Electronic Colloquium on Computational Complexity, Report No. 11 (2006)

# An Isomorphism between Subexponential and Parameterized Complexity Theory

Yijia Chen
BASICS, Department of Computer Science
Shanghai Jiaotong University
Shanghai 200030, China
yijia.chen@cs.sjtu.edu.cn

Martin Grohe
Institut für Informatik
Humboldt-Universität
Unter den Linden 6, 10099 Berlin, Germany
grohe@informatik.hu-berlin.de

**Abstract**

We establish a close connection between (sub)exponential time complexity and parameterized complexity by proving that the so-called miniaturization mapping is a reduction preserving isomorphism between the two theories.

ISSN 1433-8092

## 1. Introduction

The area of exact algorithms for hard algorithmic problem has received considerable attention in recent years (see, e.g., [15, 16]). The goal is to design exact, as opposed to approximative, algorithms for hard (usually NP-complete) problems that are better than the trivial brute force algorithms, but may still be exponential. For example, the currently best deterministic algorithm for the 3-satisfiability problem (3-SAT) due to Brueggemann and Kern [3] has a running time of $O(1.473^n)$, and the best randomized algorithm due to Iwama and Tamaki [14] has a running time of $O(1.324^n)$. Here $n$ denotes the number of variables of the input formula.

**Subexponential complexity theory.** A more qualitative question that has been recognized as central for the emerging theory of "subexponential time complexity" is whether 3-SAT can be solved in time $2^{o(n)}$. The hypothesis that this is not possible is known as the *exponential time hypothesis (*ETH*)*. It has first been studied systematically by Impagliazzo, Paturi, and Zane [13], who proved that the hypothesis is very robust and equivalent to analogous hypotheses for many other NP-complete problems. For example, ETH is equivalent to the question whether the INDEPENDENT-SET problem can be solved in time $2^{o(n)}$, where $n$ denotes the number of vertices of the input graph. The reader may wonder why the running time is measured in terms of the number of variables and number of vertices, respectively, and not in the actual input size. The main contribution of Impagliazzo et al. is to prove that the ETH is independent of the "size measure" (number of variables or actual size), that is, 3-SAT is solvable in time $2^{o(n)}$ if and only if it is solvable in time $2^{o(m)}$ for the input size $m$.[1] It is easy to see that this implies the corresponding result for INDEPENDENT-SET and a number other further problems. However, in general subexponential solvability is very sensitive with respect to the size measure, as the trivial example of the CLIQUE problem shows: Clearly, CLIQUE is solvable in time $2^{o(n)}$ if and only if INDEPENDENT-SET is, but CLIQUE is trivially solvable in time $n^{O(\sqrt{m})} = 2^{o(m)}$, where $m$ denotes the size of the input graph.

The example of the CLIQUE problem illustrates that it is reasonable for a theory of subexponential time complexity to view problems as pairs $(P, \nu)$ consisting of a decision problem $P \subseteq \Sigma^*$ over some finite alphabet $\Sigma$ and a mapping $\nu : \Sigma^* \to \mathbb{N}$, which may be viewed as the *size measure*. For technical reasons, the mapping $\nu$ is required to be polynomial time computable. Typical size measures are the number of vertices or the number of edges for graph problems, or the number of variables for satisfiability problems. Of course there is always the trivial size measure $\nu(x) = |x|$. At first sight, it seems reasonable to also require a size measure $\nu$ to be polynomially related to the input length, that is, $|x|^c \le \nu(x) \le |x|^d$ for some $c, d > 0$ and all $x \in \Sigma^*$. However, we prefer *not* to make this additional requirement, because there are problems where very natural "size measures" do not fulfill it. The most important example is the satisfiability problem SAT for arbitrary CNF formulas. The number of variables still seems one of the most natural complexity parameters, but is may be exponentially smaller than the input length. Hypergraph problems provide further natural examples; here the number of vertices is a natural parameter. Of course for such example it is no longer appropriate to call $\nu$ a "size measure", but other than that the role of $\nu$ remains the same.

But what would it mean for SAT to be subexponential with respect to "number of variables?" Clearly, SAT is not solvable in time $2^{o(n)}$, because the size $m$ of the input formula may be $2^{\Omega(n)}$. The natural question is whether SAT is solvable in time $2^{o(n)} \cdot m^{O(1)}$. More generally, we say that a problem $(P, \nu)$ is *subexponential* if it is solvable in time

$$2^{o(\nu(x))} \cdot |x|^{O(1)} \tag{1}$$

for every instance $x$. Of course, if $\nu$ is polynomially related to the input length, then the term $|x|^{O(1)}$ is dominated by $2^{o(\nu(x))}$ and can hence be omitted. We denote the class of all subexponential problems by SUBEPT. We are interested in the dividing line between exponential and subexponential solvability; the problems we consider are

[1]Let us remark that here we assume that 3-CNF formulas have no repeated clauses and hence that $m = O(n^3)$. Otherwise the input size would not be bounded in terms of $n$ and hence the problem would not be solvable in time $f(n)$ for any function $f$. Later, this problem will disappear because we will always admit a polynomial of the input size as a factor of the running time.

usually (trivially) solvable in time $2^{O(\nu(x))} \cdot |x|^{O(1)}$. Let us denote the class of all these problems by EPT.[2] The main advantage of our "parameterized" approach to subexponential complexity is that it makes it easier to compare problems that otherwise would not be by "rescaling" them. For example, PLANAR-INDEPENDENT-SET is solvable in time $2^{O(\sqrt{n})}$, and the question of whether it is solvable in time $2^{o(\sqrt{n})}$ is equivalent to the question of whether INDEPENDENT-SET is solvable in time $2^{o(n)}$. If we rescale PLANAR-INDEPENDENT-SET by letting $\nu(n) = \sqrt{n}$, we actually obtain a problem that is equivalent to INDEPENDENT-SET with the size measure "number of variables" under suitable reductions. Thus by specifying size measures, we get a more robust theory.

To develop a complexity theory, we need suitable reductions. Impagliazzo et al. [13] introduced so called *subexponential reduction families*, which are a form of Turing reductions that preserve subexponential solvability. We essentially work with these reductions, and also with a corresponding notion of many-one reductions.

The goals of *subexponential complexity theory* may now be stated as classifying concrete problems within this framework and investigating the structure of the resulting complexity classes.

**Parameterized complexity theory.** The reader familiar with parameterized complexity theory will have noticed that we are dealing with exactly the same type of parameterized problems as considered there — a *parameterized problem* is just a pair $(Q, \kappa)$, where $Q \subseteq \Sigma^*$ for some finite alphabet $\Sigma$ and $\kappa : \Sigma^* \to \mathbb{N}$, the *parameterization*, is polynomial time computable. We usually denote parameterized problems in the "subexponential world" by $(P, \nu)$ and problems in the "parameterized world" by $(Q, \kappa)$ because it will be important for us to separate the two "worlds", but formally both are the same. However, the questions asked in parameterized complexity theory are different; parameterized complexity theory's main intention is to address complexity issues in situations where the parameter is expected to be small compared to the input size, whereas in the subexponential theory the parameter was introduced as a size measure. A problem $(Q, \kappa)$ is *fixed-parameter tractable* if it can be solved in time $f(\kappa(x)) \cdot |x|^{O(1)}$, where $f$ is an arbitrary computable function. The class of all fixed-parameter tractable problems is denoted by FPT. There are corresponding notions of *fpt (many-one) reduction* and *fpt Turing reduction*.

As opposed to subexponential complexity theory, parameterized complexity theory has been developed in depth over the last fifteen years. A rich and maybe a bit unwieldy structure has emerged. The most important parameterized complexity classes are those of the *W-hierarchy*. Most natural parameterized problems are complete for one of these classes. Parameterized complexity theory almost exclusively studies problems which are in the class XP of all problems that can be solved in polynomial time for every fixed parameter value.

**Our results.** *Our main result is that subexponential and parameterized complexity theory are isomorphic.* Let us explain what we mean by this: On the subexponential side, we consider the partial order induced by subexponential reduction families on the *degrees*, that is, equivalence classes under subexponential reduction families. On the parameterized side, we consider the partial order induced by fpt-reductions on the corresponding degrees. We define a mapping, the so-called *miniaturization mapping*, that associates a problem $(Q, \kappa)$ with every problem $(P, \nu)$ and prove that this mapping is an isomorphism between the partial order of degrees inside EPT under subexponential reduction families and the partial order of degrees inside XP under fpt-reductions. This result holds for both many-one and Turing reductions. In particular, miniaturization maps the class SUBEPT (the lowest "subexponential degree") to FPT (the lowest "parameterized degree") and EPT to XP.

If we look at degrees outside of EPT, then the miniaturization mapping is still an embedding, but we prove that it is no longer onto. That is, there are parameterized degrees outside XP that are not in the image of the miniaturization mapping. Technically, this is our most difficult result.

The precise technical statement of our results requires some additional uniformity conditions. Essentially, the "little-oh" in (1) has to be interpreted "effectively" (see 2.1 for a precise statement). Alternatively, the uniformity condition in the definition of fixed-parameter tractability, requiring $f$ to be computable, can be relaxed.

The miniaturization mapping is a fairly natural mapping between parameterized problems that has been stud-

[2]This terminology has been introduced in [12] in the context of parameterized complexity theory, it will be explained in Section 2.1.

ied before [4, 7, 9], albeit not as an abstract mapping between parameterized problems, but just as a transformation between concrete problems such as vertex cover. As a matter of fact, there already is a body of work in parameterized complexity theory, starting with Abrahamson, Downey, and Fellows [1], that studies the relation between fixed-parameter tractability and subexponential time complexity [4, 5, 6, 7, 9, 11]. Notably, it follows from this work that the miniaturization mapping maps the degree of 3-SAT with the "number of variables" size measure to a class M[1] between FPT and the first level W[1] of the W-hierarchy. The degree of SAT is mapped to a class M[2] between W[1] and W[2], and the degree of the satisfiability problem for Boolean circuits under the "number of input nodes" size measure is mapped to the class W[P]. This shows that the miniaturization isomorphism is not just an abstract mapping between partial orders, but actually is a meaningful mapping between concrete problems and complexity classes.

## 2. Preliminaries

In this section we give the necessary background of subexponential and parameterized complexity. For more comprehensive details the reader is referred to [10, 11, 13].

**2.1. Subexponential complexity.** Recall that we study *parameterized problems* of the form $(P, \nu)$, where $P \subseteq \Sigma^*$ for some finite alphabet $\Sigma$ and $\nu : \Sigma^* \to \mathbb{N}$ is polynomial time computable. The mapping $\nu$ is called a *parameterization*; in the context of the subexponential theory, it is often referred to as the *size measure*. The most obvious size measure for a problem $P \subseteq \Sigma^*$ is the *length of the input* $\nu(x) = |x|$. Unfortunately, for most natural problems, the length of the input is not exactly what we think of as its "size". For example, we are used to think of the size of a graph $G$ with $n$ vertices and $m$ edges as being $(m + n)$ rather than $\Theta(n + m \cdot \log n)$, which would be the length of a reasonable encoding of $G$ over a finite alphabet. To abstract from such a heavy dependence on coding issues, we define the *size* $||G||$ of a graph $G$ with $n$ vertices and $m$ edges to $m + n$. Hence we distinguish between "size" and "length (of an encoding)" of a graph. Similarly, we define the size $||C||$ of a Boolean circuit to be the number of gates plus the number of lines. We view Boolean formulas as special circuits, so they inherit the size measures for circuits. For example, for a CNF-formula $\alpha = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} \lambda_{ij}$ this means that $||\alpha|| = \Theta(\sum_{i=1}^{m} k_i)$. We denote the parameterization of a graph problem or satisfiability problem $P$ by the input size by *s*-$P$. For example, we let

*s*-SAT

| | |
|---|---|
| *Instance:* | A CNF-formula $\alpha$. |
| *Parameter:* | $\|\|\alpha\|\|$. |
| *Problem:* | Decide whether $\alpha$ is satisfiable. |

Other natural size measures are the "number of vertices" for graph problems and the "number of variables" for satisfiability problems. As examples, consider the following problems:

*s-vert*-INDEPENDENT-SET

| | |
|---|---|
| *Instance:* | A graph $G = (V, E)$ and a $\ell \in \mathbb{N}$. |
| *Parameter:* | $\|V\|$. |
| *Problem:* | Decides whether $G$ has an independent set of cardinality $\ell$. |

*s-var*-3-SAT

| | |
|---|---|
| *Instance:* | A 3-CNF-formula $\alpha$. |
| *Parameter:* | Number of variables of $\alpha$. |
| *Problem:* | Decide whether $\alpha$ is satisfiable. |

Similarly, we can define the graph problems *s-vert*-CLIQUE, *s-vert*-VERTEX-COVER, *s-vert*-DOMINATING-SET and the satisfiability problems *s-var*-SAT (satisfiability of CNF-formulas), *s-var*-CIRCUIT-SAT (satisfiability of Boolean circuits, parameterized by the number of input gates).

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

$$f = O\left(\frac{g}{\iota}\right).$$

**Definition 1.** A parameterized problem $(P, \nu)$ is *subexponentially solvable* if there is an algorithm $\mathbb{A}$ that for every instance $x$ decides whether $x \in P$ in time

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

By $2^{o^{\text{eff}}(\nu(x))}$ we mean $2^{f(x)}$ for some function $f \in o^{\text{eff}}(\nu)$.

SUBEPT denotes the class of all subexponentially solvable problems.

As mentioned in the introduction, an example of a problem in SUBEPT is *s*-CLIQUE, the clique problem parameterized by the input size, which is solvable in time $2^{O(\sqrt{m} \cdot \log m)}$, where $m$ denotes the size of the input graph. Other, less trivial examples of problems in SUBEPT are the planar restrictions of many standard optimization problems parameterized by the number of vertices (or the size, which is equivalent for planar graphs). For example, the problems *s-vert*-PLANAR-VERTEX-COVER, *s-vert*-PLANAR-INDEPENDENT-SET, and *s-vert*-PLANAR-DOMINATING-SET are all solvable in time $2^{O(\sqrt{n})}$ and hence in SUBEPT [2]. However, most natural NP-complete problems do not seem to be in SUBEPT. To establish a completeness theory giving evidence to claims of problems not being in SUBEPT, we need an appropriate notion of reduction. The following lemma offers an alternative characterization of SUBEPT, which is the basis of the reductions we shall introduce afterwards.

**Lemma 2.** *Let $(P, \nu)$ be a parameterized problem over the alphabet $\Sigma$. The following are equivalent:*

*(1)* $(P, \nu) \in$ SUBEPT.

*(2) There is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$ and a computable function $f$ such that for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$, $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$.*

*(3) There is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$, a computable function $f$, and a constant $c \in \mathbb{N}$, such that for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$, $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{c \cdot \nu(x)/\ell} \cdot |x|^{O(1)}$.*

*Proof:* $(1) \Rightarrow (2)$: Assume $(P, \nu) \in$ SUBEPT. Let $\iota$ be a computable function that is non-decreasing and unbounded, and let $\mathbb{A}$ be an algorithm deciding $x \in P$ in time $2^{c \cdot \nu(x)/\iota(\nu(x))} \cdot |x|^{O(1)}$ for some constant $c \in \mathbb{N}$. For $\ell \in \mathbb{N}$, let $n(\ell) := \max\big(\{n \mid \iota(n) < c \cdot \ell\} \cup \{1\}\big)$ and $f(\ell) := 2^{c \cdot n(\ell)}$. Then for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ we have $2^{c \cdot \nu(x)/\iota(\nu(x))} \cdot |x|^{O(1)} \leq f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$. Let $\mathbb{A}'$ be the algorithm that, given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, simply ignores $\ell$ and simulates $\mathbb{A}$ on input $x$. Then $\mathbb{A}'$ and $f$ satisfy the conditions of (2).
The direction from (2) to (3) is trivial. We turn to $(3) \Rightarrow (1)$: Let $f : \mathbb{N} \to \mathbb{N}$ be a computable function, $c \in \mathbb{N}$ a constant, and $\mathbb{A}$ an algorithm that, given $(x, \ell) \in \Sigma^* \times \mathbb{N}$, decides if $x \in P$ in time $f(\ell) \cdot 2^{c \cdot \nu(x)/\ell} \cdot |x|^{O(1)}$. Without loss of generality, we may assume $f$ is increasing and time-constructible. Let $\iota : \mathbb{N} \to \mathbb{N}$ be the following computable function: $\iota(n) := \max\big(\{\ell \mid f(\ell) < n\} \cup \{1\}\big)$, which is clearly non-decreasing, unbounded, and computable in time polynomial in $n$. Let $\mathbb{A}'$ be the following algorithm for deciding $P$: Given $x \in \Sigma^*$, first compute $n := \nu(x)$ and $\ell := \iota(n)$, and then simulate $\mathbb{A}$ on $(x, \ell)$. The running time of $\mathbb{A}$ is bounded by

$$\begin{aligned} &|x|^{O(1)} + n^{O(1)} + f(\iota(n)) \cdot 2^{c \cdot n/\iota(n)} \cdot |x|^{O(1)} \\ \leq\ &|x|^{O(1)} + n^{O(1)} + O(n) \cdot 2^{o^{\text{eff}}(n)} \cdot |x|^{O(1)} \ = \ 2^{o^{\text{eff}}(n)} \cdot |x|^{O(1)}. \end{aligned}$$

□

Our notion of reduction is essentially that of *subexponential reduction families*, as introduced in [13]. The reduction families in [13] are Turing reductions; we also introduce a many-one version.

**Definition 3.** Let $(P, \nu)$ and $(P', \nu')$ be parameterized problems over the alphabets $\Sigma$ and $\Sigma'$, respectively.

(1) A *subexponential reduction family,* or simply serf-reduction, from $(P, \nu)$ to $(P', \nu')$ is a mapping $R : \Sigma^* \times \mathbb{N} \to (\Sigma')^*$, such that:

  (a) For all $(x, \ell) \in \Sigma^* \times \mathbb{N}$ we have $\big(x \in P \iff R(x, \ell) \in P'\big)$.

  (b) Given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, $R(x, \ell)$ is computable in time
$$f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$$
  for a computable function $f : \mathbb{N} \to \mathbb{N}$.

  (c) There is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that
$$\nu'(R(x, \ell)) \leq g(\ell) \cdot (\nu(x) + \log |x|)$$
  for all $(x, \ell) \in \Sigma^* \times \mathbb{N}$.

(2) A *subexponential Turing reduction family,* or serf Turing reduction, from $(P, \nu)$ to $(P', \nu')$ is an algorithm $\mathbb{A}$ with an oracle to $P'$ such that there are computable functions $f, g : \mathbb{N} \to \mathbb{N}$ with:

  (a) Given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathbb{A}$ decides if $x \in P$ in time
$$f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}.$$

  (b) For all oracle queries "$y \in P'$?" posed by $\mathbb{A}$ on input $(x, \ell) \in \Sigma^* \times \mathbb{N}$ we have
$$\nu'(y) \leq g(\ell) \cdot (\nu(x) + \log |x|).$$

We write $(P, \nu) \leq^{\text{serf}} (P', \nu')$ (or $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$) if there is a serf-reduction (or serf Turing reduction, respectively), from $(P, \nu)$ to $(P', \nu')$. It is clear that $(P, \nu) \leq^{\text{serf}} (P', \nu')$ implies $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$. We write $(P, \nu) \equiv^{\text{serf}} (P', \nu')$ if $(P, \nu) \leq^{\text{serf}} (P', \nu')$ and $(P', \nu') \leq^{\text{serf}} (P, \nu)$.

It is not completely trivial that serf reductions preserve subexponential solvability.

**Proposition 4.** *Let $(P, \nu)$ and $(P', \nu')$ be parameterized problems. If $(P, \kappa) \leq^{\text{serf-T}} (P', \nu')$ and $(P', \nu') \in$ SUBEPT, then $(P, \nu) \in$ SUBEPT.*

A proof can be found in Appendix B.1.

**Example 5.** For every graph problem $P$ we trivially have *s*-$P \leq^{\text{serf}}$ *s-vert*-$P$, as a reduction family we can simply use the mapping $R(x, \ell) = x$. Similarly, for every satisfiability problem $P$ we have *s*-$P \leq^{\text{serf}}$ *s-var*-$P$.

It is a highly nontrivial result due to Impagliazzo et al. [13] that for many natural graph and satisfiability problem, the converse also holds. As a matter of fact, Impagliazzo et al. proved that the following problems are all equivalent with respect to serf Turing reductions: *s*-3-SAT, *s-var*-3-SAT, *s*-INDEPENDENT-SET, *s-vert*-INDEPENDENT-SET, *s*-VERTEX-COVER, *s-vert*-VERTEX-COVER, *s*-DOMINATING-SET, *s-vert*-DOMINATING-SET, and *s-vert*-CLIQUE.

Note that *s*-CLIQUE does not appear in this list of problems. Indeed, it seems unlikely that *s*-CLIQUE is equivalent to the problems above because it is in SUBEPT, whereas the other problems are not in SUBEPT unless the exponential time hypothesis (mentioned in the introduction) fails.

While unlikely to belong to SUBEPT, all problems considered in the previous example belong to the class EPT:

**Definition 6.** A parameterized problem $(P, \nu)$ is in EPT if there is an algorithm $\mathbb{A}$ that, for every instance $x$, decides if $x \in P$ in time $2^{O(\nu(x))} \cdot |x|^{O(1)}$.

It follows from the *Time Hierarchy Theorem* that SUBEPT is a *proper* subclass of EPT. The next proposition shows that EPT is closed under serf Turing reductions.

**Proposition 7.** *Let* $(P, \nu)$ *and* $(P', \nu')$ *be parameterized problems. If* $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$ *and* $(P', \nu') \in$ EPT, *then* $(P, \nu) \in$ EPT.

A proof can be found in Appendix B.2.

The following example introduces another problem that will turn out to be complete for the class EPT under serf-reduction.

**Example 8.** Consider the halting problem for alternating Turing machines with binary alphabet parameterized by the amount of space a computation uses (halting problems parameterized by space are referred to has "compact" halting problems in the parameterized complexity literature):

| *p*-COMPACT-BIN-ATM-HALT | |
|---|---|
| *Instance:* | An alternating Turing machine $M$ with binary alphabet, $k \in \mathbb{N}$. |
| *Parameter:* | $k$. |
| *Problem:* | Decide whether $M$ accepts the empty input using at most $k$ tape cells. |

It is easy to see that *p*-COMPACT-BIN-ATM-HALT $\in$ EPT; just observe that for a given instance $(M, k)$, there are at most $N = 2^{O(k)} \cdot |M|^{O(1)}$ many relevant configurations. So we can first compute the *configuration graph* of $M$ of size $N$, and then test the accepting condition in time polynomial in $N$ by computing the *alternating reachability problem* on that graph.

**2.2. Parameterized complexity.** As mentioned in the introduction, in parameterized complexity we are dealing with the same type of problems as in subexponential complexity, namely parameterized problems $(Q, \kappa)$, where $Q \subseteq \Sigma^*$ and $\kappa : \Sigma^* \to \mathbb{N}$ is polynomial time computable. However, we study the problems from a different perspective, in parameterized complexity theory we usually assume the parameter to be small, whereas in the subexponential, the parameter is supposed to be a size measure and hence close to the size of the instance.

**Definition 9.** A parameterized problem $(Q, \kappa)$ is *fixed-parameter tractable* if there is an algorithm $\mathbb{A}$ and a computable function $f$ such that for every instance $x$ $\mathbb{A}$ decides if $x \in Q$ in time

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

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

**Definition 10.** Let $(Q, \kappa)$ and $(Q', \kappa')$ be two parameterized problems over the alphabets $\Sigma$ and $\Sigma'$, respectively.

(1) A *(many-one)* fpt-*reduction* from $(Q, \kappa)$ to $(Q', \kappa')$ is a mapping $R : \Sigma^* \to (\Sigma')^*$ such that:

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

(b) For all $x \in \Sigma^*$, $R(x)$ is computable in time

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

for a computable $f : \mathbb{N} \to \mathbb{N}$.

(c) 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 $(Q, \kappa)$ to $(Q', \kappa')$, and we write $(Q, \kappa) \equiv^{\text{fpt}} (Q', \kappa')$ if $(Q, \kappa) \leq^{\text{fpt}} (Q', \kappa')$ and $(Q', \kappa') \leq^{\text{fpt}} (Q, \kappa)$.

(2) An fpt *Turing reduction* from $(Q, \kappa)$ to $(Q', \kappa')$ is an algorithm $\mathbb{A}$ with an oracle to $Q'$ such that there are computable functions $f, g : \mathbb{N} \to \mathbb{N}$ with:

   (a) Given an instance $x \in \Sigma^*$, the algorithm $\mathbb{A}$ decides if $x \in Q$ in time

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

   (b) For all oracle queries "$y \in Q'$?" posed by $\mathbb{A}$ on input $x \in \Sigma^*$ we have $\kappa'(y) \leq g(\kappa(x))$.

   $(Q, \kappa) \leq^{\text{fpt-T}} (Q', \kappa')$ and $(Q, \kappa) \equiv^{\text{fpt-T}} (Q', \kappa')$ are understood similarly as in the case of fpt-reductions.

It is easy the see that a many-one fpt-reduction is also an fpt Turing reduction, and FPT is closed under fpt Turing reductions.

Most parameterized problems that are studied in parameterized complexity theory have the property that for every fixed parameter value, the instances with this value are decidable in polynomial time (albeit by an algorithm whose running time is bounded by a polynomial that may depend on the parameter $k$). Essentially, XP is the class of all such problems, but with a uniformity condition added.

**Definition 11.** A parameterized problem $(Q, \kappa)$ is in XP, if there is an algorithm $\mathbb{A}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathbb{A}$ decides $Q$, and the running time of $\mathbb{A}$ on input $x$ is bounded by

$$|x|^{f(\kappa(x))} + f(\kappa(x)).$$

Clearly XP is closed under fpt- and fpt Turing reductions.

**Example 12.** The *parameterized compact halting problem for alternating Turing machines* is the following parameterized problem.

| | |
|---|---|
| $p$-COMPACT-ATM-HALT | |
| *Instance:* | An *alternating* Turing machine $M$ with *arbitrary* alphabet, $k \in \mathbb{N}$. |
| *Parameter:* | $k$. |
| *Problem:* | Decide whether $M$ accepts the empty input using space $k$. |

It has been proved by Demri et al. [8] that *p*-COMPACT-ATM-HALT is complete for XP under fpt-reductions.

## 3. The miniaturization mapping

**Definition 13.** Let $(P, \nu)$ be a parameterized problem over the alphabet $\Sigma$. We define the *miniaturization* *p*-MINI$(P, \nu)$ of $(P, \nu)$ as the following parameterized problem:

| | |
|---|---|
| *p*-MINI$(P, \nu)$ | |
| *Instance:* | $x \in \Sigma^*$, $m \in \mathbb{N}$ in *unary* with $m \geq |x|$. |
| *Parameter:* | $\left\lceil \frac{\nu(x)}{\log m} \right\rceil$. |
| *Problem:* | Decide whether $x \in P$. |

In other words, *p*-MINI$(P, \nu)$ is the parameterization of the classical problem

| | |
|---|---|
| MINI$(P, \nu)$ | |
| *Instance:* | $x \in \Sigma^*$, $m \in \mathbb{N}$ in unary with $m \geq |x|$. |
| *Problem:* | Decide whether $x \in P$. |

with $\kappa(x,m) = \lceil \nu(x)/\log m \rceil$.

The ideas underlying the following result go back to [4, 9]. The theorem also follows from Theorem 17 below, but nevertheless we find it worthwhile to state and prove it separately first.

**Theorem 14.** *Let $(P,\nu)$ be a parameterized problem. Then $(P,\nu) \in$ SUBEPT $\iff$ p-*MINI$(P,\nu) \in$ FPT.

*Proof:* Let $\Sigma$ be the alphabet of $P$. Assume $(P,\nu)$ is in SUBEPT. By Lemma 2, there is an algorithm $\mathbb{A}$ expecting inputs from $\Sigma^* \times \mathbb{N}$ and a computable function $f$ such that for all $(x,\ell) \in \Sigma^* \times \mathbb{N}$, $\mathbb{A}$ decides if $x \in P$ in time $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$.

Let $\mathbb{B}$ be the following algorithm: Given an instance $(x,m) \in \Sigma^* \times \mathbb{N}$, if $|x| < m$, $\mathbb{B}$ rejects. Otherwise, $\mathbb{B}$ computes the parameter $k := \left\lceil \frac{\nu(x)}{\log m} \right\rceil$, and then simulates $\mathbb{A}$ on $(x,k)$. Since $\nu(x) \leq k \cdot \log m$, the time taken by the simulation is bounded by

$$f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)} \leq f(k) \cdot m \cdot |x|^{O(1)} \leq f(k) \cdot (|x| + m)^{O(1)}.$$

Therefore *p*-MINI$(P,\nu)$ is fixed-parameter tractable.

For the converse direction, assume there is an algorithm $\mathbb{A}$ and a computable function $f : \mathbb{N} \to \mathbb{N}$ that for every $(x,m)$ decides if $(x,m) \in$ MINI$(P,\nu)$ in time

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right) \cdot (|x| + m)^{O(1)}.$$

Without loss of generality we assume $f$ is monotone. Now let $\mathbb{B}$ be the algorithm that, for any given $(x,\ell) \in \Sigma^* \times \mathbb{N}$, first computes

$$m := \max\left\{|x|, \left\lceil 2^{\nu(x)/\ell} \right\rceil\right\},$$

which can be done in time $2^{O(\kappa(x)/\ell)} \cdot |x|^{O(1)}$. Then $\mathbb{B}$ simulates $\mathbb{A}$ on $(x,m)$, whose running time is

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right) \cdot (|x| + m)^{O(1)} \leq f(\ell) \cdot 2^{O(\nu(x)/\ell)} \cdot |x|^{O(1)}.$$

So again by Lemma 2, $(P,\nu)$ is in SUBEPT. □

As the following two examples show, there is a number of problems that have natural parameterized problems as their miniaturizations.

**Example 15.** The miniaturization of *s-var*-CIRCUIT-SAT is fpt-equivalent to the parameterized weighted circuit satisfiability problem

| *p*-W-CICUIT-SAT | |
|---|---|
| *Instance:* | A Boolean circuit $C$ and a $k \in \mathbb{N}$. |
| *Parameter:* | $k$. |
| *Problem:* | Decide whether $C$ has a satisfying assignment that sets exactly $k$ input gates to TRUE. |

Essentially, this result goes back to Abrahamson, Downey and Fellows [1] (see [11] for a proof). It derives its significance from the fact that *p*-W-CICUIT-SAT is complete for the important parameterized complexity class W[P].

**Example 16.** The miniaturization of *p*-COMPACT-BIN-ATM-HALT is fpt-equivalent to the parameterized problem *p*-COMPACT-ATM-HALT. A proof can be found in Appendix B.3.

The previous example can be generalized to the halting problems for *deterministic* and *nondeterministic* Turing machines parameterized by space. However for the moment we do not have a similar exact correspondence between "time" halting problems.

As we have seen in Theorem 14, miniaturization maps subexponential solvability of a problem exactly into the fixed-parameter tractability of its miniaturization. Indeed it can be viewed as a consequence of the following theorem.

**Theorem 17.** *Let $(P,\nu)$ and $(P',\nu')$ be parameterized problems. Then*

*(1)* $(P,\nu) \leq^{\text{serf}} (P',\nu') \iff p\text{-}\textsc{Mini}(P,\nu) \leq^{\text{fpt}} p\text{-}\textsc{Mini}(P',\nu')$.

*(2)* $(P,\nu) \leq^{\text{serf-T}} (P',\nu') \iff p\text{-}\textsc{Mini}(P,\nu) \leq^{\text{fpt-T}} p\text{-}\textsc{Mini}(P',\nu')$.

A proof can be found in Appendix B.4.

The main results of this and the following section can most elegantly be formulated in the language of *degrees* from classical recursion theory. Suppose we have some reducibility relation $\leq$ on parameterized problems, for example, $\leq^{\text{fpt}}$. In general, $\leq$ only needs to be a reflexive and transitive relation. Let us denote the corresponding equivalence relation by $\equiv$. Then the $\leq$*-degree* of a problem $(Q,\nu)$, denoted by $[\![(Q,\nu)]\!]^{\leq}$, is the $\equiv$-equivalence class of $(Q,\nu)$. For example, the $\leq^{\text{fpt}}$-degree of $p$-COMPACT-ATM-HALT is the class of all XP-complete problems. The class of all $\leq$-degrees is denoted by $\mathbf{D}_{\leq}$, and for a class C of parameterized problems that is downward closed under $\leq$, the class of all degrees in C is denoted by $\mathbf{C}_{\leq}$. The reduction $\leq$ induces a partial order on $\mathbf{D}_{\leq}$. If $\leq = \leq^{\text{fpt}}$, then to simplify the notation we speak of fpt-degrees instead of $\leq^{\text{fpt}}$-degrees and write $[\![(Q,\nu)]\!]^{\text{fpt}}$, $\mathbf{D}_{\text{fpt}}$, et cetera. The same notational convention applies to reductions $\leq^{\text{fpt-T}}$, $\leq^{\text{serf}}$, and $\leq^{\text{serf-T}}$.

Note that by Theorem 17(1), the miniaturization mapping induces a well-defined mapping $M : \mathbf{D}_{\text{serf}} \to \mathbf{D}_{\text{fpt}}$, defined by $M\big([\![(P,\nu)]\!]^{\text{serf}}\big) := \big[\!\big[p\text{-}\textsc{Mini}(P,\nu)\big]\!\big]^{\text{fpt}}$, on the serf-degrees. By Theorem 17(2), it also induces a mapping on the serf Turing degrees. The main results of this section can be summarized in the following theorem:

**Embedding Theorem.** *The miniaturization mapping induces an embedding of the partially ordered set* $(\mathbf{D}_{\text{serf}}, \leq^{\text{serf}})$ *into the partially ordered set* $(\mathbf{D}_{\text{fpt}}, \leq^{\text{fpt}})$ *and also an embedding of the partially ordered set* $(\mathbf{D}_{\text{serf-T}}, \leq^{\text{serf-T}})$ *into the partially ordered set* $(\mathbf{D}_{\text{fpt-T}}, \leq^{\text{fpt-T}})$.

## 4. An Isomorphism between XP and EPT

**Lemma 18.** *Let* $(Q,\kappa) \in \text{XP}$*. Then there exists a problem* $(P,\nu) \in \text{EPT}$ *such that* $(Q,\kappa) \equiv^{\text{fpt}} p\text{-}\textsc{Mini}(P,\nu)$.

*Proof:* In a first step we construct a problem $(Q',\kappa')$ equivalent to $(Q,\kappa)$ that is decidable in time $|x|^{O(\sqrt{\kappa'(x)})}$.

Let $\Sigma$ be the alphabet of $Q$ and suppose that $Q$ is decidable in time $|x|^{f(\kappa(x))} + f(\kappa(x))$, where without loss of generality $f$ is increasing and time constructible. Let $(Q',\kappa')$ be the following parameterized problem:

| | |
|---|---|
| *Instance:* | $x \in \Sigma^*$, $\ell \in \mathbb{N}$ in unary such that $\ell \geq f(\kappa(x))^2$. |
| *Parameter:* | $\ell$. |
| *Problem:* | Decide whether $x \in Q$. |

It is easy to see that indeed $(Q',\kappa') \equiv^{\text{fpt}} (Q,\kappa)$ and that $Q'$ is decidable in time $|x|^{O(\sqrt{\kappa'(x)})}$.

In the second step, we construct the desired problem $(P,\nu)$. Let $\Sigma'$ be the alphabet of $Q'$. We let $(P,\nu)$ be the following problem:

| | |
|---|---|
| *Instance:* | $x \in (\Sigma')^*$. |
| *Parameter:* | $\kappa'(x) \cdot \lceil \log |x| \rceil$. |
| *Problem:* | Decide whether $x \in Q'$. |

Then $P = Q'$, that is, $(P, \nu)$ is just a re-parameterization of $(Q', \kappa')$. Recall that $Q'$ is decidable in time

$$|x|^{O(\sqrt{\kappa'(x)})} = 2^{O(\sqrt{\kappa'(x)} \cdot \log |x|)} \leq 2^{O(\kappa'(x) \cdot \lceil \log |x| \rceil)} = 2^{\nu(x)}.$$

It follows that $(P, \nu) \in$ EPT. Now we claim that $p$-MINI$(P, \nu) \equiv^{\text{fpt}} (Q', \kappa')$.

This is trivially true if $Q' = \emptyset$ or $Q' = (\Sigma')^*$. So suppose that neither $Q' = \emptyset$ nor $Q' = (\Sigma')^*$ and let $x_+ \in Q'$ and $x_- \in (\Sigma')^* \setminus Q'$.

To prove that $p$-MINI$(P, \nu) \leq^{\text{fpt}} (Q', \kappa')$, we define a reduction $R$ by letting

$$R(x, m) := \begin{cases} x_+ & \text{if } m \geq |x|^{\sqrt{\kappa'(x)}} \text{ and } x \in Q', \\ x_- & \text{if } m \geq |x|^{\sqrt{\kappa'(x)}} \text{ and } x \notin Q', \\ x & \text{if } |x| \leq m < |x|^{\sqrt{\kappa'(x)}}, \\ x_- & \text{otherwise}. \end{cases}$$

Then clearly for all $(x, m) \in \Sigma^* \times \mathbb{N}$ we have $(x, m) \in$ MINI$(P, \nu) \iff R(x, m) \in Q'$. Moreover, $R(x, m)$ is computable in polynomial time, because $x \in Q'$ is decidable in time $|x|^{O(\sqrt{\kappa'(x)})}$, which is $m^{O(1)}$ if $m \geq |x|^{\sqrt{\kappa'(x)}}$.

It remains to prove that the parameter $\kappa(R(x, m))$ of the image is effectively bounded in terms of the parameter $\lceil \nu(x)/\log m \rceil$ of the argument. Let $(x, m)$ be an instance of MINI$(P, \nu)$. If either $m \geq |x|^{\sqrt{\kappa'(x)}}$ or $m < |x|$, then $\kappa(R(x, m)) \leq \max\{\kappa(x_+), \kappa(x_-)\}$, which is a constant. So let us assume that $|x| \leq m < |x|^{\sqrt{\kappa'(x)}}$. Then $\log\, m < \sqrt{\kappa'(x)} \cdot \lceil \log |x| \rceil = \nu(x)/\sqrt{\kappa'(x)}$, because $\nu(x) = \kappa'(x) \cdot \lceil \log |x| \rceil$. Thus $\kappa'(x) = \kappa'(x) \cdot \log\, m/\log\, m < \sqrt{\kappa'(x)} \cdot \nu(x)/\log\, m$ and therefore $\kappa'(x) \leq (\nu(x)/\log m)^2$. As $\kappa'(x)$ is the parameter of $x = R(x, m)$ regarded as an instance of $Q'$ and $\nu(x)/\log m$ is the parameter of $(x, m)$ regarded as an instance of MINI$(P, \nu)$, this shows that indeed $R$ is an fpt-reduction and proves $p$-MINI$(P, \nu) \leq^{\text{fpt}} (Q', \kappa')$.

For the other direction, $(Q', \kappa') \leq^{\text{fpt}} p$-MINI$(P, \nu)$, we define a reduction $R : (\Sigma')^* \to (\Sigma')^* \times \mathbb{N}$ by $R(x) = (x, 2 \cdot |x|)$. As

$$\frac{\nu(x)}{\log\,(2 \cdot |x|)} = \frac{\kappa'(x) \cdot \lceil \log |x| \rceil}{\log |x| + 1} \leq \kappa'(x),$$

$R$ is an fpt-reduction from $(Q', \kappa')$ to $p$-MINI$(P, \nu)$. □

Now we can establish a similar correspondence as Theorem 14 with respect to XP and EPT.

**Theorem 19.** *Let $(P, \nu)$ be a parameterized problem. Then $(P, \nu) \in$ EPT $\iff$ $p$-MINI$(P, \nu) \in$ XP.*

*Proof:* Let $(P, \nu) \in$ EPT, in other words, $(P, \nu)$ is decidable in time $2^{O(\nu(x))} \cdot |x|^d$ for some constant $d \in \mathbb{N}$. Let $(x, m)$ be an instance of MINI$(P, \nu)$. If $|x| > m$, it is a 'no'-instance, so let us assume that $|x| \leq m$. Let $n = \nu(x)$ and $k = \lceil n/\log m \rceil$. Then the instance is decidable in time $2^{O(\nu(x))} \cdot |x|^d \leq m^{O(\nu(x)/\log m)} \cdot m^d \leq m^{O(\lceil \nu(x)/\log m \rceil) + d}$. Hence, $p$-MINI$(P, \nu)$ is in XP.

For another direction, assume $p$-MINI$(P, \nu) \in$ XP. By Lemma 18, there is a parameterized problem $(P', \nu') \in$ EPT such that $p$-MINI$(P, \nu) \equiv^{\text{fpt}} p$-MINI$(P', \nu')$. Now Theorem 17 implies $(P, \nu) \equiv^{\text{serf}} (P', \nu')$, and by Proposition 7, $(P, \nu) \in$ EPT. □

**Corollary 20.** *Let $(P, \nu)$ be a parameterized problem. $(P, \nu)$ is* EPT- *complete (*EPT-*hard) under* serf-*reductions if and only* $p$-MINI$(P, \nu)$ *is* XP-*complete (*XP-*hard, respectively) under* fpt-*reductions.*

**Example 21.** $p$-COMPACT-BIN-ATM-HALT is complete for EPT under serf-reductions.

*Proof:* $p$-COMPACT-ATM-HALT is complete for XP under fpt-reductions [8], and the miniaturization of $p$-COMPACT-ATM-HALT is fpt-equivalent to $p$-COMPACT-BIN-ATM-HALT (see Example 16). So Corollary 20 implies $p$-COMPACT-BIN-ATM-HALT is complete for EPT under serf-reductions. □

Rephrasing the results of this section in the language of degrees introduced at the end of the previous section, we obtain the following theorem:

**Isomorphism Theorem.** *The miniaturization mapping induces an isomorphism between* $(\mathbf{EPT}_{\text{serf}}, \leq^{\text{serf}})$ *and* $(\mathbf{XP}_{\text{fpt}}, \leq^{\text{fpt}})$ *and also an isomorphism between* $(\mathbf{EPT}_{\text{serf-T}}, \leq^{\text{serf-T}})$ *and* $(\mathbf{XP}_{\text{fpt-T}}, \leq^{\text{fpt-T}})$.

The following theorem shows that the Isomorphism Theorem cannot be extended from the degrees in EPT and XP to all degrees, because outside of XP the mapping induced by the miniaturization mapping is not onto.

**Theorem 22.** *There is a parameterized problem* $(Q, \kappa)$ *that is not* fpt-T*-equivalent to* $p$-MINI$(P, \nu)$ *for any* $(P, \nu)$.

A proof can be found in Appendix B.5.

## 5. The S-Hierarchy and the W-Hierarchy

The purpose of this last section of the paper is to gather further evidence that the miniaturization mapping is not only an abstract isomorphism between the subexponential and parameterized complexity theory, but actually establishes a relation between interesting and relevant complexity classes on both sides. Specifically, we want to lay out a relation between a natural hierarchy of the subexponential theory and the W-hierarchy of parameterized complexity theory. The results in this section are not new; they go back to Abrahamson, Downey, and Fellows [1] and have been refined in [5, 6, 11]. For proofs of the results, we refer the reader to these references.

The W-hierarchy is defined in terms of *weighted satisfiability problems* for classes $\Gamma$ of propositional formulas or Boolean circuits:

| | |
|---|---|
| *p*-WSAT$(\Gamma)$ | |
| *Instance:* | $\gamma \in \Gamma$ and $k \in \mathbb{N}$. |
| *Parameter:* | $k$. |
| *Problem:* | Decide whether $\gamma$ has a satisfying assignment that sets precisely $k$ variables to TRUE. |

We denote the class of Boolean circuits by CIRC, and the class of propositional formulas by PROP. As usual PROP can be viewed as a subclass of CIRC. Note that *p*-WSAT(CIRC) is exactly the parameterized problem *p*-W-CICUIT-SAT introduced in Example 15.

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

$$\begin{aligned}
\Gamma_{0,d} &:= \big\{\lambda_1 \wedge \ldots \wedge \lambda_c \mid c \leq d, \lambda_1, \ldots, \lambda_c \text{ literals}\big\},\\
\Delta_{0,d} &:= \{\lambda_1 \vee \ldots \vee \lambda_c \mid c \leq d, \lambda_1, \ldots, \lambda_c \text{ literals}\},\\
\Gamma_{t+1,d} &:= \Big\{\bigwedge_{i \in I} \delta_i \mid I \text{ finite index set and } \delta_i \in \Delta_{t,d} \text{ for all } i \in I\Big\},\\
\Delta_{t+1,d} &:= \Big\{\bigvee_{i \in I} \gamma_i \mid I \text{ finite index set and } \gamma_i \in \Gamma_{t,d} \text{ for all } i \in I\Big\}.
\end{aligned}$$

The *W-hierarchy* of parameterized complexity theory consists of the following classes:

**Definition 23.** (1) For $t \geq 1$, W$[t] := \bigcup_{d \geq 1} \big\{(Q, \kappa) \mid (Q, \kappa) \leq^{\text{fpt}} p\text{-WSAT}(\Gamma_{t,d})\big\}$.

(2) W[SAT] $:= \big\{(Q, \kappa) \mid (Q, \kappa) \leq^{\text{fpt}} p\text{-WSAT}(\text{PROP})\big\}$.

(3) W[P] $:= \big\{(Q, \kappa) \mid (Q, \kappa) \leq^{\text{fpt}} p\text{-WSAT}(\text{CIRC})\big\}$.

This definition of the W-hierarchy establishes the role of weighted satisfiability problems as the "generic" (hard) problems of parameterized complexity theory. We propose that plain (unweighted) satisfiability problems with the "number of variables" size measure can play a similar role in subexponential complexity theory. For every class $\Gamma$ of propositional formulas or Boolean circuits, we let:

| | |
|---|---|
| *s-var*-SAT$(\Gamma)$ | |
| *Instance:* | $\gamma \in \Gamma$. |
| *Parameter:* | Number of variables of $\gamma$. |
| *Problem:* | Decide whether $\gamma$ is satisfiable. |

The *S-hierarchy* consists of the following classes:

**Definition 24.** (1) For $t \geq 1$, $\mathrm{S}[t] := \bigcup_{d\geq 1} \big\{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-}\mathrm{SAT}(\Gamma_{t,d})\big\}$.

(2) $\mathrm{S}[\mathrm{SAT}] := \big\{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-}\mathrm{SAT}(\mathrm{PROP})\big\}$.

(3) $\mathrm{S}[\mathrm{P}] := \big\{(P,\nu) \mid (P,\nu) \leq^{\mathrm{serf}} s\text{-}var\text{-}\mathrm{SAT}(\mathrm{CIRC})\big\}$.

So far, mainly the first level S[1] of the S-hierarchy has been studied, and some highly nontrivial completeness results are known. Most importantly, Impagliazzo, Paturi, and Zane [13] have proved that *s-var*-3-SAT is complete for S[1] under serf Turing reductions. Thus the exponential hypothesis (discussed in the introduction) is equivalent to S[1] $\neq$ SUBEPT.

To relate the S-hierarchy and the W-hierarchy, we consider the image of the S-hierarchy under the miniaturization mapping (the so-called *M-hierarchy*):

**Definition 25.** (1) For $t \geq 1$, $\mathrm{M}[t] := \big\{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{fpt}} p\text{-}\mathrm{MINI}(P,\nu)$ for some $(P,\nu) \in \mathrm{S}[t]\big\}$.

(2) $\mathrm{M}[\mathrm{SAT}] := \big\{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{fpt}} p\text{-}\mathrm{MINI}(P,\nu)$ for some $(P,\nu) \in \mathrm{S}[\mathrm{SAT}]\big\}$.

(3) $\mathrm{M}[\mathrm{P}] := \big\{(Q,\kappa) \mid (Q,\kappa) \leq^{\mathrm{fpt}} p\text{-}\mathrm{MINI}(P,\nu)$ for some $(P,\nu) \in \mathrm{S}[\mathrm{P}]\big\}$.

We obtain the following result:

**Theorem 26 ([1]).** *For every* $t \geq 1$, $\mathrm{M}[t] \subseteq \mathrm{W}[t] \subseteq \mathrm{M}[t+1]$. *Moreover* $\mathrm{M}[\mathrm{SAT}] = \mathrm{W}[\mathrm{SAT}]$ *and* $\mathrm{M}[\mathrm{P}] = \mathrm{W}[\mathrm{P}]$.

A schematic figure illustrating the relations between all complexity classes considered in this paper can be found in Appendix A.

## 6. Concluding Remarks

This paper is a contribution to a (not yet clearly established) subexponential complexity theory. A long term goal of such a theory might be to prove that the exponential time hypothesis is equivalent to P $\neq$ NP and thus obtain a solid basis for exponential lower bounds such as the one for 3-SAT stated by the exponential time hypothesis. However, with current methods this goal seems out of reach, and maybe such an equivalence cannot be established without actually proving that the exponential time hypothesis and hence P $\neq$ NP holds.[3]

What we can establish now is a close connection between subexponential and parameterized complexity theory. The obvious open question is whether the exponential time hypothesis is equivalent to FPT $\neq$ W[1]. Despite serious efforts, so far researchers in parameterized complexity have not been able to prove that M[1] $=$ W[1], even though this still seems quite plausible. However, it would also be compatible with our current knowledge that M[2] $=$ W[1].

The higher levels of the S-hierarchy (or equivalently, the M-hierarchy), have not yet received much attention. Specifically, no completeness results for S[2] are known, even though the class contains natural problems such as *s-var*-SAT that are not believed to be in S[1]. It may be worthwhile to study the class S[2] and develop a completeness theory for this class similar to the S[1]-completeness theory, which is based on Impagliazzo, Paturi, and Zane's Sparsification Lemma [13].

[3]However, 15 years ago an equivalence between the inapproximability of, say, MAX-SAT and P $\neq$ NP also must have seemed far out of reach.

## References

[1] K. Abrahamson, R. Downey, and M. Fellows. Fixed-Parameter Tractability and Completeness IV: On Completeness for W[P] and PSPACE Analogues. *Annals of Pure and Applied Logic*, 73(3): 235-276, 1995.

[2] J. Alber, H. Fernau, and R. Niedermeier. Parameterized complexity: exponential speed-up for planar graph problems. In *Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP'01)*, Lecture Notes in Computer Science 2076, 261-272, 2001.

[3] T. Brueggemann and W. Kern. An improved deterministic local search algorithm for 3-SAT. *Theoretical Computer Science*, 329(1-3), 303-313, 2004.

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

[5] J. Chen, B. Chor, M. Fellows, X. Huang, D. Juedes, I. Kanj, and Ge Xia. Tight Lower Bounds for Certain Parameterized NP-Hard Problems. In *Proceedings of The 19th IEEE Annual Conference on Computational Complexity (CCC'04)*, 150-160, 2004.

[6] J. Chen, X. Huang, I. Kanj, and G. Xia. Linear FPT reductions and computational lower bounds. In *Proceedings of the 36th annual ACM symposium on Theory of computing*, 212-221, 2004.

[7] Y. Chen and J. Flum. On miniaturized problems in parameterized complexity theory. To appear in *Theoretical Computer Science*, 2005.

[8] S. Demri, F. Laroussinie, and Ph. Schnoebelen. A parametric analysis of the state explosion problem in model checking (extended abstract). In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science STACS '02*, volume 2285 of *Lecture Notes in Computer Science*, 620-631, Springer-Verlag, 2002.

[9] R. Downey, V. Estivill, M. Fellows, E. Prieto, and F. Rosamond. Cutting up is hard to do: the parameterized complexity of $k$-cut and related problems. In *Proceedings of Computing: The Australasian Theory Symposium (CATS'03)*, Electronic Notes in Theoretical Computer Science, 78(0), 1-14, 2003.

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

[11] J. Flum and M. Grohe. Parameterized complexity and subexponential time. In *the Complexity Column of the Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 84, October, 2004.

[12] J. Flum, M. Grohe, and M. Weyer. Limited nondeterminism and parameterized complexity theory. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, Lecture Notes in Computer Science 3142, 555-567, 2004.

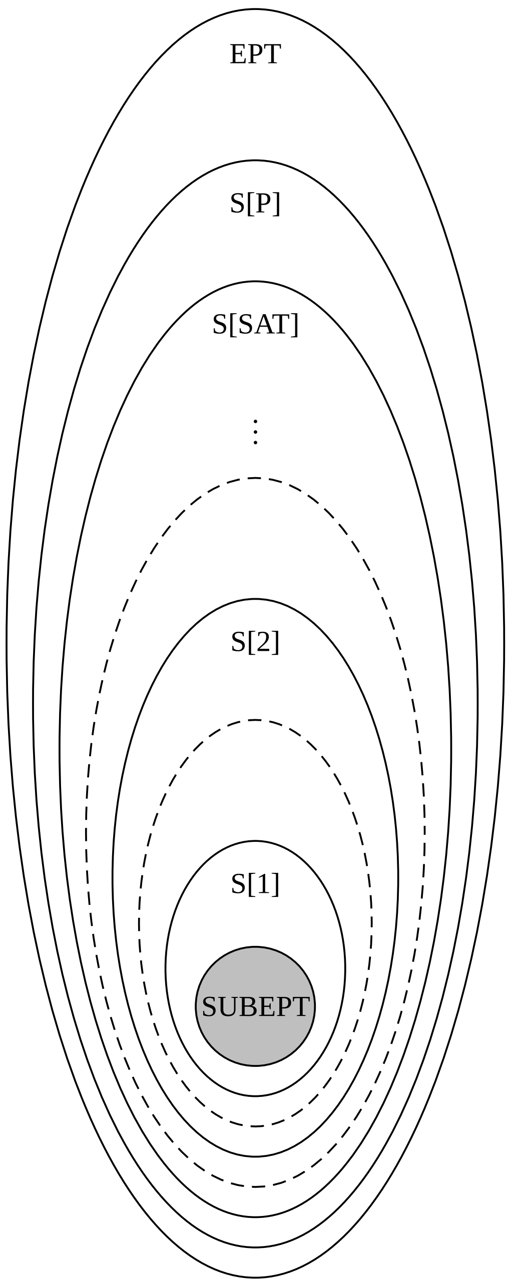
[13] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer System and Science*, 63(4):512-530, 2001.

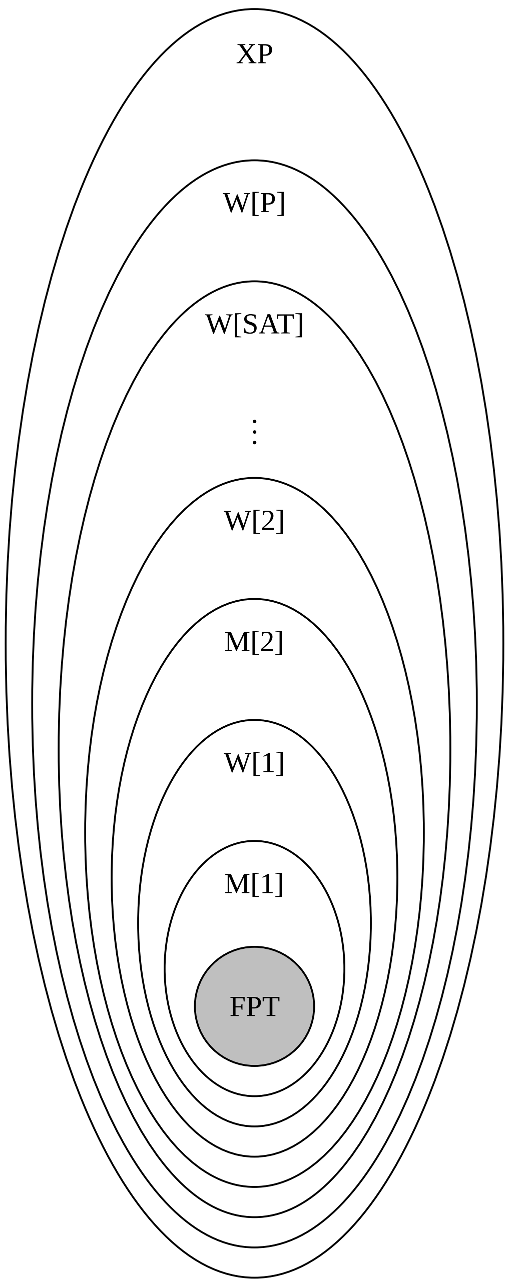
[14] K. Iwama and S. Tamaki. Improved upper bounds for 3-sat. In *Proceedings of the 15th Anuual ACM-SIAM Symposium on Discrete Algorithms*, 328-329, 2004.

[15] U. Sch¨oning: Algorithmics in Exponential Time. In *Proceedings of 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS '05)*, Lecture Notes in Computer Science 3404, 36-43, 2005.

[16] G. Woeginger. Space and Time Complexity of Exact Algorithms: Some Open Problems (Invited Talk). In *Proceedings of Parameterized and Exact Computation, First International Workshop (IWPEC '04)*, Lecture Notes in Computer Science 3162, 281-290, 2004.

## A. Overview over the complexity classes and hierarchies





## B. Proofs

### B.1. Proof of Proposition 4.

Let $\Sigma, \Sigma'$ be the alphabets of $(P, \nu)$, $(P', \nu')$, respectively. Let $\mathbb{A}$ be a serf Turing reduction from $(P, \nu)$ to $(P', \nu')$, and let $f, g$ be the functions bounding the running time and the parameter. Let $f' : \mathbb{N} \to \mathbb{N}$ be a computable function and $\mathbb{A}'$ an algorithm that, given $(x', k') \in (\Sigma')^* \times \mathbb{N}$, decides if $x' \in P'$ in time $f'(k') \cdot 2^{\nu'(x')/k'} \cdot |x'|^{O(1)}$. Such $f', \mathbb{A}'$ exist by Lemma 2.

Let $\mathbb{B}$ be the algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, simulates $\mathbb{A}$ and answers the oracle queries with instance $x'$ by simulating $\mathbb{A}'$ on input $(x', k')$, where $k' := g(k) \cdot k$. Observe that since the running time of $\mathbb{A}$ is bounded by $f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)}$, for each oracle query with instance $x'$ we have $|x'| \leq f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)}$. Also recall that by the definition of subexponential reduction families, we have $\nu'(x') \leq g(k) \cdot (\nu(x) + \log |x|)$. Then $\mathbb{B}$ decides $P$, and the running time on input $(x, k)$ is bounded by

$$\begin{aligned}
& f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)} \cdot f'(k') \cdot 2^{\nu'(x')/k'} \cdot |x'|^{O(1)} \\
\leq\ & f(k) \cdot 2^{\nu(x)/k} \cdot |x|^{O(1)} \cdot f'(g(k) \cdot k) \cdot 2^{(\nu(x)+\log |x|)/k} \cdot f(k)^{O(1)} \cdot 2^{O(\nu(x)/k)} \cdot |x|^{O(1)} \\
\leq\ & f(k)^{O(1)} \cdot f'(g(k) \cdot k) \cdot 2^{O(\nu(x)/k)} \cdot 2^{\log |x|/k} \cdot |x|^{O(1)} \\
\leq\ & h(k) \cdot 2^{O(\nu(x)/k)} \cdot |x|^{O(1)}
\end{aligned}$$

for a suitable computable function $h$. Thus by Lemma 2, $(P, \nu) \in$ SUBEPT. □

### B.2. Proof of Proposition 7.

It is not hard to see that, from the a serf Turing reduction from $(P, \nu)$ to $(P', \nu')$, we can construct an algorithm $\mathbb{A}$ with an oracle to $P'$ satisfying:

(R1) $\mathbb{A}$ decides if $x \in P$ in time $O(2^{\nu(x)} \cdot |x|^{O(1)})$ for any instance $x$.

(R2) For all oracle queries "$y \in P'$?" posed by $\mathbb{A}$ on input $x$ we have $\nu'(y) = O(\nu(x) + \log |x|)$.

Let $\mathbb{A}'$ be an algorithm that, given $x'$, decides $x' \in P'$ in time $2^{O(\nu'(x'))} \cdot |x'|^{O(1)}$.

Now we define the following algorithm $\mathbb{B}$ that, given an instance $x$, simulates $\mathbb{A}$ and answers the oracle queries with instance $x'$ by simulating $\mathbb{A}'$ on $x'$. By (R1), for each oracle query with instance $x'$ we have $|x'| = O(2^{\nu(x)} \cdot |x|^{O(1)})$. In addition, (R2) implies that $\nu'(x') = O(\nu(x) + \log |x|)$. It is clear that $\mathbb{B}$ decides $P$, and its running time on input $x$ is bounded by

$$\begin{aligned}
& O(2^{\nu(x)} \cdot |x|^{O(1)}) \cdot 2^{O(\nu'(x'))} \cdot |x'|^{O(1)} \\
=\ & O(2^{\nu(x)} \cdot |x|^{O(1)}) \cdot 2^{O(\nu(x)+\log |x|)} \cdot O(2^{O(\nu(x))} \cdot |x|^{O(1)}) \ = \ 2^{O(\nu(x))} \cdot |x|^{O(1)}.
\end{aligned}$$ □

### B.3. Proof of the statement of Example 16.

First it is easy to see the miniaturization of $p$-COMPACT-BIN-ATM-HALT is fpt-equivalent to the problem:

$p$-$k \cdot \log n$-COMPACT-BIN-ATM-HALT

| | |
|---|---|
| *Instance:* | An alternating Turing machine $M$ with binary alphabet, $n \in \mathbb{N}$ in *unary*, $k \in \mathbb{N}$. |
| *Parameter:* | $k$. |
| *Problem:* | Decide whether $M$ accepts the empty input using space $\lceil k \cdot \log n \rceil$. |

Therefore it suffices to show $p$-COMPACT-ATM-HALT $\equiv^{\text{fpt}} p$-$k \cdot \log n$-COMPACT-BIN-ATM-HALT.

$p$-COMPACT-ATM-HALT $\le^{\text{fpt}} p$-$k \cdot \log n$-COMPACT-BIN-ATM-HALT: Given an alternating machine $M$ of arbitrary alphabet, it is not very hard, albeit tedious, to construct another alternating machine $M_{\text{bin}}$ with binary alphabet by encoding each symbol in $M$ by a binary string of length $\log |M|$. Thus $M$ accepts the empty input using space $k$ if and only if $M_{\text{bin}}$ accepts the empty input using space $k \cdot \log |M|$. So

$$(M, k) \mapsto (M_{\text{bin}}, |M|, k)$$

is an appropriate fpt-reduction.

$p$-$k \cdot \log n$-COMPACT-BIN-ATM-HALT $\le^{\text{fpt}} p$-COMPACT-ATM-HALT: For an alternating machine $M$ with binary alphabet and a natural number $n$, we can construct an alternating machine $M_{\log n}$ whose alphabet is of size $n$, each corresponds a $\log n$-bit binary. Thus $M_{\log n}$ can simulate a computation of $M$ which uses space $k \cdot \log n$ by a computation using space $k$. Hence

$$(M, n, k) \mapsto (M_{\log n}, k)$$

gives the required fpt-reduction. □

**B.4. Proof of Theorem 17.**

We show (2), the easier (1) is left to the reader. Let $\Sigma$ and $\Sigma'$ be the alphabets of $P$ and $P'$, respectively.

For the forward direction, let $\mathbb{A}$ be a serf Turing reduction from $(P, \nu)$ to $(P', \nu')$ which is computable in time $f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)}$ for every $(x, \ell) \in \Sigma^* \times \mathbb{N}$ for a computable function $f : \mathbb{N} \to \mathbb{N}$, and for any oracle query "$x' \in P'$?" posed by $\mathbb{A}$

$$\nu'(x') \le g(\ell) \cdot (\nu(x) + \log |x|)$$

for a computable function $g : \mathbb{N} \to \mathbb{N}$.

Let $\mathbb{B}$ be an algorithm that, given an instance $(x, m)$ of $\text{MINI}(P, \kappa)$, if $m < |x|$, answers 'no', otherwise simulates $\mathbb{A}$ on $(x, \ell)$, where

$$\ell := \left\lceil \frac{\nu(x)}{\log m} \right\rceil,$$

and replaces each oracle query "$x' \in P'$?" posed by $\mathbb{A}$ by "$(x', m') \in \text{MINI}(P', \nu')$?" where $m' = \max(|x'|, m)$.

Note the overall running time of $\mathbb{B}$ is bounded by

$$f(\ell) \cdot 2^{\nu(x)/\ell} \cdot |x|^{O(1)} \le f\left( \left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot m \cdot |x|^{O(1)}.$$

Moreover for each oracle query "$(x', m') \in \text{MINI}(P', \nu')$?" posed by $\mathbb{B}$, we have $\nu'(x') \le g(\ell) \cdot (\nu(x) + \log |x|)$. Hence,

$$\begin{aligned} \left\lceil \frac{\nu'(x')}{\log m'} \right\rceil &\le g\left( \left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot \left\lceil \frac{\nu(x) + \log |x|}{\log m} \right\rceil \qquad (\text{since } m' \ge m) \\ &\le g\left( \left\lceil \frac{\nu(x)}{\log m} \right\rceil \right) \cdot \left( \left\lceil \frac{\nu(x)}{\log m} \right\rceil + 1 \right), \qquad (\text{since } m \ge |x|). \end{aligned}$$

Thus $\mathbb{B}$ is an fpt Turing reduction from $p$-$\text{MINI}(P, \nu)$ to $p$-$\text{MINI}(P', \nu')$.

For the backward direction, let $\mathbb{A}$ be an algorithm with an oracle to $\text{MINI}(P', \nu')$ such that there are monotone computable functions $f, g : \mathbb{N} \to \mathbb{N}$ and a constant $d \in \mathbb{N}$ with:

(F1) Given an instance $(x, m) \in \Sigma^* \times \mathbb{N}$, the algorithm $\mathbb{A}$ decides if $(x, m) \in \textsc{Mini}(P, \nu)$ in time

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right) \cdot (|x| + m)^d.$$

(F2) For all oracle queries "$(x', m') \in \textsc{Mini}(P', \nu')$?" posed by $\mathbb{A}$ on input $(x, m)$ we have

$$\left\lceil \frac{\nu'(x')}{\log m'} \right\rceil \leq g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right).$$

Now to show $(P, \nu) \leq^{\text{serf-T}} (P', \nu')$, let $\mathbb{B}$ be an algorithm with an oracle to $(P', \nu')$ such that, given a pair $(x, \ell) \in \Sigma^* \times \mathbb{N}$, $\mathbb{B}$ simulates $\mathbb{A}$ on $(x, m)$, where

$$m := \max\left\{|x|, \left\lceil 2^{\nu(x)/(d \cdot \ell)} \right\rceil\right\},$$

and replaces oracle queries "$(x', m') \in \textsc{Mini}(P', \nu')$?" posed by $\mathbb{A}$ by "$x' \in P'$?", if $m' \geq |x'|$, or answering 'no' otherwise.

Observe that

$$\frac{\nu(x)}{\log m} \leq d \cdot \ell. \tag{2}$$

So by (F1) the running time of $\mathbb{A}$ on $(x, m)$, and hence $\mathbb{B}$ on $(x, \ell)$, is bounded by

$$f\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right) \cdot (|x| + m)^d \leq f(d \cdot \ell) \cdot 2^{2 \cdot d} \cdot 2^{\nu(x)/\ell} \cdot |x|^d.$$

Note it follows that for any oracle query "$(x', m') \in \textsc{Mini}(P', \nu')$?" posed by $\mathbb{A}$, we have

$$m' \leq f(d \cdot \ell) \cdot 2^{2 \cdot d} \cdot 2^{\nu(x)/\ell} \cdot |x|^d, \tag{3}$$

since it is represented in unary.

Now for any oracle query "$x' \in P'$?" posed by $\mathbb{B}$, which comes from the simulation of $\mathbb{A}$ over an oracle query "$(x', m') \in \textsc{Mini}(P', \nu')$?", (F2) implies

$$\left\lceil \frac{\nu'(x')}{\log m'} \right\rceil \leq g\left(\left\lceil \frac{\nu(x)}{\log m} \right\rceil\right).$$

Therefore by (2) and (3)

$$\begin{aligned} \nu'(x') &\leq g(d \cdot \ell) \cdot \log\left(f(d \cdot \ell) \cdot 2^{2 \cdot d} \cdot 2^{\nu(x)/\ell} \cdot |x|^d\right) \\ &\leq h(\ell) \cdot (\nu(x) + \log |x|) \end{aligned}$$

for some suitable computable function $h$.

So $\mathbb{B}$ is a serf Turing reduction from $(P, \nu)$ to $(P', \nu')$. □

## B.5. Proof of Theorem 22.

We need some preparation before we prove the theorem.

**Definition 27.** Let $Q$ and $Q'$ be two classical problems. An algorithm $\mathbb{A}$ with an oracle to $Q'$ is a *2-exptime Turing reduction* from $Q$ to $Q'$, if for any instance $x$ of $Q$, $\mathbb{A}$ decides if $x \in Q$ in time

$$2^{2^{|x|^{O(1)}}}.$$

2-exptime Turing reductions are slightly at odds with all the usual reductions (including those introduced in this paper so far), namely they are not *transitive*. However they are closed under the composition with polynomial time Turing reductions. More precisely:

**Lemma 28.** *Let $Q$, $Q'$, $Q''$ be classical problems. There is a 2-exptime Turing reduction from $Q$ to $Q''$,*

- *if there is a 2-exptime Turing reduction from $Q$ to $Q'$, and a polynomial time Turing reduction from $Q'$ to $Q''$;*
- *or, if there is a polynomial time Turing reduction from $Q$ to $Q'$, and a 2-exptime Turing reduction from $Q'$ to $Q''$.*

We omit the routine proof.

**Lemma 29.** *There is a sequence of problems $(Q_i)_{i\in\mathbb{N}}$ such that*

- *$\big\{\{i\} \times Q_i \mid i \in \mathbb{N}\big\}$ is decidable,*
- *for all $i \in \mathbb{N}$, $Q_i$ is not 2-exptime Turing reducible to*

$$L_i \quad := \quad \big\{(j,x) \mid j \neq i, x \in Q_j\big\}.$$

*Proof:* Fix an alphabet $\Sigma$. Let $\mathbb{A}_1, \mathbb{A}_2, \dots$ be an effective enumeration of all the 2-exptime Turing reductions from problems over the alphabet $\Sigma$ to problems as the subsets of $\mathbb{N} \times \Sigma^*$.

For $S \subseteq \mathbb{N} \times \Sigma^*$, $e \in \mathbb{N}$, and $x \in \Sigma^*$, we define

$$\mathbb{A}_e^S(x) := \begin{cases} 1 & \mathbb{A}_e \text{ accepts } x \text{ with an oracle to } S, \\ 0 & \text{otherwise}, \end{cases}$$

and let

$$u(S,e,x) := \max\big\{|y| \mid \text{ in the computation of } \mathbb{A}_e^S(x), \text{ there is an oracle query "} y \in S\text{?"}\big\}.$$

Clearly with a computable $S$, $\mathbb{A}_e^S(x)$ and $u(S,e,x)$ are both computable in $e$ and $x$.

Note for given $S_1, S_2 \subseteq \mathbb{N} \times \Sigma^*$, $e \in \mathbb{N}$, and $x \in \Sigma^*$, if for all $y \in \mathbb{N} \times \Sigma^*$ with $|y| \leq u(S_1,e,x)$, ($y \in S_1 \iff y \in S_2$), then the computation of $\mathbb{A}_e^{S_1}(x)$ exactly coincides with that of $\mathbb{A}_e^{S_2}(x)$, in particular, $\mathbb{A}_e^{S_1}(x) = \mathbb{A}_e^{S_2}(x)$.

Let $\langle \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ be a reasonable bijective encoding function. We will construct a computable sequence of pairwise distinct elements $(a_j)_{j\in\mathbb{N}} \in \Sigma^*$ such that for $j = \langle i, e\rangle$, $a_j$ witnesses the fact that

$$\mathbb{A}_e \text{ is not a reduction from } Q_i \text{ to } L_i.$$

In parallel we will define inductively computable sequences $0 = \ell_0 < \ell_1 < \ell_2 \cdots \in \mathbb{N} \cup \{0\}$ and $\emptyset = P_0 \subseteq P_1 \subseteq P_2 \subseteq \cdots \subseteq \mathbb{N} \times \Sigma^*$ such that for all $j \in \mathbb{N}$ with $j = \langle i, e\rangle$,

$$\ell_{j-1} < |(i, a_j)| \leq \ell_j, \tag{4}$$

$$\text{and} \quad |P_j \backslash P_{j-1}| \subseteq \{(i, a_j)\}. \tag{5}$$

Then we set, for each $i \in \mathbb{N}$,

$$Q_i := \Big\{a \in \Sigma^* \mid \langle i, a\rangle \in \bigcup_{j\in\mathbb{N}} P_j\Big\}. \tag{6}$$

Recall $\ell_0 = 0$, $P_0 = \emptyset$. Now for $j \in \mathbb{N}$ with $j = \langle i, e\rangle$, assume $\ell_{j-1}$ and $P_{j-1}$ are already defined. Let $a_j \in \Sigma^*$ be the minimal element in the lexicographical order such that

$$|(i, a_j)| > \ell_{j-1}$$

and distinct from all $a_{j'}$ for $1 \leq j' < j$. Now let

$$T_j := \big\{(i', a) \in P_{j-1} \mid i' \neq i\big\} = P_{j-1} \backslash \big(\{i\} \times \Sigma^*\big). \tag{7}$$

and

$$\ell_j := \max\big\{u(T_j, e, a_j), |(i, a_j)|\big\}. \tag{8}$$

(i) If $\mathbb{A}_e^{T_j}(a_j) = 0$, let $P_j := P_{j-1} \cup \{(i, a_j)\}$.

(ii) Otherwise $\mathbb{A}_e^{T_j}(a_j) = 1$, let $P_j := P_{j-1}$.

This finishes the construction.

Now we show that for each $i, e \in \mathbb{N}$, $\mathbb{A}_e$ is not a reduction from $Q_i$ to

$$\begin{aligned} L_i &= \big\{(i', a) \mid i' \neq i, a \in Q_{i'}\big\} \\ &= \big\{(i', a) \mid i' \neq i, \text{there exists a } j \in \mathbb{N} \text{ such that } (i', a) \in P_j\big\} \end{aligned} \tag{9}$$

Let $j := \langle i, e\rangle$. Note it suffices to prove that it is *not* the case that

$$a_j \in Q_i \iff \mathbb{A}_e^{L_i}(a_j) = 1.$$

First observe that (7) and (9) imply $T_j \subseteq L_i$. And by (4), (5), and (8), for any $(i', a') \in L_i \backslash T_j$, we have $|(i', a')| > u(T_j, e, a_j)$. It follows that

$$\mathbb{A}_e^{L_i}(a_j) = \mathbb{A}_e^{T_j}(a_j). \tag{10}$$

Recall during the construction we have two cases for $\mathbb{A}_e^{T_j}(a_j)$:

(i) If $\mathbb{A}_e^{T_j}(a_j) = 0$, then $\mathbb{A}_e^{L_i}(a_j) = 0$ by (10). And by our construction $(i, a_j) \in P_j$, therefore $a_j \in Q_i$ by (6).

(ii) If $\mathbb{A}_e^{T_j}(a_j) = 1$, then $\mathbb{A}_e^{L_i}(a_j) = 1$. Hence $(i, a_j) \notin P_j \backslash P_{j-1}$. It follows that $(i, a_j) \notin P_{j'}$ for all $j' \in \mathbb{N}$ by (5). Consequently $a_j \notin Q_i$.

To see the decidability of $(Q_i)_{i\in\mathbb{N}}$: given an instance $(i, a) \in \mathbb{N} \times \Sigma^*$, clearly

$$a \in Q_i \iff (i, a) \in \bigcup_{j\in\mathbb{N}} P_j.$$

We compute the minimal $j \in \mathbb{N}$ such that

$$|(i, a)| \leq \ell_j,$$

and then decide if $(i, a)$ is in the finite set $P_j$.

□

*Proof of Theorem 22.* Let $(Q_i)_{i\in\mathbb{N}}$ be as stated in Lemma 29. And for each $e \in \mathbb{N}$, let $\varphi_e$ denote the $e$-th partially recursive function.

For any $e \in \mathbb{N}$ and $n \in \mathbb{N} \cup \{0\}$, let

$$C(e,n) := \begin{cases} k & k \text{ is maximum such that } \varphi_e(1), \dots, \varphi_e(k) \\ & \quad \text{are defined, together can be computed in} \\ & \quad \text{at most } n \text{ steps, and each is smaller than } n, \\ 1 & \text{if no such } k \text{ exists.} \end{cases}$$

Clearly for any fixed $e$, $C(e,n)$ can be computed in time polynomial in $n$, and $C(e,n) \leq n$. Moreover if $\varphi_e(1)$ is defined, then

$$\varphi_e(C(e,n)) \leq \max\{\varphi_e(1), n\}. \tag{11}$$

Now let

$$Q := \big\{(e,a,k) \mid e \in \mathbb{N}, a \in Q_e, k = C(e,|a|)\big\}.$$

Define the parameterization $\kappa$ by $\kappa(e,a,k) := k$.

Suppose for contradiction that

$$(Q,\kappa) \equiv^{\text{fpt-T}} p\text{-}\textsc{Mini}(P,\nu)$$

for a parameterized problem $(P,\nu)$ over some alphabet $\Sigma'$. Let

$$\mu(x,m) = \left\lceil \frac{\nu(x)}{\log m} \right\rceil$$

be the parameterization of *p*-$\textsc{Mini}(P,\nu)$.

Let $\mathbb{A}$ be an fpt-T-reduction from $(Q,\kappa)$ to *p*-$\textsc{Mini}(P,\nu)$, and let $f : \mathbb{N} \to \mathbb{N}$ be a computable function such that, for all instances $x$, the algorithm $\mathbb{A}$ decides if $x \in Q$ in time

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

Let $e \in \mathbb{N}$ be such that $\varphi_e = f$. We leave $e$ fixed for the rest of the proof.

*Claim 1.* There is a polynomial time Turing reduction from $Q_e$ to $\textsc{Mini}(P,\nu)$.

*Proof of the claim*: Let $\mathbb{B}$ be the following Turing reduction from $Q_e$ to $\textsc{Mini}(P,\nu)$: Given an instance $a \in \Sigma^*$, the algorithm $\mathbb{B}$ computes in polynomial time $k := C(e,|a|)$. Then it simulates the fpt-T-reduction $\mathbb{A}$ on $(e,a,k)$. Since $a \in Q_e \iff (e,a,k) \in Q$, this algorithm $\mathbb{B}$ correctly decides if $a \in Q_e$.

Moreover (11) implies that

$$\varphi_e(\kappa(e,a,k)) = \varphi_e(k) = O(|a|)$$

for $\varphi_e$ is total. Hence the running time of $\mathbb{A}$ (and hence of $\mathbb{B}$) on $(e,a,k)$ is

$$\varphi_e(\kappa(e,a,k)) \cdot |(e,a,k)|^{O(1)} = O(|a|) \cdot |(e,a,k)|^{O(1)} = O(|a|) \cdot |a|^{O(1)} = O(|a|^{O(1)}).$$

The second equality follows from the fact that $k = C(e,|a|) \leq |a|$ and $e$ is a constant. Thus $\mathbb{B}$ is a desired polynomial time Turing reduction, which proves claim 1.

Since $\nu$ is polynomial time computable, without loss of generality we assume

$$\nu(x) \leq 2^{|x|} \tag{12}$$

for any $x \in (\Sigma')^*$. Let

$$P' := \left\{(x, 2^{2^{|x|}}) \mid x \in P\right\}.[4]$$

It follows that for all $(x,m) \in P'$

$$m \;=\; 2^{2^{|x|}}. \tag{13}$$

*Claim 2.* $\textsc{Mini}(P,\nu)$ is 2-exptime Turing reducible to $P'$.
*Proof of the claim*: Given an instance $(x,m) \in (\Sigma')^* \times \mathbb{N}$, if $m < |x|$, it is a 'no'-instance. Otherwise

$$(x,m) \in \textsc{Mini}(P,\nu) \iff x \in P \iff (x, 2^{2^{|x|}}) \in P'.$$

This can be easily turned into a 2-exptime Turing reduction, thus claim 2 is proved.

Note that for all $(x,m) \in P'$ we have

$$\begin{aligned} \mu(x,m) &= \left\lceil \frac{\nu(x)}{\log m} \right\rceil \\ &\le \left\lceil \frac{2^{|x|}}{\log 2^{2^{|x|}}} \right\rceil \qquad \text{by (12) and (13)} \\ &= 1. \end{aligned} \tag{14}$$

*Claim 3.* There is a polynomial time Turing reduction $\mathbb{B}$ from $P'$ to $Q$. Moreover the set

$$\big\{(e,a,k) \in Q \mid \text{for some } (x,m) \in (\Sigma')^* \times \mathbb{N}, \text{there is an oracle query "}(e,a,k) \in Q\text{?"} \\ \text{in the computation of } \mathbb{B} \text{ on } (x,m)\big\}.$$

is *finite*.
*Proof of the claim*: Recall we assume $p$-$\textsc{Mini}(P,\nu) \equiv^{\text{fpt-T}} (Q,\kappa)$. So there is an algorithm $\mathbb{A}'$ with an oracle to $(Q,\kappa)$ and a computable functions $g : \mathbb{N} \to \mathbb{N}$, such that, given an instance $(x,m) \in (\Sigma')^* \times \mathbb{N}$:

(E1) $\mathbb{A}'$ decides if $(x,m) \in \textsc{Mini}(P,\nu)$ in time

$$g(\mu(x,m)) \cdot |(x,m)|^{O(1)}.$$

(E2) For each oracle query "$(e',a',k') \in Q$?" posed by $\mathbb{A}'$,

$$\kappa(e',a',k') = k' \le g'(\mu(x,m))$$

for a computable function $g' : \mathbb{N} \to \mathbb{N}$. For simplicity, we assume $g' = g$.

Given an instance $(x,m)$ of $P'$, the algorithm $\mathbb{B}$ first checks in polynomial time if $m \neq 2^{2^{|x|}}$ or $\mu(x,m) > 1$. If so, then $(x,m)$ is a 'no'-instance by (13) and (14). Otherwise $m = 2^{2^{|x|}} \ge |x|$, and $\mu(x,m) = 1$. It follows that

$$(x,m) \in P' \iff x \in P \iff (x,m) \in \textsc{Mini}(P,\nu).$$

Therefore $\mathbb{B}$ decides if $(x,m) \in P'$ by simulating $\mathbb{A}'$ on $(x,m)$, which by (E1) requires time

$$g(\mu(x,m)) \cdot |(x,m)|^{O(1)} = g(1) \cdot |(x,m)|^{O(1)},$$

therefore polynomial in $|(x,m)|$.

[4]Here $2^{2^{|x|}}$ is represented in unary.

Now for any oracle query "$(e', a', k') \in Q$?" that occurs in the simulation of $\mathbb{A}'$ on $(x, m)$, by (E2) we have

$$k' = \kappa(e', a', k') \leq g(\mu(x, m)) = g(1).$$

If $(e', a', k') \in Q$ and $e' = e$, then $k' = C(e, |a'|)$ is the maximum such that $\varphi_e(1), \ldots, \varphi_e(k')$ can be computed in at most $|a'|$ steps and all bounded by $|a'|$. As $k'$ is bounded by the fixed constant $g(1)$ independent of $x$ and $m$, there are only finitely many such $a'$ with $(e, a', k') \in Q$ for all possible instances $(x, m)$, otherwise $\varphi_e$ is not total. Thus claim 3 is proved.

*Claim 4.* There is a polynomial time Turing reduction from $P'$ to

$$L_e = \big\{(e', a) \mid e' \neq e, a \in Q_{e'}\big\},$$

*Proof of the claim*: By Claim 3 it suffices to give a polynomial time Turing reduction from

$$\{(e', a, k) \in Q \mid e' \neq e\}$$

to $L_e$: For any instance $(e', a, k)$, if $k \neq C(e', |a|)$ or $e' = e$, then it is a 'no'-instance. Otherwise

$$(e', a, k) \in \big\{(e', a, k) \in Q \mid e' \neq e\big\} \iff (e', a) \in L_e.$$

This proves claim 4.

Combining claims 1,2, and 4, we have a 2-exptime Turing reduction from $Q_e$ to $L_e$ by Lemma 28, which contradicts our construction. □