Index Sets and Parametric Reductions

Rod G. Downey*
Department of Mathematics
P.O. Box 600
Victoria University
Wellington, New Zealand
rod.downey@vuw.ac.nz

Michael R. Fellows †
Department of Computer Science
University of Victoria
Victoria, B.C. V8W 3P6, Canada
mfellows@csr.uvic.ca

December 6, 1999

Abstract

We investigate the index sets associated with the degree structures of computable sets under the parameterized reducibilities intorduced by the authors. We solve a question of the Peter Cholak and the first author by proving the fundamental index sets associated with a computable set A, $\{e : W_e \leq_q^u A\}$ for $q \in \{m, T\}$ are Σ_4^0 complete. We also show that $FPT(\leq_q^n)$, that is $\{e : W_e$ computable and $\equiv_q^n \emptyset\}$,

*Research supported by the New Zealand Marsden Fund for Basic Science

†Research supported by the National Science and Engineering Research Council of Canada.

is Σ_4^0 complete, but, paradoxically, there exist natural computable parameterized languages such that $\{e : W_e \text{ computable and } \leq_q^n A\}$ is in Δ_4^0. This last result corrects an error in our monograph (Theorem 19.23 (ii), where it is claimed that $\{e : W_e \text{ computable and } \leq_q^n A\}$ is *always* Σ_4^0 complete.) We do prove that $\{e : W_e \text{ computable and } \leq_q^n A\}$ is always Π_3^0 hard. We also look at computable presentability of these classes.

1 Introduction

The framework of parameterized computational complexity theory allows one to investigate how different aspects or different parts of the input to a problem may contribute differentially to overall problem complexity. For example, both of the well-known graph problems VERTEX COVER and DOMINATING SET take as input a graph G and a positive integer k. VERTEX COVER can be solved in time $O(f(k)n^c)$ where c is a constant independent of k, and n is the number of vertices of G, a complexity behavior termed *fixed-parameter tractability* in the theory. (For VERTEX COVER we may take $f(k) = (4/3)^k$ and $c = 1$ [BFV95].) The best that is known about DOMINATING SET is essentially that it can be solved by brute force (trying all k-subsets) in time $\Omega(n^{k+1})$.

Abstracting from these examples, the basic setting will be *parameterized languages* which are subsets L of $\Sigma^* \times \Sigma^*$. The second coordinate is called the *parameter*. For our purposes, we will consider the parameter to be an element of $\mathbb{N}$. The basic questions one considers is a problem of the form below.

Input: $\langle x, k\rangle$.
Parameter: k.
Question: Is $\langle x, k\rangle \in L$?

Intuitively, we are saying that L is "fixed parameter tractable" (formal definitions below) if it is "tractable by the slice." That is, there is a constant c *independent of* k such that, for a fixed k the basic question above is solvable in time $O(|\langle x, k\rangle|^n)$. The resulting problem classification is more or less orthogonal to classical complexity. For instance, the above examples of Vertex Cover and Dominating Set are examples of two problems whose classical complexity is identical, and the corresponding parametric complexity is apparently quite different.

We remark that a parameter may also be some aggregate of information. For example, if for the above two problems the parameter describes not only the size of the vertex set, but also the treewidth of G, then both problems are fixed-parameter tractable. The theory provides a sensitive classification tool which allows us to understand the manner *by which* a problem becomes intractable. Furthermore, like average case behaviour, in many cases there turn out to be standard techniques such as Courcelle's Theorem [Co87, Co90] which allow one to salvage some degree of tractability from a problem which is classically intractable. The theory has been shown to address many natural problems in diverse areas of computer science, and it can be regarded as one of the principal "applied" complexity theories [BDFW95, BDFHW95, BF95, BFH94, CCDF94, CW95, DEF93, DF92, DF95a, DFHKW94, FK93].

While the concrete complexity theory is quite well developed, as can be seen from the compendiums in [DF95a] and [DF99], our understanding of the underlying *metatheory* of the basic reductions is still fairly primitive. For instance, it is unknown if there is an analogue of the basic Ladner result that the polynomial time degrees of computable sets form a dense upper semi-lattice. The goal of the present paper is to contribute to our understanding of the structure of computable sets under parameterized polynomial time reductions.

A striking feature of the concrete reductions is their complexity. The seem, on the surface, much more difficult to construct than the analogous classical polynomial time reductions. Furthermore, in the two earlier investigations [DF93] and [CD94], it is noteworthy that the principal tool used for the analysis of the structure of computable sets under parametric reductions was the *priority method.* This is well contrasted with studies into the classical structure of computable sets under either Karp or Cook reducibilities which often work by *injury free* simple delayed diagonalization. Some insight into why this complexity comes about can be gleaned from an analysis of the basic definitions.

Definition 1.1 (The Fundamental Definitions, [DF92, DF93, DF95a, DF99]) Let A be a parameterized problem.

(i) We say that A is *uniformly fixed parameter tractable* if there is an algorithm Φ, a constant c, and an arbitrary function $f : \mathbb{N} \mapsto \mathbb{N}$ such that
(a) the running time of $\Phi(\langle x, k\rangle)$ is at most $f(k)|x|^c$, and
(b) $\langle x, k\rangle \in A$ iff $\Phi(\langle x, k\rangle) = 1$.

(ii) We say that A is *strongly uniformly fixed parameter tractable* if A is uniformly fixed parameter tractable via some Φ, f such that f is computable.

(iii) We say that A is *nonuniformly fixed parameter tractable* if there is a constant c, a function $f : \mathbb{N} \mapsto \mathbb{N}$, and a collection of procedures $\{\Phi_k : k \in \mathbb{N}\}$ such that for each $k \in \mathbb{N}$, the running time of $\Phi_k(\langle x, k\rangle)$ is at most $f(k)|x|^c$ and $\langle x, k\rangle \in A$ iff $\Phi_k(\langle x, k\rangle) = 1$.

The reader should realize that the deefinitions above reflect three possible interpretations to the level of uniformity by which one can compute A by the slice. Natural examples of, for instance, graph theoretical problems of types (ii) and (iii) come from applications of the Graph Minor Theorem, so the various levels of uniformity arize naturally and therefore are more than mere academic exercises.

Reductions allow us to partially order languages in terms of their computational complexity. Two languages L and L' are taken to have the same complexity from the point of view of the given reduction iff they are in the same *degree*, that is there is a reduction from L to L' and vice versa. We will need to create reductions which express the fact that two languages have the same *parameterized* complexity. What is needed are reductions that ensure that if L' is computable in time $f(k)n^n$, "by the slice" then there is a parameterized algorithm for L running in time $g(k)n^{c'}$ "by the slice." After a moment's thought, we realize that we can achieve such reductions only if we allow each slice of L to reduce to a finite number of slices of L'. The easiest way to make such a reduction is to reduce, in parameterized polynomial time, the k-th slice of L to the k'-th slice of L'. This leads us to the *working definition*, Definition 1.2 below, of a parameterized reduction for combinatorial reductions.

Definition 1.2 (Basic Working Definition) We say that L reduces to L' by a *standard* parameterized m-reduction if there are functions $k \mapsto k'$ and $k \mapsto k''$ from $\mathbb{N}$ to $\mathbb{N}$, and a function $\langle x, k\rangle \mapsto x'$ from $\Sigma^* \times \mathbb{N}$ to Σ^*, such that

$$\langle x, k\rangle \mapsto x' \text{ is computable in time } k''|x|^n, \text{ and}$$
$$\langle x, k\rangle \in L \text{ iff } \langle x', k'\rangle \in L'.$$

We remark that the basic working definition is used in all known concrete reductions. This is primarily because naturally occurring parametric languages usually (always?) have the property that for all k, the k-th slice corresponds to a trivial instance of the $k+1$-st slice. For instance, for Dominating Set, a graph G has a size k dominating set, iff G^+ has a size $k+1$ dominating set where G^+ is the result of adding a single new unattached vertex to G. One could also develop the theory with only such self encoded, or *smooth* languages considered, and then we could well only use the basic working definition. However, it seems more natural to consider general parametric languages.

Thus, while the working definition is fine for most concrete reductions, for our metatheory, we will need to be more precise. Associated with the three flavours of uniformity of Definition 1.1, there are three reductions, of differing levels of uniformity.

Definition 1.3 (Uniform fixed parameter reducibility) Let A, B be par ameterized problems. We say that A is *uniformly fixed parameter-reducible* to B if there is an oracle procedure Φ, a constant α, and an arbitrary function $f : \mathbb{N} \mapsto \mathbb{N}$ such that
(a) the running time of $\Phi(B; \langle x, k\rangle)$ is at most $f(k)|x|^\alpha$,
(b) on input $\langle x, k\rangle$, Φ only asks oracle questions of $B^{(f(k))}$ where

$$B^{(f(k))} = \bigcup_{j \le f(k)} B_j = \{\langle x, j\rangle : j \le f(k) \& \langle x, j\rangle \in B\}, \text{ and}$$

(c) $\Phi(B) = A$.

If A is uniformly fixed parameter reducible to B we write $A \le_T^u B$. Where appropriate we may say that $A \le_T^u B$ *via* f. If the reduction is many:1 (an *m-reduction*), we will write $A \le_m^u B$.

Definition 1.4 (Strongly uniform reducibility) Let A, B be parameterized problems. We say that A is *strongly uniformly fixed parameter-reducible* to B if $A \leq_T^u B$ via f where f is computable. We write $A \leq_T^s B$ in this case.

Definition 1.5 (Nonuniform reducibility) Let A, B be parameterized problems. We say that A is *nonuniformly fixed parameter-reducible* to B there is a constant α, a function $f : \mathbb{N} \mapsto \mathbb{N}$, and a collection of procedures $\{\Phi_k : k \in \mathbb{N}\}$ such that, for each $k \in \mathbb{N}$, $\Phi_k(B^{(f(k))}) = A_k$ and the running time of Φ_k is at most $f(k)|x|^\alpha$. Here we write $A \leq_T^n B$.

Note that the above are good definitions since whenever $A \leq B$ with $\leq$ any of the reducibilities, if B is fixed parameter tractable so too is A. Note also that the above definitions allow us to specify the notions of fixed parameter tractability we had before. Now nonuniformly fixed parameter tractability corresponds to being $\leq_T^n \emptyset$. *We will henceforth write $FPT(\leq)$ as the class of computable sets, fixed parameter tractable corresponding to the reducibility $\leq$. And we will denote the uppersemilattice (USL) of $\leq$ degrees of computable sets by $\mathcal{R}(\leq)$.*

If $A \leq B$ via a standard reduction in the sense of Definition 1.2 then $A \leq_m^u B$.

In [DF93], the authors first studied the structure of $\mathcal{R}(\leq)$ for the various $\leq$. For instance, for the class $\mathcal{R}(\leq_q^s)$ for $q \in \{m, T\}$, it is shown that the USL is dense, and is not a lattice. In [CD94], for the m-degree case, Cholak and Downey prove certain undecidability results. Naturally, associated with a computable language L there are *index sets* such as $\{e : W_e \leq A\}$. In the classical setting of Karp and Cook reducibilities, an index set such as this is Σ_2^0-complete if $A \notin P$. Cholak and Downey observed that the index sets in tha parametric setting seem more complex.

Theorem 1.1 (Cholak and Downey [CD94]) *Let A be a computable set. Then for $q \in \{m, T\}$,*
(i) $\{e : W_e \leq_q^s A\}$ is a Σ_3^0 set.
(ii) $\{e : W_e \leq_q^u A\}$ is a Σ_4^0 set.
(iii) $\{e : W_e$ computable and $W_e \leq_q^n A\}$ is a Σ_4^0 set.

Proof. We prove (ii) and (iii). The others are left to the reader. By the definition, for (ii), we see that

$$W \leq_q^u A \text{ iff } \exists e, c \forall k \exists u \forall z (\Phi_e(A^{(u)}; \langle z, k \rangle) = W(\langle z, k \rangle) \text{ in time } u|z|^c).$$

For (iii), we see that $W_e \leq_q^n A$ iff W_e computable and

$$\exists c \forall k \exists e, u \forall z (\Phi_e(A^{(u)}; \langle z, k \rangle) = W_e(\langle z, k \rangle \text{ in time } u|z|^c).$$

These are clearly Σ_4^0. $\square$.

In the present paper, we will prove that the index sets of Theorem 1.1 are *always* more complex than the analogous classical ones. Indeed they are as always as complex as they can be.

Theorem 1.2 *Suppose that A is computable. Let $q \in \{m, T\}$. Then the index sets*
(i) $\{e : W_e \leq_q^s A\}$ and $\{e : W_e \equiv_q^s A\}$ are Σ_3^0 complete.
(ii) $\{e : W_e \leq_q^u A\}$, $\{e : W_e \equiv_q^u A\}$ $\{e : W_e$ computable and $\leq_q^n \emptyset\}$ (i.e. $FPT(\leq_q^n)$) are Σ_4^0 complete.

As we will see, Theorem 1.2 has certain ramifications concerning the effective presentability of the computable languages reducibly to, for instance, a computable language A. See Corollary 2.10 and Corollary 2.11.

The relevance of theorem 1.2 is the following. In any normal construction analysing a degree structure, one must usually perform some diagonalization or the like, usually against potential reductions provided by the "opponent". In a classical argument, the index set is Σ_2^0, and hence the recognition of such potential reductions is relatively simple. Indeed, Hartmanis [Ha89] demonstrated how to use the simplicity of these index sets to obtain a number of classical structural results.

Our results show that in the parametric setting the arguments should be at least as complex as those used for strong reducibilities (such as *weak truth table* reducibility) on the computably enumerable sets for the reducibility $\leq_q^s$, and says that for the other parametric reducibilities, the arguments should have the same order of complexity as those used for the computable enumerable sets under classical Turing reducibility. Therefore it is natural, and probably necesary, that the principal tool should be the finite injury priority method, (in the case of strong uniform parametric reducibility) and the infinite injury method for the non-strongly uniform cases.

The next section is devoted to the proof of Theorem 1.2. The notations and terminology is drawn from Soare [So87], and Balcazaar et. al. [BDG]. We use the expressions "computable" in place of "recursive" and similarly "computable enumerable" ("c.e.") in place of "recursively enumerable"("r.e."), in line with recent changes in the subject. We shall assume that the reader is familiar with the basics of computability theory and with tree of strategies priority arguments. Thus, for instance, $\{\varphi_e : e \in \mathbb{N}\}$ denotes the e-th unary partial computable function. We will differ from Soare's notation by using $\leq_L$ denote *lexicographic* ordering.

2 Proof of Theorem 1.2

We begin with (i). We prove that $\{e : W_e \leq_T^s A\}$ is Σ_3^0 complete, the remainder of (i) being left to the reader. For ease of notation we shall first prove that $FPT(\leq_T^s)$ is Σ_3^0 complete. We will then point out the easy modifications to obtain the result for A computable. Thus let B be a Σ_3^0 set. By definition, there is a computable relation Q such that for all e

$$e \in B \text{ iff } \exists x \forall y \exists z Q(x, y, z, e).$$

We will build a collection $\mathcal{C}$ of sets $V_e = W_{f(e)}$ whose indices are given by the s-m-n theorem, such that we meet the requirements

$$R_e : V_e \leq_T^s \emptyset \text{ iff } e \in B.$$

Actually the fundamental strategy is to try to ensure that $V_e \not\leq_T^s \emptyset$, and only $fail$ to achieve this goal if $e \in B$.

Now, $V_e \leq_T^s \emptyset$ means that there should be some reduction triple consisting of a reduction (Φ_m), a total computable function φ_m and a positive integer n, such that for all $\langle w, k\rangle$ $\Phi_m(\emptyset; \langle w, k\rangle) = V_e(\langle w, k\rangle)$, and the running time of the computation is bounded by $\varphi(k)|w|^n$. Hence, provided that $e \notin B$, we will meet all the "subrequirements" below.

$R_{\langle e,m,n\rangle}$: If $e \notin B$ then
either φ_m is not total, or
for some k, w, $\Phi_m(\emptyset; \langle w, k\rangle) \neq V_e(\langle w, k\rangle)$,or
$\Phi_m(\emptyset; \langle w, k\rangle)$ does not run in time $\leq \varphi_m(k)|w|^n$.

We will suppose that each requirement occurs infinitely often. It is easiest to conceive of the construction as follows. For each potential "witness" x to the outer existential quantifier used to show that $e \in B$, we will have a *Control Device* $\mathcal{C}(x)$. The idea of the control device is that it tells us when to attempt to meet the subrequirements.

The fundamental plan is to pursue an elaboration of the strategy of Theorem 2.5 of Downey and Fellows [DF93][1]. For the present situation, our strategy is the following:

We build V_e in stages. At stage s we decide the fate of $\langle 1^s, k\rangle$ for all $k \in \mathbb{N}$. (If $k > s$ then $\langle 1^s, k\rangle \notin V_e$ by fiat.) In Downey and Fellows [DF93], we devoted the j-th slice of $V = V_e$ to meeting R_j where j will be some $\langle e, m, n\rangle$. In the present paper, this assignment will be replaced by a more flexible arrangement where $slice(j, s)$ denotes the current slice devoted to satisfying R_j. Modulo the control device (to be described) (so that we can attack the requitement freely at each stage), at stage s, the construction of V_e runs as follows:

For each $\langle e, m, n\rangle \leq s$, if $R_{e,m,n}$ is not yet declared satisfied, compute s steps in the computation of $\varphi_m(slice(\langle e, m, n\rangle, s))$. (Call this $\varphi_{m,s}(slice(\langle e, m, n\rangle, s))$.) If $\varphi_{m,s}(slice(\langle e, m, n\rangle, s)) \uparrow$ do nothing for $\langle e, m, n\rangle$ at this stage. If $\varphi_{m,s}(slice(\langle e, m, n\rangle, s)) \downarrow$ declare $R_{e,m,n}$ as satisfied and perform the following diagonalization for $\langle e, m, n\rangle$. Run $\Phi_m(\emptyset; \langle 1^s, slice(\langle e, m, n\rangle, s)\rangle)$ for $\varphi_m(slice(\langle e, m, n\rangle, s))s^n$ many steps. If this does not halt in this many steps we need do nothing since the running time is wrong. If $\Phi_m(\emptyset; \langle 1^s, slice(\langle e, m, n\rangle, s)\rangle) \downarrow$ in $\varphi_m(slice(\langle e, i, n\rangle, s))s^n$ or fewer steps set

$$V_e(\langle 1^s, slice(\langle e, i, n\rangle, s)\rangle) = 1 - \Phi_m(\emptyset; \langle 1^s, slice(\langle e, m, n\rangle, s)\rangle).$$

The enumeration of B controls the above in the following way. We say

[1]This theorem was that (i) $FPT(\leq_T^u) \subset FPT(\leq_T^n)$ even for computable sets, and (ii) $FPT(\leq_T^s) \subset FPT(\leq_T^u)$.

that x is *s-confirmed* (for e) *at stage* t if

$$\forall y \leq s \exists z \leq t Q(e,x,y,z).$$

We say that $\mathcal{C}(x)$ *asserts control of* $R_{\langle e,m,n\rangle}$ *at stage* t if
(i) x is s-confirmed at stage t,
(ii) x was not s confirmed at stage $t-1$, and
(ii) $x < \langle e,m,n\rangle < s$.

We remark that the idea here is that if the x is a least correct witness to e being in B then the inner Π_2 part of the definition of B (namely $\forall y \exists z Q(x,y,z,e)$ will "appear correct" (i.e. be confirmed) infinitely often). The principle idea is that such confirmation will allow us to move the slices we are dynamically assigning to the satisfaction of the requirements into higher slices. If e is really in B then, for all but a finite number of rows, we will see that we will kick all of the witness rows to infinity. On the other hand, if $e \notin B$ then eventually each slice, for example, $slice(j,s)$, comes to a resting place, and this is where we will get to meet R_j. More formally, we have the following.

The Full Construction: Incorporation of x-Control. Now we will modify the construction to incorporate $\mathcal{C}(x)$. At each stage t for each $x, e \leq t$, we first see if x is s confirmed for e for some $s \leq t$. If this is the case, we say that $\mathcal{C}(x)$ asserts control of various $R_{\langle e,m,n\rangle}$. In particular, for each such $\langle e,m,n\rangle$, we will reset $slice(\langle e,m,n\rangle, t+1)$ to be large and fresh. Specifically, for all $\langle m',n'\rangle \geq \langle m,n\rangle$, set $slice(\langle e,m',n'\rangle,t+1) = slice(\langle e,m',n'\rangle,t)+t+1$. Also we initialize $R_{e,m',n'}$.
End of Construction

Now the point of all the above is the following. One can easily establish by induction that
(i) $e \in B$ implies that for all but finitely many m,n, $R_{\langle e,m,n\rangle}$ is initialized infinitely often and hence $slice(\langle e,m,n\rangle,s) \to \infty$.
(ii) $e \notin B$ implies that for all m,n, $\lim_s slice(\langle e,m,n\rangle,s)$ exists.

To see that (i) holds, if $e \in B$ there is some x with $\forall y \exists z Q(e,x,y,z)$. Now such an x will be confirmed for e infinitely often. Hence it will assert control of any $R_{\langle e,m,n\rangle}$ for $\langle m,n\rangle > x$ infintely often, driving $slice(\langle e,m,n\rangle,s)$ to ∞. On the other hand, if $e \notin B$, then for each x, there is some y such that for all z, it is never the case that $Q(e,x,y,z)$. It follows that x will only be confirmed for e until it gets stuck on y. Thereafter, it can no longer initialize any $R_{\langle e,m,n\rangle}$. Since there are only j numbers $x < j$ and only $x < \langle m,n\rangle$ can initialize $R_{\langle e,m,n\rangle}$, it follows that there is a stage $t_{m,n}$ such that for all $t \geq t_{m,n}$, $R_{\langle e,m,n\rangle}$ is not initialized at stage t. Since $slice(\langle e,m,n\rangle,t)$ is only reset when $R_{\langle e,m,n\rangle}$ is initialized, it follows that $slice(\langle e,m,n\rangle,t)$ comes to a limit.

Finally we see that we meet all the requirements. If $e \notin B$ then a simple induction shows that, for $j = \langle e,m,n\rangle$, we will meet R_j in row $slice(j) = \lim_s slice(j,s)$.

If $e \in B$ then we argue as follows. Let x be least such that $\mathcal{C}(x)$ asserts control of some $R_{\langle e,m,n\rangle}$ infinitely often. Let s_0 be a stage such that for all $t > s_0$, for all $y < x$, $\mathcal{C}(y)$ does not assert control of $R_{\langle e,m,n\rangle}$ at stage t. Thus for all $\langle y,d\rangle < x$ for all $slice(\langle e,y,d\rangle,s_0) = slice(\langle e,y,d\rangle)$. For all j we know that R_j acts on row z at most once. Let $s_1 > s_0$ be a stage such that for all $\langle m',n'\rangle < x$, and for all $t > s_1$, $R_{e,m',n'}$ does not act at stage t. Then we know that for all $j < slice(x-1)$, and all p,

$$V_{e,s}(\langle p,j\rangle) = V_e(\langle p,j\rangle).$$

(Indeed we remark that for all p with $|p| > s_1$, $V_e(p) = 0$.) We now give a $FPT(\leq_T^s)$ decision procedure for determining if $u = \langle v,k\rangle \in V_e$.

If $k < slice(x-1)$ go to stage s_1 and see if $u \in V_e$. If $k \geq slice(x-1)+1$ go to the stage $s = s(k) > s_1$ where $\mathcal{C}(x)$ has asserted control of e k times. Then for all $t > s$, for all $\langle m,n\rangle \geq x$, $slice(\langle e,m,n\rangle,t) > k$. Hence for all $t > s$ and for all $\langle e,m,n\rangle$, $slice(\langle e,m,n\rangle,t) \neq k$. It follows that $u \in V_e$ iff $u \in V_{e,s}$. Notice that the computation of the stage $s(k)$ depends on k alone, and hence the procedure is independent of v. This is why the procedure is FPT in constant time by the slice. This concludes our proof that $FPT(\leq_T^s)$ is Σ_3^0 complete.

To modify the construction for the more general case proving that $\{e : W_e \leq_T^s A\}$ is also Σ_3^0 complete, the control device is the same but we must modify the diagonalization. Basically, one replaces the oracle $\emptyset$ by $A^{(\varphi_{m,s}(slice(\langle e,m,n\rangle,s)))}$. Since A is computable, the construction either make V_e in $FPT(\leq_T^s)$, or produce a language $V_e \not\leq_T^s A$. $\square$

(ii) The other result we will prove is that $\{e : W_e \leq_T^u A\}$ is Σ_4^0 complete. The remaining completeness results are totally analogous and are left to the reader, although we will indicate how to adapt the technique for the nonuniform case at the end of the proof.

Again we first take $A = \emptyset$. This proof is considerably more complex than the previous one and we need to take some care. Actually, this will need the infinite injury method, and uses a priority tree. We use the following representation theorem for Σ_4^0 sets:

Lemma 2.1 (Yates [Ya69]) *The index set $Comp = \{e : \emptyset' \leq_T W_e\}$ of complete computably enumerable sets, is Σ_4^0 complete*

For a proof of Lemma 2.1 see Soare [So87], XII §2. In the construction to follow, we will need to approximate $Comp$ as our control device. Thus, we will need to discover which W_e are T-complete. To do this we will need to guess the index of a procedure Φ_i with $\Phi_i(W_e) = K$, with $K = \emptyset'$. To this end, let

$$\ell(e, i, s) = \max\{x : \forall y < x(\Phi_i(W_{e,s}; y) = K_s(y))\}.$$

Now $\ell(e, i, s)$ can exhibit the following behaviour: We can have $\limsup_s \ell(e, i, s) \neq \infty$, or we can have $\limsup_s \ell(e, i, s) = \infty$. The latter can either mean that $\liminf \ell(e, i, s) = \infty$ (so that $\Phi_i(W_e) = K$), or that for some x, $\liminf \ell(e, i, s) = x$, and hence $\Phi_i(W_e; x) \uparrow$. Here we will use the so-called "hat" convention: We assume that if for a computably enumerable set M and procedure Φ_i if $\Phi_{i,s}(M_s; z) \downarrow$, but $z' \leq u(\Phi_{i,s}(M_s; z))$ enters $M_{s+1} - M_s$, then $\Phi_{i,s}(M_{s+1}; z) \uparrow$. The hat convention ensures that if $\Phi_i(M; z) \uparrow$ then $\exists^\infty s(\Phi_{i,s}(M_s; z'') \uparrow)$ for all $z'' \geq z$. The trick is due to Lachlan. (See Soare [So87], Ch. VIII.) On the priority tree we will need to guess if $\ell(e, i, s) \to \infty$. This will be measured at nodes τ, which we refer to as *top* nodes. Such τ nodes have outcomes (i, ∞) and (i, f) as in the standard and very well known priority tree proof (e.g Soare [So87] Ch XIV) Lachlan-Yates minimal pair theorem. The trouble is that we also need to differentiate between infinite and finite liminf behaviour. This is very difficult to achieve at a single node, and in fact we will decompose the behaviour into infinitely many guesses for the liminf below the (i, ∞) outcome of the τ node. At such a node trying to figure out such behaviour, which we refer to as a σ node, we will be considering e, i, x.

We will have two outcomes (i, x, ∞) and (i, x, u), with (i, x, u) to the left of of (i, x, ∞). Thus we have the priority tree given in Figure 1.

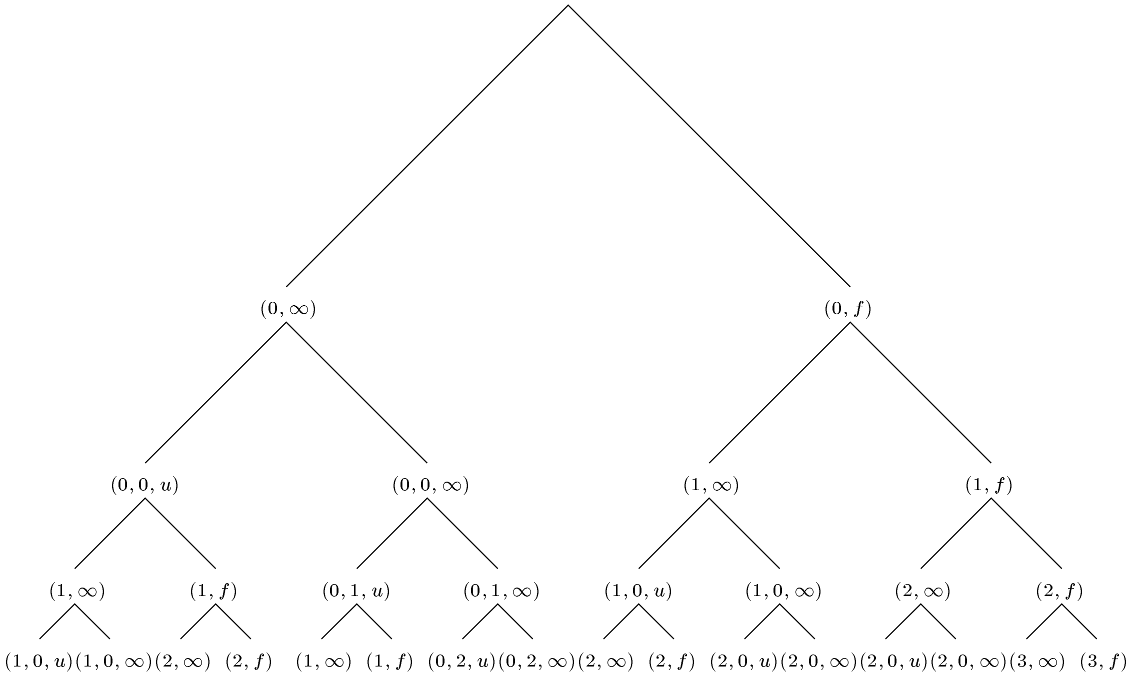

Figure 1: The Priority Tree for Theorem 1.2

The outcome (i, x, ∞) is meant to represent the fact that $\exists s \forall t \geq s(\ell(e, i, s) > x)$, whereas the outcome (i, x, u) represents the fact that infinitely often (i, x, ∞) looks correct, and yet later $\ell(e, i, s) \leq x$. (The "u" means "unbounded" here.) Thus this is the outcome that $\limsup \ell(e, i, s) \to \infty$, and yet (if x is least), $\liminf \ell(e, i, s) = x$.

In the same way we used $\mathcal{C}$ to control the strategies above, here we will use the Priority Tree PT to control our strategies. Note that we will build one version of PT for each e below. (The formal definition can be found in Definition 2.1, following some further motivational comments.)

Before we explain exactly how this control is achieved, let us turn to the actual requirements we will meet. For each e we will again build an auxiliary computable set V_e to try to meet the requirements below.

R_e: $V_e \in FPT(\leq_u^m)$ iff $e \in Comp$.

Since the V_e are all built separately, we will drop the "e" when the context is clear. Actually, as with the previous construction we will meet the following requirements.

R : Either $e \in Comp$ and $V \in FPT$ via some witness i
(Here, of course, by witness i we mean that $\Phi_i(W_e) = K$.)
or $\forall m, n R_{\langle e,m,n\rangle}$, where

$R_{\langle e,m,n\rangle}$: Either there exists an m', n' such that
$lim_s \varphi_m(\langle e, m', n'\rangle, s) =_{\text{def}} \varphi_m(e, m', n')$ fails to exist, or
there exist p, w, with
$\Phi_m(\emptyset, \langle p, w\rangle) \neq V(\langle p, w\rangle)$, or
it does not run in time $\leq \varphi_m(w)|p|^n$.

Notice that in the above, we have made the replacement of the unary partial computable function $\varphi_m(\cdot)$ by a binary partial computable function $\varphi_m(\cdot, \cdot)$ since in the uniform case we will need to guess the use and constant which will filter through the limit lemma. In fact we will assume that such φ_m are monotone in both variables. Again, we are using the requirements of Theorem 2.5 of [DF93] (ii) in modified form. We attempt to meet the $R_{\langle e,m,n\rangle}$ as we did there, but again modulo the external control this time driven by the Priority Tree. The basic strategy used to meet the $R_{\langle e,m,n\rangle}$ is the following.

At each stage s, if $R_{\langle e,m,n\rangle}$ is not as yet declared satisfied and $\langle e, m, n\rangle \leq s$ find the least unused $p \leq s$,if any, such that $p = \varphi_{m,s}(\langle e, m, n\rangle, t) \downarrow$ for some $t \leq s$. If either $\varphi_{m,s}(\langle e, m, n\rangle, t) \uparrow$ for all $t \leq s$ or there is no unused p do nothing. If p, and hence t, exist declare p as used. Now compute ps^n steps in the computation of $\Phi_m(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle)$. If this computation does not halt, do nothing else. If $\Phi_m(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle)$ halts in $\leq ps^n$ steps, win by setting

$$V_e(\langle 1^s, \langle e, m, n\rangle\rangle) = 1 - \Phi_m(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle).$$

Now again we will need to modify the above so that a version R_σ of $R_{\langle e,m,n\rangle}$ and guessing σ will work on a slice $slice(\sigma, s)$ instead of a fixed slice $\langle e, m, n\rangle$. Again we will need to argue that if there is no witness i to $e \in Comp$ and hence $e \notin Comp$, then for all σ on TP, $\lim_s slice(\sigma, s) = slice(\sigma)$ exists.

Additionally, if $e \notin Comp$, we will need to argue that for all $\langle e,m,n\rangle$ there is some $\sigma \subset TP$ devoted to solving $R_{\langle e,m,n\rangle}$. This is done as follows.

Definition 2.1 (The Priority Tree and Attachments)
Step 1. We formally define the priority tree and the attachments as follows. If α is on PT then α is of the form $\nu\hat{\ }(i,x,\infty)$, $\nu\hat{\ }(i,x,u)$, λ, $\nu\hat{\ }(i,f)$, or $\nu\hat{\ }(i,\infty)$. If α is of the form λ, $\nu\hat{\ }(i,f)$, or $\nu\hat{\ }(i,x,u)$ we say that α is a τ node. If α is of the form $\nu\hat{\ }(i,x,\infty)$ or $\nu\hat{\ }(i,\infty)$, we say that α is a σ node. Now we construct PT by induction on the length of nodes.

Case 1. If the node is a τ node then it will form the "top" of an i tree. Let $i(\tau)$ be the least i not yet assigned to any $\tau' \subset \tau$ and assign $i(\tau)$ to τ. (Hence $i(\tau) = i-1$ in all cases except $\tau = \lambda$.) The outcomes of τ will be $(i(\tau),\infty)$ and $(i(\tau),f)$ with $(i(\tau),\infty) <_L (i(\tau),f)$. Put $\tau\hat{\ }O$ on PT for the outcomes O of τ.

Case 2. If the node is a σ node so that it will be devoted to some x for some i, find the longest τ node $\subseteq \sigma$ and define $i(\sigma) = i(\tau)$ and $\tau(\sigma) = \tau$. (τ is called σ's *top*. Note that $i(\sigma) = i$ in all cases above.) It will be the case that $\tau\hat{\ }\infty \subseteq \sigma$. For all γ with $\tau\hat{\ }\infty \subseteq \gamma \subset \sigma$ we will have $\gamma'\hat{\ }(i,x',\infty) \subseteq \sigma$. Let

$$x(\sigma) = \max\{0, x'+1 : \tau\hat{\ }\infty \subseteq \gamma\hat{\ }(i,x',\infty) \subseteq \sigma\}.$$

(In all the specifying cases above it will be the case that $x = \max\{0, x'+1\}$.) Let σ have outcomes $(i,x,u) <_L (i,x,\infty)$. Put $\sigma\hat{\ }O$ on PT for each outcome O of σ.

Step 2. Now assign by induction on $\langle e,m,n\rangle$ versions of $R_{\langle e,m,n\rangle}$ to nodes of the form $\gamma = \sigma\hat{\ }(i,x,u)$ and $\gamma = \tau\hat{\ }f$ on PT. Do this in the obvious way. For each such node γ find the least $\langle e,m,n\rangle$ not attached to any node $\mu \subset \gamma$, and attach $R_{\langle e,m,n\rangle}$ to γ.

As with many tree arguments, we will need the notion of an α-stage.

Definition 2.2 (a) We define the notions α-stage, $m\ell(\alpha,s)$, and α-expansionary by induction on $|\alpha|$.
(i) Every stage s is a λ-stage.

(ii) **Case 1.** Suppose that s is a β-stage with β a top node devoted to solving the problem of whether $\ell(\beta, s) = \ell(i(\beta), s) \to \infty$. Define

$$m\ell(\beta, s) = \max\{0, \ell(\beta, t) : t \text{ is a } \beta\text{-stage } < s\}.$$

We say that s is β-expansionary if $\ell(\beta, s) > m\ell(\beta, s)$ and declare s to be a $\beta^\frown\infty$-stage.
If $\ell(\beta, s) \leq m\ell(\beta, s)$, declare that s is a $\beta^\frown f$-stage.
Case 2. Suppose that β is a σ node devoted to (i, x). If there has been a previous β-stage t let s' denote the largest β-stage $\leq s$. If $\ell(\tau(\beta), s') > x$ and there has been a stage m with $s' < m < s$ with $\ell(\tau(\beta), m) \leq x$, declare s to be a $\beta^\frown(i, x, u)$-stage. In any other case, declare s to be a $\beta^\frown(i, x, \infty)$-stage.

(b) We define TP_s, the apparent true path at stage s, to be the unique α of length s with s an α-stage.

Definition 2.3 Suppose that R_α is a version of $R_{\langle e,m,n\rangle}$ attached to α. We say that R_α *requires attention* at stage s if s is an α-stage, R_α is not yet declared satisfied, and there is some least unused $p \leq s$ such that $p = \varphi_{m,s}(slice(\alpha, s), t) \downarrow$ for some $t \leq s$.

The Construction. Having gone to all the effort above, the construction actually becomes rather easy.
Stage 0. Define $slice(\alpha, 0) = \alpha$ (that is, the Gödel number of α in some lexicographic order of the strings) for all α on PT with R_q attached.

Stage $s + 1$. Compute TP_{s+1}. For each γ with $\gamma \not\leq_L TP_{s+1}$, initialize γ and in particular reset $slice(\gamma, s)$ to be new and large. ($> s$). For each $\alpha \subseteq TP_{s+1}$, if R_α requires attention via p, declare p as used. Now compute ps^n steps in the computation of $\Phi_m(\emptyset; \langle 1^s, slice(\alpha, s)\rangle)$. If this does not halt do nothing, do nothing else. If $\Phi_m(\emptyset; \langle 1^s, slice(\alpha, s)\rangle)$ halts in $\leq ps^n$ steps, win by setting

$$V(\langle 1^s, slice(\alpha, s)\rangle) = 1 - \Phi_m(\emptyset; \langle 1^s, slice(\alpha, s)\rangle),$$

and declare R_α as satisfied.

End of Construction.
The verification is straightforward modulo all the definitions etc.

Lemma 2.2 *Suppose that $R_{\langle e,m,n\rangle}$ is attached to some node γ. Then for some τ and i, γ is of the form $\tau\hat{\ }(i,f)$ or of the form $\tau\hat{\ }(i,k,u)$ (for some k).*

Proof This is immediate by the way we attach requirements in Definition 2.1. $\square$

Let TP be the true path. (That is the leftmost path visited infinitely often.) Notice that the construction ensures that R_α only can receive attention at α-stages. (See Definition 2.3.) The initialization of γ right of TP_s means that the following is easily seen by induction:

Lemma 2.3 *(i) If $TP <_L \gamma$ then $slice(\gamma, s) \to \infty$.*
(ii) If $\gamma <_L TP$ and $\gamma \not\subset TP$ then γ is visited only finitely often, and hence there is some stage s_0 beyond which R_α will never receive attention.
(iii) If $TP \not<_L \gamma$ then $lim_s slice(\gamma, s) = slice(\gamma)$ exists.

Proof For (i), note that at stage $s+1$ after we compute TP_{s+1} we initialize all $\gamma \not\leq_L TP_{s+1}$ and in particular reset $slice(\gamma, s+1)$ to be bigger than s. Now if $TP <_L \gamma$, then for infinitely many s, $\gamma \not\leq_L TP_s$. Hence $slice(\gamma, s)$ is initialized infinitely many times and (i) follows.

(ii) follows by the definition of the TP as the leftmost path visited infinitely often and the fact that R_α can only recieve attention at an α-stage, i.e. where $\alpha \subseteq TP_s$.

For (iii), note that the *only* time that $slice(\gamma, s)$ is reset is at a stage s where $\gamma \not\leq TP_s$. If $TP \not<_L \gamma$, then either $\gamma \leq_L TP$, yet $\gamma \not\subseteq TP$, or $\gamma \subset TP$. In either case, there are only finitely many stages s where $\gamma \not\leq_L TP_s$, by definition of TP_s, and TP. $\square$

Lemma 2.4 *Suppose that for each $j < i$, j is not a witness for "$e \in Comp$". Then for each $j < i$, there is a τ and a σ such that either*
(i) $\tau\hat{\ }(j,\infty)\hat{\ }\sigma\hat{\ }(j,k,u) \subset TP$, for some $k \in \omega$, or
(ii) $\tau\hat{\ }(j,f) \subset TP$.

Proof We prove this by induction on i and j. So fix $i > 0$ and suppose the claim for each $j' < j$. Hence for some γ we have $\tau = \gamma\widehat{\ }(j-1, k', u) \subset TP$ (for some k'), or $\tau = \gamma\widehat{\ }(j-1, f) \subset TP$. By the construction of the priority tree, there are no nodes μ below τ on the priority tree with $i(\mu) = j'$ for any $j' \leq j-1$. Also by the priority tree construction, we see $i(\tau) = j$. By definition of TP_s, since $\tau \subset TP$, after some stage s_0, $\tau \leq_L TP_s$ for $s \geq s_0$, and there are infinitely many τ-stages.

Now either $\tau\widehat{\ }(j, f) \subset TP$, or $\tau\widehat{\ }(j, \infty) \subset TP$. In the former case, we are done. In the latter, we know that there are infinitely many τ-expansionary stages. However, we *also* know that j is *not* a witness for "$e \in Comp$." Hence $\text{liminf}_s \ell(e, j, s) = k$ for some k.

Then we see that we will infinitely often invoke Case 2 of Definition 2.3 with $x = k$ at $\tau\widehat{\ }(j, \infty)$-stages, since infinitely often between τ-expansionary stages, the length of agreement will drop to k. Since this cannot happen for $k' < k$, lest $\text{liminf}_s \ell(e, j, s) < k$, it follows that $\tau\widehat{\ }(j, \infty)\widehat{\ }(j, 0, \infty)\widehat{\ }...\widehat{\ }(j, k-1, \infty)\widehat{\ }(j, k, u) \subset TP$, giving the result. $\square$

Lemma 2.5 *Suppose that $e \notin Comp$.*
(i) Then for all $i \in \omega$, there is a node $\tau \subset TP$ with $i(\tau) = i$, and such that either $\tau\widehat{\ }(i, f) \subset TP$, or there is a σ, k such that $\tau\widehat{\ }(i, \infty)\widehat{\ }\sigma\widehat{\ }(i, k, u) \subset TP$.
(ii) For each m, n there is a node $\gamma \subset TP$ with $R_{\langle e,m,n\rangle}$ attached to γ. That is R_γ is $R_{\langle e,m,n\rangle}$.
(iii) All the $R_{\langle e,m,n\rangle}$ have versions $R_\gamma \subset TP$, and are met.
(iv) $V_e \notin FPT$.

Proof. (i) Follows by Lemma 2.4, since no i can witness "$e \in Comp$".
In Definition 2.1, Step 2, we attach $R_{\langle e,m,n\rangle}$ in order down paths, to nodes of the form $\sigma\widehat{\ }(i, f)$ or $\sigma\widehat{\ }(i, k, u)$. By (i) there are infinitely many such nodes on the true path, and hence all such $R_{\langle e,m,n\rangle}$ become attached to some node on TP. Thus we get (ii).
For (iii), let R_σ be the version of $R_{\langle e,m,n\rangle}$ on TP. Then by definition of TP_s, there are infinitely many σ-stages. At every such σ-stage, $R_{\langle e,m,n\rangle}$ can possibly receive attention. Now we can argue as in the basic module.

Since $\alpha \subset TP$, by Lemma 2.3, $slice(\alpha, s)$ has a limit, say $slice(\alpha)$. If R_α receives attention infinitely often, then it can only be that $p(s) =$

$\varphi_{m,s}(slice(\alpha), t)$ assumes infinitely many values. (Only unused values $p(s)$ can trigger receipt of attention, and they can only be used once. See Definition 2.3.) This means $\varphi_m(slice(\alpha), t)$ has no limit.

Otherwise, R_α receives attention only finitely often. As a consequence, either R_α is satisfied by diagonalization, since in the construction we set

$$V(\langle 1^s, slice(\alpha, s)\rangle) = 1 - \Phi_m(\emptyset; \langle 1^s, slice(\alpha, s)\rangle),$$

or each time R_α receives attention, the relevant computation does not halt in the requisite number of steps.

Now we see that (iv) follows since we meet all the $R_{\langle e,m,n\rangle}$ somewhere on the TP, and hence there is no witness to $V \in FPT$. $\square$

Lemma 2.6 *Suppose that $e \in Comp$. Let i be least such that $\Phi_i(W_e) = K$. Let $\tau^+ \subset TP$, with $\tau = \tau^{+\frown}(i-1, f) \subset TP$, or $\tau = \tau^{+\frown}(i-1, k, u) \subset TP$. (Here we are using Lemma 2.4.) Then*
(i) $\tau^\frown(i, \infty) \subset TP$, and for all $x \in \omega$,
(ii) $\tau^\frown\infty(i, 0, \infty)^\frown...^\frown(i, x, \infty) \subset TP$.

Proof (i) is immediate, since there are infinitely many τ-expansionary stages. For (ii), let s_0 be a stage such that for all $s > s_0$, $\tau^+ \leq_s TP_s$. Let $s_1 > s_0$ be a τ expansionary stage where $\ell(\tau, s_1) > x$ via correct computations. That is $\Phi_{i,s_1}(W_{e,s_1}; x') = K_{s_1}(x')$ for all $x' \leq x$ via the final computations. The $\ell(\tau, s)$ can never drop below x again, and hence if s_2 is the $\tau^\frown(i, \infty)$ stage after s_1, we see that for all stages $t > s_2$, $\tau^\frown\infty(i, 0, \infty)^\frown...^\frown(i, x, \infty) \leq_s TP_s$. Furthermore, every $\tau^\frown(i, \infty)$-stage, will also be a $\tau^\frown\infty(i, 0, \infty)^\frown...^\frown(i, x, \infty)$-stage, and hence $\tau^\frown\infty(i, 0, \infty)^\frown...^\frown(i, x, \infty) \subset TP$ as required. $\square$

Lemma 2.7 *Suppose that $e \in Comp$. Let i be least such that $\Phi_i(W_e) = K$. Let τ be as in Lemma 2.6. Let $\sigma \not\leq_L \tau$. The either σ has no $R_{\langle e,m,n\rangle}$ attached (because it is of the wrong type), or $slice(\sigma, s) \to \infty$, or σ extends τ but is strictly left of TP and extends $\tau^\frown\infty(i, 0, \infty)^\frown...^\frown(i, x-1, \infty)^\frown(i, x, u)$ for some x.*

Proof Since initialization occurs at σ whenever $\sigma \not\leq_L TP_s$, resetting $slice(\sigma, s)$ larger, the lemma follows for any node σ with $TP <_L \sigma$. By Lemma 2.6, if

$\tau \subset \sigma \subset TP$, then no $R_{\langle e,m,n\rangle}$ is ever attached to σ. Thus, the only other case is that σ is strictly left of TP, and extends τ. This can only happen if there is some x with $\tau\hat{}\infty(i,0,\infty)\hat{}...\hat{}(i,x-1,\infty)\hat{}(i,x,u) \subseteq \sigma$, and such nodes are visited only finitely often. $\square$

Lemma 2.8 *Suppose that $e \in Comp$. Then $V \in FPT(\leq_T^u)$.*

Proof Now if $e \in Comp$ then we know that there is some (least) i such that i is the witness to the fact that $e \in Comp$. Let τ be the top node on TP with $i(\tau) = i$ as in Lemma 2.7. We show that $V \in FPT(\leq_T^u)$ as follows.

Let s_0 be a stage beyond which we are never left of $\tau\hat{}(i,\infty)$. Since we ensure that at most finitely many elements ever get into any slice, we can by fiat write in all the slices associated with nodes left or above $\tau\hat{}(i,\infty)$. We consider slices V_y for y beyond such slices. Let $x > y$ and $s \geq s_0$. We know that for all nodes γ with $\gamma \not<_L \tau\hat{}(i,0,\infty)\hat{}(i,1,\infty)\hat{}...\hat{}(i,x,\infty)$, either γ is above or left of τ and so we can suppose that all elements of $slice(\gamma,s)$ to ever enter V have already done so by stage s_0, or we know that $slice(\gamma,s') > x > y$ for some $s' \geq s$. The procedure to decide if $\langle v,y\rangle \in V$ is that $\langle v,y\rangle \in V$ iff $\langle v,y\rangle \in V$ by stage t where $t = t(y)$ is a constant computed by a $\mathbf{0}'$-oracle.

$t(y)$ is approximated by the limit lemma, and the computable approximation $t(y,s)$ is computed from TP_s. Each time we play

$$\tau\hat{}(i,\infty)\hat{}(i,0,\infty)\hat{}(i,1,\infty)\hat{}...\hat{}(i,x,\infty)$$

immediately after playing

$$\tau\hat{}(i,0,\infty)\hat{}(i,1,\infty)\hat{}...\hat{}(i-1,x,\infty)\hat{}(i,x,\infty),$$

or playing

$$\tau\hat{}(i,0,\infty)\hat{}(i,1,\infty)\hat{}...\hat{}(i,z,u)$$

for any $z \leq x$, (that is,

$$TP_{s'} <_L \tau\hat{}(i,\infty)\hat{}(i,0,\infty)\hat{}(i,1,\infty)\hat{}...\hat{}(i,x,\infty),$$

so the current approximation looks wrong) we reset $t(y,s)$ to be the the amount of time needed to emulate the construction up to the current stage number. Because of the choice of i we know that $\lim_s(t(x), s)$ exists and hence we have $t(y) \leq_T \emptyset'$. Finally note that the choice of t means that the construction is respected, in the sense that we have enough time to emulate that stage of the construction, and hence the answers must be in agreement with the construction. That is, given some $\langle v, y\rangle$, if y is devoted to some R_σ right of TP, (which can be determined from τ), then we can simply wait for y to no longer be devoted to this, and hence any other, R_σ. The $\langle v, y\rangle \in V$ iff it has gotten in by now. Otherwise, assuming that $\langle v, y\rangle$ is not associated with R_γ in the parameter nodes γ left or above τ, we can only have $\langle v, y\rangle$ associated with some R_σ with $\tau\hat{}(i, 0, \infty)\hat{}(i, 1, \infty)\hat{}...\hat{}(i, z, u) \subset \sigma$, for some $z < y$. ($z < y$ by the the monotone attachment scheme.) Now $\mathbf{0}'$ can compute $t(y)$, and note that after stage t we are never strictly left of $\tau\hat{}(i, \infty)\hat{}(i, 0, \infty)\hat{}(i, 1, \infty)\hat{}...\hat{}(i, x, \infty)$ again. Furthermore t will be a $\tau\hat{}(i, \infty)\hat{}(i, 0, \infty)\hat{}(i, 1, \infty)\hat{}...\hat{}(i, x, \infty)$-stage. It follows that $\langle v, y\rangle \in V$ iff $\langle v, y\rangle \in V_t$. Thus $V \in FPT(\leq_T^u)$ with constant t given by the limit lemma, and the constant time algorithm. $\square$

This completes the proof in the case that $A = \emptyset$. To complete the proof note that the one can replace the oracle by A and the diagonalization similarly to get the relativized version, as in the $\leq_T^s$ case. The crucial point here is that there is a most one element per row ever put into V. $\square$

We now turn to the proof of Theorem 1.2 (ii) for the nonuniform case. Thus we will be showing that

$$\{e : W_e \text{ computable and } \leq_q^n \emptyset\}$$

is also Σ_4^0 complete. Actually, the proof is a mild variation of the proof above.

Thus for each e we will again build an auxiliary computable set V_e to try to meet the requirements below.

R_e: $V_e \in FPT(\leq_u^m)$ iff $e \in Comp$.

We modify the tasks of the the previous construction and now we will meet the following requirements.

$R:$ Either $e \in Comp$ and $V \in FPT$ via some witness i
(Here, of course, by witness i we mean that $\Phi_i(W_e) = K$.)
or $\forall m, n R_{\langle e,m,n \rangle}$, where

$R_{\langle e,m,n \rangle}$: Either there exists an m', n' such that
$lim_s \varphi_m(\langle e, m', n' \rangle, s) =_{\text{def}} \varphi_m(e, m', n')$ fails to exist, or
there exist p, w, with
$\Phi_{\varphi_m(w)}(\emptyset, \langle p, w \rangle) \neq V(\langle p, w \rangle)$, or
it does not run in time $\leq \varphi_m(w)|p|^n$.

Notice that in the above, this time we have replaced the uniform procedure Φ_m by the procedure $\Phi_{\varphi_m(w)}$ with approximation given by a binary partial computable function $\varphi_m(\cdot, \cdot)$ since in the nonuniform case we will need to guess the procedure as well as both the use and constant which all will filter through the limit lemma. (Again, we will assume that such φ_m are monotone in both variables.)

The skeptical reader, or the one unfamiliar with [DF93, CD94], might be dubious that all this can be computed in $\mathbf{0}'$. From [DF93] we have the following.

Lemma 2.9 (Downey and Fellows [DF93]) *Suppose that $A \leq_T^n B$ with A and B computable. Then there exists an c.e. function f*[2] *such that $A \leq_T^n B$ via f in the sense that there is a constant c such that for all k, x,*

$$\Phi(B^{(f(k))}(\langle x, k \rangle) = A(\langle x, k \rangle) \text{ in time } \leq f(k)|x|^c.$$

.

Proof Suppose that A and B are computable and $A \leq_T^n B$. Then there is a function g and a constant c so that for all k, z

$$\langle z, k \rangle \in A \text{ iff } \Phi_{g(k)}(B^{(g(k))}; \langle z, k \rangle) = 1 \text{ and runs in time } \leq g(k)|z|^c).$$

Here we can take $g(k)$ to be the same in all three places there are constants depending upon k by maximizing and padding the index of the procedure, if necessary. We claim that $\mathbf{0}'$ can compute a value that works in

[2]Here by c.e. function, we mean one with a computable approximation $\widehat{f}(x, s)$ given by the limit lemma, monotone in both variables.

place of $g(k)$ in the above. That is for each k, $\mathbf{0}'$ can compute $m = m(k)$ satisfying the excpression with m in place of g. The reason is that the expression in the scope of the universal quantifier is computable and hence the whole expression is $\leq_T \emptyset'$. (For the reader who has forgotten this sort of thing, we briefly remind them that for each pair $\langle n, k\rangle$ we can enumerate a partial recursive function $\psi_{\langle n,k\rangle} = \phi_{h(n,k)}$ whose index $h(n, k)$ is given by the s-m-n theorem with $dom\psi_{\langle n,k\rangle}$ equal to ω if there is some z with $\langle z, k\rangle \notin A$ but $\Phi_n(B^{(n)}; \langle z, k\rangle) = 1$, or $\langle z, k\rangle \in A$ and $\Phi_n(B^{(n)}; \langle z, k\rangle) = 0$, or $\Phi_n(B^{(n)}; \langle z, k\rangle)$ not running in time $n|z|^c$; and we have $\psi_{\langle n,k\rangle}$ the empty function otherwise. Now $\emptyset'$ can decide if $\langle h(n, k), h(n, k)\rangle \in \emptyset'$ and hence can compute the least n such that $dom\psi_{\langle n,k\rangle} = \emptyset$. For such an n we have that

$$A_k = \Phi_n(B^{(n)}) \text{ in running time } n|z|^c.$$

Now it is clear that we can take such an $n(k)$ via a function where values only increase and hence we can take an f to perform the role of g that is c.e.. $\square$.

The satisfaction of the new $R_{\langle e,m,n\rangle}$ is similar to that of the $\leq_T^u$ case, and runs with the *same* the external control driven by the previous Priority Tree.

The basic strategy used to meet the $R_{\langle e,m,n\rangle}$ this time is the following.

At each stage s, if $R_{\langle e,m,n\rangle}$ is not as yet declared satisfied and $\langle e, m, n\rangle \leq s$ find the least unused $p \leq s$,if any, such that $p = \varphi_{m,s}(\langle e, m, n\rangle, t)\downarrow$ for some $t \leq s$. If either $\varphi_{m,s}(\langle e, m, n\rangle, t)\uparrow$ for all $t \leq s$ or there is no unused p do nothing. If p, and hence t, exist declare p as used. Now compute ps^n steps in the computation of $\Phi_p(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle)$. (Note: this is essentially the only change here, "Φ_p" replaces "Φ_m".) If this computation does not halt, do nothing else. If $\Phi_p(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle)$ halts in $\leq ps^n$ steps, we try to win win by setting

$$V_e(\langle 1^s, \langle e, m, n\rangle\rangle) = 1 - \Phi_p(\emptyset; \langle 1^s, \langle e, m, n\rangle\rangle).$$

The key difference here is that in the preceding construction, at this stage we would declare $R_{\langle e,m,n\rangle}$ as satisfied and end its effect on a slice. However, now the opponent can change the index from p to some other $\varphi_{m,s'}(\langle e, m, n\rangle, t')$ for some $t' > t$, and $s' > s$. Because of this, the requirement has a portentially infinitary *positive* action. That is because when a new index is given we might need to again enumerate some $\langle 1^{s'}, \langle e, m, n\rangle\rangle$ into V. We need a little care here, since should $e \in Comp$ we still need to make sure that $V \in FPT(\leq_T^n)$. The reason we need some care is that there will be a finite number of rows corresponding to nodes on TP but above the top node τ as in the previous Lemmas 2.6 and 2.7. We really only need to make sure that they are each polynomial time for this to work.

The little modification we need is the following. When we see a new value $p(s) = \varphi_{m,s}(\langle e, m, n\rangle, t)$ we do not attend the requirement until a stage $u > s$ occurs where $u^{n+1} > pu^n$. Of course such a stage must occur and we can check that it is true, in time u^{c+1}.

The advantage of this is that we can decide for this row if $\langle 1^d, \langle e, m, n\rangle\rangle \in V$ in time essentially d^{c+1}.

Now modulo these modifications we do the construction virtually exactly as in the previous one. The proofs of the Lemmas are almost identical. (The astute reader would note that we actually prove that if $e \in Comp$ then $V \in FPT(\leq_T^u)$ (rather than simply $FPT(\leq_T^n)$) still, in fact, by essentially the same proof, but using the parameters on the finite number of infinite rows. To get the weaker $V \in FPT(\leq_T^n)$ is even easier than the previous argument, since any language V with a finite number of slices in polynomial time, and only finitely elements in each other slice is *automatically* in $FPT(\leq_T^n)$.) This concludes our discussion as to how to modify the proof in the nonuniform case. $\square$

Before we turn to the index set $\{e : W_e \text{ computable and } \leq_q^n A\}$, we note that Theorem 1.2 has certain consequences concerning the collection of computable sets $\leq_q^{\{s,u,n\}} A$ for a fixed A. As Ambos-Spies [AS85] first explicitly observed, for any of the classical reducibilities $\leq_q^p$, the languages $\{L : L \leq_q^p A\}$ for a fixed A are computably presentable. Here a set of languages $\mathcal{C}$ is called *computably presentable* iff there is a computable collection $\widehat{\mathcal{C}}$ of languages $\{U_e : e \in \omega\}$ such that for all $L \in \mathcal{C}$, there is a $U_i \in \widehat{\mathcal{C}}$ such that $L = U_i$, and conversely. The Cholak-Downey Theorem, Theorem 1.1, has the following corollary.

Corollary 2.10 *The collection $\mathcal{C} = \{W_e \leq_q^s A\}$ (for A computable) is computably presentable.*

Proof We can effectively enumerate 5-tuples $\langle e, i, c, j, k\rangle$ corresponding to two computable enumerable languages W_e, W_i, a constant $c \in \omega$, a partial

computable function φ_j, and a partial computable procedure (here we assume $q = T$), Φ_k. We then let $U^t_{\langle e,i,c,j,k\rangle} = W_{e,s}$ for the largest $s \leq t$ for which for all $k \leq s$
(i) $\varphi_s(k)\downarrow$
(ii) for all $x \leq s$, $\langle x, k\rangle \in W_{e,s}$ iff $\langle x, k\rangle \notin W_{i,s}$ and
(iii) for all $z, k \leq s$, $W_{e,s}(\langle z, k\rangle) = \Phi_{k,s}(A; \langle z, k\rangle)\downarrow)$ with use bounded by $\varphi_i(k)$ and time bounded by $\varphi_i(k)|z|^c$.

(There are many ways to do this.) The point is that either U_n will be finite because something is wrong, or the 5-tuple truly gives witnesses to the computability of W_e and the relevant reductions and constants reducing W_e to A. The collection $\widehat{\mathcal{C}} = \{U_n : n \in \omega\}$ is a computable presentation of $\mathcal{C}$. $\square$

On the other hand, suppose that, for instance, $\mathcal{C} = \{W_e \leq^u_T A\}$ was computably presentable, say, by $\widehat{\mathcal{C}} = \{U_n : n \in \omega\}$. Then we know that for each e,

$$W_e \in \mathcal{C} \text{ iff } \exists n(U_n = W_e).$$

We look at the complexity of "$U_n = W_e$". We note that $U_n = W_e$ happens iff for all z, $z \in U_n$ iff $z \in W_e$. That is, for all z, either for all s, $z \notin W_{e,s} \cup U_{n,s}$ or there is an s such that $z \in W_{e,s} \cap U_{n,s}$. That is, $W_e \in \widehat{\mathcal{C}}$ iff

$$\exists n \forall z \forall s \exists t[z \in W_{e,s} \cup U_{n,s} \rightarrow z \in W_{e,t} \cap U_{n,t}].$$

This last equation is Σ^0_3. Thus we see the following.

Corollary 2.11 *The collections* $\{e : W_e \leq^u_q A\}$, $\{e : W_e \equiv^u_q A\}$ $\{e : W_e$ *computable and* $\leq^n_q \emptyset\}$ *are are not computably presentable.*

Proof If they were computably presentable, the index set would have a Σ^0_3 definition, as we pointed out in the paragraph above. $\square$

3 The index sets $\{e : W_e$ computable and $\leq_q^n A\}$ $\{e : W_e$ computable and $\equiv_q^n A\}$ for A computable

It would seem that for a computable language A, the index sets $\{e : W_e$ computable and $\equiv_q^n A\}$ and $\{e : W_e$ computable and $\leq_q^n A\}$ would be at least as complex as $\{e : W_e$ computable and $\equiv_q^n \emptyset\}$. However, as we will now see, this is not always the case. It is instructive to understand where the proof outlined at the end of the last section breaks down.

In both cases, should $e \notin Comp$, there will be some top τ below which any diagonalization activity is either right of the true path, or can be figured out by $\emptyset'$. However there will be a finite number of nodes $\sigma \subset \tau$ on the true path that get to act. This action is used as a parameter in claiming that $V \in FPT$. In the uniform case, even if $A \neq \emptyset$, we know that there will be at most one element per row. This is just a parameter. In the nonuniform case, with $A = \emptyset$, this will not be the case for such σ on TP above τ. However, since A is the emptyset, we use the device of making sure that each such σ-slice is polynomial time. If A has the property that A is above every language where the slices are in P, then *no* diagonalization which can potentially succeed can have the this property. The problem is that each time $\varphi_m(k,t)$ increases, it means that the A-use increases to even more rows. This does not matter it $A = \emptyset$ since we know the answer! However, each successive row of A could be ever more complex. We remark in passing that for the reasons described in the current paragraph, we can salvage the following.

Theorem 3.1 *Suppose the A is computable. Let $\widehat{A} = \{\langle x, 0\rangle : x \in A\}$. Then the index set $\{e : W_e$ computable and $\leq_q^n A\}$ is Σ_4^0 complete.*

However, we now see that not all of the relevant index sets are Σ_4^0 complete. To prove the following we essentially exploit the failure of the strategy above. Let L_k be the canonical computable language complete for $DTIME(n^k)$. For instance, we can let M_e be the e-th machine running in time $n^k + e$, and put $\langle x, e\rangle$ into L_k iff M_e accepts x. Let X be the parameterized language

$$\langle z, k\rangle \in X \text{ iff } z \in L_k.$$

Then, the notation of Downey-Fellows [DF99], Chapter 15, this is a language complete for the class XP, consisting of languages V with the property that every slice if in P. (That is, "SLICEWISE P".)

Theorem 3.2 *The index sets $I = \{e : W_e$ computable and $\leq_q^n X\}$ and $J = \{e : W_e$ computable and $\equiv_q^n X\}$ is in Δ_4^0.*

We remark that in our monograph [DF99], Theorem 19.23 we claimed that the index sets above we always Σ_4^0 complete. This error is corrected here.

Proof We analyse the complexity of being in XP. Note that $V \in I$ iff V is computable (which is Σ_3^0) and $\forall k \exists \widehat{k} \exists e \forall x(\langle x, k\rangle \in V$ iff $\langle\langle x, e\rangle, \widehat{k}\rangle \in X)$. (This is Π_3^0).

In the J case, $V \in J$ iff $V \in I$ and

$$\forall k \exists \widehat{k} \exists f \forall x(\langle x, k \in X \text{ iff } \Phi_f(X^{(f)}; \langle x, k\rangle) \text{ accepts in time } \leq f|x|).$$

□

The crucial difference in this case is that the relevant class contains a complete language under some *fixed* exponent reducibility. We finish by proving that the index set in question is always fairly complicated.

Theorem 3.3 *For any computable A, the index sets $I = \{e : W_e$ computable and $\leq_q^n A\}$ and $J = \{e : W_e$ computable and $\equiv_q^n X\}$ are always Π_3^0 hard.*

Proof We begin with the argument for I. The argument is reminiscent of Theorem 1.2, but easier. Let Pi be a Π_3^0 complete language. Hence $e \in P$ iff $\forall x \exists s \forall t(Q(x, s, t, e)$. Again we use a priority tree to build language V_e. We meet the requirements below. (Here we concentrate upon $\leq_m^n$, the others are similar.)

R_e: $V_e \leq_m^n A$ iff $e \in Pi$.

Again we must meet the requirements

$R:$ Either $e \in Pi$ and $V \leq_m^n A$
or $\forall m, n R_{\langle e,m,n \rangle}$, where

$R_{\langle e,m,n\rangle}$: Either there exists an m', n' such that
$lim_s \varphi_m(\langle e, m', n'\rangle, s) =_{\text{def}} \varphi_m(e, m', n')$ fails to exist, or
there exist p, w, with
$\Phi_{\varphi_m(w)}(\emptyset, \langle p, w\rangle) \neq V(\langle p, w\rangle)$, or
it does not run in time $\leq \varphi_m(w)|p|^n$.

The basic strategy is identical to that of Theorem 1.2. Onle the control device changes. This time we use the tree (or in fact a subtree of) $\{\infty, f\}^{<\omega}$. We devote level x to $Q(x, \cdot, \cdot, e)$, or $Q(x, \cdot, \cdot)$ dropping the e. The top of the tree is thus devoted to solving "is 0 a witness to $e \notin Pi$?" We will play the Π_2 outcome ∞ (which is left of f) of 0 for the first time when we see $Q(x, 0, t)$ hold for some t. We play it again when we see $Q(x, 1, t')$ also hold for some t', etc. Each time we see the Π_2 outcome of 0 hold we initialize all nodes to the right. We define Π_2 activity for other levels x similarly.

We will make the tree have no nodes below an infinite outcome, since that will correspond to $e \notin Pi$. We will attach requitement R_j to infinitely many rows of V. *But,* we only let R act on level j provided we are at a stage s where we play $\sigma\widehat{\ }\infty$ for some σ with length $\leq j$. This means that if $e \in Pi$ then every slice is finite, and hence $V_e \in FPT$. Otherwise, there is some x witnessing $e \notin Pi$ and hence every row of length greater than or equal to x will get infitely many chances to act, and hence all R_j will be met.

Finally, we look at the case for $J = \{e : W_e$ computable and $\equiv_q^n A\}$. The only difference is to attach coding requirements to the ∞ outcomes. That is, for each $\sigma\widehat{\ }\infty$ on the priority tree, we will try to code A into V. For instance, if $|\sigma| = n$ then on the k-th time we visit $\sigma\widehat{\ }\infty$ we can code the first k elements of the first k slices of A as, say, the n-th to $n+k$-th slices of V. (All you really need to ensure is that, for instance, for all k the k-th slixe of A is almost equal to the $n+k$-th slice of V.) $\square$

Note that there are sets that are Π_3^0 but not Σ_3^0. Therefore we see the following corollary.

Corollary 3.4 *Let A be computable. The the index sets $\{e : W_e$ computable and $\equiv_q^n$*

$A\}$ and $\{e : W_e$ computable and $\leq_q^n A\}$ are not computably presentable.

References

[ADF95] K.A. Abrahamson, R.G. Downey and M.R. Fellows. Fixed parameter tractability and completeness IV: on completeness for W[P] and PSPACE analogs. *Annals of Pure and Applied Logic* 73 (1995), 235–276.

[AS85] K. Ambos-Spies, *On the Structure of Polynomial Time Degrees of Recursive Sets,* Habilitionsschrift, Universität Dortmund, 1985.

[BDG] J. Balcazaar,J. Diaz, and J. Gabarro, *Structural Complexity*, Volumes 1 and 2, Springer Verlag (1987,1989)

[BDFHW95] H. Bodlaender, R.G. Downey, M.R. Fellows, M. Hallett and H.T. Wareham. Parameterized complexity analysis in computational biology. *Computer Applications in the Biosciences* 11 (1995), 49–57.

[BDFW94] H. Bodlaender, R.G. Downey, M.R. Fellows and H. T. Wareham. The parameterized complexity of sequence alignment and consensus. *Proceedings of the Fifth Symposium on Combinatorial Pattern Matching (CPM)*, Springer-Verlag, Lecture Notes in Computer Science vol. 807 (1994), 15–30.

[BDFW95] H. Bodlaender, R.G. Downey, M.R. Fellows and H.T. Wareham. The parameterized complexity of the longest common subsequence problem. *Theoretical Computer Science A* 147 (1995), 31–54.

[BF95] H. Bodlaender and M.R. Fellows. On the complexity of k-processor scheduling. *Operations Research Letters* 18 (1995), 93–98.

[BFH94] H. Bodlaender, M.R. Fellows and M. Hallett. Beyond NP-completeness for problems of bounded width: hardness for the W hierarchy. *Proceedings of the ACM Symposium on the Theory of Computing (STOC)* (1994), 449–458.

[BFV95] R. Balasubramanian, M.R. Fellows, and V. Raman. An improved fixed parameter algorithm for vertex cover. To appear, *Information Processing Letters.*

[CCDF94] L. Cai, J. Chen, R.G. Downey and M.R. Fellows. On the parameterized complexity of short computation and factorization. *Archive for Mathematical Logic,* Vol. 36, No 4/5 (1997), 321-338.

[Ces96] M. Cesati. Structural aspects of the parameterized complexity. Doctoral dissertation (Informatics). University of Rome, "La Sapienza", 1996.

[CW95] M. Cesati and H.T. Wareham. Parameterized complexity analysis in robot motion planning. In: *Proc. 25th IEEE Intl. Conf. on Systems, Man and Cybernetics.*

[CD94] P. Cholak and R. G. Downey. Decidability and definability for parameterized polynomial time m-reducibilities. In *Logical Methods* (ed. Crossley, Remmel, Shore, and Sweedler) Birkhauser, Boston, 1994, 194-221.

[Co87] B. Courcelle, Recognizability and second-order definability for sets of finite graphs. Technical Report I-8634, Universite de Bordeaux, 1987.

[Co90] B. Courcelle, The monadic second order theory of graphs I : recognisable sets of finite graphs, *Information and Computation,* , Vol. 85, (1990) 12-75.

[DEF93] R.G. Downey, P.A. Evans and M.R. Fellows. Parameterized learning complexity. *Proceedings of the Sixth ACM Workshop on Computational Learning Theory* (1993), 51–57.

[DF92] R. G. Downey and M. R. Fellows. Fixed parameter intractability. In *Proceedings Structure in Complexity, Seventh Annual Conference*, IEEE Publication, (1992) 36-50.

[DF93] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness III: some structural aspects of the W-hierarchy. In *Complexity Theory: Current Research* (Ed. K. Ambos-Spies, S. Homer and U. Schöning) Cambridge University Press, (1993) 166-191.

[DF95a] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: basic results. *SIAM Journal of Computing* 24 (1995), 873–921.

[DF95b] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness II: completeness for W[1]. *Theoretical Computer Science A* 141 (1995), 109–131.

[DF95c] R.G. Downey and M.R. Fellows. Parameterized computational feasibility. *Proceedings of the Second Cornell Workshop on Feasible Mathematics*, Feasible Mathematics II, P. Clote and J. Remmel (eds.), Birkhauser Boston (1995), 219–244.

[DF99] R.G. Downey and M.R. Fellows. *Paramaterized Complexity,* Monographs in Computer Science, Springer-Verlag, 1999.

[DFHKW94] R.G. Downey, M.R. Fellows, M.T. Hallett, B.M. Kapron, and H.T. Wareham. The parameterized complexity of some problems in logic and linguistics. *Proceedings Symposium on Logical Foundations of Computer Science (LFCS)*, Springer-Verlag, Lecture Notes in Computer Science vol. 813 (1994), 89–100.

[DFT96] The parameterized complexity of relational database queries and an improved characterization of $W[1]$, (with M. Fellows and U. Taylor), in *Combinatorics, Complexity and Logic,*, Proceedings of DMTCS '96, (D. Bridges, C. Calude, J. Gibbons, S. Reeves, I Witten, Eds) Springer-Verlag (1996), 194-213.

[FHW93] M.R. Fellows, M.T. Hallett and H.T. Wareham. DNA physical mapping: three ways difficult. In *Algorithms — ESA '93*, (Proceedings of the First European Symposium on Algorithms), Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 726 (1993), 157–168.

[FK93] M.R. Fellows and N. Koblitz. Fixed-parameter complexity and cryptography. *Proceedings of the Tenth International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC'93)*, Springer-Verlag, Berlin, Lecture Notes in Computer Science vol. 673 (1993), 121–131.

[Ha89] J. Hartmanis, On the importance of being Π_2 hard, *Bulletin of the European Assoc. for Theor. Comput. Sci.,* No. 37, (1989), 117-127.

[So87] R. I. Soare. Recursively Enumerable Sets and Degees. Springer Verlag, 1987.

[Ya69] C. E. M. Yates. The degrees of index sets II. Trans. Amer. Math. Soc., Vol. 135, (1969), 249-266.