On quantum computation theory

van Dam, W.K.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 4

Quantum Bounded Queries

It is known that that a super-polynomial quantum improvement can only be obtained if we consider problems that are more structured than those in the black-box model of computation.[10] In this chapter we look at the query complexity of problems that can be computed in polynomial time with the help of, for example, an oracle for the SAT problem. It is shown how in this setting a quantum computer requires less queries than a classical computer, provided that standard complexity assumptions like $P \neq NP$ are true.

4.1 Introduction

We combine the classical notions and techniques for bounded query classes with those developed in quantum computing. We give strong evidence that quantum queries to an oracle in the class NP does indeed reduce the query complexity of decision problems. Under traditional complexity assumptions, we obtain an exponential speed-up between the quantum and the classical query complexity of function classes.

For decision problems and function classes we obtain the following results (see the appendix of this thesis for a brief overview of these complexity classes):

- $P^{NP[2^k]} \subseteq EQP^{NP[k]}$
- $P^{NP[2^{k+1} - 2]} \subseteq EQP^{NP[k]}$
- $FP^{NP[2^{k+1} - 2]} \subseteq FEQP^{NP[2^k]}$
- $FP^{NP} \subseteq FEQP^{NP[O(\log n)]}$

For sets $A$ that are many-one complete for PSPACE or EXP we show that $FP^A \subseteq FEQP^A[1]$. Sets $A$ that are many-one complete for PP have the property that $FP^A \subseteq FEQP^A[1]$. In general we prove that for any set $A$ there is a set $X$ such that $FP^A \subseteq FEQP^{X[1]}$, establishing that no set is superterse in the quantum setting.
Chapter 4. Quantum Bounded Queries

The query complexity of a function is the minimum number of queries (to some oracle) that are needed to compute one value of this function. With bounded query complexity we look at the set of functions that can be calculated if we put an upper bound on the number of queries that we allow the computer to ask the oracle. This notion has been extensively studied both in the resource bounded setting [2, 4, 5, 13, 12, 11, 17, 60, 75, 104] and in the recursive setting [15, 16]. This notion and its variants has lead to a series of techniques and tools that are used throughout complexity theory.

In this chapter we combine some of the bounded query notions with quantum computation. The main goal is to further—as was done by Fortnow and Rogers [43]—the incorporation of quantum computation complexity classes into standard classical complexity theory. We feel that the synthesis of quantum computation and classical complexity theory serves two purposes. First, it is important to know the limits of feasible quantum computation and these can be clarified by expressing them in the framework of classical computation. Second, the insights of quantum computation can be useful for classical complexity theory in turn.

We start out with the class of sets (or decision problems) that are computable in polynomial time with bounded queries to a set in NP. We consider the setting where the queries are adaptive (i.e., a query may depend on the answers to previous ones), as well as where they are non-adaptive. Classically, it is known that any decision problem that can be solved in polynomial time with \( k \) adaptive queries to a set in NP (the class \( \mathcal{P}^{NP[k]} \)) can also be solved with \( 2^k - 1 \) non-adaptive queries (the class \( \mathcal{P}'^{NP[2^k-1]} \), where "\( \forall \)" indicates the parallel or non-adaptive queries), and vice-versa [13]. In other words: \( \mathcal{P}^{NP[k]} = \mathcal{P}'^{NP[2^k-1]} \). Moreover, there is strong evidence that this trade-off is optimal in the sense that every non-adaptive class \( \mathcal{P}'^{NP[k]} \) is different for different values of \( k \). For example if \( \mathcal{P}'^{NP[2]} \subseteq \mathcal{P}^{NP[1]} \), then the polynomial hierarchy collapses [60] (see also [27, 52]).

We will see that if we allow the query machine to make use of quantum mechanical effects such as superposition and interference the situation changes. In the non-adaptive case we will show that \( 2^k \) classical queries can be simulated with only \( k \) non-adaptive ones on a quantum computer and in the adaptive case we show how to simulate \( 2^{k+1} - 2 \) classical queries with only \( k \) quantum queries. The natural quantum analog of \( \mathcal{P} \) is the class \( \mathcal{EQP} \), which stands for exact quantum polynomial time. This is the class of sets or decision problems that is computable in polynomial time with a quantum computer that makes no errors (i.e., is exact). Then, our results are that

\[
\mathcal{P}^{NP[2^k]} \subseteq \mathcal{EQP}^{NP[k]} \quad \text{and} \quad \mathcal{P}^{NP[2^k+1]-2} \subseteq \mathcal{EQP}^{NP[k]}.
\]

In particular it follows from this result that \( \mathcal{P}^{NP[2]} \subseteq \mathcal{EQP}^{NP[1]} \) (see also [36]).

In order to prove these results we combine the classical mind-change technique [13] with the one query version (see [31]) of the first quantum algorithm developed by David Deutsch [38].

Next, we turn our attention to functions that are computable with bounded queries to a set in NP. Compared to the decision problems there is probably no nice trade-off
between adaptive and non-adaptive queries for functions. This is because the following is known [17]: for any \( k \) the inclusion \( \text{FP}^{\text{NP}}[k] \subseteq \text{FP}^{\text{NP}}[k-1] \) implies that \( P = \text{NP} \). Moreover, if \( \text{FP}^{\text{NP}} \subseteq \text{FP}^{\text{NP}[O(\log n)]} \) then the polynomial time hierarchy collapses [12, 87, 98].

When the adaptive query machine is a quantum computer, things are different and we seem to get a trade-off between adaptiveness and query complexity. We show the following:

\[
\text{FP}^{\text{NP}[2^{k+1}-2]} \subseteq \text{FEQP}^{\text{NP}[2^k]} \quad \text{and} \quad \text{FP}^{\text{NP}} \subseteq \text{FEQP}^{\text{NP}[O(\log n)]}.
\]

Here \( \text{FEQP}^{\text{NP}[k]} \) is the class of functions that is computable by an exact quantum Turing machine that runs in polynomial time and is allowed to make \( k \) queries to a set in \( \text{NP} \).

The proofs of these results use our previous results on decision problems and a quantum algorithm developed by Deutsch-Jozsa [39] and Bernstein-Vazirani [22].

Using the same ideas we are able to show that for any set \( A \) there exists a set \( X \) such that \( \text{FP}^{A} \subseteq \text{FEQP}^{X[1]} \), establishing that no set is 'superterse'. Also because the complexity of \( X \) is not much harder than that of \( A \) (the problem \( X \) is Turing reducible to \( A \)), we get quite general theorems for complete sets of complexity classes.

For a complexity class \( C \) that is closed under Turing reductions, and a problem \( A \in C \) that is many-one complete for the class \( C \), the inclusion \( \text{FP}^{C} \subseteq \text{FEQP}^{A[1]} \) is proven. This holds in particular for the set \( \text{QBF} \) of the true quantified Boolean formulae which is a \( \text{PSPACE} \) complete problem, and the complete sets for the class \( \text{EXP} \). If \( C \) is a class that is closed under truth-table reductions, then it holds that \( \text{FP}^{C} \subseteq \text{FEQP}^{A[1]} \). The Theta levels of the polynomial hierarchy and \( \text{PP} \) are examples of such classes.

The ingredients for all our results are standard quantum algorithms combined with well known techniques from complexity theory. Nevertheless we feel that this combination gives a new point of view on the nature of bounded query classes and the structure of complete sets in general.

### 4.2 Classical Complexity Theory

We assume the reader to be familiar with basic notions of complexity theory such as the various complexity classes and types of reducibility as can be found in many textbooks in the area [6, 7, 46, 58]. The essentials for this chapter are mentioned below.

For a (decision problem) \( A \) we will identify \( A \) with its characteristic function. Hence for a string \( x \) we have \( A(x) \in \{0, 1\} \), and \( A(x) = 1 \) if and only if \( x \in A \). A class \( C \) consists of a set of decision problems. A problem \( A \) is many-one poly-time, or \( \leq_{m}^{p} \)-complete for a class \( C \) if for any problem \( B \in C \), there exists a polynomial-time computable function or "Karp-reduction" \( \tau \) such that \( x \in B \) if and only if \( \tau(x) \in A \). The typical example of such a complete problem is \( \text{SAT} \) (the set of satisfiable Boolean formulae) which is \( \leq_{m}^{p} \)-complete for the class \( \text{NP} \). The class \( \text{FP} \) indicates the set of functions that can be calculated on a polynomial time, deterministic Turing machine.
Chapter 4. Quantum Bounded Queries

An oracle Turing machine is non-adaptive, if it can produce a list of all of the oracle queries it is going to make before it makes the first query. For any set $A$, the elements of the class $\text{P}^A[k]$ (FP$^A[k]$) are the languages (functions) that are computable by polynomial time Turing machines that accesses the oracle $A$ at most $k$ times on each input. The class $\text{P}^A[k]$ and FP$^A[k]$ allow only non-adaptive access to $A$. The notation $\text{P}^{\text{NP}(g(n))}$ is used to indicate algorithms that might require $g(n)$ calls to an NP oracle, where $g$ is a function of the input size $n$.

The class NP can be generalized by defining the polynomial time hierarchy. We start with the definitions $\Delta_1^P = \text{P}$ and $\Sigma_1^P = \text{NP}$, and then for the higher levels continue in an inductive fashion according to $\Delta_{i+1}^P = \text{P}^{\Delta_i^P}$ and $\Sigma_{i+1}^P = \text{NP}^{\Sigma_i^P}$ for $i = 2, 3, \ldots$.

Many complexity theorists conjecture that this polynomial time hierarchy is infinite, i.e., $\Sigma_i^P \neq \Sigma_i^P$ for all $i$.

A class $C$ of languages is closed under Turing (truth-table) reduction if any decision problem that can be solved with a polynomial time Turing machine and (non-adaptive) queries to a set in $C$, is itself also an element of $C$. Examples of such classes are PSPACE, EXP, and the Delta levels $\Delta_i^P$. The classes PP and $\Theta_i^P = \Sigma_i^P \cap \Sigma_i^P$ (Theta levels of the polynomial hierarchy) are for example closed under this truth-table-reduction.

4.3 Quantum Complexity Classes

The class EQP is the collection of those sets that can be computed by a quantum Turing machine that runs in polynomial time and accepts every string $j$ with probability 1 or 0. Likewise, we define the class of functions FEQP as the class of functions that can be computed exactly by some quantum Turing machine that runs in polynomial time. The output of the Turing machine is the function value (rather than a single decision bit).

We model oracle computation as follows (see also [19]). An oracle Turing machine has a special query tape, and during the computation the Turing machine may enter a special pre-query state to make a query to the oracle set $A$. Suppose the query tape contains the state $|i\rangle|b\rangle$ ($i$ represents the query and $b$ is a bit meant to receive the answer to the query). The result of this operation is that after the call the machine will go into a special state called the post-query state and that the query tape has changed into $|i\rangle|A(i) \oplus b\rangle$, where $\oplus$ is the EXCLUSIVE OR. We will denote this unitary operation by $U_A$. Note that $U_A$ only changes the contents of the special query answer bit $b$, and leaves all the other registers unchanged.

As with classical oracle computation, we make the distinction between adaptive and non-adaptive quantum oracle machines. We call a quantum oracle machine non-adaptive if on every computation path a list of all the oracle queries (on this path) is generated before the first query is made.

The class EQP$^A[k]$ are the sets recognized by an exact quantum Turing machine that runs in polynomial time and makes at most $k$ adaptive queries to the oracle for
4.4. Decision Problems

A. Likewise, we define classes like $\text{EQP}^A[q(n)]$, $\text{FEQP}^A[q(n)]$, and $\text{FEQP}^A[q(n)]$, for non-adaptive decision, adaptive function, and non-adaptive function classes respectively (with $q(n)$ a function that gives an upper bound on the number of queries and $n$ the size of the input string).

4.4 Decision Problems

In this section we will investigate the extra power that a polynomial time, exact quantum computer yields compared to classical deterministic computation when querying a set in the class $\text{NP}$. In the case of deterministic computation the following equality between adaptive and non-adaptive queries to $\text{NP}$ is well known.

**Fact 6** [13, 29, 104]

1. For all $k \geq 0$ we have $P^{\text{NP}[2^k-1]}_n = P^{\text{NP}[k]}_n$.

2. For any polynomial $q(n) > 1$ the equality $P^{\text{NP}[q(n)]}_n = P^{\text{NP}[O(\log(q(n)))]}_n$ holds.

**Proof:** Both items are proved in a similar way which has two parts. The first part shows that computing a function in $P^{\text{NP}[2^k-1]}_n$ can be reduced to computing the parity of $2^k - 1$ other queries to $\text{NP}$. The second part then proceeds by showing that using binary search one can compute the parity of $2^k - 1$ $\text{NP}$-queries with $k$ adaptive queries to $\text{SAT}$. On the other hand, it is trivial to see that any computation with $k$ adaptive queries can be simulated exhaustively with $2^k - 1$ non-adaptive oracle calls.

There is also strong evidence that the above trade-off is tight (see [14, 60]). It follows for example that if $P^{\text{NP}[2]}_n = P^{\text{NP}[1]}_n$ then the polynomial hierarchy collapses [60]. (See [27] for the latest developments with respect to this question.)

Perhaps surprisingly the situation changes when the query machine is quantum mechanical. Using the one-call-parity trick of Fact 3, we will show that a quantum Turing machine can compute decision problems with half the number of non-adaptive queries.

**Theorem 3** For all $k \geq 0$ we have the inclusion $P^{\text{NP}[2k]}_n \subseteq \text{EQP}^{\text{NP}[k]}_n$.

**Proof:** Without loss of generality we will assume that the queries are made to $\text{SAT}$, and that the predicate that is computable with $2k$ queries to $\text{SAT}$ is $f(x)$. Let $\psi_1, \psi_2, \ldots, \psi_{2k}$ be the queries that the computation of $f(x)$ makes. We will use the proof technique of Fact 6 (also called mind-change technique) which enables us to compute $f(x)$ by calculating the single bit $\text{SAT}(\phi_1) \oplus \cdots \oplus \text{SAT}(\phi_{2k})$. Here the new formulae $\phi_1, \ldots, \phi_{2k}$ can be computed in polynomial time from $\psi_1, \ldots, \psi_{2k}, f$, and $x$, but without having to consult $\text{SAT}$.

Next, we use Fact 3 to compute the parity $\text{SAT}(\phi_i) \oplus \text{SAT}(\phi_{i+1})$ for odd $i$ ($1 \leq i < 2k$) with $k$ non-adaptive queries to $\text{SAT}$. Finally we compute the parity of these answers, thus obtaining the necessary information for calculating $f(x)$. 

\[\square\]
**Lemma 3** \( \text{P}^{\text{NP}[2]} \subseteq \text{EQP}^{\text{NP}[1]} \) (see [36]).

We do not know whether this is tight. It would be interesting to either improve this result to \( \text{P}^{\text{NP}[2]} \subseteq \text{EQP}^{\text{NP}[1]} \) or to show as a consequence of this that the polynomial time hierarchy collapses.

Fact 6 relates adaptive query classes to non-adaptive ones, thereby establishing an exponential gain in the number of queries (\( 2^k - 1 \) versus \( k \) queries). We will now show how to use the Deutsch trick to improve this result slightly in the quantum case.

**Theorem 4** \( \text{P}^{\text{NP}[2k+1-2]} \subseteq \text{EQP}^{\text{NP}[k]} \) for all \( k \geq 0 \).

**Proof:** The proof is by induction on \( k \). For \( k = 1 \) we return to the situation of Lemma 3. Let the predicate \( f(x) \) be computable with \( 2^{k+1} - 2 \) non-adaptive queries to SAT. As in the proof of Theorem 3 we reduce the \( 2^{k+1} - 2 \) queries \( \psi_i \) that \( f(x) \) makes, to the calculation of the parity-bit SAT(\( \phi_1 \)) \( \oplus \cdots \oplus \) SAT(\( \phi_{2^{k+1}-2} \)). Next, we construct \( 2^{k+1} \) - 2 new formulae \( \chi_1, \ldots, \chi_{2^{k+1}-2} \) according to:

\[
\chi_i \text{ is satisfiable } \iff \left| \{ \phi_1, \ldots, \phi_{2^{k+1}-2} \} \cap \text{SAT} \right| \geq i.
\]

The construction of each such \( \chi_i \) can be done in polynomial time. Consider the non-deterministic polynomial time Turing machine \( M \) that on input \( \langle i, \phi_1, \ldots, \phi_{2^{k+1}-2} \rangle \), accepts if and only if it can find for \( i \) of the formulae a satisfying assignment. Cook and Levin [34, 66]—proving that SAT is \( \leq_p \) -complete for NP—showed that any polynomial time non-deterministic Turing machine computation \( M(x) \) in polynomial time can be transformed into a formula that is satisfiable if and only if \( M(x) \) has an accepting computation. Let \( \chi_i \) be the result of this Cook-Levin reduction.

Note the following two properties of those formulae \( \chi_i \):

1. The parity SAT(\( \phi_1 \)) \( \oplus \cdots \oplus \) SAT(\( \phi_{2^{k+1}-2} \)) is the same as the parity SAT(\( \chi_1 \)) \( \oplus \cdots \oplus \) SAT(\( \chi_{2^{k+1}-2} \)).

2. For every \( i \) we have SAT(\( \chi_i \)) \( \geq \) SAT(\( \chi_{i+1} \)).

Now we are ready to make the first query. We compute the parity of \( \chi_{2^{k-1}} \) and \( \chi_{2^{k-1}+2^{k-1}} \). This can be done in one query using Fact 3. By doing this we have at the cost of one query reduced the question of computing the parity of \( 2^{k+1} - 2 \) formulae to computing the parity of \( 2^k - 2 \). These we can solve using \( k - 1 \) queries using the induction hypothesis. To see this observe the following. For convenience set \( a = 2^{k-1} \) and \( b = 2^{k-1} + 2^k - 1 \).

Suppose the parity of \( \chi_a \) and \( \chi_b \) is odd, with \( a < b \). From the second property above, it follows that \( \chi_a = 1 \) and \( \chi_b = 0 \), and hence that \( \chi_1, \ldots, \chi_a \) are all satisfiable and \( \chi_b, \ldots, \chi_{2^{k+1}-2} \) are all unsatisfiable. Also note that \( a \) is even, so the parity of \( \chi_1, \ldots, \chi_{2^{k+1}-2} \) is the same as the parity of \( \chi_{a+1}, \ldots, \chi_{b-1} \) (these are \( 2^k - 2 \) many formulae).

On the other hand assume that the parity of \( \chi_a \) and \( \chi_b \) is even. This means (again using property 2 above) that \( \chi_a, \ldots, \chi_b \) are all either satisfiable or unsatisfiable and
4.5. Functions computable with queries to NP Oracles

hence have even parity. So again the question reduces to the parity of the remaining
formulæ: $\chi_1, \ldots, \chi_{a-1}$ and $\chi_{b+1}, \ldots, \chi_{2^k-1}$. Which happen to be $2^k - 2$ many
formulæ.

In essence the above technique seems to boil down to searching in an ordered list
$X_1, \ldots, X_{2^k-2}$. In [56] it has been shown that this can not be done with less than
\(\frac{\log n}{\log e} - O(1)\) queries. On the other hand, results by Farhi et al. [42] and [56] indicate
that the query complexity of the ordered search problem is upper bounded by \(\frac{1}{\alpha} \log n + O(1)\), with \(\alpha\) at least 1.88\ldots. Using these results it is likely that we can strengthen the
above theorem to $P^{NP[2^k+O(1)]} \subseteq EQP^{NP[k]}$.

4.5 Functions computable with queries to NP Oracles

Now we turn our attention to function classes where the algorithm can output bit strings
rather than single bits. We will see that in this scenario the difference between classical
and quantum computation becomes more pronounced.

We start out by looking at functions that are computable with queries to a complete
set for the class NP. Classically the situation is not as well understood as the class of
decision problems. There is strong evidence that the analog of Fact 6 is not true.

Fact 7 The following holds for the classical, exact computation of functions:

1. If for some $k \geq 0$ we have $FP^{NP[k+1]}_n \subseteq FP^{NP[k]}_n$, then $P = NP$ [17].

2. If for all polynomials $q(n)$ (with $n$ the size of the input string): $FP^{NP[q(n)]}_n \subseteq FP^{NP[O(\log n)]}_n$, then $NP = R$ (and the polynomial hierarchy collapses) [12, 87, 98].

When we allow the adaptive query machine to be quantum mechanical the picture be-
comes again quite different. We will show for example that the inclusion $FP^{NP[q(n)]}_n \subseteq$ $FEQP^{NP[2\log(q(n))]}_n$ holds (and this does not imply $NP = R$ as far as we know). In order
do so we will use Fact 4.

Let us turn back now to our setting of bounded query classes. Using the quantum
tricks of Sections 2.4 and 2.5 we can establish the following result.

Theorem 5 For exact function calculation with the use of an oracle in NP it holds that

1. $FP^{NP[2k+1−2]}_n \subseteq FEQP^{NP[2k]}_n$ for any $k \geq 0$,

2. $FP^{NP}_n \subseteq FEQP^{NP[O(\log n)]}_n$.

Proof: Fix $k \geq 0$, the input $z$ of length $m$ and let $g$ be the function in $FP^{SAT[2^{k+1−2}]}_n$.

Suppose that $g(z) = (a_1 \cdots a_n) = a$ with $n = m^c$ for some $c$ depending on $g$. The
Chapter 4. Quantum Bounded Queries

goal is to obtain the state:

$$|\text{Output}\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{(x,a)} |x\rangle,$$  \hspace{1cm} (4.1)

since with this state one application of $H^\otimes n$ will give us $a = g(z)$ (cf. 2.6). Clearly, we can obtain this state if we have access to a function $f$ with the property

$$f_z(x) = \sum_{i=1}^{n} a_i x_i \mod 2,$$  \hspace{1cm} (4.2)

for every $x \in \{0,1\}^n$.

The goal thus is to transform the function we have access to—SAT in our case—into one that resembles the one in Equation 4.2. The way to do this is to make use of a quantum subroutine. Observe the following: the binary function $f_z(x) = (x,a)$ is in $P^{\text{SAT}(2^{k+1} - 2)}_u$ because we can first compute $g(z) = a$ with $2^{k+1} - 2$ queries to SAT and then determine $(x,a)$. By Theorem 4 this function is computable in EQP$^{\text{SAT}[k]}$. Hence, when we use this adaptive EQP algorithm in superposition we have the desired function $f$. There is however one problem with this approach. The algorithm that comes out of Theorem 4 leaves several of the registers in states depending on the input $x$ and SAT. For example the algorithm that computes the parity of two function calls in one generates a phase of $(-1)$ depending on the value of the first function call (see Equation 2.4). These changes in registers and phase shifts obstruct our base quantum machine and as a consequence the sum computed in Equation 2.6 does not work out the way we want (i.e., the interference pattern is different and terms do not cancel out as nice as before.)

The solution to this kind of ‘garbage’ problem is as follows:

1. Compute $f_z(x)$ with $k$ queries to SAT.

2. Copy the outcome onto an extra auxiliary qubit (by setting the auxiliary bit $b$ to the EXCLUSIVE OR of $b$ and the outcome).

3. Reverse the computation of $f_z(x)$ making another $k$ queries to SAT.

Observe that when we compute $f_z(x)$ in this way, all the phase changes and registers are reset and are in the same state as before computing $f$, except for the auxiliary qubit that contains the answer. Since the subroutine was exact (i.e., in EQP) the answer bit is a classical bit and will not interfere with the rest of the computation. Note that this corresponds exactly to one oracle call to $f$. Thus we simulated 1 call to $f$ with $2k$ queries to SAT and hence have established a way of producing the desired state of Equation 4.1.

The second part of the theorem is proved in a similar way now using part 2 of Fact 6. \hspace{1cm} \Box
4.6. **Terseness, and other Complexity Classes**

The quantum techniques described above are quite general and can be applied to sets outside of NP. Classically the following question has been studied (see [12] for more information). For any set $A$ define the function $F_n^A(x_1, \ldots, x_n) = (A(x_1) \cdots A(x_n))$ which is an $n$ bit vector telling which of the $x_i$'s is in $A$ and which ones are not. A basic question now is: how many queries to $A$ do we need to compute $F_n^A$? Sets for which $F_n^A$ cannot be computed with less than $n$ queries to $A$ (i.e., $F_n^A \not\in \text{FP}^{A[n-1]}$) are called $P$-terse. We call the decision problem $A \text{ P-supertese}$ if $F_n^A \not\in \text{FP}^X[n-1]$ for any set $X$. The next theorem shows that this last notion is not useful in the quantum setting.

**Theorem 6** Let $A$ be a subset of $\mathbb{N}$ and let the function $F_n^A : \mathbb{N}^n \to A^n$ be defined by $F_n^A(x_1, \ldots, x_n) := (A(x_1), \ldots, A(x_n))$, where $A(x) = 0$ if $x \notin A$ and $A(x) = 1$ if $x \in A$. For any set $A$ there exists a set $X \subseteq \mathbb{N}$ such that for all $n$ we have $F_n^A \in \text{FEQP}^{X[1]}$.

**Proof:** Let $X$ be the following set:

$$X = \{ (z_1 \cdots z_n, x_1 \cdots x_n) \mid (F_n^A(z_1, \ldots, z_n), x_1 \cdots x_n) = 1 \}.$$

Using the same approach as the proof of Theorem 5 it is not hard to see that $F_n^A$ can be computed relative $X$ with only a single query. □

Using the same idea we can prove the following general theorem about oracles for complexity classes other than NP.

**Theorem 7** Let $C$ be a complexity class and the set $A \triangleq_m \text{P-complete}$ for $C$.

1. If $C$ is closed under $\leq_m \text{P}$-reductions then $\text{FP}^C = \text{FP}^A \subseteq \text{FEQP}^{[1]} = \text{FEQP}^C[1]$.

2. If $C$ is closed under $\leq_m \text{P}$-reductions then $\text{FP}_m^C = \text{FP}_m^A \subseteq \text{FEQP}^{[1]} = \text{FEQP}^C[1]$.

**Proof:** Let $f$ be the function we want to compute relative to $A$. Without loss of generality we assume that $\ell(f(z)) = \ell(z)^c$ for some $c$ depending only on $f$. As before we construct the following set:

$$X = \{ (z, y) \mid (f(z), y) = 1, \text{ and } \ell(y) = \ell(z)^c = \ell(f(z)) \}.$$

As in Theorem 6 it follows that $f(z)$ is computable with one quantum query to $X$. Since $C$ is closed under $\leq_m \text{P}$-reductions and $X \leq_m \text{P} A$, it follows that $X \in C$. Furthermore, since $A$ is $\leq_m \text{P}$-complete for $C$ it also follows that $X \leq_m \text{P} A$. Thus the quantum query can be made to $A$ itself instead of $X$. The proof of the second part of the theorem is analogous to the first. □

This last theorem gives us immediately the following two lemmas about quantum computation with oracles for some known complexity classes.
Lemma 4

\[
\begin{align*}
FP^{PSPACE} & \subseteq FEQ^{PSPACE}[1] \\
FP^{EXP} & \subseteq FEQ^{EXP}[1] \\
FP^{\Delta^p_t} & \subseteq FEQ^{\Delta^p_t}[1]
\end{align*}
\]

for the Delta levels \(\Delta^p_t\) in the polynomial time hierarchy.

Lemma 5

\[
\begin{align*}
FP^{PP} & \subseteq FEQ^{PP}[1] \\
FP^{\Theta^p_t} & \subseteq FEQ^{\Theta^p_t}[1]
\end{align*}
\]

with \(\Theta^p_{t+1} = P^{\Theta^p_t}_t\).

The first lemma holds in particular for \(A = \text{QBF}\) (the set of true quantified Boolean formulae) which is PSPACE-complete. Observe also that the situation is quite different in the classical setting, since for EXP-complete sets the above is simply not true.

4.7 Conclusions and Open Problems

We have combined techniques from complexity theory with some of the known quantum algorithms. In doing so we showed that a quantum computer can compute certain functions with fewer queries than classical deterministic computers. Many questions however remain. Is it possible to get trade-off results between the adaptive class \(E^{P_{NP}(x)}\) and the non-adaptive \(E^{P_{NP}(x)}_{\leq x-1}\) for quantum machines? Are the results we present here optimal? (Especially the recent results on exact searching in an ordered list\([42]\) and [56] deserve further analysis as they seem to suggest a reduction of the quantum query complexity of Theorems 4 and 5 by a factor of two.)

What can one deduce from the assumption that \(P^{NP} \subseteq EQ^{NP}[1]\)? Is it true that for any set \(A\) we have \(P^{A} \subseteq EQ^{A}[1]\) or are there sets where this is not true? A random set would be a good candidate where more than one quantum query is necessary.