The Parameterized Complexity of Relational Database Queries and an Improved Characterization of $W[1]$

Rod G. Downey[1]*
Michael R. Fellows[2] ** and Udayan Taylor[2]

[1] Department of Mathematics,
Victoria University
P.O. Box 600
Wellington, New Zealand
rod.downey@vuw.ac.nz

[2] Department of Computer Science
University of Victoria
Victoria, B.C. V8W 3P6, Canada
{mfellows,udi}@csr.uvic.ca

Abstract. It is well known that for a fixed relational database query ϕ in m free variables, it can be determined in time polynomial in the size n of the database whether there exists an m-tuple x that belongs to the relation defined by the query. For the best known algorithms, however, the exponent of the polynomial is proportional to the size of the query. We study the data complexity of this problem parameterized by the size $k = |\phi|$ of the query, and answer a question recently raised by Yannakakis [Yan95]. Our main results show: (1) the general problem is complete for the parametric complexity class $AW[*]$, and (2) when restricted to monotone queries, the problem is complete for the fundamental parametric complexity class $W[1]$. The practical significance of these results is that unless the parameterized complexity hierarchy collapses, there are unlikely to be algorithms that solve this problem (even under the restriction to monotone queries) in time $f(k)n^c$ where f is an arbitrary function of k and c is a constant independent of k.
An important consequence of the proof of (2) is a significantly improved characterization of the parameterized complexity class $W[1]$. Previous results by Downey and Fellows characterize $W[1]$ in terms of the k-Weighted Circuit Satisfiability problem, for families of circuits that satisfy: (1) the depth of the circuits is bounded by a constant c, (2) on any input-output path there is at most one gate having unbounded fan-in (termed a *large* gate), with all other gates having fan-in bounded by c (that is, *small* gates). We show that the definition can be broadened by allowing circuits of depth bounded by an arbitrary function $f(k)$. If we denote this parameterized complexity class $W^*[1]$, then our corollary

* Research supported by the New Zealand Marsden Fund for Basic Science
** Research supported by the National Science and Engineering Research Council of Canada.

is: $W^*[1] = W[1]$. As a consequence, the k-Weighted Satisfiability problem for Boolean expressions that are products of k-sums is complete for $W[1]$. Part of our argument establishes a general normalization lemma for $W^*[t]$ for all t.

1 Introduction

Many natural computational problems have input that consists of several elements of information. It is natural to consider some of these elements as a *parameter* and evaluate their relative contribution to the overall complexity of the problem. In many practical applications of computational problems only a small set of parameter values may be significant. In these cases, complexity that is exponential in the parameter, but otherwise a fixed polynomial, may be acceptable.

A framework in which to study this complexity issue has been developed in [DF95a, DF95b, DF95c, ADF95] and applied to problems in diverse areas of computer science [BDFW95, BDFHW95, BF95, BFH94, CCDF94, CW95, DEF93, DFHKW94, FK93]. 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 $c = 1$ [BFR95].) The best that is known about Dominating Set is essentially that it can be solved by brute force (trying all k-subsets) in time $O(n^{k+1})$.

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 has been shown to address many natural problems in diverse areas of computer science, and it must be regarded as one of the principal "applied" complexity theories [BDFW95, BDFHW95, BF95, BFH94, CCDF94, CW95, DEF93, DF95a, DFHKW94, FK93].

Part of the theory of parameterized complexity is concerned with general methodologies for demonstrating parametric tractability. On the other hand, as with classical complexity, if we cannot seem to be able to demonstrate parametric tractability, then we would like to prove, or at least give strong evidence for, parametric *intractability.* As we will see in the next section, to this end, Downey and Fellows developed a hierarchy of classes of parameterized languages called the W-hierarchy, and has the form

$$W[1] \subseteq W[2] \subseteq ...W[SAT] \subseteq W[P] \subseteq AW[1] = AW[*] \subseteq AW[P] \subseteq XP.$$

The parameterized complexity class $W[1]$ is particularly interesting and important, for several reasons. First of all, a demonstration of hardness for $W[1]$

can be regarded as the principal "working" reference point for likely fixed-parameter intractability for the reason that it provides a convincing "minimum" demonstration. Secondly, there is a substantial list of very natural and important problems that are precisely $W[1]$-complete, including: Clique, Independent Set, Vapnik-Chervonenkis Dimension, Monotone Data Complexity for Relational Databases (from this paper), Square Tiling, k-Step Derivation for Context Sensitive Grammars, m-Length Common Subsequence for k Sequences, k-Length Post Correspondence, and the k-Step Halting Problem for Single-Tape Nondeterministic Turing Machines (Short Turing Machine Acceptance [Ces96, DEF93, DF95b, DFHKW94, DFT96]. The last result is of particular significance since it is kind of a miniaturization of Cook's Theorem. The class $W[1]$ the the calls of parameterized languages which have the complexity of the problem below.

SHORT TURING MACHINE ACCEPTANCE
Input: A nondeterministic Turing machine M.
Parameter: A positive integer k.
Question: Does M have a computation path accepting the empty string in $\leq k$ steps?

However in the case of classical TURING MACHINE ACCEPTANCE, (i.e. the problem above with no parameter, and asking if the Turing Machine accepts in $\leq |M|^n$ steps for some fixed n) the point is that a nondeterministic Turing machine is such an opaque and generic object that it simply does not seem reasonable that we should be able to decide if a given one on a given input has some accepting path without looking at the paths. Cook's Theorem therefore gives powerful evidence that $P \neq NP$. It seems to us that if one accepts the philosophical argument that TURING MACHINE ACCEPTANCE is intractable then the same reasoning would suggest that SHORT TURING MACHINE ACCEPTANCE is *fixed parameter* intractable. Surely we would need to look at all tha paths of length k. In this way, it is natural to regard the class $W[1]$ which has the problem SHORT TURING MACHINE ACCEPTANCE complete for it, as being a hardness measure for parametric intractability.

In this paper we report on applications of this perspective to problems of database theory, a line of investigation recently proposed by Yannakakis [Yan95]. We prove the following theorems.

Theorem 1. *The* Query Non-Emptiness *problem for relational databases is complete for the parameterized complexity class $AW[*]$.*

Theorem 2. *The* Monotone Query Non-Emptiness *problem for relational databases is $W[1]$ complete.*

The main Technical Lemma establishes a significantly broadened logical characterization of $W[1]$ than that proved in [DF95b]. The importance of this result is that it makes it easier to prove "upper bounds" on the complexity of problems with respect to the W hierarchy. In the present paper we use this characterization to prove $W[1]$ membership for the emptiness problem for monotone queries to relational databases, which together with the hardness arguments there, shows that this very natural problem is $W[1]$-complete.

Previous results by Downey and Fellows characterize $W[1]$ in terms of the k-WEIGHTED CIRCUIT SATISFIABILITY problem, for families of circuits that satisfy: (1) the depth of the circuits is bounded by a constant c, (2) on any input-output path there is at most one gate having unbounded fan-in (termed a *large* gate), with all other gates having fan-in bounded by c (that is, *small* gates). We show that the definition can be broadened by allowing circuits of depth bounded by an arbitrary function $f(k)$. If we denote this parameterized complexity class $W^*[1]$.

Theorem 3. $W[1] = W^*[1]$

As a consequence, the k-WEIGHTED SATISFIABILITY problem for Boolean expressions that are products of k-sums is complete for $W[1]$. Part of our argument establishes a general normalization lemma for $W^*[t]$ for all t. We suspect that Theorem 3 can be extended to the entire $W[t]$-hierarchy ($W^*[t] = W[t]$). As a step in the direction of this conjecture, we prove an important normalization theorem for $W^*[t]$, analogous to results concerning $W[t]$ in [DF95a].

In the next section we give a brief introduction to the formal framework of parameterized complexity theory. In §3 we introduce the terminology of relational database queries and prove Theorem 1. In §4 we prove Theorem 2, assuming Theorem 3. Finally in §5 and §6 we prove the extended "normalization" theorem, and prove the crucial technical Theorem 3. The final section discusses some related open problems.

2 Background on Parameterized Complexity

The formal framework is sketched as follows. We consider that input to a computational problem consists of two parts, one of which is expected to be relatively small. Thus we consider a *parameterized language* L to be a set of pairs of strings, $L \subseteq \Sigma^* \times \Sigma^*$. For notational convenience, we may equivalently consider a parameterized language L to be a subset of $\Sigma^* \times \mathbb{N}$, where for $(x, k) \in L$, k is the *parameter*. Note that a parameterized language can always be considered "classically" as well.

The fundamental concept of the theory is the notion of *fixed-parameter tractability*. We say that a parameterized language L is *fixed parameter tractable* (FPT) if it can be determined in time $f(k)n^c$ whether $(x, k) \in L$, where c is a constant independent of the parameter k and $n = |x|$ is the size of x. There are many examples of parameterized problems that are FPT, yet when considered classically are *NP*-complete, *PSPACE*-complete or even undecidable. A corresponding notion of problem reduction is defined as follows[3].

[3] Definition 4 is not the most general notion of parametric reduction, but is sufficient for our purposes. We refer the reader to [DF95a, DFta] for more details

Definition 4. Let A, B be parameterized decision problems. We say that A is (standardly) *reducible* to B if there is an algorithm Φ which transforms (x, k) into $(x', g(k))$ in time $f(k)|x|^\alpha$, where $f, g : N \to N$ are arbitrary functions and α is a constant independent of k, so that $(x, k) \in A$ if and only if $(x', g(k)) \in B$.

It is easy to see that if A reduces to B and B is fixed parameter tractable then so too is A.

Note that if $P = NP$ then a variety of natural parameterized NP-complete problems would be fixed-parameter tractable. Thus a completeness program is reasonable for establishing apparent fixed-parameter intractability. We remark that parametric reductions are much "finer" than classical ones, and hence instead of the classical observation that "normal" problems in NP being either in P or NP complete, there tend to be a large collection of hardness classes. The definitions to follow establish the basic parametric complexity classes. While they seem a little involved the reader should keep in mind that the basic problems complete for the classes are simply analogues of the classical SATISFIABILITY problem. The parameterized version(s) of SATISFIABILITY is based upon the notion of *weight*, where the *weight* of a binary string is the number of 1's in it. We ask if a formula has a satisfying assignment of fixed weight k, which is the parameter. The w-hierarchy below is based on the belief that WEIGHTED CNF SATISFIABILITY and WEIGHTED 3SAT are of differing complexity, and the underlying reason for this differing complexity is the difference in logical depth of the formulae. Thus if CNF is thought of as products-of-sums of literals, then we can define a propositional formula to be *t-normalized* as follows.

Definition 5. We say that a propositional formula X is *t-normalized* if X is of the form products-of-sums-of-products... of literals with t-alternations.

Thus we see according to Definition 5, 2-normalized is the same as CNF. Now we believe that for all t, WEIGHTED t-NORMALIZED SATISFIABILITY, that is, satisfiability for t-normalized formulae, is strictly easier than WEIGHTED $t+1$-NORMALIZED SATISFIABILITY. This suggests a hierarchy based on the complexity of t-normalized formulae. We can consider t-normalized formula as special examples of certain classes of circuits, namely the weft t boolean circuits below.

Definition. A Boolean circuit is of *mixed type* if it consists of circuits having gates of the following kinds.

(1) *Small gates*: $\neg$ gates, $\wedge$ gates and $\vee$ gates with bounded fan-in. We will assume that the bound on fan-in is 2 for *and* gates and *or* gates, and 1 for *not* gates.

(2) *Large gates*: $\wedge$ gates and $\vee$ gates with unrestricted fan-in.

Definition. The *depth* of a circuit C is defined to be the maximum number of gates on an input-output path in C. The *weft* of a circuit C is the maximum number of large gates on an input-output path in C.

Definition. Let F be a family of decision circuits. We allow that F may have many different circuits with a given number of inputs. To F we associate the parameterized language $L_F = \{(C, k) : \exists \text{ an input vector } x \text{ of weight } k \text{ accepted by } C\}$.

Definition. A parameterized language L belongs to $W[t]$ if L reduces to $L_{F(t,h)}$ for the family $F(t,h)$ of mixed type decision circuits of weft at most t, and depth at most h, for some constant h.

In the above definition, the weight k vector x is governed by a single existential quantification. This can be generalized.

Definition. A k-structured circuit C is a circuit with inputs partitioned into k disjoint sets $I_1, ..., I_k$, each with an associated logical quantifier $Q_i \in \{\exists, \forall\}$ for $i = 1, ..., k$.

Definition. In a manner analogous to the definition of L_F we define

$$
\begin{aligned}
L_F^A = \{(C,k) : & \ C \text{ is a } k\text{-structured circuit} \\
& Q_1 i_1 \in I_1,\ Q_2 i_2 \in I_2,\ \cdots Q_k i_k \in I_k, \text{ such that} \\
& \text{the weight } k \text{ input } x \text{ with } x_{i_1} = x_{i_2} = \cdots = x_{i_k} = 1 \text{ is accepted by } C\}
\end{aligned}
$$

Definition. A parameterized language L belongs to $AW[t]$ if L reduces to $L^A_{F(h,t)}$ for the family $F(h,t)$ of structured circuits of weft at most t and depth at most h.

This gives us the following hierarchy of parameterized complexity classes:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq AW[1] \subseteq AW[2] \subseteq \cdots$$

In [ADF95] and [DFR96] it is shown that the $AW[t]$ hierarchy collapses to $AW[1]$. In [DF95a] the basic Normalization Theorem is proven:

Basic Normalization Theorem *For all* $t \geq 1$, WEIGHTED t-NORMALIZED SATISFIABILITY *is complete for* $W[t]$.

We will improve upon the Normalization Theorem in the present paper[4].

[4] The reader who is unfamiliar with parametric complexity might wonder if there are some connections with classical notions. There are two answers to this yes and no! In some sense, the theory is *orthogonal* to classical complexity since we are dealing with parameterized languages. Thus we are *definitely not* dealing with something inside of, say NP or $PSPACE$. For instance if A is *any* language however classically difficult, the language $A^P = \{(x,x) : x \in A\}$ is FPT but clearly has the same classical complexity as A. Hoewver, the event that, say, $W[1] = FPT$ does have classical consequences. We illustrate that for $W[P]$ and refer the reader to [ADF95] for further details. $NP[f(n)]$ denotes the class of decision problems that are solvable in P-time with an algorithm using only $f(n)$ bits of nondeterminism. Abrahamson, Downey and Fellows defined the class $SUBEXPTIME(f(n))$ to be the set of languages L for which there is a polynomial $p(n)$ such that L is accepted in $DTIME(p(n)2^{g(n)})$ for some function g in $o(f(n))$. (That is, $\lim_{n\to\infty} f(n)/g(n) = \infty$.) They established the following result connecting parameterized and classical complexity.
Theorem [Abrahamson, Downey and Fellows] *(i) For the standard reducibility,* $W[P] = FPT$ *iff for every* P*-time function* f *with* $f(n) \geq \log n$*, there is a recursive function* h *such that for every* $L \in NP[f(n)]$ $h(L)$ *computes machine* M *which witnesses that* $L \in SUBEXPTIME(f(n))$*. (We say that* $NP[f(n)]$ *is constructively a subset of* $SUBEXPTIME(f(n))$*.)*
(ii) Consequently, for the standard reducibility, $W[P] = FPT$ *implies that for all* P*-time* f*,* $NP[f(n)] \subseteq SUBEXPTIME(f(n))$*.*

3 Parametric Data Complexity of Relational Queries

We assume the reader is familiar with the basic ideas of relational database theory, as for example in [Ul88].

A relational database is essentially just a finite set of finite relations on a finite set U. Each relation can be viewed as a set of tuples listed in a table for the relation. The size of a database is the length of a description of all the relations by listing the contents of the relational tables.

Formally, there is a finite set U of attributes, each with an associated domain. A *relational schema* R is a subset of U. A *database schema* $D = (R_1, ..., R_n)$ is a set of relation schemas. A relation r_i over a relation schema R_i is a set of R_i-tuples, where a R_i-tuple is a mapping from the attributes of R_i to their domains. A database d over the schema D is a set $\{r_1, ..., r_n\}$ of finite relations.

We consider here the complexity of queries formulated in the *relational calculus.* Formulas of the calculus are built from atomic formulas of the form $x_1 = x_2$ and $R_i(x_1, ..., x_m)$, where the x_i are variables or constants of the appropriate domains, using Boolean connectives and quantifiers. A formula ϕ with free variables $y_1, ..., y_k$ specifies a query relation

$$r_\phi = \{(a_1, ..., a_k) : \phi(a_1, ..., a_k) \text{ is true in } d\}$$

A formula will be said to be in *normal form* if all quantifiers are up front and the body of the formula is in conjunctive normal form. For example, the query

$$\phi = \exists a \forall b \forall c\, (R_1(a, b, d) \vee \neg R_2(a, c) \vee \neg R_1(a, c, e)) \wedge (\neg R_1(a, d, e) \vee R_2(d, e))$$

is in normal form.

Various problems may be considered concerning the complexity of computations associated with a query ϕ. The following problem is simply stated, and a number of other problems can be easily reduced to it. (For example, the problem of determining whether a target tuple x belongs to the relation defined by the query is easily reduced to the non-emptiness problem.)

Query Non-Emptiness
Input: A database d, and a query ϕ.
Parameter: $|\phi|$
Question: Is the query relation r_ϕ nonempty?
Proposition 1. Query Non-Emptiness *is in* $AW[1]$.

Proof. At the cost of a blow-up exponential in the size of ϕ we may assume that ϕ is in the normal form:

$$\phi = Q_1 x_1 Q_2 x_2 \cdots Q_r x_r E$$

where the the Q_i are logical quantifiers and E is in conjunctive normal form. Using multiplicative notation for convenience:

$$E = \prod_{i=1}^{l} \sum_{j=1}^{m} E_{ij}$$

where each E_{ij} is either a relational atomic formula or its negation. Let $y_1, ..., y_s$ denote the free variables of ϕ, and let U denote the base set over which the database d is defined, $|U| = n$.

We describe an appropriate k-structured circuit C to exhibit membership in $AW[4]$. Since the AW hierarchy collapses to $AW[1]$, this is sufficient for our purposes. The parameter k is obliged to be purely a function of ϕ. We may take $k = r + s$, the number of variables (both free and bound) occuring in ϕ.

The inputs to C are partitioned into k blocks of size n. Let J_i for $i = i, ..., r$ denote the blocks corresponding to the r bound variables of ϕ, and let J'_i for $i = 1, ..., s$ denote the blocks corresponding to the free variables of ϕ. Let

$$J_i = \{v[i, u]; u \in U\} \text{ for } i = 1, ..., r$$

and let

$$J'_i = \{w[i, u] : u \in U\} \text{ for } i = 1, ..., s$$

To each block J_i for $i = 1, ..., r$ we associate the quantifier Q_i. To each block J'_i for $i = 1, ..., s$ we associate the existential quantifier.

The circuit C for our reduction corresponds to the logical expression:

$$F = F_1 \cdot F_2 \cdot F_3$$

where

$$F_1 = \prod_{i=1}^{r} \prod_{u,u' \in U,\ u \neq u'} (\neg v[i, u] + \neg v[i, u'])$$

$$F_2 = \prod_{i=1}^{s} \prod_{u,u' \in U,\ u \neq u'} (\neg w[i, u] + \neg w[i, u'])$$

and

$$F_3 = \prod_{i=1}^{l} \sum_{j=1}^{m} F_{ij}$$

where the subexpressions F_{ij} correspond to the relational formulas (or negations of these) E_{ij} of E in the following way (which we will explain by example).

Suppose $E_{ij} = R(x_2, x_3, y_5)$. Then

$$F_{ij} = \sum_{(u,u',u'') \in R} (v[2, u] \cdot v[3, u'] \cdot w[5, u''])$$

The expression F_{ij} is thus simply a sum-of-products expression that evaluates R in the blocks corresponding to the variables occuring in the atomic relational formula (or its negation, as the case may be).

The reader can easily check that this construction works correctly, as it is little more than a notational translation. The circuit C has depth and therefore weft 4. Since $AW[4] = AW[1]$ we are done.

Proposition 2. Query Non-Emptiness *is hard for* $AW[*]$.

Proof. We reduce from the problem SHORT NODE KAYLES that is shown to be complete for $AW[*]$ in [ADF95]. This problem takes as input a graph $G = (V, E)$. The parameter is a positive integer k, and the question is whether the first player has a k-move winning strategy in the following game. Players alternate pebbling a vertex that is not adjacent to any previously pebbled vertex. The first player with no move loses.

The database d for our reduction has as a base set the vertex set of the graph, and it has two relations: (1) the binary adjacency relation E of the graph (where we consider also that a vertex is adjacent to itself), and (2) a unary padding relation $T = \{(u) : u \in V$.

It is straightforward to concoct a query that will define a non-empty relation if and only if the first player has a winning strategy. For example, the following query will serve when $k = 3$ and is easily generalized to arbitrary k.

$$\phi = \exists x_1(\forall y_1(E(x_1, y_1) \vee \exists x_2(\forall y_2(E(x_1, y_2) \vee E(x_2, y_2)\vee$$
$$\exists x_3(\forall y_3(E(x_1, y_3) \vee E(x_2, y_3) \vee E(x_3, y_3))))))) \wedge T(x)$$

The only free variable of ϕ is x. The unary relation defined by ϕ coincides with T when the first player has a 3-move winning strategy, and is otherwise empty.

Note that the size of ϕ depends only on k.

Together, Propositions 1 and 2 prove the theorem:

Theorem 1. QUERY NON-EMPTINESS *is* $AW[1]$*-complete.*

4 Monotone Queries

In this section we consider the complexity of the restriction of the QUERY NON-EMPTINESS problem to monotone queries. A query ϕ is termed *monotone* if: (1) if it has only existential quantifications, and (2) no negations.

Proposition 3. MONOTONE QUERY NON-EMPTINESS *belongs to* $W[1]$.

Proof. Use the same circuit construction as in the proof of Proposition 1. In this case, there are no blocks of input variables (as the circuit is now "ordinary" as opposed to k-structured). The subexpressions F_1 and F_2 enforce, however, that in any weight k input vector accepted by the circuit, there is exactly one input variable in each block with value 1, which is equivalent to an implicit existential quantification, as required for the correctness of the reduction from a monotone query.

An examination of the circuit C shows that on any input-output path there is at most one gate with unbounded fan-in, all other gates having fan-in bounded by a function of k. To complete the proof, we will need the technical result Theorem 3:

Theorem ($W[1]$ Characterization), Theorem 3*In the definition of* $W[1]$ *in terms of circuits, the definition of small gates can be relaxed to allow gates of fan-in* $f(k)$*, where* f *is an arbitrary function of the parameter* k.

Thus by Theorem 3, we have established membership in $W^*[1] = W[1]$.

Proposition 4. Monotone Query Non-Emptiness *is hard for* $W[1]$.

Proof. A reduction from Clique is straightforward and can be sketched as follows. Let $G = (V, E)$ be the graph for which we wish to determine whether it contains a k-clique. Let V be the base set of the database d that has binary relation R recording the edges of the graph. Order V arbitrarily (denoted $<$), and let:

$$R = \{(u, v) : u < v \text{ and } uv \in E\}$$

The database d also has a "padding" unary relation T:

$$T = \{(u) : u \in V\}$$

The query to determine whether there is a k-clique is just:

$$\phi = \exists x_1 \exists x_2 \cdots \exists x_k R(x_1, x_2) \wedge R(x_1, x_3) \wedge \cdots \wedge R(x_{k-1}, x_k) \wedge T(x)$$

Note that the size of ϕ is a function only of k.

The only free variable in ϕ is x. If there is a k-clique in G, then the unary relation R_ϕ defined by ϕ is:

$$R_\phi = \{(v) : v \in V\}$$

and if there is no k-clique in G then $R_\phi = \emptyset$. The ordering of V used in defining R implicitly forces a successful instantiation of the existentially quantified variables to represent k distinct vertices of G. Note that ϕ is monotone.

Remark. The expression of the Clique problem as a database query problem is really quite straightforward. One can easily work out a similar representation for the Dominating Set problem — but not as a monotone query.

Propositions 3 and 4 together prove:

Theorem 2. Monotone Query Non-Emptiness *is complete for* $W[1]$.

5 W^* Normalization

Now we turn to our improvements to the Normalization Theorem of [DF95a], and the proof of Theorem 3. We need a definition.

Definition 6. Fix a function f and consider a gate to be *small if it has fanin $\leq f(k)$. The definition of *small allows us, as in §2 to define a notion of *weft. Then we say that a parameterized language L belongs to $W^*[t]$ if L reduces to $L_{F^*(t,h)}$ for the family $F^*(t, h)$ of mixed type decision circuits of *weft at most t, and depth at most $h(k)$, for some arbitrary function h. (Note that if we wish to consider families of circuits where *small* gates may have fan-in $f(k)$, then we can equivalently substitute a faster-growing function h.)

It is clear that $W^*[t]$ might well be a substantially larger class of parameterized languages than $W[t]$, and that trivially $W[t] \subseteq W^*[t]$.

In this section we sketch the proof of an important lemma concerning $W^*[t]$ that says essentially that the circuits can be put in a normal form that corresponds to a certain kind of Boolean formula. The normal form is defined recursively as follows.

Definition. A Boolean expression E is in 1-alternating normal form with respect to n and k if E is either an n-product of k-sums or an n-sum of k-products. A Boolean expression E is in t-alternating normal form with respect to n and k, for $t \geq 2$, if E either is of the form:

$$E = \prod_{i=1}^{n} \sum_{j=1}^{k} E_{ij}$$

or of the form:

$$E = \sum_{i=1}^{n} \prod_{j=1}^{k} E_{ij}$$

where each sub expression E_{ij} is $(t-1)$-alternating normal with respect to n and k.

Note that the above definition can be equivalently phrased in terms of circuits in a natural way. Define a tree circuit C to be in *t-alternating normal form with respect to n and k* if it has $2t$ layers of gates, with the fan-in alternating between n and k starting with fan-in n for the output gate.

Corresponding to this form we have the following parameterized problem.

t-Alternating Weighted Satisfiability

Input: A Boolean expression E in t-alternating normal form with respect to n and k

Parameter: k

Question: Is there a weight k truth assignment that satisfies E?

Lemma (Normalization). t-Alternating Weighted Satisfiability *is complete for $W^*[t]$ for all $t \geq 1$.*

Proof. (sketch) Assume that the given circuit C has an output large gate, since an output small gate subcircuit can be removed by employing additional nondeterminism in the manner used in [DF95a]. The idea is basically to progressively analyze and "copy" the circuit C into the required form, working progressively downward from the input level. In the first step, we locate the topmost large gates. Let g denote such a gate, and let $a_1, ..., a_r$ denote the inputs to g. Suppose g is a $\wedge$ gate. Each a_i is computed by a subcircuit consisting only of small gates, and therefore there the number of inputs to C on which a_i depends is bounded by a function of k. Thus, a_i can be re-expressed (at a cost that is bounded by a function of k) as a product-of-sums of total size bounded by a function of k. Thus g can be replaced by a product-of-sums "copy-gate" g', where there are at most $f(k)n$ sums (for some function f) and each sum has size bounded by some function $g(k)$. Make a copy of g' for each fan-out line of g.

For the case where g is a large $\vee$ gate, we replace each argument a_i with an equivalent sum-of-products, to obtain (for g') an $f(k)n$-sum of $g(k)$-products.

We next identify the topmost layer of large gates *below* the inputs to C and the copy gates created so far. Note that each copy gate has fan-out 1. We repeat the above recipe, creating two more layers of copy gates ($f'(k)n$-products of $g'(k)$-sums and $f'(k)$-sums of $g'(k)$-products, for new functions f' and g'). Eventually the entire circuit C is replaced by the copy gates, and the resulting copy gate circuit has the required form.

6 An Improved Characterization of $\boldsymbol{W}[1]$

In this section we prove the main technical result, Theorem 3.
Theorem 1. $W^*[1] = W[1]$.

Proof. By the Normalization Lemma, we just have to show that 1-Alternating Weighted Satisfiability is in $W[1]$. The only issue is for expressions that are n-products of k-sums, since the case of n-sums of k-products is in FPT. Let E denote the Boolean expression. We view E as consisting of a set $\mathcal{C}$ of clauses

$$\mathcal{C} = \{C_1, ..., C_n\}$$

where each clause is a set of k literals. Let X denote the set of variables of E,

$$X = \{x_1, \ldots, x_m\}$$

Our argument consists of a description of how to produce a Boolean expression E' over a set of variables X' that is a product of sums of size bounded by a constant, such that E' is satisfiable by a weight $k' = k + 2^{k+1}$ truth assignment if and only if E is satisfiable by a weight k truth assignment. By the results of [DF95b] this is sufficient to conclude the theorem.

Let $X_i \subseteq X$ denote the set of variables occurring in the clause C_i. Let X_i^+ denote the variables that occur positively in C_i and let X_i^- denote the set of variables that occur negated in C_i. (Below we will make the trivial assumption that $X_i^+ \cap X_i^- = \emptyset$).

Suppose $X_i^- = \{x_1, x_3, x_5, x_8\}$ The sequence (family) of increasingly larger sets beginning with the empty set, and progressively including the variable of least index among those remaining, we will call the i^{th} *trail* $\mathcal{T}_i$. Thus for our example X_i^-, we would have

$$\mathcal{T}_i = \{\emptyset, \{x_1\}, \{x_1, x_3\}, \{x_1, x_3, x_5\}, \{x_1, x_3, x_5, x_8\}\}$$

Let $\mathcal{T}$ be the union of the trails

$$\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$$

For $T \in \mathcal{T}$ the set X_T of *successors* of T is the set of variables $x = x_j$ such that:
(1) $x_j \notin T$,

(2) for all $x_i \in T$, $j > i$, and
(3) $T' = T \cup \{x_j\} \in \mathcal{T}$.
Thus if x is a successor of T, then it will have the largest index in T'. Note also that the size of $\mathcal{T}$ is bounded by a quadratic polynomial in the size of E.

If $T \in \mathcal{T}$ then since T is a set of indexed variables, T has an ordering inherited from the indices. We will write $T(s)$ to denote the s^{th} variable of T in this ordering.

For each $T \in \mathcal{T}$ define $\mathcal{C}_T$ to be the set of clauses for which:
(1) the set of negated variables in the clause is contained in T, and
(2) the set of positive variables of the clause is disjoint from T.
Formally:

$$\mathcal{C}_T = \{C_i \in \mathcal{C} : X_i^- \subseteq T \text{ and } X_i^+ \cap T = \emptyset\}$$

For example, $\mathcal{C}_\emptyset$ is the set of monotone clauses of E.

For each set of clauses $\mathcal{C}_T$ for $T \in \mathcal{T}$ we compute a rooted labeled tree Ω_T as follows. Each node of the tree is labeled by a pair $(\mathcal{A}, V)$ where $\mathcal{A}$ is a subset of $\mathcal{C}_T$, and V is a set of variables. The root of Ω_T is labeled by the pair $(\mathcal{C}_T, \emptyset)$. The tree is generated by beginning at the root and progressively expanding nodes, creating children in the following way.

In order for a node labeled $(\mathcal{A}, V)$ to be expandable, there are two conditions that must be met:
(1) $\mathcal{A}$ must be nonempty, and
(2) $|V| \leq k$.

The expansion of a node labeled $(\mathcal{A}, V)$ is achieved as follows. Suppose that the least indexed clause in $\mathcal{A}$ is C_j. One child is created for each variable x_s in X_j^+, that is, for each positive variable of the clause C_j. The child node is labeled $(\mathcal{A}', V')$ where

$$V' = V \cup \{x_s\})$$

and

$$\mathcal{A}' = \{C_i \in \mathcal{A} : X_i^+ \cap V' = \emptyset\}$$

The tree Ω_T is created by beginning with the root node, and expanding nodes of the tree until no more expansion is possible. Note that by condition (2), the height of Ω_T is at most k, and since the clauses of E have size k, the branching factor of Ω_T is also bounded by k. Consequently there are at most k^k leaves.

A leaf of Ω_T is *interesting* if it is labeled by a pair $(\mathcal{A}, V)$ with $\mathcal{A} = \emptyset$. Let $\mathcal{L}_T$ denote the family of sets of variables V that occur in the labels of interesting leaves of Ω_T:

$$\mathcal{L}_T = \{V : (\emptyset, V) \text{ is the label of an interesting leaf of } \Omega_T\}$$

Let $\mathcal{L}$ denote the union of the $\mathcal{L}_T$:

$$\mathcal{L} = \bigcup_{T \in \mathcal{T}} \mathcal{L}_T$$

The usefulness of the sets $\mathcal{L}_T$ is illustrated by the special case of $T = \emptyset$. In this case $\mathcal{C}_T$ is the set of monotone clauses of E, that is, the clauses having only positive literals. The following claim is easy to verify from the construction of $\Omega_\emptyset$.

Claim 1. E is satisfiable by a weight k truth assignment only if $\exists L \in \mathcal{L}_\emptyset$ such that all of the variables in L are assigned the value *true*.

For an arbitrary $T \in \mathcal{T}$, the set of clauses $\mathcal{C}_T$ must be satisfied "positively" in any satisfying truth assignment for E that assigns all of the variables of T to be *true*. By the construction of Ω_T the following claim holds.

Claim 2. The clauses of $\mathcal{C}_T$ are positively satisfied by a weight k truth assignment that sets all of the variables of T to *true*, only if $\exists L \in \mathcal{L}_T$ such that all of the variables in L are assigned the value *true*.

We next describe the set of variables X', which consists of three principal parts:

$$X' = X'[1] \cup X'[2] \cup X'[3]$$

For convenience, let $[k]$ denote the set $\{1, 2, \ldots, k\}$ and let $2^{[k]}$ (as usual) denote the set of all subsets of $[k]$. If $J \in 2^{[k]}$ then J is naturally ordered. We will write $J(s)$ to denote the s^{th} element of J.

In the definition of $X'[2]$ and $X'[3]$ we employ a "dummy" index $\perp$. In order for our indexing scheme to work properly, we define

$$\mathcal{L}_\perp = \{\perp\}$$

The constituent sets of variables are:

$$X'[1] = \{a[i, x] : 1 \leq i \leq k,\ x \in X\}$$

$$X'[2] = \{b[J, T] : J \in 2^{[k]},\ T \in (\mathcal{T} \cup \{\perp\})\}$$

$$X'[3] = \{c[J, T, L, J'] : J, J' \in 2^{[k]},\ T \in (\mathcal{T} \cup \{\perp\}),\ L \in \mathcal{L}_T\}$$

The expression E' is the product of the following boolean expressions. To simplify the notation, the indices in the products implicitly range over the various important sets used to define the variables of X'. In particular: i and i' range over $[k]$; J, J' and J'' range over $2^{[k]}$; x and x' range over X; T and T' range over $\mathcal{T}$ (but not $\perp$); L and L' range over $\mathcal{L}$.

$$E'[1] = \prod_i \prod_{x \neq x'} (\neg a[i, x] \vee \neg a[i, x'])$$

$$E'[2] = \prod_{i \neq i'} \prod_x (\neg a[i, x] \vee \neg a[i', x])$$

$$E'[3] = \prod_{i < i'} \prod_{1 \leq j < j' \leq m} (\neg a[i, j'] \vee \neg a[i', j])$$

$$E'[4] = \prod_{J} \prod_{T \neq T'} (\neg b[J,T] \vee \neg b[J,T'])$$

$$E'[5] = \prod_{J,T:|J| \neq |T|} (\neg b[J,T])$$

$$E'[6] = b[\emptyset, \emptyset]$$

$$E'[7] = \prod_{J,T} \prod_{i > \max(J)} \prod_{x \in X_T} ((b[J,T] \wedge a[i,x]) \rightarrow b[J \cup \{i\}, T \cup \{x\}])$$

$$E'[8] = \prod_{J,T:|J|=|T|} \prod_{s=1}^{|J|} (b[J,T] \rightarrow a[J(s), T(s)])$$

$$E'[9] = \prod_{J''} \prod_{(T,L,J) \neq (T',L',J')} (\neg c[J'', T, L, J] \vee \neg c[J'', T', L', J'])$$

$$E'[10] = \prod_{J,T,L} \prod_{J':|J'| \neq |L|} (\neg c[J,T,L,J'])$$

$$E'[11] = \prod_{J,J'} \prod_{T \neq T'} \prod_{L} (\neg b[J,T] \vee \neg c[J,T',L,J'])$$

$$E'[12] = \prod_{J,T,J'} (\neg b[J,T] \vee \neg c[J, \perp, \perp, J'])$$

$$E'[13] = \prod_{J,T,L,J':|L|=|J'|} \prod_{s=1}^{|L|} (c[J,T,L,J'] \rightarrow a[J'(s), L(s)])$$

The variables of X' and the subexpressions of E' all have intuitive semantic interpretations that we next discuss, in order to clarify the argument.

Note that the variables of $X'[1]$ can be group into k *choice blocks* by the first index, each block having $|X|$ variables. Thus the variables of $X'[1]$ represent the k choices of a variable of X to set to *true* in a weight k truth assignment for E.

The subexpression $E'[1]$ enforces that in each block, at most one choice is indicated in any satisfying truth assignment for E'. The subexpression $E'[2]$ enforces that the indicated choices are distinct, and $E'[3]$ enforces that the choices of variables of X to be set *true* are made in increasing order according to the index of the variables.

The variables of $X'[2]$ pair sets of choice blocks with sets of variables in X. For example, the variable $b[\{2,5,7\}, \{x_3, x_4, x_6\}]$ has the intended meaning: "x_3 is chosen in block 2, x_4 in block 5, and x_6 in block 7".

The subexpression $E'[4]$ enforces that for any subset of the k choice blocks, at most one corresponding "macro" set of variables is indicated. Note that it is not possible to supply all possible macros of size at most k since there are too many.

Only those that occur in a certain way are available (or needed) as second indices for the variables of X'_2. When a macro corresponding to the choices made in a set of blocks is "not available", the set of blocks can be paired with the dummy index $\perp$. The subexpression $E'[5]$ enforces that in the pairing of a set of blocks with a set of variables, the sets have the same size, as required by the intended meaning.

The subexpression $E'[6]$ is essentially an artifact having to do with the availability of the dummy index. We require that the empty set of blocks is paired with the empty set of variables. Together with the subexpression $E'[7]$, this forces the variables of $X'[2]$ to do their intended job (the formal argument for this is made below). The subexpression $E'[8]$ forces any pairings asserted by the boolean variables of $X'[2]$ to be represented in the truth assignment for $X'[1]$.

The variables of $X'[3]$ each represent a fairly complicated predicate that is perhaps best illustrated by example. Suppose $J = \{2, 3, 5\}$ so that J represents the set of choice blocks indexed 2,3 and 5. By some variable of $X'[2]$ this is paired with (if it is available as an index, suppose this is so) the set of variables (of X) chosen in these blocks. Suppose that this set of variables is $T = \{x_4, x_6, x_8\}$. There are then various clauses of E that are forced to be positively satisfied in any truth assignment that makes the variables of T *true*, and this situation is analyzed in the tree Ω_T. An index $L \in \mathcal{L}_T$ indicates a choice of further variables to set *true* in order to satisfy these clauses.

In summary, the variable $c[J, T, L, J']$ represents the assertion: "The variables of L are chosen to be *true* in the blocks indicated by J' in order to solve the positive-satisfaction problem created by the variables of T being set *true* in the blocks indicated by J."

The subexpression $E'[9]$ enforces that there is at most one such assertion made for any set of blocks J (by a satisfying truth assignment). $E'[10]$ insures that the right number of blocks are pointed to in the solution of the positive-satisfaction problem. $E'[11]$ and $E'[12]$ enforce that the solution indicated for the positive satisfaction problem created by T must be a solution computed in Ω_T. $E'[13]$ forces the indicated solution to be "implemented" in the variables of $X'[1]$.

We next argue more formally for the correctness of the reduction. Suppose τ is a weight k truth assignment that satisfies E. Let K denote the set of variables set to *true* and let $K(i)$ denote the i^{th} variable of K (ordered by the indices of the variables in K).

For $J \subseteq [k]$, let $K(J) = \{x : \exists j \in J,\ K(j) = x\}$.

For $T \in \mathcal{T}$, let $L(T, K)$ denote the lexicographically smallest element of $S = \{K' \subseteq K : K' \in \mathcal{L}_T\}$.

Claim 3. $L(T, K)$ is well-defined.

Proof. It is enough to argue that the set S is nonempty. By Claim 2, any truth assignment for X that satisfies E must be a superset of at least one element of $\mathcal{L}_T$, otherwise a clause of $\mathcal{C}_T$ would not be satisfied.

Note that $L(T, K)$ is necessarily disjoint from T.

Let τ' be the weight k' truth assignment to the variables of X' that assigns the following variables the value *true*:

(1) $\{a[i, K(i)] : 1 \leq i \leq k\}$

(2) $\{b[J, K(J)] : J \in 2^{[k]},\ K(J) \in \mathcal{T}\}$

(3) $\{b[J, \perp] : J \in 2^{[k]},\ K(J) \notin \mathcal{T}\}$

(4) $\{c[J, K(J), L(K(J), K), J'] : J \in 2^{[k]},\ K(J) \in \mathcal{T},\ K(J') = L(K(J), K)\}$

(5) $\{c[J, \perp, \perp, \emptyset] : J \in 2^{[k]},\ K(J) \notin \mathcal{T}\}$

Claim 4. If τ satisfies E then τ' satisfies E'.

Proof. To see that this is so, we just need to check that each of the subexpressions of E' is satisfied. The only case that is nontrivial is $E'[7]$. If one of the implications of this subexpression were false, then we would have a situation where τ' has assigned the value *true* to all possible true assertions represented by the variables of $X'[2]$ of the form: "the variables of T are chosen (in ascending order) in the blocks indicated by J", and where for one of these true assertions represented by $b[J, T]$, and for a further choice of a variable x set *true* by τ' in a block i (of larger index than any block in J), with x a successor of T, τ' has assigned the value *false* to $b[J \cup \{i\}, T \cup \{x\}]$. Since $T \cup \{x\} \in \mathcal{T}$, this is a contradiction, since this variable represents a statement that is clearly true.

That completes (the easier) half of the argument for the correctness of the reduction.

Now suppose that υ' is a weight k' truth assignment to the variables of X' that satisfies E'. We argue that necessarily υ has a certain restricted form, from which we can then describe a corresponding weight k truth assignment to the variables of X that satisfies E.

Claim 5. The truth assignment υ must assign exactly one variable of $X'[1]$ to be true in each choice block of $X'[1]$.

Proof. Since $E'[1]$ is satisfied, there can at most one variable in each block assigned *true*. The satisfaction of $E'[4]$ and $E'[9]$ imply that that no more than 2^{k+1} variables are assigned *true* in $X'[2] \cup X'[3]$. This forces k to be assigned *true* in $X'[1]$ and the claim follows.

Let υ denote the truth assignment for the variables of X that is indicated in the natural manner (as per Claim 5) by υ'. Our goal is to show that the truth assignment υ satisfies E. Part of our argument is based on the details of how υ' satisfies E'.

Claim 6. Necessarily υ' assigns *true* to all possible variables of $X'[2]$ that represent true statements (about υ) of the form: "the variables of T are chosen (in ascending order) in the blocks indicated by J".

Proof. For any $T \in \mathcal{T}$ the proof is a simple induction along the trail leading to T, based on the satisfaction of $E'[6]$ (the base step), and $E'[7]$.

Claim 7. The truth assignment υ satisfies E.

Proof. Suppose on the contrary that some clause C_i is not satisfied. Then all of the variables in C_i^- must be assigned the value *true* by υ. Also, $C_i^- \in \mathcal{T}$. By Claim 6, there is some J such that the variable $b[J, C_i^-]$ is assigned the value *true* by υ'. Since υ' satisfies the subexpressions $E'[10]$, $E'[11]$ and $E'[12]$, it must assign *true* to some variable $c[J, C_i^-, L, J']$ with $|L| = |J'|$ and $L \in \mathcal{L}_{C_i^-}$. Since υ' satisfies $E'[13]$, υ assigns the value *true* to the variables of L. Since $C_i \in \mathcal{C}_{C_i^-}$, the clause C_i must be satisfied by υ, a contradiction.

That completes the proof of the theorem.

7 Conclusions

The "concrete" results of this paper indicate that the size of a relational database query is not likely to be that can be effectively exploited in isolation. It might be interesting to study various parameterizations of the *structure* of database queries in addition to their size to determine if there may be some opportunities for fixed-parameter tractability.

An interesting open problem concerning the W hierarchy is whether there are any *natural* degrees other than

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$$

(The four named in this tower are abundantly natural.) For example, are there natural problems that constitute a likely "$W[1/2]$" or "$W[3/2]$" degree? A promising place to look is in variations of the weighted t-normalized satisfiability problems that are shown to be complete for $W[t]$ in [DF95a]. Our main theorem here shows that one promising candidate ($W^*[1]$) for a representative of a "$W[3/2]$" degree is in fact no more than $W[1]$. It would be interesting to know if $W^*[t] = W[t]$ for all t.

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.

[BFR95] R. Balasubramanian, M. Fellows and V. Raman, An improved fixed parameter algorithm for vertex cover, To appear, *Information Processing Letters,* 1995.

[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.

[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.

[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.

[CCDF94] L. Cai, J. Chen, R.G. Downey and M.R. Fellows. On the parameterized complexity of short computation and factorization. To appear in *Arch. for Math. Logic.*

[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.*

[DEF93] R.G. Downey, P.A. Evans and M.R. Fellows. Parameterized learning complexity. *Proceedings of the Sixth ACM Workshop on Computational Learning Theory (COLT'93)*, 51–57,

[DF95a] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: basic theory. *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.

[DFta] R.G. Downey and M.R. Fellows. Paramaterized Complexity. Monograph, Springer-Verlag, to appear.

[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.

[DFR96] R.G. Downey, M.R. Fellows and K. Regan. Parameterized circuit complexity and the W hierarchy. *Theoretical Computer Science*, to appear.

[DFT96] R.G.Downey, M.R. Fellows and U. Taylor. An improved characterization of $W[1]$. Manuscript. February 1996.

[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.

[GJ79] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W.H. Freeman and Company, San Francisco, 1979.

[Ul88] J.D. Ullman. Database and Knowledge-base Systems. Computer Science Press, 1988.

[Yan95] M. Yannakakis. Perspectives on database theory. Proceedings FOCS 95, 224-246. Computer Science Press, 1988.