# An Application of the Deutsch-Jozsa Algorithm to Formal Languages and the Word Problem in Groups

Michael Batty[1], Andrea Casaccino[2,⋆], Andrew J. Duncan[1], Sarah Rees[1], and Simone Severini[3]

[1] Department of Mathematics, University of Newcastle upon Tyne, United Kingdom
[2] Information Engineering Department, University of Siena, Italy
ndr981@tin.it
[3] Institute for Quantum Computing and Department of Combinatorics and Optimization, University of Waterloo, Canada

**Abstract.** We adapt the Deutsch-Jozsa algorithm to the context of formal language theory. Specifically, we use the algorithm to distinguish between trivial and nontrivial words in groups given by finite presentations, under the promise that a word is of a certain type. This is done by extending the original algorithm to functions of arbitrary length binary output, with the introduction of a more general concept of parity. We provide examples in which properties of the algorithm allow to reduce the number of oracle queries with respect to the deterministic classical case. This has some consequences for the word problem in groups with a particular kind of presentation.

## 1 The Deutsch-Jozsa Algorithm and Formal Languages

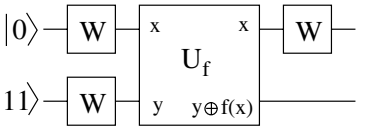
We apply a direct generalization of the Deutsch-Jozsa algorithm to the context of formal language theory. More particularly, we extend the algorithm to distinguish between trivial and nontrivial words in groups given by finite presentations, under the promise that a word is of a certain type. For background information, we refer the reader to [1] and [2].

The Deutsch-Jozsa algorithm concerns maps $f : \{0,1\}^n \longrightarrow \{0,1\}$, which we may think of as words of length $2^n$ in a two-letter alphabet. Instead, let us consider words of length $2^n$ in a four-letter alphabet $\mathcal{A} = \{a, b, c, d\}$. We identify the letters with binary strings of length 2: $a \leftrightarrow 00$, $b \leftrightarrow 01$, $c \leftrightarrow 10$ and $d \leftrightarrow 11$. In this way we can look at words of length $2^n$ as in one-to-one correspondence with maps $f : \{0,1\}^n \longrightarrow \{00, 01, 10, 11\}$.

First, consider the case $n = 1$, that is when the words have length 2. As with the standard formulation of the Deutsch-Jozsa algorithm, the promise will be vacuous in this case. We use the quantum circuit represented below. The circuit essentially implements the Deutsch's algorithm, but with input $|11\rangle$ rather than $|1\rangle$:

⋆ Corresponding author.

Y. Kawano and M. Mosca (Eds.): TQC 2008, LNCS 5106, pp. 57–69, 2008.




After applying the Hadamard gates, the state of the system is

$$W \otimes W_2(|0\rangle \otimes |11\rangle) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)^{\otimes 2}.$$

If $x \in \{0,1\}$ then we have

$$\begin{aligned} & U_f \left( |x\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)^{\otimes 2} \right) \\ &= |x\rangle \otimes \frac{1}{2} \left( |00 \oplus f(x)\rangle - |01 \oplus f(x)\rangle - |10\rangle \oplus f(x)\rangle + |11 \oplus f(x)\rangle \right) \\ &= (-1)^{p(f(x))} |x\rangle \otimes \left(\frac{|1\rangle - |0\rangle}{\sqrt{2}}\right)^{\otimes 2}. \end{aligned}$$

For a binary string $y$, we denote by $p(y)$ the *parity* of $y$, that is if $m$ is the number of 1s in $y$ then $p(y) = m(mod 2)$. After querying the oracle $U_f$, the state is

$$\frac{(-1)^{p(f(0))}|0\rangle + (-1)^{p(f(1))}|1\rangle}{\sqrt{2}} \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)^{\otimes 2}.$$

Finally, after the last Hadamard gate, the first qubit is in the state $|0\rangle$ if $p(f(0)) = p(f(1))$ or $|1\rangle$ if $p(f(0)) \neq p(f(1))$. We shall say that $f$ is *parity constant* if $p(f(0)) = p(f(1))$; *parity balanced*, otherwise. By measuring the final state, we obtain $|0\rangle$ with probability 1 if $f$ is parity balanced and $|1\rangle$ with probability 1 if $f$ is parity constant.

Let us now introduce some terminology related to formal languages. Given a word $w : \{0,1\}^n \longrightarrow \{a,b,c,d\}$, an *anagram* of $w$ is a word of the form $w \circ \phi$, where $\phi : \{0,1\}^n \longrightarrow \{0,1\}^n$ is a permutation. We write $[w]$ for the set of all anagrams of $w$. More formally, let $F$ denote the free monoid on $\{a,b,c,d\}$ and let $M$ denote the free commutative monoid on $\{a,b,c,d\}$. Let $R$ denote the natural map from $F$ to $M$ and suppose that $w \in M$. Then $R(w) = [w]$, the set of all anagrams of $w$. It is clear that the definition of parity balanced and parity constant words extends to $M$ in this way. The set of parity constant words is then a union of sets of anagrams

$$\mathcal{C}_1^{11}(a,b,c,d) = [aa] \cup [bb] \cup [cc] \cup [dd] \cup [bc] \cup [ad].$$

Similarly, the set of parity balanced words is

$$\mathcal{B}_1^{11}(a,b,c,d) = [ab] \cup [ac] \cup [bd] \cup [cd].$$

Note that both the terms of the alphabet in the bracket have the same parity with the notation $a \leftrightarrow 00$, $b \leftrightarrow 01$, $c \leftrightarrow 10$ and $d \leftrightarrow 11$.

Suppose not to input $|11\rangle$ into the auxiliary workspace, but rather some other number $0 \leq n \leq 3$. How does this affect the sets of words we can distinguish between? It is interesting to observe that we may define as follows a more general type of "parity". Let $x$ be a nontrivial element of $\{00, 01, 10, 11\}$, where the latter set is considered in the natural way as the vector space $(\mathbb{Z}_2)^2 = \mathbb{Z}_2 \oplus \mathbb{Z}_2$. Define $p^x(y)$ to be equal to 0, if $y$ is in the subspace $\langle x \rangle = \{00, x\}$ and equal to 1, otherwise. With this notation, $p^{11}(y) = p(y)$, the usual parity function. A similar circuit, taking the auxiliary input $\neg(x)$, that is the binary complement of $x$, will distinguish between whether the word is $x$-constant or $x$-balanced (where these terms have the obvious meaning). Again, measurement of the state will yield this information with certainty. It is clear that if $x = 00$ then the output of the circuit is independent of $f$, and so this is of no use. Let us now suppose that $x = 01$. Then $x$-constant means that the output of $f$ are in the same coset of $\{0, 1\}$ in $(\mathbb{Z}_2)^2$ and $x$-balanced means that $f(0)$ and $f(1)$ are in different cosets, or, in other words, both in or out the subspace $\langle x \rangle = \{00, x\}$. The set of 01-constant words is

$$\mathcal{C}_1^{01}(a, b, c, d) = [aa] \cup [bb] \cup [cc] \cup [dd] \cup [ab] \cup [cd]$$

and the set of 01-balanced words is

$$\mathcal{B}_1^{01}(a, b, c, d) = [ac] \cup [ad] \cup [bc] \cup [bd].$$

With the same notation,

$$\mathcal{C}_1^{10}(a, b, c, d) = [aa] \cup [bb] \cup [cc] \cup [dd] \cup [ac] \cup [bd]$$

and

$$\mathcal{B}_1^{10}(a, b, c, d) = [ab] \cup [ad] \cup [bc] \cup [cd].$$

As before, the first term and the second term in the bracket represent the first output and the second output of the function, respectively. Also the parity is the same as described before. Note that when the set is parity constant both terms are in or out the subspace $\langle x \rangle$, while in the parity balanced case one term is in the subspace and the other one is out.

It is now useful to introduce some general notation. Let $x \in \{0, 1\}^2$. We denote the set of *$x$-constant* words of length $2^n$ over $\mathcal{A}$ by $\mathcal{C}_n^x(\mathcal{A})$ and the set of *$x$-balanced* words of length $2^n$ over $\mathcal{A}$ by $\mathcal{B}_n^x(\mathcal{A})$. We write $\mathcal{F}_n^x(\mathcal{A}) = \mathcal{C}_n^x(\mathcal{A}) \cup \mathcal{B}_n^x(\mathcal{A})$ and call this the set of *$x$-feasible* words of length $2^n$. The following fact is important in the context of our discussion.

**Theorem 1.** *Given a word $w \in \mathcal{F}_n^x$, we can decide with a single quantum query whether it is in $\mathcal{C}_n^x$ or $\mathcal{B}_n^x$.*

It is useful to observe that already in the seminal work [3], it was pointed out that a classical randomized algorithm solves the Deutsch-Jozsa task with 3 classical queries on average, whereas the quantum approach solve it with probability 1 using one single quantum query (see also [4]). Here the output of the function $f$ is no more a single bit but a bit string. Particulary, the possible output of

the function is an $n$ bit string where $n$ is $\log_2$ (the alphabet symbol length). Therefore, first of all, a word is given by $k$ repeated random output of the function, where $k$ is the length of the word. In other terms, a word is like a sequence obtained by tossing a dice with $j$ faces, where $j$ is the number of letters in the alphabet. The parity constant output and the parity balanced output are not mutually exclusive over a fixed word of length $k$ and do not cover all the possible $k$ output combinations of the function. This means that it is possible to see a word as obtained by $k$ repeated queries to the function (*e.g.*, given a word in a certain set it is possible to deduce the parity of the function).

From this point of view, it is easy to see that the probability of being constant over all possible anagrams, interpreting the output binary string of $k$ queries as anagrams of $k$ letters, is higher than the balanced case, when $k$ is small and the difference decreases while the number of queries to the function increases. As long as any possible parity function partitions into two classes the function co-domain, it is clear that an higher number of possible outputs is not relevant to the number of classical queries required to distinguish between the parity constant and parity balanced cases. What is remarkable about this method is that it allows to extend the Deutsch-Jozsa algorithm to functions with output of any dimension, $f : \{0,1\}^n \longrightarrow \{0,1\}^n$. Defining appropriate parities, like for groups or code membership problems, could give arise to potentially interesting applications.

It is clarifying to fully work out an example for $n = 2$. If $X$ is a subset of $\{01, 10, 11\}$ then we write $\mathcal{F}_n^X(\mathcal{A})$ for $\cap_{x\in X}\mathcal{F}_n^x(\mathcal{A})$. We have

$$\begin{aligned}
\mathcal{C}_2^{11}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [aaad] \\
&\quad \cup [aadd] \cup [addd] \cup [bbbc] \cup [bbcc] \cup [bccc], \\
\mathcal{B}_2^{11}(a,b,c,d) &= [aabb] \cup [aacc] \cup [bbdd] \cup [ccdd] \\
&\quad \cup [aabc] \cup [bcdd] \cup [abbd] \cup [accd] \cup [abcd], \\
\mathcal{C}_2^{01}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [aaab] \\
&\quad \cup [aabb] \cup [abbb] \cup [cccd] \cup [ccdd] \cup [cddd], \\
\mathcal{B}_2^{01}(a,b,c,d) &= [aacc] \cup [aadd] \cup [bbcc] \cup [bbdd] \\
&\quad \cup [aacd] \cup [bbcd] \cup [abcc] \cup [abdd] \cup [abcd], \\
\mathcal{F}_2^{\{01,11\}} &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [aabb] \\
&\quad \cup [aacc] \cup [aadd] \cup [bbcc] \cup [bbdd] \cup [ccdd] \cup [abcd].
\end{aligned}$$

When $n = 2$, in the parity balanced case half of the output is of the same parity. We also have

$$\begin{aligned}
\mathcal{B}_2^{11}(a,b,c,d) \cap \mathcal{B}_2^{01}(a,b,c,d) &= [abcd] \cup [aacc] \cup [bbdd], \\
\mathcal{C}_2^{11}(a,b,c,d) \cap \mathcal{B}_2^{01}(a,b,c,d) &= [aadd] \cup [bbcc], \\
\mathcal{B}_2^{11}(a,b,c,d) \cap \mathcal{C}_2^{01}(a,b,c,d) &= [aabb] \cup [ccdd], \\
\mathcal{C}_2^{11}(a,b,c,d) \cap \mathcal{C}_2^{01}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd].
\end{aligned}$$

Therefore, given a word in $\mathcal{F}_2^{\{01,11\}}$, we can decide with two quantum queries in which of these four languages the words is. The remaining possibilities for $x$ are

$$\begin{aligned}
\mathcal{C}_2^{10}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [aaac] \\
&\quad \cup [aacc] \cup [accc] \cup [bbbd] \cup [bbdd] \cup [bddd], \\
\mathcal{B}_2^{10}(a,b,c,d) &= [aabb] \cup [aadd] \cup [bbcc] \cup [ccdd] \cup [aabd] \\
&\quad \cup [bccd] \cup [abbc] \cup [acdd] \cup [abcd], \\
\mathcal{F}_2^{\{10,11\}} &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [aabb] \cup [aacc] \\
&\quad \cup [aadd] \cup [bbcc] \cup [bbdd] \cup [ccdd] \cup [abcd].
\end{aligned}$$

We then have

$$\mathcal{F}_2^{\{01,11\}} = \mathcal{F}_2^{\{10,11\}}.$$

It can be checked that this is also equal to $\mathcal{F}_2^{\{01,10\}}$. However, the three possibilities $X = \{01, 11\}, \{10, 11\}$ and $\{01, 10\}$ all distinguish between different languages, since we have

$$\begin{aligned}
\mathcal{B}_2^{11}(a,b,c,d) \cap \mathcal{B}_2^{10}(a,b,c,d) &= [abcd] \cup [aabb] \cup [ccdd], \\
\mathcal{C}_2^{11}(a,b,c,d) \cap \mathcal{B}_2^{10}(a,b,c,d) &= [aadd] \cup [bbcc], \\
\mathcal{B}_2^{11}(a,b,c,d) \cap \mathcal{C}_2^{10}(a,b,c,d) &= [aacc] \cup [bbdd], \\
\mathcal{C}_2^{11}(a,b,c,d) \cap \mathcal{C}_2^{10}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd], \\
\mathcal{B}_2^{01}(a,b,c,d) \cap \mathcal{B}_2^{10}(a,b,c,d) &= [abcd] \cup [aadd] \cup [bbcc], \\
\mathcal{C}_2^{01}(a,b,c,d) \cap \mathcal{B}_2^{10}(a,b,c,d) &= [aabb] \cup [ccdd], \\
\mathcal{B}_2^{01}(a,b,c,d) \cap \mathcal{C}_2^{10}(a,b,c,d) &= [aacc] \cup [bbdd], \\
\mathcal{C}_2^{01}(a,b,c,d) \cap \mathcal{C}_2^{10}(a,b,c,d) &= [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd].
\end{aligned}$$

This provides an improvement over the classical deterministic setting, where we need three queries to distinguish any of these sets of four languages.

We have then seen that

$$\mathcal{F}_2^{\{01,11\}} = \mathcal{F}_2^{\{10,11\}} = \mathcal{F}_2^{\{01,10\}} = \mathcal{F}_2^{\{01,10,11\}}.$$

It may be interesting to consider larger alphabets:

$$\{a, b, c, d, e, f, g, h\} \to \{000, 001, 010, 011, 100, 101, 110, 111\}.$$

Here, it still possible to define a parity, based on the even number of 1s, like $p^{11}$. This is equivalent to determine if a word $w$ is in the subspace {000,011,101,110}, also denoted $p^{adfg}$. In this case, the set of parity constant and parity balanced words can be obtained using the auxiliary input $|111\rangle$ in the circuit described before:

$$
\begin{aligned}
& U_f\left(|x\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{3}}\right)^{\otimes 3}\right) \\
&= |x\rangle \otimes \frac{1}{2}(|000 \oplus f(x)\rangle - |001 \oplus f(x)\rangle - |010 \oplus f(x)\rangle + |011 \oplus f(x)\rangle \\
&- |100 \oplus f(x)\rangle + |101 \oplus f(x)\rangle + |110 \oplus f(x)\rangle - |111 \oplus f(x)\rangle) \\
&= (-1)^{p(f(x))}|x\rangle \otimes \left(\frac{|1\rangle - |0\rangle}{\sqrt{3}}\right)^{\otimes 3}.
\end{aligned}
$$

As result of this input, $U_f$ gives (*i.e*, it is possible to recognize the membership of a letter from the plus sign in front of term) the following set

$$
\begin{aligned}
\mathcal{C}_1^{adfg}(a,b,c,d,e,f,g) &= [aa] \cup [bb] \cup [cc] \cup [dd] \cup [ee] \cup [ff] \cup [gg] \cup [hh] \cup [ad] \cup [af] \\
&\cup [ag] \cup [df] \cup [dg] \cup [fg] \cup [bc] \cup [be] \cup [bh] \cup [ce] \cup [ch] \cup [eh].
\end{aligned}
$$

Similarly, the set of parity balanced words is

$$
\begin{aligned}
\mathcal{B}_1^{adfg}(a,b,c,d,e,f,g) = &[ab] \cup [ac] \cup [bd] \cup [cd] \cup [ae] \cup [ah] \cup [de] \cup [dh] \\
&\cup [bf] \cup [bg] \cup [cf] \cup [cg] \cup [fe] \cup [fh] \cup [ge] \cup [gh].
\end{aligned}
$$

Other parities can be defined considering different set of vectors. For our purposes it is sufficient to define a set composed by the elements $p^{abcd} = \{000, 001, 010, 011\}$. This plays the same role as $p^{01}$. In this case, the set of parity constant word can be obtained by using $|100\rangle$ as auxiliary input. The circuit has the following output:

$$
\begin{aligned}
& U_f\left(|x\rangle \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{3}}\right)\left(\frac{|0\rangle + |1\rangle}{\sqrt{3}}\right)^{\otimes 2}\right) \\
&= |x\rangle \otimes \frac{1}{2}(|000 \oplus f(x)\rangle + |001 \oplus f(x)\rangle + |010\rangle \oplus f(x)\rangle + |011 \oplus f(x)\rangle \\
&- |100 \oplus f(x)\rangle - |101 \oplus f(x)\rangle - |110 \oplus f(x)\rangle - |111 \oplus f(x)\rangle) \\
&= (-1)^{p(f(x))}|x\rangle \otimes \left(\frac{|1\rangle - |0\rangle}{\sqrt{3}}\right)\left(\frac{|1\rangle + |0\rangle}{\sqrt{3}}\right)^{\otimes 2}.
\end{aligned}
$$

As discussed before this procedure gives the following sets:

$$
\begin{aligned}
\mathcal{C}_1^{abcd}(a,b,c,d,e,f,g) &= [aa] \cup [bb] \cup [cc] \cup [dd] \cup [ee] \cup [ff] \cup [gg] \cup [hh] \cup [ab] \cup [ac] \\
&\cup [ad] \cup [bc] \cup [bd] \cup [cd] \cup [ef] \cup [eg] \cup [eh] \cup [fg] \cup [fh] \cup [gh].
\end{aligned}
$$

The set of parity balanced words is

$$
\begin{aligned}
\mathcal{B}_1^{abcd}(a,b,c,d,e,f,g) = &[ae] \cup [af] \cup [ag] \cup [ah] \cup [be] \cup [bf] \cup [bg] \cup [bh] \\
&\cup [ce] \cup [cf] \cup [cg] \cup [ch] \cup [de] \cup [df] \cup [dg] \cup [dh].
\end{aligned}
$$

For reasons that will be clear later, it is important to define also the parity, based on the subspace $p^{adeh} = \{000, 011, 101, 111\}$, for which the set of parity constant words is obtained by setting as auxiliary input the state $|011\rangle$:

$$
\begin{aligned}
& U_f \left( |x\rangle \otimes \left( \frac{|0\rangle + |1\rangle}{\sqrt{3}} \right) \left( \frac{|0\rangle - |1\rangle}{\sqrt{3}} \right)^{\otimes 2} \right) \\
&= |x\rangle \otimes \frac{1}{2}(|000 \oplus f(x)\rangle - |001 \oplus f(x)\rangle - |010\rangle \oplus f(x)\rangle + |011 \oplus f(x)\rangle + \\
&- |100 \oplus f(x)\rangle + |101 \oplus f(x)\rangle - |110 \oplus f(x)\rangle + |111 \oplus f(x)\rangle) \\
&= (-1)^{p(f(x))}|x\rangle \otimes \left( \frac{|1\rangle - |0\rangle}{\sqrt{3}} \right)^{\otimes 2} \left( \frac{|1\rangle + |0\rangle}{\sqrt{3}} \right).
\end{aligned}
$$

The sets produced are represented by

$$
\begin{aligned}
\mathcal{C}_1^{adeh}(a,b,c,d,e,f,g) \;=\; & [aa] \cup [bb] \cup [cc] \cup [dd] \cup [ee] \cup [ff] \cup [gg] \cup [hh] \cup [ad] \cup [ae] \\
& \cup [ah] \cup [de] \cup [dh] \cup [eh] \cup [bc] \cup [bf] \cup [bg] \cup [cf] \cup [cg] \cup [fg];
\end{aligned}
$$

for the balanced case, we have

$$
\begin{aligned}
\mathcal{B}_1^{adeh}(a,b,c,d,e,f,g) = & [ab] \cup [ac] \cup [af] \cup [ag] \cup [db] \cup [dc] \cup [df] \cup [dg] \\
& \cup [eb] \cup [ec] \cup [ef] \cup [eg] \cup [hb] \cup [hc] \cup [hf] \cup [hg].
\end{aligned}
$$

It is indeed possible to generalize the circuit for an arbitrary length binary function co-domain. In particular, the length of the output binary string will be determined by $log_2$ of the cardinality of the alphabet considered (for example, two bits for a 4-elements alphabet). Moreover, to each parity function subspace corresponds a unique input to be fed into the circuit shown before. The Hadamard gate transforms each bit of the input binary string into the state $|+\rangle$ or $|-\rangle$ depending on the value of the bit. For the generic input $|0 \ldots 1\rangle$, we have

$$
U_f \left( |x\rangle \otimes \left( \frac{|0\rangle + |1\rangle}{\sqrt{n}} \right) \ldots \left( \frac{|0\rangle - |1\rangle}{\sqrt{n}} \right) \right) = (-1)^{p(f(x))}|x\rangle \otimes \left( \frac{|1\rangle + |0\rangle}{\sqrt{n}} \right)^{\otimes n} \left( \frac{|1\rangle + |0\rangle}{\sqrt{n}} \right).
$$

Now we are going to analyze longer words. For example, if $n = 2$, for $p^{adfg}$, the set of parity balanced words is

$$
\begin{aligned}
\mathcal{C}_2^{adfg}(a,b,c,d,e,f,g) = & [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [eeee] \\
& \cup [ffff] \cup [gggg] \cup [hhhh] \cup [aaad] \cup [aadd] \\
& \cup [addd] \cup [aaaf] \cup [aaff] \cup [afff] \cup [aaag] \\
& \cup [aggg] \cup [dddf] \cup [ddff] \cup [dfff] \cup [ffgg] \\
& \cup [ffgg] \cup [fggg] \cup [bbbc] \cup [bbcc] \cup [bccc] \\
& \cup [bbbe] \cup [bbee] \cup [beee] \cup [ccce] \cup [ccee] \\
& \cup [ceee] \cup [bbbh] \cup [bbhh] \cup [bhhh] \cup [ccch] \\
& \cup [cchh] \cup [chhh] \cup [eeeh] \cup [eehh] \cup [ehhh];
\end{aligned}
$$

while the set of parity balanced words is

$$\begin{aligned}\mathcal{B}_2^{adfg}(a,b,c,d,e,f,g) = {} & [aabb] \cup [aacc] \cup [aaee] \cup [aahh] \cup [ddbb] \\ & \cup [ddcc] \cup [ddee] \cup [ddhh] \cup [ffbb] \cup [ffcc] \\ & \cup [ffee] \cup [ffhh] \cup [ggbb] \cup [ggcc] \cup [ggee] \\ & \cup [gghh] \cup [adbc] \cup [afce] \cup [agbc] \cup [agbe] \\ & \cup [agce] \cup [adce] \cup [adbe] \cup [adhe] \cup [agch] \\ & \cup [afce] \cup [afch] \cup [adbe] \cup [adbh] \cup [afbc] \\ & \cup [afbh] \cup [agbh] \cup [ageh] \cup [afeh] \cup [afbe].\end{aligned}$$

For $p^{abcd}$, the set of parity balanced words is

$$\begin{aligned}\mathcal{C}_2^{abcd}(a,b,c,d,e,f,g) = {} & [aaaa] \cup [bbbb] \cup [cccc] \cup [dddd] \cup [eeee] \\ & \cup [ffff] \cup [gggg] \cup [hhhh] \cup [aaab] \cup [aabb] \\ & \cup [abbb] \cup [aaac] \cup [aacc] \cup [accc] \cup [aaad] \\ & \cup [aadd] \cup [addd] \cup [bbbc] \cup [bbcc] \cup [bccc] \\ & \cup [bbbd] \cup [bbdd] \cup [bddd] \cup [cccd] \cup [ccdd] \\ & \cup [cddd] \cup [eeef] \cup [eeff] \cup [efff] \cup [eeeg] \\ & \cup [eegg] \cup [eggg] \cup [eeeh] \cup [eehh] \cup [ehhh] \\ & \cup [fffg] \cup [ffgg] \cup [ffgh] \cup [ffhh] \cup [fhhh] \\ & \cup [gggh] \cup [gghh] \cup [ghhh];\end{aligned}$$

the set of parity balanced words is

$$\begin{aligned}\mathcal{B}_2^{abcd}(a,b,c,d,e,f,g) = {} & [aaee] \cup [aaff] \cup [aagg] \cup [aahh] \cup [bbee] \\ & \cup [bbff] \cup [bbgg] \cup [bbhh] \cup [ccee] \cup [ffcc] \\ & \cup [ccgg] \cup [cchh] \cup [ddee] \cup [ddff] \cup [ddgg] \\ & \cup [ddhh] \cup [abef] \cup [abeg] \cup [abeh] \cup [acef] \\ & \cup [aceg] \cup [aceh] \cup [adef] \cup [adeg] \cup [adeh] \\ & \cup [abfg] \cup [abfh] \cup [acfg] \cup [acfh] \cup [adfg] \\ & \cup [adfh] \cup [agbh] \cup [agch] \cup [adgh].\end{aligned}$$

The same reasoning carried on for a four-letters alphabet can be applied to form the set of words

$$\mathcal{F}_n^x(\mathcal{A}) = \mathcal{C}_n^x(\mathcal{A}) \cup \mathcal{B}_n^x(\mathcal{A})$$

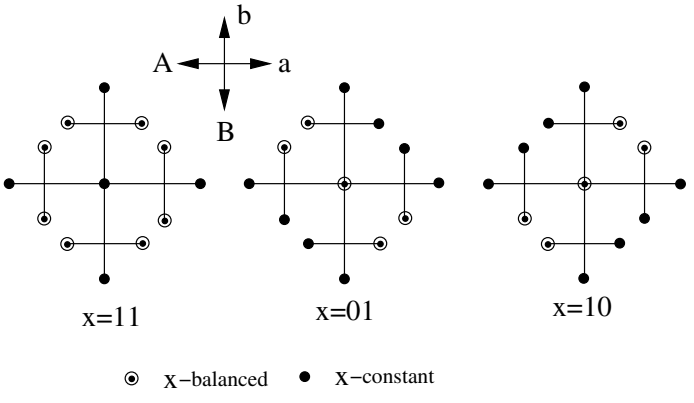
and the relative intersections. A potential generalization comes from error correcting codes. This could be based on introducing an encoding in which the letters of the alphabet are associated to the codewords of a subspace quantum error correcting code. A form of parity could be defined by considering the remaining subspaces.

## 2 Application to the Word Problem in Groups

Let $\{a, b, c = B, d = A\}$ be a paired alphabet, where $A$ represents $a^{-1}$ and $B$ represents $b^{-1}$. We first consider words of length 2. Parity constant words are "character constant", *i.e.* consist of only one letter, whether it be lower or upper case. Parity balanced words are "character balanced". The words corresponding to the parity constant case are $aa$,$aA$,$bb$,$bB$,$Bb$,$BB$,$Aa$,$AA$. Those corresponding to the parity balanced case are $ab$,$aB$,$ba$,$bA$,$Ba$,$BA$,$Ab$,$AB$. The words $w$ in the first list all satisfy $w \in \langle a \rangle \cup \langle b \rangle$ (in fact we have $w \in \langle a^2 \rangle \cup \langle b^2 \rangle$), whereas those $w$ in the second list all satisfy $w \notin \langle a \rangle \cup \langle b \rangle$. Thus, for words of length 2, we can determine with a single measurement whether or not $w \in \langle a \rangle \cup \langle b \rangle$.

If $x = 01$ then the $x$-constant words are $aa$,$ab$,$ba$,$bb$,$BA$,$BB$,$AA$,$AB$ and the $x$-balanced words are $aB$,$aA$,$bB$,$bA$,$Aa$,$Ab$,$Ba$,$Bb$. So, 01-constant and 01-balanced may be thought of as "case constant" and "case balanced" where the case can be upper or lower. For example, a commutator word (reduced or not) is always case balanced. "Case constant" and "case balanced" are properties of $\overline{w}$, rather than $w$. This is not the case for "parity constant" and "parity balanced".

If $x = 10$ then the $x$-constant words are $aa$,$aB$,$bb$,$bA$,$Ba$,$BB$,$Ab$,$AA$ and the $x$-balanced words are $ab$,$aA$,$ba$,$bB$,$Bb$,$BA$,$AB$,$Aa$. This does not seem to have any nice interpretation, while the 10-balanced corresponds to the cyclic subgroup generated by $ab$ and the 11-constant set solve a problem of union of subgroup membership for $\langle a \rangle \cup \langle b \rangle$. The elements represented by these words are depicted on the following Cayley graph portions:




Note that $w$ is 11-constant but not 01-constant and $w$ is 11-constant and not 10-constant then $w = F_2^{11}$. This gives a method of solving the word problem for words of length 2 using two quantum queries.

**Definition 1.** *We say that words in $\mathcal{F}_1^{11} \cap \mathcal{F}_1^{01}$ are $\mathcal{QWP}$-*feasible *(quantum word problem feasible). We write the set of such words as $\mathcal{QWP}_1$.*

For $n = 1$, if we are promised that $w$ is $\mathcal{QWP}$-feasible then the quantum query complexity of the property "is $w$ trivial?" seems to be 2. But this is not a

reduction in complexity from the classical case. However, there is hope that an analogous method might be an improvement in quantum query complexity for longer words. We have the following

**Proposition 1.** *For all $n$, if we are promised that the word $w$ of length $2^n$ is 11-feasible then the quantum query complexity of the property "Does $w$ represent an element of $\langle a\rangle \cup \langle b\rangle$?" is 1.*

This is directly analogous to the Deutsch-Jozsa algorithm, and the proof is the same. It is unclear how to extend the definition of $\mathcal{QWP}$ beyond two letters. Here are examples of two groups where we require different promises:

**Proposition 2.** *Consider the free abelian group $G = \langle a, b \mid ab = ba\rangle$. Let $w$ be a four-letter word in $\mathcal{A}$ which is in $\mathcal{F}_1^{11} \cap \mathcal{F}_2^{01} \cap \mathcal{F}_2^{10}$. Then the quantum query complexity of the question "Does $w$ represent the trivial element of $G$?" is at most 3.*

*Proof.* The first query asks whether $w \in \mathcal{C}_2^{01}$ or $w \in \mathcal{B}_2^{01}$. If the former is true then $w$ is not trivial so stop. If $w \in \mathcal{B}_2^{01}$ then proceed to the second query, which is whether $w \in \mathcal{C}_2^{11}$ or $w \in \mathcal{B}_2^{11}$. If the former is true then $w$ is trivial so stop. Otherwise we know that $w \in \mathcal{B}_2^{11} \cap B_2^{01}$ and we may proceed to the third query. There are two possibilities. The first possibility is that we have a word with two $A$s and two $b$s or a word with two $a$s and two $B$s. That is, $w$ is a cyclic rotation of $(AAbb)^{\pm 1}$. The second possibility is that we have one each of $A$, $b$, $a$ and $B$. In the first case, $w$ is nontrivial and in $\mathcal{C}_2^{10}$; in the second case, $w$ is trivial and in $\mathcal{B}_2^{01}$. So our third query is whether $w \in \mathcal{C}_2^{10}$ or $w \in \mathcal{B}_2^{10}$; this solves the word problem provided $w$ is as promised. ∎

It is indeed possible to generalize this theorem to the 8-letters alphabet introduced earlier, by considering the four-paired alphabet $\{a, b, c, d, e = D, f = C, g = B, h = A\}$, where the upper-case $A, B, C, D$ letters represent respectively $a^{-1}, b^{-1}, c^{-1}, d^{-1}$. In particular, we have the following statement:

**Proposition 3.** *Consider the free group $G = \langle a, b, c, d \mid abcd = dcba\rangle$. Let $w$ be a 8-letter word in $\mathcal{A}$ which is in $\mathcal{F}_3^{adfg} \cap \mathcal{F}_3^{abcd} \cap \mathcal{F}_3^{adeh}$. Then the quantum query complexity of the question "Does $w$ represent the trivial element of $G$?" is at most 3.*

*Proof.* The first query asks whether $w \in \mathcal{C}_3^{abcd}$ or $w \in \mathcal{B}_3^{abcd}$. If the former is true then $w$ is not trivial so stop. If $w \in \mathcal{B}_3^{abcd}$ then proceed to the second query, which is whether $w \in \mathcal{C}_3^{adfg}$ or $w \in \mathcal{B}_3^{adfg}$. If the former is true then $w$ is trivial so stop. Otherwise we know that $w \in \mathcal{B}_3^{adfg} \cap B_3^{abcd}$ and we may proceed to the third query. There are two possibilities. The first possibility is that we have a word with two $A$s two $D$s and two $as$ and two $ds$ or a word with two $C$s two $B$s, two $c$s and two $b$s. That is, $w$ is a cyclic rotation of $(AADDaadd)^{\pm 1}$ or $(BBCCbbcc)^{\pm 1}$. The second possibility is that we have one each of $A$, $b$, $a$ $B$, $C$, $d$,$c$, and $D$ . In the first case, $w$ is nontrivial and in $\mathcal{C}_3^{adeh}$; in the second, $w$ is trivial and in $\mathcal{B}_3^{abcd}$. Our third query is whether $w \in \mathcal{C}_3^{adeh}$ or $w \in \mathcal{B}_3^{abcd}$; this solves the word problem provided $w$ is as promised. ∎

Looking at the first two queries it seems possible to generalize this result for every paired alphabet of dimension $2^{n-1}$ and words of length $2^n$, by defining parities based on the even number of ones, like $p^{adfg}$. This is always possible because of the equipartition of the binary strings with respect to the number of ones and on the subspaces formed by the first $2^{n-1}$ vectors labeled from zero to $2^n$. The last parities required is the one used to identify words that are cyclic permutations of elements of the alphabet, for example, $p^{adeh}$. Moreover it's also easy to see that the setting is independent on the dimension of the alphabet as long it's possible to define a parity function that splits in two part the number of symbols and it's an efficient way of recognizing the membership of an element to a given subset. It does not seem easy to distinguish between trivial and nontrivial four-letter words in the free group of rank 2 using less than 4 quantum queries. However, the first indication that classical query complexity can be improved upon in a nonabelian finitely presented group is the following:

**Proposition 4.** *Consider the group presented by* $G = \langle a, b \mid a^2 = b^2 \rangle$. *Suppose we are given a word* $w$ *of length* 4 *in* $\mathcal{A}$ *such that* $w \in \mathcal{F}_2^{11} \cap \mathcal{F}_2^{01}$. *Then the quantum query complexity of the question "Does* $w$ *represent the trivial element of* $G$*?" is at most* 3.

*Proof.* The first two queries are as in the proof of the last proposition. So we can assume that if we do not already know whether or not $w$ is trivial, $w \in \mathcal{B}_2^{11} \cap B_2^{01}$ and we may proceed to the third query. For this, we construct a "syllable function"

$$f : \{0,1\} \to \{aa, ab, aB, aA, ba, bb, bB, bA, Ba, Bb, BB, Ba, Aa, Ab, AB, AA\}.$$

It maps $AA$,$BB$,$Aa$,$aA$,$Ab$,$AB$,$ab$,$aB$ to 0 and $Bb$,$bB$,$BA$,$bA$,$Ba$,$ba$,$aa$,$bb$ to 1. Note that, since $w \in \mathcal{B}_2^{11} \cap B_2^{01}$, $w$ is either a cyclic rotation of $(AAbb)^{\pm 1}$ or $w$ is an anagram of $AaBb$. Words in the first case are all trivial, because $a^2 = b^2$ is a relation in $G$, and these words are all balanced under the syllable function. Words in the second case are nontrivial if and only if they are nontrivial commutators. Commutators are constant under the syllable function. Words in the second case which are trivial (*i.e.*, not commutators) are all balanced under the syllable function. Thus a third query of "is $w$ syllable-balanced or syllable-constant" will complete the solution of the word problem. The following table lists all 0-syllabs and 1-syllabs:

| 0-syllabs | 1-syllabs |
|---|---|
| $AA$ | $aa$ |
| $BB$ | $bb$ |
| $Aa$ | $Bb$ |
| $aA$ | $bB$ |
| $Ab$ | $bA$ |
| $AB$ | $BA$ |
| $ab$ | $ba$ |
| $aB$ | $Ba$ |

■

While the group $G$ in the last proposition is nonabelian, it can be shown to have a free abelian subgroup of rank 2 and index 4; it is an extension of $\mathbb{Z} \oplus \mathbb{Z}$ by the Klein 4-group.

**Proposition 5.** *Consider the group presented by* $G = \langle a, b, c, d \mid a^2b^2 = b^2a^2 \rangle$. *Suppose we are given a word* $w$ *of length* 8 *in* $\mathcal{A}$ *such that* $w \in \mathcal{F}_3^{adfg} \cap \mathcal{F}_3^{abcd}$. *Then the quantum query complexity of the question "Does* $w$ *represent the trivial element of* $G$*?" is at most* 3.

*Proof.* The first two queries are as in the proof of the last proposition. So we can assume that if we do not already know whether or not $w$ is trivial, $w \in \mathcal{B}_3^{adfg} \cap B_3^{abcd}$ and we may proceed to the third query. For this, we construct an extended syllable function whose output has a cardinality of $2^{n-1}$. Some of the elements are listed below:

$$\begin{aligned} f : \{0,1\} \longrightarrow \{&aaaa, bbbb, BBBB, AAAA, aaab, aabb, abbb,\\ &aaaB, aaBB, aBBB, aaaA, aaAA, aAAA, aaAA,\\ &aAAA, bbbB, bbBB, bBBB, bbbA, bbAA, bAAA,\\ &bbba, bbaa, baaa, BBBa, BBaa, Baaa, BBBb,\\ &BBbb, Bbbb, BBBA, BBAA, BAAA, AAAa, AAaa,\\ &Aaaa, AAAb, AAbb, Abbb, AAAB, AABB, ABBB, ...\} \end{aligned}$$

Examples of this map are

$$AAAA, BBBB, Abbb, AAaa, aBBB, aaBB, aaaB, abAB, ABab, ABab, AABB \ldots \text{to } 0$$

and

$$aaaa, bbbb, Bbbb, BBbb, bBBB, bAAA, BBAA, BAAA, baaa, bAAA, aabb, bbaa \ldots \text{to } 1.$$

Note that since $w \in \mathcal{B}_3^{adfg} \cap B_3^{abcd}$, $w$ is either a cyclic rotation of $(AABBaabb)^{\pm 1}$ or $w$ is an anagram of $AAaaBBbb$. Words in the first case are all trivial, because $a^2b^2 = b^2a^2$ is a relation in $G$, and these words are all balanced under the syllable function. Words in the second case are nontrivial if and only if are nontrivial sequence of letters, that is not commutator-like sequence with respect to the presentation. Words in the second case which are trivial (*i.e.*, not trivial sequence) are all balanced under the extended syllable function. Thus a third query of "is $w$ syllable-balanced or syllable-constant" will complete the solution of the word problem. ∎

The same considerations can be made by looking at different sets of generators or relations like $G = \langle a, b, c, d \mid c^2d^2 = d^2c^2 \rangle$ and $G = \langle a, b, c, d \mid b^2c^2 = c^2b^2 \rangle$. It is important to notice that all the alternate sets of relations five groups isomorphic to the group considered in Proposition 5. To see this, it is sufficient to relabel the generators. The relation in $G$ is in fact very general and it is possible to obtain the same result with a whole family of similar relations. This can be done by varying the parity function used for the queries, choosing the presentation accordingly. Moreover such a group is a free group of rank 2 with $G = \langle a, b, c, d \mid a^2b^2 = b^2a^2 \rangle$.

It is simple to see that since the other two generators, $c$ and $d$, are not involved in the proof, it is possible to take the free product of $G$ with any free group and get to the same conclusion. In particular it is possible to extend the free product with *any* group and see the invariance of those three quantum queries under free products.

Notice that the choice of some particular kind of relations and an higher number of generators in the setting of the problem may increase the number of queries required. The reason of this is the exponential growth in the number of permutations, in particular, in those cases where splitting the words in parity balanced and parity constant does not help. Generalize to other different sets of generators and possibly for free products, and limiting to commutator words might give interesting promises. It is also useful to remark the importance of having the correct and certain answer to each of the queries used to prove the above propositions. As we have mentioned before, although it is possible to obtain *on average* with three classical queries the same results given by the Deutsch-Jozsa algorithm (see [4]), we assume for our proofs to have three certain answers to the queries. This means that on average the solution to the problems proposed requires at least nine classical queries (*i.e.*, three classical queries for each quantum one). This makes explicit the gain in number of queries of the quantum setting with respect to the classical deterministic one.

## 3 Conclusions

We have extended the original Deutsch-Jozsa algorithm to functions of arbitrary length binary output with the introduction of a more general concept of parity. This setting allows to consider a mapping between a binary string and the elements of an alphabet. The mapping helps to solve some instances of the word problem, using small alphabets and free groups, in a reduced number of queries with respect to the deterministic classical case. Extensions to more general groups and presentations may give interesting promises.

*Acknowledgments.* The authors would like to thank Andrew Childs for useful remarks. Part of this work as been carried out while Andrea Casaccino was attending "The Seventh Canadian Summer School on Quantum Information", hosted by Perimeter Institute and the Institute for Quantum Computing, at the University of Waterloo.

## References

1. Lyndon, R.C., Schupp, P.E.: Combinatorial group theory. Classics in Mathematics. Springer, Heidelberg (2001) (reprint of the 1977 edition)
2. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, Cambridge (2000)
3. Deutsch, D., Jozsa, R.: Rapid Solution of Problems by Quantum Computation. Proc. R. Soc. of London A 439, 553–558 (1992)
4. Farhi, E., Goldstone, J., Gutmann, S., Sipser, M.: Limit on the speed of quantum computation on determining parity. Phys. Rev. Lett. 81, 5552–5554 (1998)