{\text{p}}$.

The exponential time classes E and NE denote "linear exponential time," that is, $\text{E} = \text{DTIME}[2^{O(n)}]$ and $\text{NE} = \text{NTIME}[2^{O(n)}]$. We will also use the class UE of unambiguous linear exponential time, which means that the nondeterministic machines that decide languages in this class may have at most one accepting path for each input. The classes of the *exponential hierarchy* are $\Sigma_i^{\text{e}} = \text{NE}^{\Sigma_{i-1}^{\text{p}}}$ and $\Pi_i^{\text{e}} = \text{coNE}^{\Sigma_{i-1}^{\text{p}}}$ and $\Delta_i^{\text{e}} = \text{E}^{\Sigma_{i-1}^{\text{p}}}$. Note that the oracles in the definition of the exponential hierarchy are the same as for the polynomial hierarchy; only the "bases" differ. The exponential hierarchy itself is the union over all these classes, that is, $\text{EH} = \text{NE}^{\text{PH}} = \text{coNE}^{\text{PH}} = \text{E}^{\text{PH}}$.

Advice classes, a concept due to Karp and Lipton [20], model the concept of precomputation. The idea is that for each word length we have access to a small amount of hard-to-compute *advice*. Formally, given a complexity class C and an advice length function $l\colon \mathbb{N} \to \mathbb{N}$, the class C/l contains the following languages A: There must exist a $B \in C$ and an *advice function* $f\colon \mathbb{N} \to \{0,1\}^*$ with $\forall n\colon |f(n)| = l(n)$, such that $x \in A$ iff $\langle x, f(|x|)\rangle \in B$.

Function Classes. A *tally function* is a function of the form $f\colon \{1\}^* \to \Sigma^*$. We use the notation FP for the class of all functions $f\colon \Sigma^* \to \Sigma^*$ that are computable in polynomial time. For the definition of nondeterministic function classes, we follow the lines of Selman [31]. A *multi-valued function* f is a "function" that can map a single value to more than one value and also, possibly, to no value at all. Formally, a multi-valued function f is a relation on two sets A and B and we write $f\colon A \to B$ to denote it. Let set-$f(x)$ denote the set of all values in B that f relates to $x \in A$. For a polynomially time-bounded nondeterministic Turing machine M and an input x, we write $\text{outputs}_M(x)$ for the set of all outputs produced by M on accepting computation paths. We say that *M computes the multi-valued function $f\colon \Sigma^* \to \Sigma^*$* if set-$f(x) = \text{outputs}_M(x)$ holds for all words x.

The class NPMV contains all multi-valued functions that can be computed (in the above sense) by polynomially time-bounded nondeterministic Turing machines. The class NPSV is the restriction to functions f with $|\text{set-}f(x)| \leq 1$ for all $x \in \Sigma^*$. The class NPFewV is the restriction to functions f with $|\text{set-}f(x)| \leq |x|^{O(1)}$. The class $\mathrm{NPSV_g}$ is the restriction of NPSV to functions f whose graph $\{\langle x, y\rangle \mid x \in \Sigma^*, y \in \text{set-}f(x)\}$ lies in P. The relativized versions of these classes are defined in the obvious manner.

In order to be able to compare classes like NPFewV to the class FP, let us extend the definition of FP so that it also contains all multi-valued functions f for which a polynomially time-bounded Turing machine can, on any input x, output the code $\langle\text{set-}f(x)\rangle$ of $\text{set-}f(x)$.

Optimization classes can be defined in different ways, we use the more theory-oriented formalism of Krentel [23] in the present paper. For a polynomially time-bounded nondeterministic Turing machine M, let $\text{max-output}_M(x)$ denote lexicographically maximal element of $\text{outputs}_M(x)$. If M has not accepting computation path, let $\text{max-output}_M(x)$ be undefined. The class MaxP contains all functions $f\colon \Sigma^* \to \Sigma^*$ for which there is a polynomially time-bounded nondeterministic Turing machine M with $\forall x \in \Sigma^*\colon f(x) = \text{max-output}_M(x)$. One can define the class MinP in the same way and define OptP = MinP ∪ MaxP. All results proved in the following hold regardless of whether one uses OptP, MinP, or MaxP, and we use MaxP for simplicity. The definition relativizes readily, so let us write MaxP^X for the class that results when the machines M have access to the oracle X.

P-Selectivity. Before we define P-selective sets, let us first define *standard left cuts.* A word $w \in \{0,1\}^*$ induces a rational number r_w if we consider the bits of w as the fractional digits of the "number $0.w$." Formally, if $w = w_1 \dots w_{|w|}$ with $w_i \in \{0,1\}$, then $r_w = \sum_{i=1}^{|w|} w_i \cdot 2^{-i}$. For a real number r with $0 \leq r \leq 1$ the *standard left cut of* r is the set $\{w \in \Sigma^* \mid r_w \leq r\}$.

A function $f\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ is a *selector* if it has the following properties:

1. For all $x, y \in \Sigma^*$ we have $f(x,y) \in \{x, y\}$.
2. For all $x, y \in \Sigma^*$ we have $f(x,y) = f(y,x)$.
3. It is computable in polynomial time.

Strictly speaking, the above selectors should be called "commutative P-selectors" or even "commutative FP-selectors," but we will not use any other kind of selector in the present paper. The commutativity property does not change the power of selectors, see [15] for a detailed account of the algebraic properties of selectors.

A language A is P-*selective* if there exists a selector f such that for all words $x, y \in \Sigma^*$ the following implication holds: $x \in A \vee y \in A \implies f(x,y) \in A$. A language A is *lengthwise* P-*selective* if there exists a selector f such that for all words $x, y \in \Sigma^*$ with $|x| = |y|$ the same implication holds.

A selector f induces tournaments in the following manner. For a word length $n \geq 0$, the tournament T_f^n has the vertex set Σ^n and the edge set $\bigl\{\bigl(f(x,y), y\bigr) \mid x, y \in \Sigma^n\bigr\}$. This means that there is an edge from x to y in the tournament iff $f(x,y) = x$. A simple, but important observation is the following:

Lemma 2.1. *Let f witness that A is lengthwise* P*-selective. Let $n \geq 0$ and let p be a path in T_f^n. If the last vertex of the path lies in A, so do all vertices on the path.*

3 Generalizations of the Hartmanis–Immerman–Sewelson Theorem

3.1 Function Classes and Upward and Downward Collapses

In the present section we show how upward and downward collapses link the exponential hierarchy to the complexity of maximization problems and nondeterministic function classes. Recall that a tally function is a function of the form $f\colon \{1\}^* \to \Sigma^*$. Also recall that when we write "all tally functions in NPFewV lie in FP," then this means that all tally functions mapping an input to the coded set of outputs of an NPFewV-machine lie in FP.

Theorem 3.1. *Let X and Y be oracles. Then:*

1. $\mathrm{UE}^X \subseteq \mathrm{E}^Y$ *if, and only if, all tally functions in* $\mathrm{NPSV}_{\mathrm{g}}^X$ *lie in* FP^Y.
2. $\mathrm{NE}^X \subseteq \mathrm{E}^Y$ *if, and only if, all tally functions in* NPFewV^X *lie in* FP^Y.
3. $\mathrm{E}^{\mathrm{NP}^X} \subseteq \mathrm{E}^Y$ *if, and only if, all tally functions in* MaxP^X *lie in* FP^Y.

For the proof some lemmas are needed. We start with the well-known padding lemma.

Lemma 3.2 (Padding Lemma). *Let $A \subseteq \Sigma^*$ and let $A' = \{1^n \mid \mathrm{bin}'(n) \in A\}$. Then we have $A \in \mathrm{E} \iff A' \in \mathrm{P}$ and $A \in \mathrm{UE} \iff A' \in \mathrm{UP}$ and $A \in \mathrm{NE} \iff A' \in \mathrm{NP}$ and $A \in \mathrm{E}^{\mathrm{NP}} \iff A' \in \mathrm{P}^{\mathrm{NP}}$.*

The lemma relativizes. The next lemma is a direct consequence of the padding lemma.

Lemma 3.3 (Upward and Downward Collapse Lemma). *Let X and Y be oracles.*

1. $\mathrm{UE}^X \subseteq \mathrm{E}^Y \iff \mathrm{UP}^X \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY}$.
2. $\mathrm{NE}^X \subseteq \mathrm{E}^Y \iff \mathrm{NP}^X \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY}$.
3. $\mathrm{E}^{\mathrm{NP}^X} \subseteq \mathrm{E}^Y \iff \mathrm{P}^{\mathrm{NP}^X} \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY}$.

The collapse lemma and the following lemma together imply the theorem.

Lemma 3.4. *Let X and Y be oracles.*

1. $\mathrm{UP}^X \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY} \iff$ *all tally functions in* $\mathrm{NPSV}_{\mathrm{g}}^X$ *lie in* FP^Y.
2. $\mathrm{NP}^X \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY} \iff$ *all tally functions in* NPFewV^X *lie in* FP^Y.
3. $\mathrm{P}^{\mathrm{NP}^X} \cap \mathrm{TALLY} \subseteq \mathrm{P}^Y \cap \mathrm{TALLY} \iff$ *all tally functions in* MaxP^X *lie in* FP^Y.

Proof of Lemma 3.4. We start with the first claim. For the implication from left to right, let a tally function $f \in \mathrm{NPSV}_{\mathrm{g}}^X$ be given. We define a *tally coding* $\mathrm{tc}(f)$ of f, which is a tally language that contains all information about f. In detail, $\mathrm{tc}(f)$ contains 1^n if, and only if, $\mathrm{bin}'(n) = \langle \mathrm{bin}(m), \mathrm{bin}(k), b\rangle$, where m, k are nonnegative integers and $b \in \{0,1\}$ is a bit, such that (a) $f(1^m)$ is defined and (b) $f(1^m)$ has length at most k and (c) the kth bit of $f(1^m)$ is b. We claim $\mathrm{tc}(f) \in \mathrm{UP}^X$. A UP^X-machine can, on input 1^n with $\mathrm{bin}'(n) = \langle \mathrm{bin}(m), \mathrm{bin}(k), b\rangle$, guess $f(1^m)$, verify that the guess is correct, and then accept if the kth bit is b. By assumption, this implies $\mathrm{tc}(f) \in \mathrm{P}^Y$, from which we can conclude $f \in \mathrm{FP}^Y$: On input 1^m an FP^Y-machine can check for all $k \in \{1, \dots, m\}$ and all $b \in \{0,1\}$ whether $1^{n(k,b)} \in \mathrm{tc}(f)$ holds, where $n(k,b)$ has the property $\mathrm{bin}'\big(n(k,b)\big) = \langle \mathrm{bin}(m), \mathrm{bin}(k), b\rangle$. Note that $n(k,b)$ is polynomial in m and, hence, the length of $1^{n(k,b)}$ is polynomial in the length m of the input word. For the implication from right to left, start with a tally set in UP^X. Then the function f for which set-$f(1^n)$ contains all accepting computations of the UP^X-machine (of which there can be at most one) lies in $\mathrm{NPSV}_{\mathrm{g}}^X$ and, hence, in FP^Y. This implies that the tally set lies in P^Y.

Let us next prove the last claim. For the implication from left to right, start with a tally function $f \in \mathrm{MaxP}^X$. We claim that $\mathrm{tc}(f) \in \mathrm{P}^{\mathrm{NP}^X}$: On input 1^n with $\mathrm{bin}'(n) = \langle \mathrm{bin}(m), \mathrm{bin}(k), b\rangle$ a $\mathrm{P}^{\mathrm{NP}^X}$-machine can compute $f(1^m)$ by doing a prefix search, each time querying its NP^X oracle whether $f(1^m)$ starts with the prefix computed so far. Once $f(1^m)$ has been computed, we can check whether the kth bit is b. By assumption, we now know $\mathrm{tc}(f) \in \mathrm{P}^Y$ and, hence, $f \in \mathrm{FP}^Y$. For the other direction, start with a tally set in $L \in \mathrm{P}^{\mathrm{NP}^X}$. As shown by Krentel [23, Theorem

4], L can be written as $L = \{x \in \{1\}^* \mid \langle x, f(x)\rangle \in R\}$ for some $R \in \mathrm{P}$ and some (tally) function $f \in \mathrm{MaxP}^X$. Then, by assumption, $f \in \mathrm{FP}^Y$, from which can conclude $L \in \mathrm{P}^Y$ as claimed.

Let us now prove the second claim. For the direction from left to right, let a function $f \in \mathrm{NPFewV}^X$ be given. This time, the tally coding alone will not suffice. Let L be the following set: It contains all words 1^n such that $\mathrm{bin}'(n)$ is of one of the two forms $\langle 0, \mathrm{bin}(m), \mathrm{bin}(s)\rangle$ or $\langle 1, \mathrm{bin}(m), \mathrm{bin}(s), \mathrm{bin}(k), b\rangle$, with $m, s, k \geq 0$ and $b \in \{0,1\}$, and such that the following holds: If $\mathrm{bin}'(n)$ is of the first form, then $|\text{set-}f(1^m)| \leq s$. If $\mathrm{bin}'(n)$ is of the second form, then there must exist a subset $S \subseteq \text{set-}f(1^m)$ such that (a) $|S| = s$, (b) the length of the coding $\langle S\rangle$ is at most k, and (c) the kth bit of $\langle S\rangle$ is b. We claim that $L \in \mathrm{NP}^X$ holds. On input 1^n an NP^X-machine can easily determine the form of $\mathrm{bin}'(n)$. If it is of the first form, it guesses s different outputs, verifies them, and accepts. If it is of the second form, it also guesses k different outputs, verifies them, and checks whether the appropriate bit of the coding is, indeed, b. By assumption, we have $L \in \mathrm{P}^Y$. To show $f \in \mathrm{FP}^Y$, on input 1^m a P^Y-machine can first find the maximum s such that $\langle 0, \mathrm{bin}(m), \mathrm{bin}(s)\rangle \in L$. Once this number has been found, we can do a prefix search using queries of the second kind to compute $f(1^n)$. For the second proof direction, the function f for which $\text{set-}f(1^n) = \{1\}$ if there is an accepting computation and for which $\text{set-}f(1^n) = \emptyset$ otherwise lies in NPSV^X and, hence, in FP^Y. □

3.2 Upward and Downward Collapses for Standard Left Cuts

The Hartmanis–Immerman–Sewelson result states that $\mathrm{NE} = \mathrm{E} \iff \mathrm{NP} \cap \mathrm{SPARSE} \subseteq \mathrm{P}$. We now explore what happens when we replace SPARSE by the class STANDARD-LEFT-CUTS. One might expect that a similar result holds or that standard left cuts might cause a weaker collapse since, in some sense, there is "less information stored" in each level of a standard left cut compared to a sparse set. A bit surprisingly, a *stronger* collapse happens: $\mathrm{E}^{\mathrm{NP}} = \mathrm{E} \iff \mathrm{NP} \cap \text{STANDARD-LEFT-CUTS} \subseteq \mathrm{P}$.

Theorem 3.5. *Let X and Y be oracles. Then*

$$\mathrm{E}^{\mathrm{NP}^X} \subseteq \mathrm{E}^Y \iff \mathrm{NP}^X \cap \text{STANDARD-LEFT-CUTS} \subseteq \mathrm{P}^Y.$$

Proof. By Theorem 3.1 it suffices to show that all tally functions in MaxP^X lie in FP^Y iff all standard left cuts in NP^X lie in P^Y.

For the first direction, let a standard left cut $L \in \mathrm{NP}^X$ be given and assume that all tally functions in MaxP^X lie in FP^Y. Consider the tally function f that maps 1^n to the lexicographically maximal $x \in \{0,1\}^n$ with $x \in L$ (such an x must exist since $0^n \in L$ holds for all n). We claim $f \in \mathrm{MaxP}^X$. This holds since a machine M^X can on input 1^n guess a word $x \in \{0,1\}^n$, then guess a certificate for $x \in L$, and – if this certificate turns out to be correct – output x. The maximum output of this nondeterministic machine will be $f(x)$. This shows that f is a tally function in MaxP^X and hence, by assumption, $f \in \mathrm{FP}^Y$. However, this implies $L \in \mathrm{P}^Y$ since on input $x \in \{0,1\}^n$ a P^Y-machine can compute $f(1^{|x|})$ and then accept the input x iff $x \leq_{\mathrm{lex}} f(1^{|x|})$.

For the second direction, let a tally function $f \in \mathrm{MaxP}^X$ be given and assume that every standard left cut in NP^X lies in P^Y. Our aim is to show that $f \in \mathrm{FP}^Y$ holds. To this end, we will construct a special standard left cut $L \in \mathrm{NP}^X$ such that this standard left cut contains "all the information" about f. Let $f \in \mathrm{MaxP}^X$ via the machine $M^{()}$.

The basic idea (which will have to be modified to actually work) is the following: Suppose that $|f(1^n)| = n$ were to hold for all n and suppose that we were supposed to find a "lengthwise left cut," that is, suppose we could decide on a new cut number for each word length n. Then, we could setup the language L in the following way: It contains exactly the words $y \in \Sigma^n$ with

$y \leq_{\text{lex}} f(1^n)$. The $L \in \mathrm{NP}^X$ since on input y we can guess an accepting computation path of M^X and accept if the output is lexicographically at least y. Furthermore, this language will be such a "lengthwise left cut." Then, by assumption, we will have $L \in \mathrm{P}^Y$. We claim that this implies $f \in \mathrm{FP}^Y$: On input 1^n a P^Y-machine can do a prefix search on the words of length n to find the lexicographically largest word in the language of this length – and this word is exactly $f(1^n)$.

To make the basic idea work, we must make some technical modifications. First, $f(1^n)$ might have a varying length. Let p be a strictly increasing polynomial such that $|f(x)| \leq p(|x|)$ holds for all $x \in \Sigma^*$. Let $f'(x)$ be defined by $f'(x) = \langle f(x), 0^{p(|x|)-|f(x)|}\rangle$. Recall that we defined the pairing function $\langle .,. \rangle$ such that each bit of the first word is prefixed by a 0-bit and each bit of the second word is prefixed by a 1-bit. This implies that if $f(x) \leq_{\text{lex}} f(y)$ then $f'(x) \leq_{\text{lex}} f'(y)$. Also note that $|f'(x)| = 2p(|x|)$.

Second, we must solve the problem that we need a single standard left cut, not a collection of lengthwise cuts. We define the real number r that induces the standard left cut L as follows: Let $r = \sum_{i=1}^{\infty} b_i \cdot 2^{-i}$ where b_i is the ith bit of the infinite bitstring $f'(0)f'(1)f'(2)\cdots$.

We claim $L \in \mathrm{NP}^X$. Let $q(n) = \sum_{i=0}^{n} 2p(n)$. For a word x of length $q(n)$, let x^0 consist of the first $2p(0)$ bits of x, let x^1 consists of the following $2p(1)$ bits of x, let x^2 consists of the following $2p(2)$ bits, and so on. Then $x = x^0 \dots x^n$ and $|x^i| = 2p(i)$. For a word x that is not of length $q(n)$ for any n, we pad x at the end with zeros until x has such a length and then define $x^1, \dots, x^n$ as above for this word $x0^j$.

Now, let x be an input for which an NP^X-machine N should decide whether $x \in L$ holds. Let $x0^j = x^0 \dots x^n$ as described above. For each $i \in \{0, \dots, n\}$, the machine N guesses a computation path of M^X for the input 1^i. Then it checks that all of these computation paths are accepting. Let o^i be the output produced by M^X for the input 1^i on the guessed computation path and let $\bar{o}^i = \langle o^i, 0^{p(i)-|f(1^i)|}\rangle$. Then N accepts the input x iff $x = x^0 \dots x^n \leq_{\text{lex}} \bar{o}^0 \dots \bar{o}^n$. To see that N does, indeed, accept the standard left cut L just note that the lexicographically maximal sequence $\bar{o}^0 \dots \bar{o}^n$ that is nondeterministically produced by N on any computation path is exactly $f'(0) \dots f'(n)$.

We have constructed a standard left cut $L \in \mathrm{NP}^X$. By assumption this implies $L \in \mathrm{P}^Y$. It remains to show how we can use L to show $f \in \mathrm{FP}^Y$. A deterministic polynomial-time algorithm for computing $f(1^n)$ works as follows: On input 1^n, we start a prefix search on L to compute $f'(0) \dots f'(n)$. This is done by, starting with the prefix $p = \epsilon$, repeatedly testing whether $p10^{q(n)-|p|-1} \in L$ holds and, if so, continuing with $p1$ and otherwise with $p0$. Once $f'(0) \dots f'(n)$ has been computed, we can extract $f(n)$ from the last $2p(n)$ bits. □

3.3 Downward Collapses for P-sel and P/poly

In the present section we prove downward collapses where instead of sparse sets or standard left cuts we use P-selective sets or sets in P/poly. We do not prove new upward collapses, but note that STANDARD-LEFT-CUTS $\subsetneq$ P-sel $\subsetneq$ P/poly implies that the upward collapse for standard left cuts also holds for P-selective sets and for P/poly.

Theorem 3.6. *Let X and Y be oracles. If* $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}^X}} \subseteq \mathrm{E}^Y$, *then* $(\mathrm{NP}^X \cup \mathrm{coNP}^X) \cap \text{P-sel} \subseteq \mathrm{P}^Y$.

Proof. Assume $\mathrm{E}^{\mathrm{NP}^{\mathrm{NP}^X}} \subseteq \mathrm{E}^Y$. By Theorem 3.1, this implies that all tally functions in $\mathrm{MaxP}^{\mathrm{NP}^X}$ lie in FP^Y. Let a set $L \in \mathrm{NP}^X \cap \text{P-sel}$ be given. We will argue that $L \in \mathrm{P}^Y$ holds. Since P-sel is closed under complement, this suffices to prove the claim.

Let $M^{()}$ witness $L \in \mathrm{NP}^X$ and let $f\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ be the selector function that witness $L \in$ P-sel. Let M_{sel} compute f.

Our aim is to construct a function $g \in \text{MaxP}^{\text{NP}^X}$ with the following property: Let a word length n be given and let T be the subtournament of T_f^n (recall that this is the tournament induced by the selector f on words of length n) whose vertex set is the set of words in L and whose edges are inherited from T_f^n. Then $g(1^n)$ should be the code of a dominating set T; and if T is empty, then $g(1^n)$ should encode the empty set.

Before we proceed, let us have a look at how g may help us. Suppose such a tally function $g \in \text{MaxP}^{\text{NP}^X}$ exists. As pointed out above, this implies $g \in \text{FP}^Y$. But, then, $L \in \text{P}^Y$ holds: On input of a word x, we can compute $g(1^{|x|})$ in polynomial time and, then, it suffices to check whether x is dominated by one of the vertices encoded by $g(1^{|x|})$. If this is the case, then $x \in L$ holds; otherwise $x \notin L$ holds.

It remains to argue that we can construct the function g with the desired properties. For this, consider the following nondeterministic Turing machine M: On input 1^n it nondeterministically guesses a subset S of Σ^n of size at most $\log_2(n+1)$. This subset can also have size less than $\log_s(n+1)$ and can even be empty. Next, for each element $s \in S$ the machine M guesses a certificate and then verifies that $s \in L$ holds. Provided that S passes all these verifications, the machine M poses an oracle query, described below, and if the oracle query is positive, then $\langle S \rangle$ is output and M accepts.

The verifications done up to know ensure that $S \subseteq L$ holds. The oracle query is used to verify that S is a dominating set of the vertices in $L \cap \Sigma^n$. For this we must verify that *for all words* $w \in \Sigma^n$ one of the following two alternatives is the case: (a) there is an edge from one of the elements of S to w in T_f^n, which implies that w is dominated by S, or (b) that $w \notin L$ holds. The second test amounts to checking that there is no witness for $w \in L$, that is, that for all possible candidate witnesses it turns out that they are not witnesses, after all. To sum up, we can compile a single oracle query to a coNP-oracle by asking whether for all words $w \in \Sigma^n$ and for all candidate witnesses it is the cases that (a) there is an edge from one of the elements of S to w or (b) the candidate is not a witness for $w \in L$.

We claim that all outputs of M on input 1^n are dominating sets of $L \cap \Sigma^n$. Clearly, we have setup the checks in such a way that M will output *only* dominating sets. Furthermore, M always does output such a set since every tournament has a dominating set of logarithmic size. □

Corollary 3.7.

1. *If* $\Delta_{i+2}^{\text{e}} = \text{E}$, *then* $\Sigma_i^{\text{p}} \cap \text{P-sel} = \text{P}$.
2. *If* $\Sigma_i^{\text{p}} \cap \text{P-sel} = \text{P}$, *then* $\Delta_{i+1}^{\text{e}} = \text{E}$.

Theorem 3.8. *Let X and Y be oracles. If* $\text{E}^{\text{NP}^{\text{NP}^X}} \subseteq \text{E}^Y$, *then* $\text{NP}^X \cap \text{coNP}^X \cap \text{P/poly} \subseteq \text{P}^Y$.

Proof. Assume $\text{E}^{\text{NP}^{\text{NP}^X}} \subseteq \text{E}^Y$. By Theorem 3.1, this implies that all tally functions in $\text{MaxP}^{\text{NP}^X}$ lie in FP^Y. Let a set $L \in \text{NP}^X \cap \text{coNP}^X \cap \text{P/poly}$ be given. We will argue that $L \in \text{P}^Y$ holds.

Let $M_{\text{NP}}^{()}$ witness $L \in \text{NP}^X$ and let $M_{\text{coNP}}^{()}$ witness $L \in \text{coNP}^X$. Let $f \colon \mathbb{N} \to \Sigma^*$ and M_{adv} be the advice function and the deterministic polynomially time-bounded machine that witness $L \in \text{P/poly}$, that is, $x \in L$ iff M_{adv} accepts the input $\langle x, f(|x|) \rangle$. Let p be a polynomial such that $|f(n)| \le p(n)$ holds for all n.

Our aim is to construct a function $f' \in \text{MaxP}^{\text{NP}^X}$ that outputs advice strings. These advice strings need not be the original advice strings output by f, they only need to "behave" in the same way. The functions f will only be needed to ensure that such an advice string always exists.

For a word length $n \geq 0$, let us call a bitstring b of length at most $p(n)$ *good advice* if it has the following property:

$$\forall x \in \{0,1\}^n \colon x \in L \iff M_{\text{adv}}\big(\langle x, b\rangle\big) \text{ accepts} \tag{$*$}$$

We claim that there is a tally function in $\text{MaxP}^{\text{NP}^X}$ that maps every 1^n to a good advice. To see this, consider the following nondeterministic Turing machine: On input 1^n it nondeterministically guesses a candidate for a good advice string. Then, it checks whether this string satisfies $(*)$ – we will see in a moment how this is done. If this test is passed, the bitstring is output. This means that all outputs of the machine will be good advices and, hence, so is the lexicographically largest one.

It remains to describe how we can decide, by querying an NP^X-oracle Z, whether $(*)$ holds for a given bitstring b. A machine accepting the complement of Z works as following: On input b it checks whether the following predicate holds:

$$\forall x \in \{0,1\}^n \colon \Big(\big(M_{\text{adv}}\big(\langle x, b\rangle\big) \text{ accepts} \implies M^X_{\text{coNP}}(x) \text{ accepts}\big) \land \\ \big(M_{\text{adv}}\big(\langle x, b\rangle\big) \text{ rejects} \implies M^X_{\text{NP}}(x) \text{ rejects}\big)\Big)$$

We claim that this predicate can, indeed, be checked by a coNP^X-machine: It branches universally nondeterministically over all $x \in \{0,1\}^n$ and then branches universally over either (a) all computation paths of $M^X_{\text{coNP}}(x)$ and checks that they are all accepting or (b) over all computation paths of $M^X_{\text{NP}}(x)$ and checks that they are all rejecting. The desired NP^X-oracle Z can now be defined as the complement of the words accepted by the coNP^X-machine just described.

We now know that there is a tally function in $\text{MaxP}^{\text{NP}^X}$ that always outputs good advice. This implies that this function lies in FP^Y. We can then decide L in polynomial time with access to the oracle Y using the following algorithm: On input x we compute good advice and then accept if M_{adv} accepts relative to this advice. ☐

Corollary 3.9.

1. *If* $\Delta^{\text{e}}_{i+2} = \text{E}$*, then* $\Sigma^{\text{p}}_i \cap \Pi^{\text{p}}_i \cap \text{P/poly} = \text{P}$.
2. *If* $\Sigma^{\text{p}}_i \cap \text{P/poly} = \text{P}$*, then* $\Delta^{\text{e}}_{i+1} = \text{E}$.

Note that this result suggests that the following strong generalization of the Hartmanis–Immerman–Sewelson theorem presumably does *not* hold: $\text{NE} = \text{E} \iff \text{NP} \cap \text{P/poly} = \text{P}$. Rather, $\text{NP} \cap \text{P/poly} = \text{P}$ is linked to the exponential hierarchy, but to other levels.

Corollary 3.10. $\text{EH} = \text{E} \iff \text{PH} \cap \text{P/poly} = \text{P}$.

Corollary 3.11. *If* $\Delta^{\text{e}}_3 = \text{E}$*, then* $\text{BPP} = \text{P}$*. If* $\Delta^{\text{e}}_2 = \text{E}$*, then* $\text{ZPP} = \text{P}$.

Proof. It is well known, see for instance [29], that $\text{BPP} \subseteq \Pi^{\text{p}}_2 \cap \Sigma^{\text{p}}_2 \cap \text{P/poly}$ and $\text{ZPP} \subseteq \text{NP} \cap \text{coNP} \cap \text{P/poly}$. ☐

4 The Complexity of Infinite Subsets of P-Selective Sets

4.1 The Complexity of Top-Toda Languages

In the present section we explore the complexity of top-Toda languages. The study of these languages, which began in [12], is motivated by the following observation: Consider a P-selective language A and a vertex v of a tournament T^n_f such that all vertices of the tournament are reachable from v. Then, by Lemma 2.1, if any word of length n is in A, so is v. This makes such words interesting if, as in the present paper, one is "looking for words in A."

As proposed in [12], let us call two vertices u and v of T_f^n *Toda-equivalent* if there is a path from u to v in T_f^n and also a path from v to u. The Toda equivalence classes of T_f^n are its strongly connected components. Since T_f^n is a tournament, there exists a *top Toda equivalence class*. Its elements have the property that all vertices of the tournament are reachable from all of these elements. The language TOP-TODA_f, called a *top-Toda language* in the following, is defined by: $\text{TOP-TODA}_f = \{u \in \Sigma^* \mid \text{all vertices of } T_f^{|u|} \text{ are reachable from } u\}$.

By the intuition given above, the languages in $\text{TOP-TODA} = \{\text{TOP-TODA}_f \mid f \text{ is a selector}\}$ are good candidates for infinite subsets of P-selective sets. (Note that all top-Toda languages are populated, which implies that they are all infinite.) We will explore this relationship in more detail in Section 4.3; in the present section we will just try understand the complexity of these sets. We begin with some comparatively simple observations that prepare the central result of this section, Theorem 4.2, which links the complexity of top-Toda languages to the exponential hierarchy.

As was already noted in [12], $\text{TOP-TODA} \subseteq \text{PSPACE}$ since we can decide in polynomial space whether all words in the tournament $T_f^{|u|}$ can be reached from u. This was improved in the technical report [10], where it was shown that $\text{TOP-TODA} \subseteq \Pi_2^{\text{P}}$. The main observation in this report was that one can apply a result of Nickelsen and Tantau [27] to the tournament $T_f^{|u|}$, namely that the reachability problem for succinctly represented tournaments is Π_2^{P}-complete.

The next observation, which was also presented in the technical report [10], shows that top-Toda languages are presumably quite "simple": For all selectors f the language TOP-TODA_f is lengthwise P-selective. The reason is that if all words of a tournament can be reached from u and there is an edge from v to u, then all words in the tournament can also be reached from v. Note that it is not clear whether all top-Toda languages are also P-selective (without the "lengthwise" restriction). Although all tally sets are lengthwise P-selective and, hence, there are lengthwise P-selective sets that are not P-selective as a simple diagonalization argument shows, proving the existence of top-Toda languages that are not P-selective would imply $\text{P} \neq \text{NP}$: All languages in P are trivially P-selective and all top-Toda languages lie in Π_2^{P}.

The fact that all top-Toda languages are lengthwise P-selective has different consequences for their complexity. For P-selective sets (without the "lengthwise" modifier) it is well known that they lie in P/n^2, see [21], and also in $\text{NP}/n+1$, but not necessarily in NP/n, see [14]. All of these bounds also apply to lengthwise P-selective sets: For the lower bound this is trivial since all P-selective sets are also lengthwise P-selective. For the upper bounds, one has to revisit the proofs in [21] and [14] and become convinced that the proofs argue for each word length separately (which they do). This shows that for all selectors f we have $\text{TOP-TODA}_f \in \text{NP}/n+1$ and $\text{TOP-TODA}_f \in \text{P}/n^2$. This refutes the conjecture of Hemaspaandra et al. [12] that there are top-Toda languages that do not lie in NP/linear.

The observation $\text{TOP-TODA} \subseteq \text{P/poly}$ is a strong indicator that top-Toda languages cannot be NP-hard, because by the Karp–Lipton theorem NP-hard languages cannot have polynomially-sized circuits unless the polynomial hierarchy collapses, see [20] and also the newer results [5, 22, 6]. However, for P-selective sets a much stronger result is known [1, 3, 28]: If a P-selective set is $\leq_{o(n)\text{-tt}}^{\text{P}}$-hard for NP, then $\text{P} = \text{NP}$.

This result does *not* seem to carry over to lengthwise P-selective sets since the reduction can query words of different word lengths and it is unclear what should be done in such a case. However, in the technical report [10] it is shown that one can at least prove that if some lengthwise P-selective set is $\leq_{\text{m}}^{\text{P}}$-hard for NP, then $\text{P} = \text{NP}$. This result can be improved further as the following theorem shows:

Theorem 4.1. *If some lengthwise* P*-selective set is* $\leq^{\mathrm{p}}_{\text{1-tt}}$*-hard for* NP*, then* P = NP.

Proof. Our argument uses a standard self-reduction tree pruning technique. Assume that SAT $\leq^{\mathrm{p}}_{\text{1-tt}} A$ holds for some lengthwise P-selective set A. Let f be a selector for A and let p be a polynomial time bound on the running time of f. We present a polynomial-time algorithm for deciding SAT.

On input of the code of an n-variable formula ϕ, the algorithm creates a list that contains only ϕ initially. This list will change in the phases of the algorithm, but the following invariant will always be true: ϕ is satisfiable if, and only if, at least one formula in the list is satisfiable. The algorithm consists of two alternating phases: During the expansion phase, a still unprocessed variable in ϕ is chosen. Then all formulas in the list are replaced by two formulas, one with the variable set to *true* and one with the variable set to *false.* Clearly, this expansion step will not violate the invariant and it will at most double the length of the list. During the pruning phase, to be described later, the list is shortened, so that after the pruning the list length is at most $2p(|\phi|)$. After at most $|\phi|$ alternations of the expansion and pruning phases, the list will contain only formulas without any variables. We then test, in polynomial time, whether any formula is satisfiable and accept if this is the case; otherwise we reject.

The pruning phase works as follows: If the length of the list is less than $2p(|\phi|)$, nothing needs to be done. Otherwise, for each formula ϕ_i in the list, we compute the word x_i to which ϕ_i is mapped by the reduction. Note that $|x_i| \leq p(|\phi|)$. This implies that, since there are more than $2p(|\phi|)$ members in the list, that there exist three different indices i, j, and k such that $|x_i| = |x_j| = |x_k|$, that is, three different formulas ϕ_i, ϕ_j, and ϕ_k must be mapped to the same word length l. Applying the selector f to all pairings of these words, we can sort them, so let us assume that $\chi_A(x_i) \leq \chi_A(x_j) \leq \chi_A(x_k)$. This implies that $\chi_A(x_i, x_j, x_k) \in \{000, 001, 011, 111\}$. From this we can conclude that $\chi_{\text{SAT}}(\phi_i, \phi_j, \phi_k) \in \{000 \oplus a, 001 \oplus a, 011 \oplus a, 111 \oplus a\}$ where $a \in \{0,1\}^3$ models the bitflips performed by the reduction. By checking all eight possibilities for a one can easily verify that $\{000 \oplus a, 001 \oplus a, 011 \oplus a, 111 \oplus a\} \not\supseteq \{001, 010, 100\}$. This implies that for one of the three bitstrings $b = 001$, $b = 010$, or $b = 100$ we know that $\chi_{\text{SAT}}(\phi_i, \phi_j, \phi_k) \neq b$. But, then, we know one variable which, if it is satisfiable, it is not the only one. This means that we can remove this variable from the list without violating the invariant. The pruning phase repeats this process until the length of the list has dropped to at most $2p(|\phi|)$. ☐

The results that we have established up to know already tell us a lot about the complexity of top-Toda languages: They "live in Π^{p}_2, but more or less at the bottom and they cannot even be NP-hard." This begs the question of whether we can further improve the upper bound on the complexity of top-Toda languages. For instance, are all top-Toda languages already in P? The following main result of the present section states upward and downward collapses linking top-Toda languages to the exponential hierarchy.

Theorem 4.2.

1. TOP-TODA $\subseteq$ P *implies* $\mathrm{E}^{\Sigma^{\mathrm{p}}_2} = \mathrm{E}$.
2. $\mathrm{UE}^{\Sigma^{\mathrm{p}}_2} = \mathrm{E}$ *implies* TOP-TODA $\subseteq$ P.

For both claims we will use the characterization of Theorem 3.1. For one direction we use a special construction that draws on the ideas of Nickelsen and Tantau's [27] proof that the succinct tournament reachability problem is Π^{p}_2-complete. For the other direction we introduce the new concept of *splitting pairs.*

For the upward collapse in Theorem 4.2, by Theorem 3.1 it suffices to prove the following implication:

$$\text{TOP-TODA} \subseteq \mathrm{P} \implies \text{all tally functions in } \mathrm{MaxP}^{\mathrm{NP}} \text{ lie in FP.}$$

A first step towards proving this implication is to get a better "grip" on the class $\mathrm{MaxP}^{\mathrm{NP}}$. For this, we derive a Stockmeyer-like quantifier characterization [32] for this class. The proof uses standard arguments.

Lemma 4.3. *Let $f \in \mathrm{MaxP}^{\mathrm{NP}}$ and let $\forall x \in \Sigma^*\colon |f(x)| = p(|x|)$ for some polynomial p. Then there exists a ternary relation $R \in \mathrm{P}$ and a polynomial q such that for all words $x \in \Sigma^*$ the following holds: Let $y \in \Sigma^{q(|x|)}$ be lexicographically maximal such that $\forall z \in \Sigma^{q(|x|)}\colon \langle x, y, z\rangle \in R$. Then the first $p(|x|)$ bits of y are $f(x)$.*

Proof. Let $f \in \mathrm{MaxP}^X$ with $X \in \mathrm{NP}$ via some nondeterministic oracle Turing machine $M^{()}$. Let r be a polynomial bounding the running time of $M^{()}$ and let s be a polynomial bounding the running time of the machine witnessing $X \in \mathrm{NP}$. The language R contains all words $\langle x, y, z\rangle$ for which the following is true: The first $p(|x|)$ bits of y are the output of M^X on the computation path where the nondeterministic choices are taken according to the next $r(|x|)$ bits of y and the oracle queries are answered according to the next $r(|x|)$ bits of y, called the *oracle query bits* in the following, and this computation path is accepting. The next bits of y are $r(|x|)$ many blocks, each of length $s(r(|x|))$. If the ith oracle query bit is 1, then the ith block must contain a certificate that the ith oracle query of $M^{()}$ on the above computation path is an element of X. Finally, z must consist of $r(|x|)$ many blocks, each of which is also of length $s(r(|x|))$, plus some zeros so that $|z| = |y|$. If the ith oracle query bit is 0, then the ith block in z may not be a certificate that the ith query made by $M^{()}$ on the above computation path is an element of X.

The construction shows that, clearly, $R \in \mathrm{P}$. Furthermore, if there an accepting computation path of M^X on which it outputs $f(x)$, then there exists a y starting with this output, continuing with the correct nondeterministic choices, the correct oracle answers, and the correct certificates for these answers, such that for all z we have $\langle x, y, z\rangle \in R$. The other way round, if a y has the property that for all z we have $\langle x, y, z\rangle \in R$, then the oracle queries bits are all correct (as shown be the certificates) and, thus, there is an accepting computation path of M^X on which the first $p(|x|)$ bits of y are output. $\square$

For a ternary relation $R \in \mathrm{P}$ and a polynomial q as above, let $\text{max-y}_R(x)$ be the lexicographically largest $y \in \Sigma^{q(n)}$ such that for all $z \in \Sigma^{q(n)}$ we have $\langle x, y, z\rangle \in R$. In the following definition, for a word $x \in \Sigma^*$ we construct a tournament whose top-Toda equivalence class tells us a lot about $\text{max-y}_R(x)$.

Definition 4.4. Let R be a ternary relation and q a polynomial. Let $x \in \Sigma^*$ be a word. We construct a tournament $T_R(x)$ as follows: Its vertex set $V = \{s, s'\} \cup \left(\Sigma^{q(|x|)} \times \Sigma^{q(|x|)}\right)$ consists of the special *seed vertices s and s'* (the first of which will always be part of the top-Toda equivalence class) and a set of pairs $(y, z) \in \Sigma^{q(|x|)} \times \Sigma^{q(|x|)}$. We imagine these pair vertices to be arranged in a grid so that pairs with the same y-component lie on the same row and pairs with the same z-component lie in the same column. For two vertices (y, z) and (y', z') we say that the first pair is *above* the second pair if y is lexicographically larger than y'. We also say that the seed vertices are above all other vertices. The edges of the tournament are setup according to the following rules:

1. There is an edge from s to s'.
2. For vertices on the same row, that is, for two vertices (y, z) and (y, z'), the edges are setup in such a way that all vertices on the row are in the same Toda equivalence class. This can be accomplished, for instance, by having the edge point from (y, z) to (y, z') whenever z is lexicographically larger than z', except for the minimal and maximal z and z', where the edge points in the other direction.

3. If the vertex v is above the vertex u, then there is an edge from v to u; except in the following case: For a pair (y,z) with $\langle x,y,z\rangle \notin R$, there is an edge from (y,z) to the vertex directly above it in the grid or to s if the vertex is on the top row. More precisely, if $\langle x,y,z\rangle \notin R$, the edge between (y,z) and (y',z), where y' is the lexicographic successor of y, points from (y,z) to (y',z); and if y is lexicographically maximal, then there is an edge form (y,z) to s.

The intuition behind the tournament $T_R(x)$ is the following: The seeds are in the top-Toda equivalence class. Starting from the seed s, we join the lexicographically largest row to the top-Toda equivalence class if there is some z that refutes that for the lexicographically maximal y all z have the property $\langle x,y,z\rangle \in R$. Then we join the second largest row also to the top-Toda equivalence class if there is also a counterexample z for this second largest y. Then we join the third row, and so on. In particular, the first row that is *not* part of the top-Toda equivalence class has the following property: It is the lexicographically largest y such that for all z we have $\langle x,y,z\rangle \in R$. Comparing this with the definition of $\text{max-y}_R(x)$ we see that this exactly what we where interested in. Naturally, that means that we are actually more interested in the complement of the top-Toda equivalence class, but this will not matter in the proof of the upward collapse. The following lemma summarizes these observations.

Lemma 4.5. *Let R be a ternary relation and q be a polynomial. Let $x \in \Sigma^*$ be a word. Let $y \in \Sigma^{q(|x|)}$ be lexicographically maximal such that $(y, 0^{q(|x|)})$ is not in the top-Toda equivalence class of $T_R(x)$. Then $y = \text{max-y}_R(x)$ (if either side of the equation is undefined, so it the other). Furthermore, for all $y_\leq \leq_{\text{lex}} y$ the word $(y_\leq, 0^{q(|x|)})$ is also not in the top Top equation class; and for all $y_> >_{\text{lex}} y$ the word $(y_>, 0^{q(|x|)})$ is in the top-Toda equivalence class.*

Proof. The only edge leading to s' is the edge from s to s'. This means that any vertex v can be in the top-Toda equivalence class only if there is a path from v to s. On the other hand, if this is the case, then v is automatically part of the top-Toda equivalence class since all vertices other than s are then reachable from s' in one additional step. By the construction, the only way to get from a vertex $(y, 0^{q(|x|)})$ to s is to "brave all rows": There are no edges leading from any row to a row more than one level above. This means that we can get from $(y, 0^{q(|x|)})$ to s if, and only if, for all $y' \geq_{\text{lex}} y$ there exists a z such that $\langle x,y',z\rangle \in R$ does not hold. The other way round, this implies that for the lexicographically largest y such that there is no path from $(y, 0^{q(|x|)})$ to s, for all $z \in \Sigma^{q(|x|)}$ we have $\langle x,y,z\rangle \in R$. By definition, this y is $\text{max-y}_R(x)$. □

With the above lemma and definition we have established a link between $\text{max-y}_R(x)$ and the top-Toda equivalence class of a single tournament. The words x we are interested in, namely for which we would like to compute the leading bits of $\text{max-y}_R(x)$, are the words $x \in \{1\}^*$. To turn all the tournaments into a top-Toda language, we just have to place these tournaments on different word length levels. The number of words available on this level must be large enough to store the whole tournament $T_{R,f}(x)$ and we must also take care of the extra words that are available for the word length. Definition 4.6 shows how all of this is done.

Definition 4.6. Let R be a ternary relation and let q be a monotone polynomial. For each word length $l \geq 0$ we define a tournament T_R^l on Σ^l as follows: We distinguish two cases, depending on whether there exists an $n \geq 1$ such that $l = n + q(n)^2 + 1$.

1. If there exists no such n, the tournament T_R^l is arbitrary. So, say, let there be an edge from u to v in T_R^l if u is lexicographically larger than v.
2. If there exists such an n, let $x = 1^n$. We embed the tournament $T_R(x) = (V,E)$ into the tournament $T_R^l = (\Sigma^l, E')$ using an injective (one-to-one) mapping $\iota\colon V \to \Sigma^l$. Let $\iota((y,z)) = 0yz0^n$, let $\iota(s) = 110^{l-2}$ and $\iota(s') = 1110^{l-3}$.

The edges in T^l_R are setup as follows: (a) for $u, v \in V$ we have $(\iota(u), \iota(v)) \in E'$ iff $(u, v) \in E$, (b) for $u \in V$ and $b \in \Sigma^l - \{\iota(v) \mid v \in V\}$ let there be an edge from u to b and (c) for $b, c \in \Sigma^l - \{\iota(v) \mid v \in V\}$ let there be an edge from b to c exactly if b is lexicographically smaller than c.

The following lemma states that the embedding of the above definition faithfully captures the structure of the top-Toda equivalence classes. It follows directly from the definitions.

Lemma 4.7. *Let R be a ternary relation and q a monotone polynomial. Let $x = 1^n$. Then $(y, z) \in \Sigma^{q(n)} \times \Sigma^{q(n)}$ is in the top-Toda equivalence class of $T_R(x)$ if, and only if, $0yz0^n$ is in the top-Toda equivalence class of $T^{n+q(n)^2+1}$.*

The last step is to prove the implication stated at the beginning of this section:

Lemma 4.8. *If* TOP-TODA $\subseteq$ P*, then all tally functions in* $\mathrm{MaxP}^{\mathrm{NP}}$ *lie in* FP.

Proof. Let $h \colon \{1\}^* \to \Sigma^*$ in $\mathrm{MaxP}^{\mathrm{NP}}$ be given. We must show $h \in \mathrm{FP}$. Let R and q be the relation and polynomial constructed in Lemma 4.3 for h. Consider the tournaments T^l_R from Definition 4.6. We claim that there exists a selector f such that $T^l_f = T^l_R$ holds for all word lengths l.

The selector f gets two words u and v as input. If they have different lengths it is not really important what the selector does (let f choose the shorter one), so assume $|u| = |v|$. By Definition 4.6 the selector should first check whether there exists an n such that the length of both words is exactly $n + q(n)^2 + 1$. If no such n exists, then Definition 4.6 states that f should select the lexicographically larger word. Otherwise f finds out which of the three cases (a), (b), or (c) from Definition 4.6 applies to u and v. Again, due to the simplicity of the function ι, this check is easy to perform in polynomial time. For the cases (b) and (c) it is immediately clear which of the two words u and v should be picked by f, so assume that case (a) occurs. In this case, $u = 0yz0^n$ and $v = 0y'z'0^n$ for appropriate $y, y', z, z' \in \Sigma^{q(n)}$ or either of u and v or even both are seed vertices. The selector f must then find out how the edge between the corresponding vertices of $T_R(x)$ are directed according to Definition 4.4. Clearly, the first two cases of the definition are easily handled. For the third case, if it turns out that $z = z'$ and y' is the lexicographic successor of y, then the selector f must check whether $\langle x, y, z\rangle \notin R$ holds. The same is true if y is lexicographically maximal and the other vertex is the seed vertex s. Since $R \in \mathrm{P}$ and since $|y|$ and $|z|$ are bounded by $|u|$, this check can also be performed in polynomial time.

Consider the top-Toda language induced by the selector f. By assumption, $\textsc{top-toda}_f \in \mathrm{P}$ via some machine M. We now show how h can be computed in polynomial time. We know that $h(1^n)$ is equal to the first $p(n)$ bits of $\text{max-y}_R(1^n)$, so it suffices to compute this. By Lemma 4.5 we must compute the lexicographically maximal $y \in \Sigma^{q(n)}$ such that $(y, 0^{q(n)})$ is not in the top-Toda equivalence class of $T_R(1^n)$. By Lemma 4.7, this is the same as computing the lexicographically maximal $y \in \Sigma^{q(n)}$ such that $0y0^{q(n)}0^n$ is in the top-Toda equivalence class of T^l_R for $l = n + q(n)^2 + 1$. Finally, by the construction of the selector f, this is the same as computing the lexicographically maximal $y \in \Sigma^{q(n)}$ such that $0y0^{q(n)}0^n \notin \textsc{top-toda}_f$. In the following, let us write $\gamma(y)$ for $0y0^{q(n)}0^n$.

The computation of y is based on a prefix search. We start with the empty prefix $p = \epsilon$ and now wish to extend p by one bit. For this, we test whether $\gamma(p10^{q(n)-1-|p|})$ lies in $\textsc{top-toda}_f$. If we find out that this is the case, we know that y cannot start with the prefix $p1$ and we continue with $p0$. If we find out the opposite, then y starts with $p1$. After at most $q(n)$ steps we will have determined the complete y and we are done. ☐

For the downward collapse in Theorem 4.2, by Theorem 3.1 it suffices to prove the following implication:

$$\text{All tally functions in } \mathrm{NPSV}_{\mathrm{g}}^{\Sigma_2^{\mathrm{P}}} \text{ lie in FP} \implies \text{TOP-TODA} \subseteq \mathrm{P}.$$

The idea for proving the implication is the following: In order to show TOP-TODA $\subseteq$ P, for a given selector f we construct a function $h\colon \{1\}^* \to \Sigma^*$ that will provide us with some "advice." For a word x, once we know the advice $h(|x|)$, it will be easy to decide x. However, unlike in Ko's [21] proof that TOP-TODA $\subseteq$ P/poly, the advice will not "appear by magic." Rather, it will be computable using a function in $\mathrm{NPSV}_{\mathrm{g}}^{\Sigma_2^{\mathrm{P}}}$. Then, by our assumption that this class lies in FP, we can conclude that the advice can actually be computed in polynomial time. The next definition will be useful for defining the advice for top-Toda languages.

Definition 4.9. Let $T = (V, E)$ be a tournament. A *splitting pair* is a pair (A, D) consisting of two vertex sets $A, D \subseteq V$ such that

1. $|A|, |D| \leq \log_2(n + 1)$.
2. anti-dom(A) and dom(D) form a partition of V, that is, every vertex of V is an element of exactly one of these two sets.
3. For every $a \in$ anti-dom(A) and every $d \in$ dom(D) we have $(a, d) \in E$, that is, edges point from the anti-dominated set to the dominated set.

The intuition behind this definition is the following: Imagine the tournament T to be drawn on a page such that vertices in the same Toda equivalence class are on the same height; and for vertices $u, v \in V$ in different Toda equivalence classes, if $(u, v) \in E$, then u is above v on the page. Then the top-Toda equivalence class is at the top of the page. The splitting pair splits the vertex set into two disjoint subsets anti-dom(A) and dom(D). On the page, all vertices in the first set will be above the vertices in the second set. If nonempty, the set A contains at least one vertex "at the bottom" of anti-dom(A) and D contains at least one vertex "at the top" of dom(D). The following lemmas make these ideas precise.

Lemma 4.10. *Let (A, D) be a splitting pair of T. Then both* anti-dom(A) *and* dom(D) *can be expressed as unions of Toda equivalence classes of T.*

Proof. The set dom(D) is closed under reachability since any edge "leaving" this set would have to end in anti-dom(A), which is forbidden by the properties of a splitting pair. This implies that dom(D) is also closed under Toda equivalence and, hence, it is the union of all the Toda equivalence classes of its elements. The same argument applies to anti-dom(A) if we reverse all edges. ☐

The lemma implies, in particular, that for any two splitting pairs (A, D) and (A', D') only one of the following three situations can arise:

1. dom$(D) \subsetneq$ dom(D') and anti-dom$(A) \supsetneq$ anti-dom(A').
2. dom$(D) =$ dom(D') and anti-dom$(A) =$ anti-dom(A').
3. dom$(D) \supsetneq$ dom(D') and anti-dom$(A) \subsetneq$ anti-dom(A').

This allows us to define a linear ordering on splitting pairs as follows.

Definition 4.11. Given a tournament $T = (\Sigma^n, E)$ and two splitting pairs (A, D) and (A', D'), we write $(A, D) < (A', D')$ exactly in the following cases:

1. anti-dom$(A) \subsetneq$ anti-dom(A')

2. anti-dom(A) = anti-dom(A') and
$$\langle a_1,\ldots,a_{|A|},d_1,\ldots,d_{|D|}\rangle <_{\text{lex}} \langle a'_1,\ldots,a'_{|A'|},d'_1,\ldots,d'_{|D'|}\rangle.$$

Here, a_i, d_i, a'_i, and d'_i are the elements of A, D, A', and D', respectively.

Lemma 4.12. *Let $T = (V,E)$ be a tournament and let $U \subseteq V$ be its top Toda equivalence class. Then there exists a splitting pair (A,D) of T with* anti-dom$(A) = U$.

Proof. The top-Toda equivalence class has an anti-dominating set A of size at most $\log_2(n+1)$. Likewise, the complement of the top-Toda equivalence class has a dominating set D also of size $\log_2(n+1)$. These two sets form the claimed splitting pair. ☐

Lemma 4.13. *For every top-Toda language B there exists a tally function $h \in \mathrm{NPSV}_{\mathrm{g}}^{\Sigma_2^{\mathrm{p}}}$ such that $x \in B$ can be decided in polynomial time if $h(|x|)$ is given.*

Proof. Let $B = \text{TOP-TODA}_f$ for some selector f. For a given word length n, consider the tournament T_f^n induced by f. This tournament has different splitting pairs (for instance, the pair $(A,\emptyset)$ where A is an anti-dominating set of Σ^n of logarithmic size is such a pair). Let us call the largest (with respect to the linear ordering from Definition 4.11) splitting pair (A,D) with $A \neq \emptyset$ the *top splitting pair.* This top splitting pair (or, rather, its code) will be the advice $h(1^n)$.

We use the top splitting pair to decide B on length n as follows: For an input x, let (A,D) be the top splitting pair that the advice function has provided. We claim $x \in B$ iff $x \in$ anti-dom(A), that is, anti-dom(A) is exactly the top-Toda equivalence class of T_f^n. This follows from Lemma 4.12 and the fact that (A,D) is maximal. So, in order to decide B, all we need to do is to check whether $x \in$ anti-dom(A) holds, that is, whether $x \in A$ or whether there exists some $a \in A$ such that (x,a) is an edge of T_f^n. Because the size of A is $O(\log 2^n) = O(n)$, this test can be done in polynomial time.

It remains to specify an oracle machine $M^{()}$ and an oracle $L \in \Sigma_2^{\mathrm{p}}$ such that the only output of this machine on input 1^n is $g(1^n)$, that is, the top splitting pair. Furthermore, we must be able to check in polynomial time, but using the oracle L, that a given input is this top splitting pair $g(1^n)$.

Let L_1 contain all codes of splitting pairs of all T_f^n. We claim $L_1 \in$ coNP. To see this, consider a pair (A,D) that is given as input. We must then check whether the three properties from Definition 4.9 are satisfied. The first property is trivial to check. For the second property we do a universal quantification over all vertices v of T_f^n and check, for each v, whether exactly one of the following two cases occurs: (a) $v \in A$ or $\exists a \in A\colon (v,a) \in E$ or (b) $v \in D$ or $\exists d \in D\colon (d,v) \in E$. Here, E is the edge set of T_f^n. Again, because A and D both have logarithmic size, the "$\exists a \in A$" and "$\exists d \in D$" checks can be done in polynomial time. For the third property, we also quantify universally, but this time over two vertices u and v, and do a similar check.

Next, define a language L_2: It contains all splitting pairs (A,D) with $A \neq \emptyset$ such that there exists a splitting pair (A',D') with $A' \neq \emptyset$ and $(A,D) < (A',D')$. We claim $L_2 \in \Sigma_2^{\mathrm{p}}$. On input (A,D) we first check that (A,D) is a splitting pair using the coNP-oracle L_1, then guess a pair (A',D'), and then verify, using L_1 once more, that this pair is a splitting pair. Then, we verify that $(A,D) < (A',D')$ holds (we do not need the oracle for this).

The last step is the definition of the oracle machine $M^{()}$, which uses the join of L_1 and L_2 as its oracle. On input 1^n the machine M nondeterministically guesses a pair (A,D) with $A \neq \emptyset$. It verifies that this pair is a splitting pair using L_1 and then that it does not lie in L_2 (which is only the case for the top splitting pair). If both tests pass, the code of (A,D) is output. Note that the same procedure can be used to verify that a given pair is the top splitting pair. ☐

4.2 Infinite Subsets of King Languages

In the present section we study the immunity of king languages. A *king* of a tournament is a vertex such that all vertices of the tournament are reachable in at most two steps. This means, in particular, that a king is always an element of the top-Toda equivalence class of a tournament. It is well known that every tournament has a king [24]; for instance, every vertex of maximal outdegree is a king. These observations show that the following *king language* KING_f is always a populated (and hence infinite) subset of TOP-TODA_f:

$$\text{KINGS}_f = \{u \in \Sigma^* \mid \text{all vertices of } T_f^{|u|} \text{ are reachable from } u \text{ is at most two steps}\}.$$

Intuitively, king languages should be "easier" than top-Toda languages. After all, to decide whether a vertex u is an element of a top-Toda equivalence class we have to check whether all vertices of the tournament are reachable in any number of steps – for kings we only have to check whether they are reachable in at most two steps. This intuition seemed to be supported by the results in [12], where it was shown that TOP-TODA $\subseteq$ PSPACE and KINGS $\subseteq \Pi_2^{\mathrm{p}}$. However, the results of [10, 11] and the present paper show that the intuition seems to be wrong: the top-Toda languages are "pretty low" inside the class Π_2^{p}, while every language in Π_2^{p} is first-order equivalent to a king language. In particular, there exist Π_2^{p}-complete king languages.

Even though there are Π_2^{p}-complete king languages, we may nevertheless ask how difficult infinite subsets of king languages can become. Clearly, every infinite king language has an infinite subset in Π_2^{p} (namely itself), but it is already unclear whether it will also have an infinite subset in, say, Σ_2^{p}. It is known that, in general, we cannot always expect this to happen. For instance, Vereshchagin [33] has shown that there exists an oracle such that some NP-complete does not have an infinite subset in coNP.

Our first result of the present section is that every king language has a populated subset in Σ_2^{p}. This implies, also, that every top Toda language has an infinite subset in Σ_2^{p}. In the next section we shall see how this can be used to show that the P-selective sets are not $\Sigma_2^{\mathrm{p}}/1$-immune. For the proof of the result we need a definition and a lemma.

Definition 4.14. Let $T = (V, E)$ be a tournament. A vertex $k \in V$ is a *superking* if there exists a set $U \subseteq V$ such that

1. $|U| \leq \log_2(|V| + 1)$,
2. $\{k\}$ dominates U,
3. U dominates V.

Lemma 4.15. *Every nonempty tournament has a superking.*

Proof. We inductively construct the set $U = \{u_1, \dots, u_m\}$ as follows: $T_1 = T$ must have a vertex of maximal outdegree and, since the sum of all out-degrees minus all in-degrees adds up to zero, one vertex u_1 must have an outdegree that is at least as large as its indegree. Let u_1 be this vertex. Then $\{u_1\}$ dominates at least half of the vertices in T_1. Let T_2 be obtained from T_1 by removing all vertices in $\text{dom}(\{u_1\})$ (and all pending edges). Then the size of T_2 is at most half the size of T_1. For this tournament we repeat the process of finding a vertex u_2 of maximal outdegree and removing the vertices that it dominates to obtain T_3. We repeat the construction until T_{m+1} is empty.

We claim that the last vertex u_m that was put into the set $U = \{u_1, \dots, u_m\}$ is a superking. First, U has a size that is logarithmic in the size of V, because the size of the tournament is halved in each step. Second, u_m has the property that it was not put into U in any earlier step. In particular, when any u_i with $i < m$ was put into U, there was no edge from u_i to u_m for, otherwise, we would already have removed u_m. But, then, there must be an edge from u_m to

u_i. Third, U clearly dominates V since every vertex of V can be reached in one step from some u_i (or it is one of the u_i already). ☐

The construction of the set U in the proof is a standard construction that can both be used for showing that every tournament has a logarithmically small dominating set and that it has a king. The new observation is that the resulting king can be characterized in terms of a $\exists\forall$-property – normally kings are characterized in terms of a $\forall\exists$-property.

Theorem 4.16. *Every set in* KINGS *has a populated subset in* Σ_2^{p}.

Proof. Let f be a selector and let KINGS_f be given. The desired populated subset is given by the set of superkings of the tournament family specified by f. By Lemma 4.15 this set is, indeed, populated. To decide whether a given vertex k is a superking of $T_f^{|k|}$, we first existentially guess a set U of size at most $\log_2\bigl(|\Sigma^{|k|}|+1\bigr) \leq |k|+1$. Then we check that k dominates U, which can be done in polynomial time by checking for each element $u \in U$ whether the selector f selects k on input (k,u). Finally, we use a universal quantifier to verify that U dominates V, namely by checking that for all words $v \in V$ one of the polynomially many $u \in U$ has the property $f(u,v) = u$. ☐

The theorem has an interesting consequence, stated in the following corollary, which improves the result of Hemaspaandra et al. [12] that every king language has a finder in $\mathrm{FP}^{\Sigma_3^{\mathrm{p}}}$.

Corollary 4.17. *Every king language has a finder in* $\mathrm{MaxP}^{\mathrm{NP}} \subseteq \mathrm{FP}^{\Sigma_2^{\mathrm{p}}}$.

Proof. Let f be a selector and KING_f be its king language. The finder will output a superking for the word length. In detail, define the following nondeterministic oracle Turing machine M: On input 1^n it guesses a word w and a set U and verifies, using its coNP-oracle, that w is a superking of T_f^n, as witnessed by U. If the test is passed, w is output. Then all outputs of M are superkings and, hence, so is the lexicographically maximal one. Thus, the machine witnesses that KINGS_f has a finder in $\mathrm{MaxP}^{\mathrm{NP}}$. ☐

An immediate corollary of the above result is that every top-Toda language has a finder in $\mathrm{FP}^{\Sigma_2^{\mathrm{p}}}$, which was already proved in [12]. A natural question is, can we improve this even further? The following theorem states how difficult, exactly, it will be to get the complexity down to FP.

Theorem 4.18.

1. *If* $\Delta_3^{\mathrm{e}} = \mathrm{E}$*, then every king language has a finder in* FP.
2. *If every king language has a finder in* FP*, then* $\mathrm{E} = \mathrm{NE}$.

Proof. The first claim follows directly from Theorem 3.1 and the above corollary. For the second claim, assume that every king language has a finder in FP and let any top-Toda language A be given. Let f be a selector for A. By assumption, KINGS_f has a finder in FP. Because $\mathrm{KINGS}_f \subseteq \mathrm{TOP\text{-}TODA}_f$, this is also a finder for A. Thus, every top Toda language has a finder in FP. This implies $\mathrm{E} = \mathrm{NE}$ as shown by Hemaspaandra et al. [12]. ☐

As a corollary we obtain that if TOP-TODA $\subseteq$ P, then every set in KINGS has a finder in FP. Hence, TOP-TODA $\subseteq$ P implies that KINGS is not P-immune. Indeed, an even stronger claim is possible:

Theorem 4.19. *If* TOP-TODA $\subseteq$ P*, then every king language contains a top-Toda language as a subset.*

Proof. Observe that the image of a finder is a top-Toda language: A selector can simply always select the one element in the language on each word level. What it does on the other elements does not matter. ☐

Two remarks concerning the theorem are in order. First, note that every top-Toda language contains a king language as a subset; the theorem states that the inverse is also true under a certain assumption. Second, note that the assumption is, in some sense, that all top-Toda languages are "easy." One would expect that if top Toda language could also be more difficult, then it should be even easier to find a top-Toda language as a subset of every king language. Thus, the following conjecture seems plausible:

Conjecture 4.20. *Every king language contains a top-Toda language as a subset.*

4.3 Immunity of P-Selective Sets

In this section we return to the original question of this section, namely how difficult infinite subsets of infinite P-selective sets must be. As stated in the introduction, there exist P-selective sets that are bi-immune and, thus, there is no hope of finding an easy infinite subset in every P-selective set. Nevertheless, in the previous sections we saw that top-Toda and king languages, which are in some sense "nearly" infinite subsets of P-selective sets, are not too hard to decide. All we need to know is whether there exists at least one word on a word length – once we know that this is the case, both the top-Toda equivalence class and the kings set contain examples of such words.

In the following we first present two lemmas that show how infinite subsets of top-Toda or king languages and how finders for top-Toda and king languages help us in finding infinite subsets of P-selective sets. We then apply these lemmas to the results established in the previous sections.

Lemma 4.21. *If every top-Toda language has a populated subset in C, then every infinite lengthwise* P*-selective set has an infinite subset in $C/1$.*

Proof. Let an infinite lengthwise P-selective set A be given. Let f be a selector for A. By assumption, TOP-TODA_f has a populated subset $Q \in C$. Clearly, $Q \cap A \subseteq A$. Furthermore, $Q \cap A$ is infinite as it contains at least one word for each of the infinitely many word lengths for which A contains an element. Finally, $Q \in C/1$ since for each word length n we either have $Q \cap A \cap \Sigma^n = Q \cap \Sigma^n$ or $Q \cap A \cap \Sigma^n = \emptyset$ and one bit of advice can tell us which is the case. ☐

Note that it is unclear how one could prove the converse of the above theorem, but at least the following weaker statement clearly holds: If every infinite lengthwise P-selective set has an infinite subset in C, then every top-Toda language has an infinite subset in C.

Lemma 4.22. *Let* FC *be a function class.*

1. *If every set in* KINGS *has a finder in FC, so does every set in* TOP-TODA.
2. *Every set in* TOP-TODA *has a finder in FC if, and only if, every lengthwise* P*-selective set does.*

Proof. Let f be a selector. A finder for KINGS_f is also a finder for TOP-TODA_f because $\text{KINGS}_f \subseteq \text{TOP-TODA}_f$. It is also a finder for every lengthwise P-selective set A for which f is a selector (recall that if there is a word in A of length n at all, then $\text{TOP-TODA}_f \cap \Sigma^n \subseteq A$). Finally, note that every top-Toda language is lengthwise P-selective. ☐

The first of these lemmas has the following corollary, which follows from Corollary 4.17.

Corollary 4.23. P-*sel is not* $\Sigma_2^{\mathrm{p}}/1$-*immune.*

Combining the second of lemma with Corollary 4.17 yields the following.

Corollary 4.24. *Every lengthwise* P-*selective set has a finder in* $\mathrm{MaxP}^{\mathrm{NP}} \subseteq \mathrm{FP}^{\Sigma_2^{\mathrm{p}}}$.

This corollary begs the question of whether this can be improved, that is, does every P-selective set have a finder in FP?

Corollary 4.25.

1. *If* $\Delta_3^{\mathrm{e}} = \mathrm{E}$, *then every lengthwise* P-*selective set has a finder in* FP.
2. *If every lengthwise* P-*selective set has a finder in* FP, *then* $\mathrm{NE} = \mathrm{E}$.

Proof. This follows from Theorem 4.18 and Lemma 4.22. □

Corollary 4.26. *If* $\Delta_3^{\mathrm{e}} = \mathrm{E}$, *then* P-sel *is not* P/1-*immune.*

Proof. $\Delta_3^{\mathrm{e}} = \mathrm{E}$ implies, by Theorem 4.18, that every king language has a finder in FP and, by Lemma 4.22, so does every top-Toda language. The image of the finder is a populated subset of the top-Toda language in P. By Lemma 4.21 this implies the claim. □

5 Conclusion

In this paper we presented generalizations of the Hartmanis–Immerman–Sewelson theorem. For the applications studied in the second part of the paper, the characterization of collapses in the exponential-time realm in terms of function classes turned out to be especially useful. The results of the present paper entail a number of interesting open problems.

Can we characterize $\mathrm{NP} \cap \text{P-sel} = \mathrm{P}$ or $\mathrm{NP} \cap \mathrm{P/poly} = \mathrm{P}$ equivalently in terms of a collapse in the exponential-time realm? By our results on standard left cuts this collapse will be above Δ_2^{e}, and it will be below Δ_4^{e}, but is unclear where, exactly.

Can we improve the characterization of TOP-TODA $\subseteq$ P? One plausible conjecture is TOP-TODA $\subseteq \mathrm{P} \iff \Delta_3^{\mathrm{e}} = \mathrm{E}$.

Can we show that (perhaps lengthwise) P-sel not being P/1-immune has a consequences like E = NE? Such an upward collapse seems hard to prove because we start with a nonuniform assumption. However, here is a promising approach: If lengthwise P-sel is not P/1-immune, neither is TOP-TODA. Then every set in TOP-TODA must have a finder in $\mathrm{FP}^{\mathrm{NP}}$ since for each of the two possible advice strings we can search for a word that is presumably in the language and then apply the selector and output the more likely one. Finally, one would have to show (and this is where the argument breaks down) that every set in TOP-TODA having a finder in $\mathrm{FP}^{\mathrm{NP}}$ implies an upward collapse. For this an improvement would be needed over Hemaspaandra et al.'s result like the following conjecture: If every top-Toda language has a finder in FP, then $\mathrm{NE}^{\mathrm{NP}} = \mathrm{E}$.

The notion of superkings is new and little is known about them. For instance, how many superkings can a tournament have? A better understanding of their properties might help in answering some of the above questions.

References

[1] Manindra Agrawal and Vikraman Arvind. Quasi-linear truth-table reductions to P-selective sets. *Theoretical Computer Science*, 158(1–2):361–370, 1996.

[2] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[3] Richard Beigel, Martin Kummer, and Frank Stephan. Approximable sets. *Information and Computation*, 120(2):304–314, 1995.

[4] R. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM Journal on Computing*, 13(3):461–487, 1984.

[5] N. Bshouty, R. Cleve, S. Kannan, R. Gavaldà, and C. Tamon. Oracles and queries that are sufficient for exact learning. *Journal of Computer and System Sciences*, 52(3):421–433, 1996.

[6] Jin-Yi Cai. $\mathrm{S}_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.

[7] Judy Goldsmith, Deborah Joseph, and Paul Young. A note on bi-immunity and p-closeness of p-cheatable sets in p/poly. *Journal of Computer and System Sciences*, 46(3):349–362, 1993.

[8] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP − P: EXPTIME versus NEXPTIME. *Information and Control*, 65:158–181, 1985.

[9] Edith Hemaspaandra, Lane A. Hemaspaandra, and Harald Hempel. What's up with downward collapse: using the easy-hard technique to link boolean and polynomial hierarchy collapses. *SIGACT News*, 29(3):10–22, 1998.

[10] Edith Hemaspaandra, Lane A. Hemaspaandra, Till Tantau, and Osamu Watanabe. On the complexity of kings. Technical Report TR905, Computer Science Department, University of Rochester, December 2006.

[11] Edith Hemaspaandra, Lane A. Hemaspaandra, Till Tantau, and Osamu Watanabe. On the complexity of kings. In E. Csuhaj-Varj and Z. Ésik, editors, *Proceedings of FCT 2007*, volume 4639 of *Lecture Notes in Computer Science*, pages 328–340. Springer-Verlag, 2007.

[12] L. Hemaspaandra, M. Ogihara, M. Zaki, and M. Zimand. The complexity of finding top-Toda-equivalence-class members. *ACM Transactions on Computer Systems*, 39(5):669–684, 2006.

[13] L. Hemaspaandra and L. Torenvliet. P-selectivity, immunity, and the power of one bit. In *Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science*, volume 3881 of *Lecture Notes in Computer Science*, pages 323–331. Springer-Verlag, January 2006.

[14] Lane Hemaspaandra and Leen Torenvliet. Optimal advice. *Theoretical Computer Science*, 154(2):367–377, 1996.

[15] Lane A. Hemaspaandra, Harald Hempel, , and Arfst Nickelsen. Algebraic properties for selector functions. *SIAM Journal on Computing*, 33(6):1309–1337, 2004.

[16] Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer-Verlag, 2002.

[17] Lane A. Hemaspaandra, Leen Torenvliet, and L. Torenvliet. *Theory of Semi-Feasible Algorithms*. Springer-Verlag, 2002.

[18] Carl G. Jockusch. *Reducibilities in Recursive Function Theory*. PhD thesis, Massachusetts Institute of Technology, 1966.

[19] Carl G. Jockusch. Semirecursive sets and positive reducibility. *Transactions of the American Mathematical Society*, 131:420–436, 1968.

[20] R. Karp and R. Lipton. Some connections between uniform and non-uniform complexity classes. In *Proc. 12th ACM Symp. on Theory of Computing*, pages 302–309, 1980.

[21] Ker-I Ko. On self-reducibility and weak P-selectivity. *Journal of Computer and System Sciences*, 26(2):209–221, 1983.

[22] J. Köbler and O. Watanabe. New collapse consequences of np having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1998.

[23] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[24] H. Landau. On dominance relations and the structure of animal societies, III: The condition for score structure. *Bulletin of Mathematical Biophysics*, 15(2):143–148, 1953.

[25] Arfst Nickelsen. *Polynomial Time Partial Information Classes*. Wissenschaft und Technik Verlag, 2001. Dissertation, Technische Universität Berlin, 1999.

[26] Arfst Nickelsen and Till Tantau. Partial information classes. *SIGACT News*, 34(1):32–46, 2003.

[27] Arfst Nickelsen and Till Tantau. The complexity of finding paths in graphs with bounded independence number. *SIAM Journal on Computing*, 34(5):1176–1195, 2005.

[28] Mitsunori Ogihara. Polynomial-time membership comparable sets. *SIAM Journal on Computing*, 24(5):1068–1081, 1995.

[29] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

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

[31] Alan L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[32] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[33] N. Vereshchagin. Np-sets are co-NP-immune relative to a random oracle. In *Third Israel Symposium on Theory of Computing and Systems*, pages 40–45., Los Alamitos, CA, USA, 1995. IEEE Computer Society.