Parameterized Complexity and Subexponential Time

Jörg Flum* Martin Grohe†

1. Introduction

Over the last 15 years, the theory of fixed-parameter tractability [13] has developed into a well-established branch of algorithm design and complexity theory. In this theory, the running time of algorithms is analyzed not only in terms of the input size, but also in terms of an additional parameter of problem instances. An algorithm is called *fixed-parameter tractable (fpt)* if its running time is possibly super-polynomial in terms of the parameter of the instance, but polynomial in the size. More precisely, an algorithm is fpt if its running time is

$$f(k) \cdot n^{O(1)} \tag{1.1}$$

for some computable function f, where n denotes the size of the input and k the parameter. The idea is to choose the parameterization in such a way that the parameter is small for problem instances appearing in a concrete application at hand. Since $f(k)$ is expected to be moderate for small k, fixed-parameter tractability is a reasonable approximation of practical tractability for such problem instances.

Fixed-parameter tractability is thus a specific approach to the design of *exact algorithms for hard algorithmic problems*, an area which has received much attention in recent years (see, for example, [17, 27]). Well known examples of non-trivial exact algorithms are the ever improving algorithms for the 3-satisfiability problem [24, 25, 8, 20], the currently best being due to Iwama and Tamaki [20] with a running time of roughly 1.324^n, where n is the number of variables of the input formula. In this article, we are mainly

*Abteilung für Mathematische Logik, Albert-Ludwigs-Universität Freiburg, Eckerstr. 1, 79104 Freiburg, Germany. Email: flum@uni-freiburg.de

†Institut für Informatik, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany. Email: grohe@informatik.hu-berlin.de

interested in lower bounds for exact algorithms. For example, is there an algorithm that solves the 3-satisfiability problem in time $2^{o(n)}$? The assumption that there is no such algorithm is known as the *exponential time hypothesis (ETH)*. The exponential time hypothesis and related assumptions have been studied from a complexity theoretic point of view in [14, 19, 18, 26]. Most notably, Impagliazzo, Paturi, and Zane [19] have started to develop a theory of hardness and completeness for problems with respect to subexponential time solvability. An ultimate goal of such a theory would be to show the equivalence of assumptions such as (ETH) with more established assumptions such as P $\neq$ NP. Of course it is not clear at all if such an equivalence can be proved without actually proving (ETH). Overall, we believe that it is fair to say that subexponential time complexity is not very well understood.

What singles out fixed-parameter tractability among other paradigms for the design of exact algorithms for hard algorithmic problems is that it is complemented by a very well-developed theory of intractability. It is known for quite a while that this intractability theory has close connections with subexponential time complexity and the exponential time hypothesis [1]. But only recently have these connections moved to the center of interest of researchers in parameterized complexity theory [3, 10, 4, 5, 6]. This shift of interest was caused by attempts to prove lower bounds for the parameter dependence (the function f in (1.1)) of fpt-algorithms [3] and the investigations of *miniaturized problems* in this context [10].

The purpose of this article is to explain these connections between parameterized and subexponential complexity. The intention is not primarily to survey the most recent developments, but to explain the technical ideas in sufficient detail. (For a recent survey on parameterized complexity theory, see, for example, [9].) The main technical results are reductions between the satisfiability problem and the weighted satisfiability problem, which asks for satisfying assignments setting a specific number k of the variables to TRUE. We call an assignment setting exactly k variables to TRUE a *weight k assignment*. The reductions are based on a simple idea known as the *k-log-n trick*: Specifying a weight k assignment to a set of n variables requires $k \cdot \log n$ bits. This can be used to reduce weighted satisfiability of a formula with n variables to unweighted satisfiability of a formula with only $k \cdot \log n$ variables. A similar reduction can be used in the converse direction. To obtain reasonably tight reductions for specific classes of propositional formulas, some care is required. The construction is carried out in the proof of Theorem 4.4.

All results presented in this article are known (essentially, they go back to [1]), and they are not very deep. Nevertheless, we believe it is worth while to present the results in a uniform and introductory manner to a wider audience. Our presentation may be slightly unfamiliar for the experts in the area,

as it is based on a new *M-hierarchy* of parameterized complexity classes. We show that this hierarchy is entangled with the familiar W-hierarchy. The M-hierarchy is a translation of a natural hierarchy of satisfiability problems into the world of parameterized complexity, and fixed-parameter tractability of the M-classes directly translates to subexponential complexity of the corresponding satisfiability problems. Let us emphasize that even though we will develop the theory in the setting of parameterized complexity, it directly applies to subexponential complexity. The connection will be made explicit in the last section of the article.

The article is organized as follows: After introducing our notation, we start with a brief introduction into parameterized complexity theory. In Section 4, we introduce the M-hierarchy and establish the connections between the M-hierarchy and subexponential time complexity on the one hand, and between the M-hierarchy and the W-hierarchy on the other hand. In Section 5, we study the miniaturized problems that originally led to the introduction of the class M[1]. We prove a number of completeness results for M[1], which are based on a combinatorial lemma known as the *Sparsification Lemma* [19]. (The proof of the Sparsification Lemma itself is beyond the scope of this article.) We put these results in the wider context of the syntactically defined complexity class SNP in Section 6. Finally, in Section 7, we translate the results back to the world of classical complexity theory and the exponential time hypothesis.

One nice aspect of this area is that it has a number of very interesting open problems. We conclude this article by listing a few of them.

2. Notation

The set of natural numbers (that is, positive integers) is denoted by $\mathbb{N}$. For integers n, m, we let $[n, m] = \{n, n+1, \ldots, m\}$ and $[n] = [1, n]$. Unless mentioned explicitly otherwise, we encode integers in binary.

We use $\log n$ to denote the binary (base 2) logarithm of $n \in \mathbb{N}$.

For computable functions $f, g : \mathbb{N} \to \mathbb{N}$, we say that f is *effectively little-oh of* g and write $f \in o^{\text{eff}}(g)$ if there exist $n_0 \in \mathbb{N}$ and a computable function $\iota : \mathbb{N} \to \mathbb{N}$ that is non-decreasing and unbounded such that for all $n \geq n_0$,

$$f(n) \leq \frac{g(n)}{\iota(n)}.$$

We mostly use the letter ι to denote computable functions that are non-decreasing and unbounded (but possibly growing very slowly).

Throughout this paper we work with the effective version of "little-oh". In particular, we require subexponential algorithms to have a running time of $2^{o^{\text{eff}}(n)}$ and not just $2^{o(n)}$. The reason for this is that it gives us a correspondence between "strongly uniform" fixed-parameter tractability and subexponential complexity. A similar correspondence holds between "little-oh" instead of "effective little-oh" and "uniform fixed-parameter tractability" instead of "strongly uniform fixed-parameter tractability". We prefer to work with strongly uniform fixed-parameter tractability as it has a more robust theory.

2.1. Propositional Logic

Formulas of *propositional logic* are built up from *propositional variables* X_1, $X_2, \ldots$ by taking conjunctions, disjunctions, and negations. The negation of a formula α is denoted by $\neg\alpha$. We distinguish between *small conjunctions*, denoted by $\wedge$, which are just conjunctions of two formulas, and *big conjunctions*, denoted by $\bigwedge$, which are conjunctions of arbitrary finite sequences of formulas. Analogously, we distinguish between *small disjunctions*, denoted by $\vee$, and *big disjunctions*, denoted by $\bigvee$.

The set of variables of a formula α is denoted by $\text{var}(\alpha)$. An *assignment* for a formula α is a mapping $\mathcal{V} : \text{var}(\alpha) \to \{\text{TRUE}, \text{FALSE}\}$, and we write $\mathcal{V} \models \alpha$ to denote that $\mathcal{V}$ satisfies α.

We use a similar notation for *Boolean circuits*. In particular, we think of the input nodes of a circuit γ as being labeled with variables, use $\text{var}(\gamma)$ to denote the set of these variables, and for an assignment $\mathcal{V} : \text{var}(\gamma) \to \{\text{TRUE}, \text{FALSE}\}$ we write $\mathcal{V} \models \gamma$ to denote that γ computes TRUE if the input nodes are assigned values according to γ.

The class of all propositional formulas is denoted by PROP, and the class of all Boolean circuits by CIRC. Usually, we do not distinguish between formulas and circuits, that is, we view PROP as a subclass of CIRC.

For $t \geq 0$, $d \geq 1$ we inductively define the following classes $\Gamma_{t,d}$ and $\Delta_{t,d}$ of propositional formulas:[1]

$$\Gamma_{0,d} = \{\lambda_1 \wedge \ldots \wedge \lambda_c \mid c \leq d,\ \lambda_1, \ldots, \lambda_c \text{ literals}\},$$
$$\Delta_{0,d} = \{\lambda_1 \vee \ldots \vee \lambda_c \mid c \leq d,\ \lambda_1, \ldots, \lambda_c \text{ literals}\},$$
$$\Gamma_{t+1,d} = \{\bigwedge_{i \in I} \delta_i \mid I \text{ finite index set and } \delta_i \in \Delta_{t,d} \text{ for all } i \in I\},$$

[1] We prefer to use Γ and Δ instead of the more common Π and Σ to denote classes of propositional formulas (Γ for conjunctions, Δ for disjunctions). The reason is that we want to reserve Π and Σ for classes of formulas of predicate logic. Often in parameterized complexity, it is necessary to jump back and forth between propositional and predicate logic, and it is helpful to keep them strictly separated on the notational level.

$$\Delta_{t+1,d} = \{\bigvee_{i \in I} \gamma_i \mid I \text{ finite index set and } \gamma_i \in \Gamma_{t,d} \text{ for all } i \in I\}.$$

$\Gamma_{2,1}$ is the class of all formulas in *conjunctive normal form*, which we often denote by CNF. For $d \geq 1$, $\Gamma_{1,d}$ is the class of all formulas in *d-conjunctive normal form*, which we denote by d-CNF.

The *size* $|\gamma|$ of a circuit γ is the number of nodes plus the number of edges; thus for formulas the size is O(number of nodes). We usually use the letter m to denote the size of a formula or circuit and the letter n to denote the number of variables.

3. Fundamentals of Parameterized Complexity Theory

3.1. Parameterized Problems and Fixed-Parameter Tractability

As it is common in complexity theory, we describe decision problems as languages over finite alphabets Σ. To distinguish them from parameterized problems, we refer to problems $Q \subseteq \Sigma^*$ as *classical problems*.

A *parameterization* of Σ^* is a mapping $\kappa : \Sigma^* \to \mathbb{N}$ that is polynomial time computable. A *parameterized problem* (over Σ) is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ and a parameterization κ of Σ^*. If (Q, κ) is a parameterized problem over the alphabet Σ, then we call strings $x \in \Sigma^*$ *instances* of Q or of (Q, κ) and the numbers $\kappa(x)$ the corresponding *parameters*. Slightly abusing notation, we call a parameterized problem (Q, κ) a *parameterization* of the classical problem Q.

Usually, when representing a parameterized problem we do not mention the underlying alphabet explicitly and use a notation as illustrated by the following examples.

Example 3.1. Recall that a *vertex cover* in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that for each edge $\{u, v\} \in E$, either $u \in S$ or $v \in S$. The *parameterized vertex cover problem* is defined as follows:

p-Vertex-Cover	
Instance:	A graph G and a natural number $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide if G has a vertex cover of size k.

Example 3.2. The *parameterized satisfiability problem for Boolean circuits* is defined as follows:

p-SAT(CIRC)	
Instance:	A Boolean circuit γ.
Parameter:	$\lvert\mathrm{var}(\gamma)\rvert$.
Problem:	Decide if γ is satisfiable.

More generally, for a class Γ of circuits or formulas, we let p-SAT(Γ) denote the restriction of p-SAT(CIRC) to instances $\gamma \in \Gamma$.

p-SAT(Γ) is a parameterization of the classical problem SAT(Γ). There are other interesting parameterizations of SAT(Γ), and we will see some later.

Example 3.3. The *weight* of an assignment $\mathcal{V}$ is the number of variables set to TRUE by $\mathcal{V}$. A circuit γ is *k-satisfiable*, for some $k \in \mathbb{N}$, if there is a satisfying assignment $\mathcal{V}$ of weight k for γ. The *weighted satisfiability problem* WSAT(Γ) for a class Γ of circuits asks whether a given circuit $\gamma \in \Gamma$ is k-satisfiable for a given k. We consider the following parameterization:

p-WSAT(Γ)	
Instance:	$\gamma \in \Gamma$ and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide if γ is k-satisfiable.

Definition 3.4. Let Σ be a finite alphabet and $\kappa : \Sigma^* \to \mathbb{N}$ a parameterization.

(1) An algorithm $\mathbb{A}$ with input alphabet Σ is an *fpt-algorithm with respect to κ* if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that the running time of $\mathbb{A}$ on input x is

$$f\big(\kappa(x)\big) \cdot |x|^{O(1)}.$$

(2) A parameterized problem (Q, κ) is *fixed-parameter tractable* if there is an fpt-algorithm with respect to κ that decides Q.

FPT denotes the class of all fixed-parameter tractable problems.[2]

Example 3.5. p-SAT(CIRC) is fixed-parameter tractable.

Indeed, the obvious brute-force search algorithm decides if a circuit γ of size m with n variables is satisfiable in time $O(2^n \cdot m)$.

We leave it to the reader to show that p-VERTEX-COVER is also fixed-parameter tractable. On the other hand, p-WSAT(2-CNF) does not seem to be fixed-parameter tractable. We shall now introduce the theory to give evidence for this and other intractability results.

[2]The notion of fixed-parameter tractability we introduce here is known as "strongly uniform fixed-parameter tractability." The alternative notion "uniform fixed-parameter tractability" does not require the function f to be computable.

3.2. Reductions

Definition 3.6. Let (Q, κ) and (Q', κ') be parameterized problems over the alphabets Σ and Σ', respectively. An *fpt-reduction* (more precisely, *fpt many-one reduction*) from (Q, κ) to (Q', κ') is a mapping $R : \Sigma^* \to (\Sigma')^*$ such that:

(1) For all $x \in \Sigma^*$ we have $x \in Q \iff R(x) \in Q'$.

(2) R is computable by an fpt-algorithm with respect to κ.

(3) There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$ for all $x \in \Sigma^*$.

We write $(Q, \kappa) \leq^{\text{fpt}} (Q', \kappa')$ if there is an fpt-reduction from (Q, κ) to (Q', κ'), and we write $(Q, \kappa) \equiv^{\text{fpt}} (Q', \kappa')$ if $(Q, \kappa) \leq^{\text{fpt}} (Q', \kappa')$ and $(Q', \kappa') \leq^{\text{fpt}} (Q, \kappa)$. We let $\big[(Q, \kappa)\big]^{\text{fpt}}$ be the class of parameterized problems fpt-reducible to (Q, κ), that is,

$$\big[(Q, \kappa)\big]^{\text{fpt}} = \big\{(Q', \kappa') \bigm| (Q', \kappa') \leq^{\text{fpt}} (Q, \kappa)\big\}.$$

For every class C of parameterized problems, we define C-hardness and C-completeness of a parameterized problem (Q, κ) in the usual way.

Example 3.7. Recall that an independent set in a graph is a set of pairwise non-adjacent vertices and consider the *parameterized independent set problem*:

p-Independent-Set	
Instance:	A graph G and $k \in \mathbb{N}$.
Parameter:	k.
Problem:	Decide if G has an independent set of size k.

Then p-Independent-Set $\leq^{\text{fpt}}$ p-WSat(2-CNF), where 2-CNF denotes the class of all propositional formulas in 2-conjunctive normal form.

To see this, let $G = (V, E)$ be a graph. For every vertex $v \in V$ we introduce a propositional variable X_v whose intended meaning is "v belongs to the independent set". We let

$$\gamma = \bigwedge_{\{v,w\} \in E} (\neg X_v \vee \neg X_w).$$

Then α is k-satisfiable if and only if G has an independent set of size k. (There is one detail here that requires attention: If v is an isolated vertex of G, then the variable X_v does not occur in γ. Thus the claimed equivalence is

true for graphs without isolated vertices. We leave it to the reader to reduce the problem for arbitrary graphs to graphs without isolated vertices.)

The converse also holds, that is,

$$p\text{-WSAT}(\text{2-CNF}) \leq^{\text{fpt}} p\text{-INDEPENDENT-SET},$$

but is much harder to prove [12]. By reversing the argument above, it is easy to show that p-WSAT(2-CNF$^-$) $\leq^{\text{fpt}}$ p-INDEPENDENT-SET, where 2-CNF$^-$ denotes the class of all 2-CNF-formulas in which only negative literals occur.

We also need a notion of parameterized Turing reductions:

Definition 3.8. Let (Q, κ) and (Q', κ') be parameterized problems over the alphabets Σ and Σ', respectively. An *fpt Turing reduction* from (Q, κ) to (Q', κ') is an algorithm $\mathbb{A}$ with an oracle to Q' such that:

(1) $\mathbb{A}$ decides (Q, κ).

(2) $\mathbb{A}$ is an fpt-algorithm with respect to κ.

(3) There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all oracle queries "$y \in Q'$?" posed by $\mathbb{A}$ on input x we have $\kappa'(y) \leq g(\kappa(x))$.

We write $(Q, \kappa) \leq^{\text{fpt-T}} (Q', \kappa')$ if there is an fptTuring reduction from (Q, κ) to (Q', κ'), and we write $(Q, \kappa) \equiv^{\text{fpt-T}} (Q', \kappa')$ if $(Q, \kappa) \leq^{\text{fpt-T}} (Q', \kappa')$ and $(Q', \kappa') \leq^{\text{fpt-T}} (Q, \kappa)$.

3.3. The W-Hierarchy

Recall the definitions of the classes $\Gamma_{t,d}$ of propositional formulas.

Definition 3.9. (1) For $t \geq 1$, W[t] is the class of all parameterized problems fpt-reducible to a problem p-WSAT($\Gamma_{t,d}$) for some $d \geq 1$, that is,

$$\text{W}[t] = \bigcup_{d \geq 1} \big[p\text{-WSAT}(\Gamma_{t,d})\big]^{\text{fpt}}.$$

(2) W[SAT] is the class of all parameterized problems fpt-reducible to p-WSAT(PROP), that is,

$$\text{W[SAT]} = \big[p\text{-WSAT(PROP)}\big]^{\text{fpt}}.$$

(3) W[P] is the class of all parameterized problems fpt-reducible to p-WSAT(CIRC), that is,

$$\text{W[P]} = \big[p\text{-WSAT(CIRC)}\big]^{\text{fpt}}.$$

Observe that

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{SAT}] \subseteq \mathrm{W}[\mathrm{P}].$$

One of the fundamental structural results of parameterized complexity theory is the following normalization theorem for the W-hierarchy. For $t, d \geq 1$ we let $\Gamma^+_{t,d}$ be the class of $\Gamma_{t,d}$-formulas in which all literals are positive (that is, no negation symbols occur) and $\Gamma^-_{t,d}$ be the class of $\Gamma_{t,d}$-formulas in which all literals are negative

Theorem 3.10 (Downey and Fellows [12, 11]).

(1) $\mathrm{W}[1] = \left[\mathrm{WSAT}(\Gamma^-_{1,2})\right]^{\mathrm{fpt}}$.

(2) For even $t \geq 2$, $\mathrm{W}[t] = \left[\mathrm{WSAT}(\Gamma^+_{t,1})\right]^{\mathrm{fpt}}$.

(3) For odd $t \geq 3$, $\mathrm{W}[t] = \left[\mathrm{WSAT}(\Gamma^-_{t,1})\right]^{\mathrm{fpt}}$.

Many natural parameterized problems are complete for the first two levels of the W-hierarchy. For example, p-INDEPENDENT-SET is complete for W[1] [11], and the parameterized dominating set problem is complete for W[2] [12].

3.4. W[P] and Limited Nondeterminism

We close this introductory section by presenting two results that establish a very clean connection between the class W[P] and limited nondeterminism [22, 16]. The first is a machine characterization of W[P]:

Theorem 3.11 ([2, 7]). *A parameterized problem* (Q, κ) *over the alphabet* Σ *is in* W[P] *if and only if there are computable functions* $f, h : \mathbb{N} \to \mathbb{N}$, *a polynomial* $p(X)$, *and a nondeterministic Turing machine* $\mathbb{M}$ *deciding* Q *such that for every input* x *on every run the machine* $\mathbb{M}$*:*

(1) performs at most $f(k) \cdot p(n)$ *steps;*

(2) performs at most $h(k) \cdot \log n$ *nondeterministic steps.*

Here $n = |x|$ *and* $k = \kappa(x)$.

Let $f : \mathbb{N} \to \mathbb{N}$. A problem $Q \subseteq \Sigma^*$ is in NP$[f]$ if there is a polynomial p and a nondeterministic Turing machine $\mathbb{M}$ deciding Q such that for every input x on every run the machine $\mathbb{M}$

(1) performs at most $p(|x|)$ steps;

(2) performs at most $f(|x|)$ nondeterministic steps.

There is an obvious similarity between the characterization of W[P] given in Theorem 3.11 and the (classical) classes NP[f]. The next theorem establishes a formal connection:

Theorem 3.12 ([2]). *The following statements are equivalent:*

(1) FPT = W[P].

(2) There is a computable function $\iota : \mathbb{N} \to \mathbb{N}$ *that is non-decreasing and unbounded such that* PTIME = NP[$\iota(n) \cdot \log n$].

The techniques used to prove this result are similar to those introduced in the next section. Indeed, the direction (1) $\Longrightarrow$ (2) is an easy consequence of Theorem 4.4.

The connection between parameterized complexity and limited nondeterminism can be broadened if one considers *bounded parameterized complexity theory*, where some bound is put on the growth of the dependence of the running time of an fpt-algorithm on the parameter (see [15]).

4. The M-Hierarchy

4.1. A New Parameterization of the Satisfiability Problem

In the following, we will consider different parameterizations of the satisfiability problem Sat(CIRC). We denote the input circuit by γ, its size by m, and its number of variables by n. Without loss of generality we can always assume that $m \leq 2^n$, because if $m > 2^n$ we can easily decide if γ is satisfiable in time $m^{O(1)}$. However, in general m can still be much larger than n.

If we parameterize Sat(CIRC) by n then we obtain the fixed-parameter tractable problem p-Sat(CIRC). Let us now see what happens if we decrease the parameter. Specifically, let us consider the parameterizations $\big(\text{Sat(CIRC)}, \kappa_h\big)$, where

$$\kappa_h(\gamma) = \left\lceil \frac{n}{h(m)} \right\rceil$$

for computable functions $h : \mathbb{N} \to \mathbb{N}$. For constant $h \equiv 1$, κ_h is just our old parameterization p-Sat(CIRC) $\in$ FPT. At the other end of the scale, for $h(m) \geq m \geq n$ we have $\kappa_h(\gamma) = 1$, and essentially $\big(\text{Sat(CIRC)}, \kappa_h\big)$ is just the NP-complete unparameterized problem Sat(CIRC). But what happens if we consider functions between these two extremes?

If $h(m) \in o^{\text{eff}}(\log m)$, then $\big(\text{Sat(CIRC)}, \kappa_h\big)$ is still fixed-parameter tractable. (To see this, use that Sat(CIRC) is trivially solvable in time $m^{O(1)}$ for

instances with $m \geq 2^n$.) If $h(m) \in \omega^{\text{eff}}(\log m)$ then for large circuits of size close to 2^n, but still $2^{o^{\text{eff}}(n)}$, the parameter is 1 and fixed-parameter tractability coincides with polynomial time computability. The most interesting range from the perspective of parameterized complexity is

$$h(m) \in \Theta(\log m).$$

These considerations motivate us to introduce the following parameterization of the satisfiability problem for every class Γ of circuits.

p-log-SAT(Γ)	
Instance:	$\gamma \in \Gamma$ of size m with n variables.
Parameter:	$\left\lceil \frac{n}{\log m} \right\rceil$.
Problem:	Decide if γ is satisfiable.

Obviously, *p-log*-SAT(Γ) is solvable in time

$$2^n \cdot m^{O(1)} \leq 2^{k \cdot \log m} \cdot m^{O(1)} = m^{k+O(1)},$$

where $k = \left\lceil \frac{n}{\log m} \right\rceil$ is the parameter. Intuitively it seems unlikely that the problem is fixed-parameter tractable.

To phrase our first result in its most general form, we introduce a simple closure property of classes of circuits: We call a class Γ *paddable* if for every $\gamma \in \Gamma$ and for every $m' \geq |\gamma|$ there is a circuit $\gamma' \in \Gamma$ such that $\text{var}(\gamma') = \text{var}(\gamma)$, the circuits γ and γ' are equivalent, and $m' \leq |\gamma'| \leq O(m')$. We call Γ *efficiently paddable* if, in addition, there is an algorithm that computes γ' for given γ and $m' \geq |\gamma|$ in time $(m')^{O(1)}$. Most natural classes of formulas and circuits are efficiently paddable, in particular all classes $\Gamma_{t,d}$ and the classes PROP and CIRC. For example, for the $\Gamma_{1,2}$-formula

$$\gamma = \bigwedge_{i=1}^{m} (\lambda_{i1} \vee \lambda_{i2}),$$

we can let $\lambda_{ij} = \lambda_{mj}$ for $m < i \leq m'$ and $j = 1, 2$, and

$$\gamma' = \bigwedge_{i=1}^{m'} (\lambda_{i1} \vee \lambda_{i2}).$$

Proposition 4.1 ([3, 10]). *Let Γ be an efficiently paddable class of circuits. Then*

$$\textit{p-log}\text{-SAT}(\Gamma) \in \text{FPT} \iff \text{SAT}(\Gamma) \in \text{DTIME}\big(2^{o^{\text{eff}}(n)} \cdot m^{O(1)}\big),$$

where $n = |\text{var}(\gamma)|$ is the number of variables and $m = |\gamma|$ the size of the input circuit γ.

Proof: Suppose first that *p-log*-SAT(Γ) $\in$ FPT. Let $f : \mathbb{N} \to \mathbb{N}$ be a computable function and $\mathbb{A}$ an fpt-algorithm that decides *p-log*-SAT(Γ) in time

$$f(k) \cdot m^{O(1)},$$

where $k = \lceil n/\log m \rceil$ is the parameter. Without loss of generality, we may assume that f is increasing and time constructible, which implies that $f(i) \leq j$ can be decided in time $O(j)$. Let $\iota : \mathbb{N} \to \mathbb{N}$ be defined by

$$\iota(n) = \max\Big(\{1\} \cup \{i \in \mathbb{N} \mid f(i) \leq n\}\Big).$$

Then ι is non-decreasing and unbounded, $f(\iota(n)) \leq n$ for all but finitely many n, and $\iota(n)$ can be computed in time $O(n^2)$.

We shall prove that SAT(Γ) $\in$ DTIME$\big(2^{O(n/\iota(n))} \cdot m^{O(1)}\big)$.

Let $\gamma \in \Gamma$, $m = |\gamma|$ and $n = |\mathrm{var}(\gamma)|$. Assume first that $m \geq 2^{n/\iota(n)}$. Note that

$$k = \left\lceil \frac{n}{\log m} \right\rceil \leq \iota(n).$$

Thus $f(k) \leq f(\iota(n)) \leq n$, and we can simply decide $\gamma \in$ SAT(Γ) with the fpt-algorithm $\mathbb{A}$ in time

$$f(k) \cdot m^{O(1)} \leq n \cdot m^{O(1)} = m^{O(1)}.$$

Assume next that $m < 2^{n/\iota(n)}$. Let $m' = 2^{\lceil n/\iota(n) \rceil}$. Let $\gamma' \in \Gamma$ such that $\mathrm{var}(\gamma') = \mathrm{var}(\gamma)$, the circuits γ and γ' are equivalent, and $m' \leq |\gamma'| \leq O(m')$. Since Γ is efficiently paddable, such a γ' can be computed in time polynomial in m', that is, time $2^{O(n/\iota(n))}$. Let $k' = n/\log|\gamma'|$. Then $k' \leq \iota(n)$. We decide $\gamma' \in$ SAT(Γ) with the fpt-algorithm $\mathbb{A}$ in time

$$f(k') \cdot (m')^{O(1)} \leq n \cdot 2^{O(n/\iota(n))}.$$

This completes the proof of the forward direction.

For the backward direction, let $\mathbb{B}$ be an algorithm solving SAT(Γ) in DTIME$\big(2^{O(n/\iota(n))} \cdot m^{O(1)}\big)$ for some computable function $\iota : \mathbb{N} \to \mathbb{N}$ that is non-decreasing and unbounded. Let f be a non-decreasing computable function with $f(\iota(n)) \geq 2^n$ for all $n \in \mathbb{N}$. We claim that

$$\textit{p-log}\text{-SAT}(\Gamma) \in \mathrm{DTIME}(f(k) \cdot m^{O(1)}).$$

Let $\gamma \in \Gamma$, $m = |\gamma|$, $n = |\mathrm{var}(\gamma)|$, and $k = \lceil n/\log m \rceil$. If $m \geq 2^{n/\iota(n)}$ then algorithm $\mathbb{B}$ decides $\gamma \in$ SAT(Γ) in time $m^{O(1)}$. If $m < 2^{n/\iota(n)}$, then

$$k = \left\lceil \frac{n}{\log m} \right\rceil \geq \iota(n)$$

and thus $f(k) \geq 2^n$. Thus we can decide $\gamma \in \text{SAT}(\Gamma)$ by exhaustive search in time $O(f(k) \cdot m)$. □

In the following, we shall say that $\text{SAT}(\Gamma)$ is *subexponential (with respect to the number of variables)* if it is solvable in $\text{DTIME}\big(2^{o^{\text{eff}}(n)} \cdot m^{O(1)}\big)$.

4.2. The M-Hierarchy

Motivated by Proposition 4.1, we define another hierarchy of parameterized complexity classes in anlogy to Definition 3.8:

Definition 4.2. (1) For every $t \geq 1$, we let $\text{M}[t] = \bigcup_{d \geq 1} \big[p\text{-}log\text{-SAT}(\Gamma_{t,d})\big]^{\text{fpt}}$.

(2) $\text{M}[\text{SAT}] = \big[p\text{-}log\text{-SAT}(\text{PROP})\big]^{\text{fpt}}$.

(3) $\text{M}[\text{P}] = \big[p\text{-}log\text{-SAT}(\text{CIRC})\big]^{\text{fpt}}$.

Then by Proposition 4.1:

Corollary 4.3. *(1) For $t \geq 1$, $\text{M}[t] = \text{FPT}$ if and only if $\text{SAT}(\Gamma_{t,d})$ is subexponential for all $d \geq 1$.*

(2) $\text{M}[\text{SAT}] = \text{FPT}$ if and only if $\text{SAT}(\text{PROP})$ is subexponential.

(3) $\text{M}[\text{P}] = \text{FPT}$ if and only if $\text{SAT}(\text{CIRC})$ is subexponential.

The following theorem is essentially due to Abrahamson, Downey, and Fellows [1] (also see [13]).

Theorem 4.4. *For every $t \geq 1$,*

$$\text{M}[t] \subseteq \text{W}[t] \subseteq \text{M}[t+1].$$

Furthermore, $\text{M}[\text{SAT}] = \text{W}[\text{SAT}]$ and $\text{M}[\text{P}] = \text{W}[\text{P}]$.

Proof: We first prove $\text{M}[t] \subseteq \text{W}[t]$. For simplicity, let us assume that t is odd. Fix $d \geq 1$ such that $t + d \geq 3$. We shall prove that

$$p\text{-}log\text{-SAT}(\Gamma_{t,d}) \leq^{\text{fpt}} p\text{-WSAT}(\Gamma_{t,d}). \tag{4.1}$$

Let $\gamma \in \Gamma_{t,d}$. We shall construct a $\Gamma_{t,d}$-formula β such that

$$\gamma \text{ is satisfiable} \iff \beta \text{ is } k\text{-satisfiable}. \tag{4.2}$$

Let $m = |\gamma|$, $n = |\text{var}(\gamma)|$. To simplify the notation, let us assume that $\ell = \log m$ and $k = n/\log m$ are integers. Then $n = k \cdot \ell$. Let $\mathcal{X} = \text{var}(\gamma)$, and let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be a partition of $\mathcal{X}$ into k sets of size ℓ.

For $1 \leq i \leq k$ and every subset $S \subseteq \mathcal{X}_i$, let Y_i^S be a new variable. Let $\mathcal{Y}_i$ be the set of all Y_i^S and $\mathcal{Y} = \bigcup_{i=1}^k \mathcal{Y}_i$. Call a truth value assignment for $\mathcal{Y}$ *good* if for $1 \leq i \leq k$ exactly one variable in $\mathcal{Y}_i$ is set to TRUE. There is a bijection f between the truth value assignments $\mathcal{V}$ for $\mathcal{X}$ and the good truth value assignments for $\mathcal{Y}$ defined by

$$f(\mathcal{V})(Y_i^S) = \text{TRUE} \iff \forall X \in \mathcal{X}_i : \big(\mathcal{V}(X) = \text{TRUE} \iff X \in S\big)$$

for all $\mathcal{V} : \mathcal{X} \to \{\text{TRUE}, \text{FALSE}\}$, $1 \leq i \leq k$, and $S \subseteq \mathcal{X}_i$.

Let β'' be the formula obtained from γ by replacing, for $1 \leq i \leq k$ and $X \in \mathcal{X}_i$, each occurrence of the literal X by the formula

$$\bigwedge_{S \subseteq \mathcal{X}_i \text{ with } X \notin S} \neg Y_i^S$$

and each occurrence of the literal $\neg X$ by the formula

$$\bigwedge_{S \subseteq \mathcal{X}_i \text{ with } X \in S} \neg Y_i^S.$$

Then an assignment $\mathcal{V} : \mathcal{X} \to \{\text{TRUE}, \text{FALSE}\}$ satisfies γ if and only if $f(\mathcal{V})$ satisfies β''. Thus γ is satisfiable if and only if β'' has a good assignment. Note that the size of each of the sets $\mathcal{Y}_i$ is $2^\ell = m$. Thus the size of β'' is polynomial in m. Moreover, β'' can easily be computed from γ in polynomial time.

β'' is not a $\Gamma_{t,d}$-formula: The transformation from γ to β'' has turned the small disjunctions $(\lambda_1 \vee \ldots \vee \lambda_d)$ on the bottom level of γ into formulas

$$\bigwedge_i \nu_{1i} \vee \ldots \vee \bigwedge_i \nu_{di}.$$

Applying the distributive law to all these subformulas turns them into big conjunctions of disjunctions of at most d literals, and since t is odd, it turns the whole formula β'' into a $\Gamma_{t,d}$-formula β'. Since d is fixed, the size only increases polynomially, and β' can be computed from β'' in polynomial time. And we still have: γ is satisfiable if and only if β' has a good assignment.

All that remains to do is add a subformula stating that all assignments of weight k are good. We let

$$\alpha = \bigwedge_{i=1}^{k} \bigwedge_{\substack{S,T \subseteq \mathcal{X}_i \\ S \neq T}} (\neg Y_i^S \vee \neg Y_i^T)$$

and $\beta = \alpha \wedge \beta'$. Then β is (equivalent to) a $\Gamma_{t,d}$-formula that satisfies (4.2).

Next, we prove $\mathrm{W}[t] \subseteq \mathrm{M}[t+1]$. For simplicity, let us assume again that t is odd. Let $d = 2$ if $t = 1$ and $d = 1$ otherwise. Recall that by Theorem 3.10, $\mathrm{WSAT}(\Gamma^-_{t,d})$ is $\mathrm{W}[t]$-complete. We shall prove that

$$p\text{-}\mathrm{WSAT}(\Gamma^-_{t,d}) \leq^{\mathrm{fpt}} p\text{-}log\text{-}\mathrm{SAT}(\Gamma_{t+1,1}). \tag{4.3}$$

We simply reverse the idea of the proof that $\mathrm{M}[t] \subseteq \mathrm{W}[t]$.

Let $\beta \in \Gamma^-_{t,d}$ and $k \geq 1$, say,

$$\beta = \bigwedge_{i_1 \in I_1} \bigvee_{i_2 \in I_2} \cdots \bigwedge_{i_t \in I_t} \delta(i_1, \ldots, i_t), \tag{4.4}$$

where each $\delta(i_1, \ldots, i_t)$ is a disjunction of at most d negative literals. Let $n = |\mathrm{var}(\beta)|$ and $\ell = \log n$, and let us assume again that ℓ is an integer. Furthermore, we assume that the variables of β are indexed with subsets of $\{1, \ldots, \ell\}$, or more precisely, that

$$\mathrm{var}(\beta) = \mathcal{Y} = \big\{ Y^S \;\big|\; S \subseteq \{1, \ldots, \ell\} \big\}.$$

For $1 \leq i \leq k$ and $1 \leq j \leq \ell$, let X_{ij} be a new variable. As above, let $\mathcal{X}_i = \{X_{ij} \mid 1 \leq j \leq \ell\}$ and $\mathcal{X} = \bigcup_{i=1}^k \mathcal{X}_i$. The idea is that every assignment to the variables in $\mathcal{X}_i$ corresponds to a subset $S_i \subseteq \{1, \ldots, \ell\}$ and hence to a variable Y^{S_i}. Thus an assignment to all variables in $\mathcal{X}$ corresponds to a subset $\{Y^{S_1}, \ldots, Y^{S_k}\} \subseteq \mathcal{Y}$ and hence to an assignment to the variables in $\mathcal{Y}$ of weight at most k ("at most" because the S_i are not necessarily distinct).

Formally, let g be the following mapping from the assignments for $\mathcal{X}$ to the assignments for $\mathcal{Y}$ of weight at most k: For every $\mathcal{V} : \mathcal{X} \to \{\textsc{true}, \textsc{false}\}$, we let $g(\mathcal{V}) : \mathcal{Y} \to \{\textsc{true}, \textsc{false}\}$ the assignment that sets $Y^{S_1}, \ldots, Y^{S_k}$ to TRUE and all other variables to FALSE, where for $1 \leq i \leq k$

$$S_i = \{j \mid \mathcal{V}(X_{ij}) = \textsc{true}\}.$$

Let γ'' be the formula obtained from β by replacing each literal $\neg Y^S$ by the subformula

$$\chi_S = \bigwedge_{i=1}^{k} \Big(\bigvee_{j \in S} \neg X_{ij} \vee \bigvee_{j \in \{1,\ldots,\ell\} \setminus S} X_{ij} \Big).$$

(Remember that all literals in β are negative.) Then for every assignment $\mathcal{V} : \mathcal{X} \to \{\textsc{true}, \textsc{false}\}$,

$$\mathcal{V} \text{ satisfies } \gamma'' \iff g(\mathcal{V}) \text{ satisfies } \beta.$$

The translation from β to γ'' turns every $\delta = \delta(i_1, \ldots, i_t)$ in (4.4) into a disjunction δ' of at most d formulas χ_S. Say,

$$\delta' = \big(\chi_{S_1} \vee \ldots \vee \chi_{S_d}\big)$$

By applying the distributive law, this formula can be turned into a conjunction χ of k^d disjunctions of $d \cdot \ell$ literals. Applying this operation to every δ' in γ'', we obtain an equivalent $\Gamma_{t+1,1}$-formula γ'. Then for every assignment $\mathcal{V} : \mathcal{X} \to \{\text{TRUE}, \text{FALSE}\}$,

$$\mathcal{V} \text{ satisfies } \gamma' \iff g(\mathcal{V}) \text{ satisfies } \beta.$$

This almost completes the proof. The only problem that remains to be solved is that not all assignments $g(\mathcal{V})$ have weight exactly k, because some of the induced S_i may be identical. Let

$$\alpha' = \bigwedge_{1 \le i < i' \le k} \bigvee_{j=1}^{\ell} \neg(X_{ij} \leftrightarrow X_{i'j}).$$

Then for every $\mathcal{V} : \mathcal{X} \to \{\text{TRUE}, \text{FALSE}\}$ that satisfies α', the assignment $g(\mathcal{V})$ has weight exactly k. Thus g induces a mapping from the assignments for $\mathcal{X}$ that satisfy α' onto the weight k assignments for $\mathcal{Y}$. Note that α' is equivalent to a $\Gamma_{2,1}$-formula α of size $O(k^2 \cdot 2^{2\ell}) = O(k^2 \cdot n^2)$. Furthermore, given k, n, such a formula α can be computed in time polynomial in k and n.

We let $\gamma = \alpha \wedge \gamma'$. Then γ is satisfiable if and only if β is k-satisfiable. The size m of γ is polynomial in the size of β, and the number of variables is $k \cdot \ell$, where $\ell = \log n \le \log m$. By adding dummy variables (to the outermost conjunction of γ) we can adjust the number of variables in such a way that $k = \lceil |\text{var}(\gamma)| / \log m \rceil$.

It remains to prove M[SAT] = W[SAT] and M[P] = W[P]. We can simply carry out the preceding constructions without worrying about the form of the resulting formulas. □

Corollary 4.5. *Let $t, d \ge 1$.*

(1) If $\text{W}[t] = \text{FPT}$ *then* $\text{SAT}(\Gamma_{t,d})$ *is subexponential.*

(2) If $\text{SAT}(\Gamma_{t+1,1})$ *is subexponential then* $\text{W}[t] = \text{FPT}$.

By a more refined argument based on the same idea, Chen et al. [4] strengthened part (1) of the corollary as follows:

Theorem 4.6 ([4]). *Let $t, d \geq 1$ such that $t + d \geq 3$. If*

$$\mathrm{WSAT}(\Gamma_{t,d}) \in \mathrm{DTIME}(f(k) \cdot n^{o^{\mathrm{eff}}(k)} \cdot m^{O(1)})$$

for some computable function f, then $\mathrm{SAT}(\Gamma_{t,d})$ is subexponential.

In [5], this has further been strengthened by restricting the range of values k for which the hypothesis is needed.

Let us briefly return to the connections between W[P] and limited nondeterminism. Recall Theorem 3.12. We encourage the reader to prove the direction (1) $\Longrightarrow$ (2); it follows easily from W[P] = M[P]. (The converse direction of the theorem is also not hard to prove.) Let us summarize our three characterizations of W[P] vs FPT in a corollary:

Corollary 4.7. *The following three statements are equivalent:*

(1) W[P] = FPT.

(2) SAT(CIRC) *is subexponential.*

(3) PTIME = NP$[\iota(n) \cdot \log n]$ *for some computable function $\iota : \mathbb{N} \to \mathbb{N}$ that is non-decreasing and unbounded.*

5. M[1] and Miniaturized Problems

Originally, the class M[1] was defined in terms of so-called *parameterized miniaturizations* of NP-complete problems [10]. Let $Q \subseteq \Sigma^*$ be any decision problem. We define:

p-mini-Q	
Instance:	$x \in \Sigma^*$ and $m \in \mathbb{N}$ in unary.
Parameter:	$\left\lceil \frac{\lvert x \rvert}{\log m} \right\rceil$.
Problem:	Decide if $x \in Q$.

We call *p-mini-Q* the "miniaturization" of Q, because if we assume the parameter $k = \lceil |x| / \log m \rceil$ to be small, then the size $|x| = \lfloor k \cdot \log m \rfloor$ of the actual instance is very small compared to the "padded size" $|x| + m$. There is an equivalent way of formulating the problem making this explicit:

Instance:	$x \in \Sigma^*$ and $k, m \in \mathbb{N}$ in unary such that $\lvert x \rvert = \lfloor k \cdot \log m \rfloor$.
Parameter:	k.
Problem:	Decide if $x \in Q$.

The main reason that we are interested in these strange problems is the following equivalence:

Proposition 5.1. *Let Σ be a finite alphabet and $Q \subseteq \Sigma^*$. Then*

$$p\text{-}mini\text{-}Q \in \mathrm{FPT} \iff Q \in \mathrm{DTIME}(2^{o^{\mathrm{eff}}(n)}),$$

where $n = |x|$ denotes the length of the instance x of Q.

We skip the proof, which is very similar to the proof of Proposition 4.1.

The main combinatorial tool in the development of a M[1]-completeness theory is the Sparsification Lemma due to Impagliazzo, Paturi, and Zane [19]. The lemma says that the satisfiability problem for d-CNF-formulas can be reduced to the satisfiability problem for d-CNF-formulas whose size is linear in the number of variables by a suitable reduction that preserves subexponential time solvability.

Lemma 5.2 (Sparsification Lemma [19]). *Let $d \geq 2$. There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that for every $k \in \mathbb{N}$ and every formula $\gamma \in d$-CNF with $n = |\mathrm{var}(\gamma)|$ variables there is a $\Delta_{2,d}$-formula*

$$\beta = \bigvee_{i=1}^{p} \beta_i$$

such that:

(1) β is equivalent to γ,

(2) $p \leq 2^{n/k}$,

(3) $|\beta_i| \leq f(k) \cdot n$ for $1 \leq i \leq p$.

Furthermore, there is an algorithm that, given γ and k, computes β in time $2^{n/k} \cdot |\gamma|^{O(1)}$.

The idea of using the Sparsification Lemma in this context goes back to Cai and Juedes [3].

Theorem 5.3 ([3, 10]). *p-mini-SAT(3-CNF) is M[1]-complete under fpt Turing reductions.*

Proof: To prove that p-mini-SAT(3-CNF) $\in$ M[1], we show that

$$p\text{-}mini\text{-}\mathrm{SAT}(3\text{-CNF}) \leq^{\mathrm{fpt}} p\text{-}log\text{-}\mathrm{Sat}(3\text{-CNF}).$$

Let (γ, m) be an instance of *p-mini*-SAT(3-CNF) and $k = |\gamma|/\log m$. By padding γ we obtain an equivalent formula γ' such that $\mathrm{var}(\gamma') = \mathrm{var}(\gamma)$ and $m' = |\gamma'| \geq m$. Then

$$k' = \frac{|\mathrm{var}(\gamma')|}{\log m'} \leq \frac{|\mathrm{var}(\gamma)|}{\log m} \leq \frac{|\gamma|}{\log m} = k.$$

Thus $(\gamma, m) \mapsto \gamma'$ is an fpt-reduction from *p-mini*-SAT(3-CNF) to *p-log*-Sat(3-CNF).

To prove hardness, we show that

$$\textit{p-log}\text{-SAT}(d\text{-CNF}) \leq^{\text{ftp-T}} \textit{p-mini}\text{-SAT}(3\text{-CNF})$$

for all $d \geq 3$. Fix $d \geq 3$.

Let $\gamma \in d$-CNF. Let $m = |\gamma|$, $n = |\mathrm{var}(\gamma)|$, and $k = \lceil n/\log m \rceil$.

Choose $f : \mathbb{N} \to \mathbb{N}$ (depending on d) and $\beta = \bigvee_{i=1}^{p} \beta_i$ (depending on γ and k) according to the Sparsification Lemma 5.2. Since

$$2^{n/k} = 2^{\frac{n}{\lceil n/\log m \rceil}} \leq m,$$

we have $p \leq m$, and β can be computed in time $m^{O(1)}$. Let $1 \leq i \leq p$. Since β_i has at most $f(k) \cdot n$ clauses, there is a 3-CNF-formula β_i' with at most $f(k) \cdot d \cdot n$ variables and of length $|\beta_i'| \in O(f(k) \cdot d \cdot n)$ such that

$$\beta_i \text{ is satisfiable} \iff \beta_i' \text{ is satisfiable.}$$

Thus γ is satisfiable if and only if there exists an $i, 1 \leq i \leq p$, such that β_i' is satisfiable.

For $1 \leq i \leq p$ we have

$$k_i' = \left\lceil \frac{|\mathrm{var}(\beta_i')|}{\log m} \right\rceil = O\left(\frac{f(k) \cdot d \cdot n}{\log m} \right) = O(f(k) \cdot d \cdot k).$$

The desired Turing reduction decides if $\gamma \in$ SAT(d-CNF) by querying the instances (β_i', m), for $1 \leq i \leq p$, of *p-mini*-SAT(3-CNF). □

Corollary 5.4. *p-log*-SAT(3-CNF) *is* M[1]*-complete under ftp Turing reductions.*

Proof: We have noted in the proof of Theorem 5.3 that *p-mini*-SAT(3-CNF) is fpt-reducible to *p-log*-SAT(3-CNF). □

Polynomial time reductions between problems do not automatically give fpt-reductions between their miniaturizations. Let us a call a polynomial time reduction R from a problem $Q \subseteq \Sigma^*$ to a problem $Q' \subseteq (\Sigma')^*$ *size preserving* if for all $x \in \Sigma^*$ we have $|R(x)| \in O(|x|)$.

Lemma 5.5. *Let $Q \subseteq \Sigma^*$ and $Q' \subseteq (\Sigma')^*$ such that there is a size preserving polynomial time reduction from from Q to Q'. Then there is an fpt-reduction from p-mini-Q to p-mini-Q'.*

Proof: If R is a size-preserving polynomial time reduction from Q to Q' then $(x, m) \mapsto (R(x), m)$ defines an fpt-reduction from *p-mini-*Q to *p-mini-*Q'. □

Corollary 5.6. *The following problems are* M[1]*-complete under fpt Turing reductions:*

(1) p-mini-d-COLORABILITY *for every* $d \geq 3$,

*(2) p-mini-*SAT(d-CNF) *for every* $d \geq 3$ *and p-mini-*SAT(CIRC).

Proof: The standard polynomial time reductions between d-COLORABILITY, SAT(CIRC), SAT(3-CNF) are size preserving. □

Lemma 5.7. *There is a size preserving polynomial time reduction from* WSAT(CIRC) *to* SAT(CIRC).

We skip the proof. The main idea is to use a linear size circuit to count the number of variables set to TRUE.

Corollary 5.8. *The following problems are* M[1]*-complete under fpt Turing reductions:*

*(1) p-mini-*INDEPENDENT-SET,

*(2) p-mini-*VERTEX-COVER,

*(3) p-mini-*WSAT(d-CNF) *for every* $d \geq 2$ *and p-mini-*WSAT(CIRC).

Proof: The standard reductions from SAT(3-CNF) to INDEPENDENT-SET, from INDEPENDENT-SET to VERTEX-COVER and vice versa, from INDEPENDENT-SET to WSAT(2-CNF), from WSAT(2-CNF) to *p-mini-*WSAT(CIRC) are all size preserving. Thus the equivalence of the problems here and in Corollary 5.6 follows from Lemma 5.7. □

It is an open problem if the miniaturization *p-mini-*SHORT-NTM-HALT of the following problem SHORT-NTM-HALT is M[1]-complete. The input Turing machine is supposed to have just one work tape (or a fixed number), but may have an arbitrary alphabet. The parameterization of this problem by n is known to be W[1]-complete [12].

SHORT-NTM-HALT	
Instance:	A nondeterministic Turing machine M and an $n \in \mathbb{N}$ in unary.
Problem:	Decide if M, started with the empty tape, halts in at most n steps.

Note that the standard reduction between CLIQUE and INDEPENDENT-SET is not size preserving. Actually, we have:

Corollary 5.9. *p-mini*-CLIQUE $\in$ FPT.

Proof: Observe that CLIQUE $\in$ DTIME$(n^{O(\sqrt{n})})$, where n is the size of the input. The reason is that a clique of size ℓ has $\Omega(\ell^2)$ edges and thus can only exist in a graph of size $\Omega(\ell^2)$.

By Proposition 5.1, this implies that *p-mini*-CLIQUE $\in$ FPT. □

Corollary 5.9 highlights how sensitive the whole theory is to the specific encoding of the input and our "size measure". For example, for graph problems it would also be natural to define the "size" of an instance to be the number of vertices. Then, obviously, there are "size"-preserving reductions between CLIQUE and INDEPENDENT-SET. To investigate the role of size measures, we define a *size measure* on Σ^* to be a polynomial time computable function $\nu : \Sigma^* \to \mathbb{N}$. Of course a size measure is just another parameterization. For now, we use a different term and different symbols to avoid confusion between the two. We will discuss the relation between size measures and parameterizations below.

Obviously, the actual input size, that is, $\nu(x) = |x|$ is a size measure, which we call the *standard size measure*. Other natural size measures are the number of vertices of a graph, that is,

$$\nu_{\text{vert}}(x) = \begin{cases} |V| & \text{if } x \text{ is the encoding of a graph } (V, E), \\ |x| & \text{otherwise,} \end{cases}$$

and the number of variables of a formula or circuit, that is,

$$\nu_{\text{var}}(x) = \begin{cases} |\text{var}(\gamma)| & \text{if } x \text{ is the encoding of a circuit } \gamma, \\ |x| & \text{otherwise,} \end{cases}$$

We let

p-mini[ν]-Q	
Instance:	$x \in \Sigma^*$ and $m \in \mathbb{N}$ in unary.
Parameter:	$\left\lceil \frac{\nu(x)}{\log m} \right\rceil$.
Problem:	Decide if $x \in Q$.

By essentially the same proof as that of Propositions 4.1 and 5.1, we obtain the following slightly more general result.

Proposition 5.10. *Let Σ be a finite alphabet and $Q \subseteq \Sigma^*$. Then*

$$p\text{-}mini[\nu]\text{-}Q \in \mathrm{FPT} \iff Q \in \mathrm{DTIME}(2^{o^{\mathrm{eff}}(\nu(x))} \cdot |x|^{O(1)}).$$

By using the Sparsification Lemma, it can be proved that:

$$\begin{aligned}
p\text{-}mini[\nu_{var}]\text{-}\textsc{Sat}(d\text{-CNF}) &\equiv^{\text{fpt-T}} p\text{-}mini\text{-}\textsc{Sat}(d\text{-CNF}) \\
p\text{-}mini[\nu_{var}]\text{-}\textsc{WSat}(d\text{-CNF}) &\equiv^{\text{fpt-T}} p\text{-}mini\text{-}\textsc{WSat}(d\text{-CNF}) \\
p\text{-}mini[\nu_{vert}]\text{-}d\text{-}\textsc{Colorability} &\equiv^{\text{fpt-T}} p\text{-}mini\text{-}d\text{-}\textsc{Colorability} \\
p\text{-}mini[\nu_{vert}]\text{-}\textsc{Independent-Set} &\equiv^{\text{fpt-T}} p\text{-}mini\text{-}\textsc{Independent-Set} \\
p\text{-}mini[\nu_{vert}]\text{-}\textsc{Vertex-Cover} &\equiv^{\text{fpt-T}} p\text{-}mini\text{-}\textsc{Vertex-Cover}.
\end{aligned}$$

Furthermore, we clearly have

$$p\text{-}mini[\nu_{vert}]\text{-}\textsc{Clique} \equiv^{\text{fpt-T}} p\text{-}mini[\nu_{vert}]\text{-}\textsc{Independent-Set}.$$

Thus, by Corollaries 5.6 and 5.8, all these problems are M[1]-complete. It is not known if *p-mini*[ν_{var}]-SAT(CIRC) or just *p-mini*[ν_{var}]-SAT(CNF) is reducible to *p-mini*-SAT(CIRC).

Let us re-iterate that a size measure and a parameterization are really the same thing (though introduced with different intentions). This becomes most obvious for the problems p-SAT(Γ), whose parameterization is just the size measure ν_{var} for SAT(Γ). Proposition 5.10 can be read as stating that *p-mini*[ν]-$Q \in$ FPT if and only if the parameterized problem (Q, ν) can be solved by a *subexponential fpt-algorithm*, that is, an fpt-algorithm whose running time is $2^{o^{\mathrm{eff}}(k)} \cdot n^{O(1)}$, where k is the parameter and n the input size.

A starting point for the whole theory was the question of whether the parameterized vertex cover problem p-VERTEX-COVER (cf. Example 3.1), which is easily seen to be solvable in time $O(2^k \cdot |G|)$, has a subexponential fpt-algorithm. Note that the parameterization of p-VERTEX-COVER is *not* the same as the size measure ν_{vert}. Nevertheless, it can be proved:

Theorem 5.11 ([3, 10]). p-VERTEX-COVER *has a subexponential fpt-algorithm if and only if* M[1] = FPT.

6. Miniaturizations of Problems in SNP

There is a more general principle behind the results of the previous section, which becomes apparent if we look at the *syntactic form* of the problems considered there: They all belong to the syntactically defined complexity class SNP [23]. In this section, we shall prove that essentially, the miniaturizations of all problems in SNP are in M[1]. Some care needs to be taken with regards to the size measure.

Let us first recall the definition of the class SNP. Instances of problems in SNP are *relational structures* such as graphs. Propositional formulas or circuits can also be encoded by relational structures. A problem is in SNP if it is *definable* by a formula φ of second-order logic of the form

$$\exists X_1 \dots \exists X_k \, \forall y_1 \dots \forall y_\ell \, \psi(X_1, \dots, X_k, y_1, \dots, y_\ell). \tag{6.1}$$

Here $X_1, \dots, X_k$ are *relation variables*, each with a prescribed arity, which range over relations over the universe of the input structure. $y_1, \dots, y_\ell$ are *individual variables*, which range over elements of the input structure. $\psi(X_1, \dots, X_k, y_1, \dots, y_\ell)$ is a *quantifier free formula*, that is, a Boolean combination of *atomic formulas* of the form $Rz_1 \dots z_r$ or $z_1 = z_2$, where $z_1, \dots, z_r \in \{y_1, \dots, y_\ell\}$ and R is either one of the relation variables $X_1, \dots, X_k$ or a relation symbol representing one of the relations of the structure (such as the edge relation of a graph). $Rz_1 \dots z_r$ is true in a structure under some interpretation of the variables if the tuple $(a_1, \dots, a_r)$ of elements interpreting the individual variables $(z_1, \dots, z_r)$ is contained in the relation interpreting R. Then the meaning of the whole formula is defined inductively using the usual rules for Boolean connectives and quantifiers.

For a structure A we write $A \models \varphi$ if φ holds in A. We can associate the following problem with φ:

D_φ	
Instance:	Structure A.
Problem:	Decide if $A \models \varphi$.

Slightly abusing notation, we use SNP to denote both the class of formulas of the form (6.1) and the class of all problems D_φ, where φ is a formula of the form (6.1).

Example 6.1. Let $d \geq 1$. The following SNP-formula χ states that a graph is d-colorable:

$$\underbrace{\exists X_1 \ldots \exists X_d}_{\substack{X_i \text{ is the set of} \\ \text{elements of color } i}} \forall x \forall y \Big(\underbrace{\bigvee_{i=1}^{d} X_i x \wedge \bigwedge_{1 \leq i < j \leq d} \neg(X_i x \wedge X_j x)}_{\text{“Each element has exactly one color.”}} \wedge \underbrace{\bigwedge_{1 \leq i \leq d} \big(Exy \to \neg(X_i x \wedge X_i y)\big)}_{\text{“Adjacent elements do not have the same color.”}} \Big).$$

Thus $\mathrm{D}_\chi = d$-Colorability and hence d-Colorability $\in$ SNP.

An SNP-formula as in (6.1) is *monadic* if the relation variables $X_1, \ldots, X_k$ are all unary, that is, range over subsets of the structure. MSNP denotes the class of all monadic SNP-formulas and at the same time the class of all problems defined by such formulas. For example, the formula in Example 6.1 is monadic, and thus d-Colorability $\in$ MSNP for all $d \geq 1$. It is also not hard to see that Sat(d-CNF) $\in$ MSNP for all $d \geq 1$.

We generalize the size measures ν_{vert} and ν_{var} to arbitrary input structures by letting

$$\nu_{\text{elt}}(x) = \begin{cases} n & \text{if } x \text{ is the encoding of a structure } A \text{ with } n \text{ elements,} \\ |x| & \text{otherwise,} \end{cases}$$

Then on graphs, $\nu_{\text{elt}} = \nu_{\text{vert}}$, and on d-CNF-formulas (if represented by structures in a standard way), $\nu_{\text{elt}} = \nu_{\text{var}}$.

Proposition 6.2. *For every problem $Q \in$ MSNP, the miniaturized problem p-mini[ν_{elt}]-Q is contained in the closure of* M[1] *under fpt Turing reductions.*

It shown in [6] that for every problem $Q \in$ MSNP the miniaturized problem *p-mini*[ν_{elt}]-Q is contained in W[1].

Problems such as Independent-Set or Vertex-Cover, at least if represented naturally, are not in SNP simply because the problem instances are not just structures (graphs), but pairs consisting of graphs and natural numbers. For such problems, we define a variant of SNP: Instead of formulas (6.1) we consider formulas $\varphi(X_0)$ of the form

$$\exists X_1 \ldots \exists X_k \, \forall y_1 \ldots \forall y_\ell \, \psi(X_0, X_1, \ldots, X_k, y_1, \ldots, y_\ell), \tag{6.2}$$

which have one additional relation variable occurring freely. Say, X_0 is s-ary. For a structure A and an s-ary relation S on A we write $A \models \varphi(S)$ if φ holds in A if X_0 is interpreted by S. We can associate the following problem with $\varphi(X_0)$:

WD_φ	
Instance:	A structure A and $k \in \mathbb{N}$.
Problem:	Decide if there is an s-ary relation S on A of size $\|S\| = k$ such that $A \models \varphi(S)$.

We use W-SNP to denote the class of all problems WD_φ, where φ is an SNP-formula of the form (6.2), and W-MSNP to denote the subclass of all problems WD_φ, where φ is an MSNP-formula.

Example 6.3. The following formula witnesses that Independent-Set $\in$ W-MSNP:

$$\forall y_1 \forall y_2 \Big(\big(X_0 y_1 \wedge X_0 y_2 \big) \rightarrow \neg E y_1 y_2 \Big),$$

where the binary relation symbol E represents the edge relation of the input graph.

Similarly, it can be shown that Vertex-Cover, Clique, and $\mathrm{WSat}(d\text{-CNF})$ for $d \geq 1$ are in W-MSNP.

Proposition 6.4. *For every problem $Q \in$ W-MSNP, the miniaturized problem p-mini$[\nu_{elt}]$-Q is contained in the closure of* M[1] *under fpt Turing reductions.*

Propositions 6.2 and 6.4 can be generalized to arbitrary (not necessarily monadic) SNP-formulas, but only under an unnatural size measure. For $r \geq 1$, let

$$\nu^r_{\mathrm{elt}}(x) = \begin{cases} n^r & \text{if } x \text{ is the encoding of a structure } A \text{ with } n \text{ elements,} \\ |x| & \text{otherwise,} \end{cases}$$

Call a formula of the form (6.1) or (6.2) *r-ary* if the maximum arity of the relations $(X_0), X_1, \ldots, X_k$ is r. Observe that monadic formulas are 1-ary. Propositions 6.2 and 6.4 generalize to r-ary formulas for every $r \geq 1$, but only under the size measure ν^r_{elt}.

For a thorough discussion of miniaturized problems (in particular syntactically defined problems such as those in SNP and W-SNP) under various size measures we refer the reader to [6].

7. The Exponential Time Hypothesis

We are now ready to apply the results of the previous sections in a more conventional setting.

In this section, we assume that d-CNF-formulas contain no repeated clauses and are thus of size $m = O(n^d)$ (for fixed d).[3] In particular, for every computable function $f(n) \in \Omega(\log n)$, this implies that

$$\textsc{Sat}(d\text{-CNF}) \in \mathrm{DTIME}(2^{O(f(n))}) \Longleftrightarrow \textsc{Sat}(d\text{-CNF}) \in \mathrm{DTIME}\big(2^{O(f(n))} \cdot m^{O(1)}\big).$$

We are concerned here with the "effective" version of the exponential time hypothesis:

$$\textsc{Sat}(3\text{-CNF}) \notin \mathrm{DTIME}(2^{o^{\mathrm{eff}}(n)}) \tag{ETH}$$

Recall that by Proposition 5.1 and Corollary 5.6 we have

$$(\mathrm{ETH}) \iff \mathrm{M}[1] \neq \mathrm{FPT}.$$

We say that a problem $Q \subseteq \Sigma^*$ is *subexponential with respect to a size measure* $\nu : \Sigma^* \to \mathbb{N}$ if there is an algorithm deciding $x \in Q$ in time

$$2^{o^{\mathrm{eff}}(\nu(x))} \cdot |x|^{O(1)}.$$

The negation of (ETH) will be denoted by $\neg$(ETH). The results of the previous two sections yield the following two corollaries:

Corollary 7.1 ([19]). *$\neg$(ETH) is equivalent to either of the following problems being subexponential:*

(1) $\textsc{Sat}(d\text{-CNF})$ *for $d \geq 3$ with respect to the standard size measure and ν_{var}.*

(2) $\textsc{WSat}(d\text{-CNF})$ *for $d \geq 2$ with respect to the standard size measure and ν_{var}.*

(3) $\textsc{Sat}(\mathrm{CIRC})$ *and* $\textsc{WSat}(\mathrm{CIRC})$ *with respect to the standard size measure.*

(4) d-$\textsc{Colorability}$ *for $d \geq 3$ with respect to the standard size measure and ν_{vert}.*

(5) $\textsc{Independent-Set}$ *with respect to the standard size measure and ν_{vert}.*

(6) $\textsc{Clique}$ *with respect to ν_{vert}.*

[3] Some care needs to be taken with this assumption because the proofs of some of the earlier results involve padding arguments that are no longer available if we make the assumption. The reader may be assured that we take care here.

(7) VERTEX-COVER *with respect to the standard size measure and* ν_{vert}.

It can further be proved that INDEPENDENT-SET restricted to graphs of degree at most 3 is equivalent to INDEPENDENT-SET on arbitrary graphs with respect to subexponential solvability [21].

Corollary 7.2 ([19]). $\neg$*(ETH) implies that all problems in MSNP and W-MSNP are subexponential with respect to* ν_{elt}.

As Propositions 6.2 and 6.4, Corollary 7.2 can be generalized from monadic to arbitrary SNP-problems for the size measures ν^r_{elt}.

The fixed-parameter tractable Turing reductions between the miniaturized problems that we gave in Section 5 can be translated to "subexponential" reductions between the corresponding classical problems (so-called *SERF-reductions* as defined in [19]), and it follows that the problems mentioned in Corollary 7.1 are complete for MSNP or W-MSNP, respectively, under such reductions.

In view of the previous section, there is a natural generalization of (ETH) to $t \geq 1$:

$$\exists d \geq 1 : \ \text{SAT}(\Gamma_{t,d}) \notin \text{DTIME}(2^{o^{\text{eff}}(n)} \cdot m^{O(1)}) \qquad (\text{ETH}_t)$$

Then (ETH) = (ETH_1). By Corollary 4.3, for all $t \geq 1$ we have

$$(\text{ETH}_t) \iff \text{M}[t] = \text{FPT}.$$

Not much is known about (ETH_t) for $t \geq 2$. As a matter of fact, it is not even known if

$$\text{SAT}(\text{CNF}) \in \text{DTIME}(2^{\varepsilon \cdot n} \cdot m^{O(1)}) \qquad (7.1)$$

for some constant $\varepsilon < 1$. Suppose that (ETH) holds. Then for every $d \geq 3$ there exists a positive constant

$$\varepsilon_d = \inf \left\{ \varepsilon > 0 \;\middle|\; \text{SAT}(d\text{-CNF}) \in \text{DTIME}(2^{\varepsilon \cdot n} \cdot m^{O(1)}) \right\}.$$

It is known that $\varepsilon_d < 1$ for all $d \geq 3$ and that the sequence is $(\varepsilon_d)_{d \geq 3}$ is non-decreasing and not ultimately constant [18]. The latter is a fairly deep result; its proof combines the Sparsification Lemma [19] with techniques for the SAT(d-CNF)-algorithm due to Paturi, Pudlak, Saks, and Zane [24].

It is an open problem if $\lim_{d \to \infty} \varepsilon_d = 1$. Of course, if $\lim_{d \to \infty} \varepsilon_d = 1$ then there is no constant $\varepsilon < 1$ satisfying (7.1). It is not known if the converse of this statement also holds.

8. Open Problems

First of all, it would be very nice to prove that the W-hierarchy and the M-hierarchy coincide on each level. In particular, if $\mathrm{M}[1] = \mathrm{W}[1]$ then the exponential time hypothesis would be equivalent to $\mathrm{FPT} \neq \mathrm{W}[1]$, which we may interpret as new evidence for the exponential time hypothesis. While the question of whether $\mathrm{M}[1] = \mathrm{W}[1]$ has received a lot of attention in the parameterized complexity community, the question of whether $\mathrm{M}[t] = \mathrm{W}[t]$ for $t \geq 2$ has not been looked at very intensely (and may in fact be easier, as the classes get more robust on higher levels). It is also conceivable that $\mathrm{M}[t+1] = \mathrm{W}[t]$ for $t \geq 1$.

A second interesting question is whether the M[1]-completeness of the problem *p-log*-$\textsc{Sat}(\Gamma_{1,3})$ (which may be viewed as a normal form result for M[1]) has analogues for higher levels of the hierarchy. The result one would hope for is that *p-log*-$\textsc{Sat}(\Gamma_{t,1})$ is $\mathrm{M}[t]$-complete for $t \geq 2$. Essentially the same question is whether (ETH_t) is equivalent to the statement

$$\textsc{Sat}(\Gamma_{t,1}) \notin \mathrm{DTIME}(2^{o^{\mathrm{eff}}(n)} \cdot m^{O(1)})$$

Proving such a result would probably require some form of a Sparsification Lemma for the higher levels, an interesting problem in itself. Of course one could also try to eliminate the use of the Sparsification Lemma from the proof of the M[1]-completeness of *p-log*-$\textsc{Sat}(\Gamma_{1,3})$ and possibly even prove completeness under fpt many-one reductions (instead of Turing reductions).

And finally, it is a notorious open question in parameterized complexity theory if a collapse such as $\mathrm{W}[t] = \mathrm{FPT}$ on some level t of the W-hierarchy has any implications for the higher levels (ideally, implies $\mathrm{W}[t'] = \mathrm{FPT}$ for all t'). In view of the entanglement of the W-hierarchy and the M-hierarchy, one possible approach to this question would be to prove a corresponding result for the M-hierarchy. An equivalent formulation of the question for the M-hierarchy is whether $\neg(\mathrm{ETH}_t)$ implies $\neg(\mathrm{ETH}_{t'})$ for $t' > t$.

References

[1] 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*, 73:235–276, 1995.

[2] L. Cai, J. Chen, R.G. Downey, and M.R. Fellows. On the structure of parameterized problems in NP. *Information and Computation*, 123:38–49, 1995.

[3] L. Cai and D. Juedes. On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences*, 67(4):789–807, 2003.

[4] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and G. Xia. Tight lower bounds for certain parameterized NP-hard problems. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 150–160, 2004.

[5] J. Chen, X. Huang, I. Kanj, and G. Xia. Linear fpt reductions and computational lower bounds. In *Proceedings of the 36th ACM Symposium on Theory of Computing*, pages 212–221, 2004.

[6] Y. Chen and J. Flum. On miniaturized problems in parameterized complexity theory. In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation*, 2004.

[7] Y. Chen, J. Flum, and M. Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, pages 13–29, 2003.

[8] E. Dantsin, A. Goerdt, E. A. Hirsch, R. Kannan, J. M. Kleinberg, Ch. H. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*, 289(1):69–83, 2002.

[9] R. Downey. Parameterized complexity for the skeptic. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, 2003.

[10] R. Downey, V. Estivill-Castro, M. Fellows, E. Prieto-Rodriguez, and F. Rosamond. Cutting up is hard to do: the parameterized complexity of k-cut and related problems. In J. Harland, editor, *Proceedings of the Australian Theory Symposium*, volume 78 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science Publishers, 2003.

[11] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24:873–921, 1995.

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

[13] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer-Verlag, 1999.

[14] U. Feige and J. Kilian. On limited versus polynomial nondeterminism. *Chicago Journal of Theoretical Computer Science*, 1997. Available at http://cjtcs.cs.uchicago.edu/.

[15] J. Flum, M. Grohe, and M. Weyer. Bounded fixed-parameter tractability and $\log^2 n$ nondeterministic bits. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming*, volume 3142 of *Lecture Notes in Computer Science*, pages 555–567. Springer-Verlag, 2004.

[16] J. Goldsmith, M. Levy, and M. Mundhenk. Limited nondeterminism. *SIGACT News*, 1996.

[17] J. Hromkovič. *Algorithmics for Hard Problems.* Springer-Verlag, 2nd edition, 2003.

[18] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

[19] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

[20] K. Iwama and S. Tamaki. Improved upper bounds for 3-sat. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 328, 2004.

[21] D.S. Johnson and M. Szegedy. What are the least tractable instances of max independent set? In *Proceedings of the 10th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 927–928, 1999.

[22] C. Kintala and P. Fischer. Refining nondeterminism in relativised polynomial time bounded computations. *SIAM Journal on Computing*, 9:46–53, 1980.

[23] C.H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[24] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 628–637, 1998.

[25] U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 410–414, 1999.

[26] R. E. Stearns and H. B. Hunt III. Power indices and easier hard problems. *Mathematical Systems Theory*, 23:209–225, 1990.

[27] G.J. Woeginger. Exact algorithms for NP-hard problems: A survey. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization - Eureka, You Shrink!, Papers Dedicated to Jack Edmonds, 5th International Workshop*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer Verlag, 2001.