A Generalized Quantum Branching Program

Debajyoti Bera
Department of Computer Science, IIIT-D, New Delhi, India

Tharrmashastha SAPV
Department of Computer Science, IIIT-D, New Delhi, India

Abstract

Classical branching programs are studied to understand the space complexity of computational problems. Prior to this work, Nakanishi and Ablayev had separately defined two different quantum versions of branching programs that we refer to as NQBP and AQBP. However, none of them, to our satisfaction, captures the intuitive idea of being able to query different variables in superposition in one step of a branching program traversal. Here, we propose a quantum branching program model, referred to as GQBP, with that ability. To motivate our definition, we explicitly give examples of GQBP for n-bit Deutsch-Jozsa, n-bit Parity, and 3-bit Majority with optimal lengths. We then show several equivalences, namely, between GQBP and AQBP, GQBP and NQBP, and GQBP and query complexities (using either oracle gates or a QRAM to query input bits). In a way, this unifies the different results that we have for the two earlier branching programs and also connects them to query complexity. We hope that GQBP can be used to prove space and space-time lower bounds for quantum solutions to combinatorial problems.

2012 ACM Subject Classification Theory of computation → Quantum computation theory; Theory of computation → Quantum query complexity; Theory of computation → Quantum complexity theory

Keywords and phrases Quantum computing, quantum branching programs, quantum algorithms, query complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.31

1 Introduction

The currently popular NISQ model of quantum computing highlights the importance of quantum space, i.e., qubit usage of quantum algorithms. Motivated by the need to study the quantum space complexity of combinatorial problems, we naturally looked towards quantum branching programs. A branching program is like a "random access finite-state machine where only one input character can be looked up in any state" [4]. In the classical domain, branching programs captures the non-uniform space complexity of a problem in a very natural way [7]. A branching program of size s, i.e., with s states, can be simulated by any (non-uniform) model with $\log(s)$ space to store the state space. Similarly, if a computation involves c distinct internal states or configurations (e.g., a $\log(c)$-space-bounded Turing machine), it is possible to simulate the computation by a c-size branching program. So, e.g., polynomial-size branching programs are equivalent to the non-uniform class $LOGSPACE/poly$.

Our interest in quantum branching programs is also related to a "more practical" motivation related to a current trend in machine learning – that of explainability. A decision tree is arguably the most intuitive machine learning model that can be explained in a straightforward manner. Masek, in his Master's thesis, studied branching programs in the name of decision graphs, which are essentially compact forms of decision trees. We believe that quantum branching programs may be a worthy machine learning model that can also be explained.

We are aware of two prior attempts to define a quantum version of branching programs. They, as well as ours, follow the general idea of changing the current state (which could be a superposition of basis states) by querying some input bit; the difference primarily lies

in how evolution happens. In the first version defined by Nakanishi et al. [6] (we refer to this as NQBP), the entire state is measured after each step, and evolution is continued if neither an accept nor a reject state is observed. In the version defined by Ablayev et al. [1], in each step of the evolution, a fixed bit is queried; we refer to these programs as AQBP). A closely associated model of quantum decision diagrams was also studied by Willie et al. in [9]. However, their study focuses on the design and validation of quantum circuits.

Quite a few results are known for NQBPs and AQBPs. The first advantage of quantum branching programs, more specifically NQBP, was demonstrated by Nakanishi et al. in [6]. They defined a function, namely *Half Hamming weight function*, that checks if the Hamming weight of the input is exactly $n/2$. They showed that there exists an ordered bounded width NQBP (also known as an NQOBDD) that can compute the half Hamming weight function while no ordered bounded width PBP can compute that function.

This was followed by the work of Ablayev et al [1] in which they demonstrated that an $(O(\log p_n), n)$ AQBP can compute the Mod_{p_n} function with one-sided error. However, any stable probabilistic OBDD needs width at least p_n. Here, Mod_{p_n} function is the function that decides if the Hamming weight of the input is divisible by a prime p_n. They also gave the seminal result that the class of languages in $NC1$, the languages that are decidable by uniform Boolean circuits of $O(\log n)$ depth and polynomial number of gates with at most two inputs, lies within the class of languages decidable by width-2 AQBPs [2].

Later, Sauerhoff et al. in [8] proved two results that exhibit the power and limitation of NQBP. First, they showed that the *permutation matrix test function*, the function that tests if the given matrix is a permutation matrix, can be computed using a small sized NQOBDD but needs a large deterministic OBDD. Alternatively, they also proved that any NQOBDD that computes the *disjointness function* requires $O(2^{\Omega(n)})$ size, but there exists a small sized deterministic OBDD that computes the function. More recently, in [5], Maslov et al. demonstrated that any symmetric Boolean function can be computed by $(2, O(n^2))$-AQBP. This is a striking result because there are symmetric functions that need at least 3 bits of computational space to be computed in polynomial time with arbitrary error and symmetric functions with non-trivial Fourier spectrum are not computable in polynomial time. Observe that a trivial lower bound on AQBP for computing any non-constant symmetric function is $\Omega(n)$ because the function is oblivious to the exact input and depends only on the Hamming weight of the input.

Despite the above encouraging results, we observed that AQBP suffers from the restriction of querying the same input bit within the different superposition paths in any single level. Further, the idea of measuring after every evolution step in an NQBP was not a very agreeable one. Therefore, we defined a new version which we refer to as GQBP (Figure 1 shows an example) that combines the idea of completely independent superposition paths as in NQBP with the idea of performing one measurement right at the end (as in AQBP), thus allowing interference to show its might. We use the complexity measures width (to denote the largest number of states or nodes in any level) and length (to denote the length of the longest path in the state diagram. Even though we do not consider size as a measure in this work, our results can be interpreted in terms of size by using the trivial upper bound of $size \leq width \times length$.

This paper introduces the GQBP model of branching program and relates it to a few other models like AQBP, NQBP, and query complexity. We are able to show the following.

- We show explicit general branching programs for *Parity* of n-bits, the problem of *Deutsch-Jozsa*, and *Majority* of 3 bits. These branching programs are optimal with respect to their lengths, and the optimality follows from a relation between the length of the shortest GQBP to the polynomial degree of a problem.

- We prove that an AQBP is a special form of GQBP (in which the same variable is queried in each level), and thus, the former cannot compute functions that depend non-trivially on every input bit using less than n length. We present a GQBP for Parity of length $n/2$, and this shows that GQBP does not suffer from the same limitation and is strictly more powerful, especially in the sub-linear length regime.
- Due to the above point, all the results concerning AQBP now also hold for GQBP, e.g., any problem in NC^1 can now be solved using a width-2 GQBP, and any symmetric Boolean function can be computed using a width-2, $O(n^2)$-length GQBP.
- Despite their structural difference, it is not difficult to show that a GQBP can be viewed as an NQBP. We further show that an NQBP can be simulated by a GQBP without any overhead of size and length.
- Next, we turn to quantum circuits. First, we show how to design an AQBP (and hence a GQBP) with width 2^q and length t to simulate a quantum query circuit in the QRAM model making q queries and consuming t qubits. The QRAM query model provides an alternative to implementing the oracle (in the oracle query model) by having the input in query-only registers (*aka.* QRAM) that can be queried using CNOT (or similar) gates. We want to point out that even though it may be possible to obtain a GQBP for $Parity_n$ using this approach, our handcrafted GQBP for the same uses lesser width.
- Next, we show the reverse direction, i.e., how to design a quantum circuit (using either an oracle or a QRAM to access input bits) for a given GQBP using reasonably tight depth and space (i.e., qubits). This and the above point allows us to interpret the upper and the lower bounds in quantum query complexities in terms of a GQBP and possibly derive new insights into quantum complexities of problems.

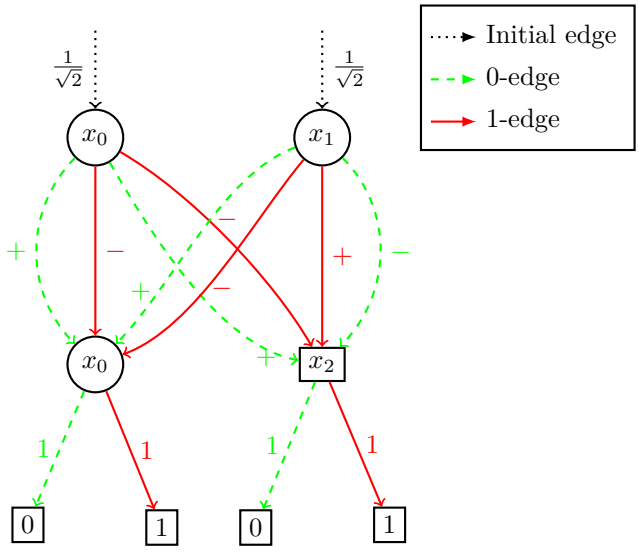

Figure 1 A length-2 and width-4 GQBP to (exactly) compute the majority of 3 bits x_0, x_1, x_2. Here, $+$ and $-$ denotes the amplitudes $\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$, respectively.

2 Background: Quantum Branching Programs

In this section, we familiarize the reader with the existing branching program frameworks. We then introduce a new variant of quantum branching programs.

The most basic branching programs are the deterministic branching programs which are defined as follows.

▶ **Definition 1** (Deterministic Branching Program). *A deterministic branching program is a tuple* $(Q, E, v_0, L, \delta, F)$ *along with an evolution operator* E *where*

- Q *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *form a directed acyclic multi-graph.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $v_0 \in Q$ *is the start node.*
- $\delta : Q \otimes \{0, 1\} \to Q$ *is transition function.*
- $F \subseteq Q$ *is the set of sinks such that* $F = F_a \cup F_r$ *where* F_a *and* F_r *are the accept and reject sinks respectively.*

Given an input x, starting from the node v_0, one iteration of the evolution of the deterministic branching program consists of two steps:

1. Let the current node be v. Query the value of $x_{L(v)}$ and call it j. Move to the node $\delta(v, j) = v'$.
2. If $v' \in F_a$ then accept the input, else if $v' \in F_r$ then reject the input. Else, continue.

The complexity of a branching program is primarily categorized by two measures: *size* and *length* of a branching program.

▶ **Definition 2** (Size and length of a branching program).

- *The* ***size*** *of a branching program is the total number of nodes in the branching program.*
- *The* ***length*** *of a branching program is the length of the longest path from the start node to any sink node.*

It is well known that any classical branching program can be transformed into an equivalent levelled branching program, i.e., a branching program where all the transitions are such that they go from a node at a level i only to a node at level $i + 1$ for each i. From now on, we assume that the branching program is levelled unless otherwise stated.

For levelled branching program, another well-used measure of complexity is the *width* of a branching program.

▶ **Definition 3** (width of a leveled branching program). *The* ***width*** *of a levelled branching program is the maximum number of nodes in any level of the branching program.*

A straightforward generalization of the deterministic branching programs is the probabilistic branching programs. But before we define the probabilistic branching programs, we establish well-behaved transition functions. A transition function $\delta : Q \otimes \{0, 1\} \times Q \to \mathbb{R}$ is said to be **classically well-behaved** if for any $v \in Q$ and $a \in \{0, 1\}$,

$$\sum_{v' \in Q} \delta(v, a, v') = 1.$$

Now, the probabilistic branching programs are defined as below.

▶ **Definition 4** (Probabilistic Branching Program). *A probabilistic branching program is a tuple* $(Q, E, v_0, L, \delta, F)$ *along with an evolution operator* E *where*

- Q *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *form a directed acyclic multi-graph.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $v_0 \in Q$ *is the start node.*

- $\delta : Q \otimes \{0,1\} \times Q \to \mathbb{R}$ *is a classically well-behaved transition function.*
- $F \subseteq Q$ *is the set of sinks such that* $F = F_a \cup F_r$ *where* F_a *and* F_r *are the accept and reject sinks respectively.*

Given an input x, starting from the node v_0, one iteration of the evolution of the deterministic branching program consists of two steps:

1. Let the current node be v. Query the value of $x_{L(v)}$ and call it j. Move to the node v' with probability $\delta(v, j, v')$.
2. If $v' \in F_a$ then accept the input, else if $v' \in F_r$ then reject the input. Else, continue.

An input string is *accepted* by a branching program if the evolution ends in an accept state. In this paper, we are concerned with *exact computation*, i.e., the probability of acceptance should be 1 for good inputs and 0 for bad inputs.

In general, branching programs are complex structures to analyse. For ease of analysis, various variants of the branching programs were introduced. Some of the variants are as follows.

- **Oblivious branching programs:** These are levelled branching programs in which the variable to be queried depends on the level of the current node rather than the node itself.
- **Ordered binary decision diagrams (OBDD):** These are branching programs with a fixed ordering of querying the indices along any path.
- **Read-k branching programs:** These are branching programs in which the number of times any variable is queried is at most k along any path from the source node to a sink node.

The notion of quantum branching programs was first introduced by Nakanishi et al. in [6] as an extension of the classical probabilistic branching programs.

▶ **Definition 5** (Nakanishi's Quantum Branching Program (NQBP)). *A quantum branching program is a tuple* $(Q, E, v_0, L, \delta, O)$ *where*

- $Q = Q_{acc} \cup Q_{rej} \cup Q_{non}$ *is a set of nodes.*
- E *is the set of edges on* Q *such that* (Q, E) *is a directed multi-graph.*
- v_0 *is the start node.*
- $L : Q \to \{1, 2, \cdots, n\}$ *assigns a variable to each node.*
- $\delta : Q \otimes \{0,1\} \otimes Q \to \mathbb{C}$ *is a quantumly well-behaved transition function.*
- O *is the projective measurement induced by the set* $\left\{\Pi_{acc}, \Pi_{rej}, \Pi_{non}\right\}$ *where* Π_i *is the projection operator onto the space* $V_i = span\{|v\rangle : v \in Q_i\}$ *for each* $Q_i \in \{Q_{acc}, Q_{rej}, Q_{non}\}$.

For any fixed input x, let O_x denote an oracle defined as $O_x(i) = x_i$. Given an oracle O_x to access the input, the i^{th} iteration of the evolution of NQBP comprises of the following steps:

1. Define an operator $U^O : Q \to Q$ as

$$U^{O_x} |v\rangle = \sum_{v' \in Q} \delta(v, O_x(L(v)), v') |v'\rangle .$$

2. Apply U^{O_x} on $|\psi_i\rangle = U^{O_x} |\psi_{i-1}\rangle$, $|\psi_0\rangle = |v_0\rangle$.
3. Observe $|\psi_i\rangle$ with respect to O.
4. If the post-measurement state is in the subspace spanned by $|E_{non}\rangle$, continue. Else accept or reject depending on the outcome.

A transition function $\delta : Q \otimes \{0,1\} \otimes Q \to \mathbb{C}$ is said to be **quantumly well-behaved** if for any $v_1, v_2 \in Q$ and all inputs $x \in \{0,1\}^n$,

$$\sum_{v' \in Q} \delta(v_1, x_{L(v_1)}, v')\delta^*(v_2, x_{L(v_2)}, v') = \begin{cases} 1 & \text{if } v_1 = v_2 \\ 0 & \text{if } v_1 \neq v_2 \end{cases}$$

Closely followed by this work, Ablayev et al., in [1], proposed a new model for quantum branching programs that was significantly different from NQBPs. Before we define Ablayev's model, we first define a quantum transformation as defined by Ablayev et al.

▶ **Definition 6** (Quantum Transformation). *Given an input string $x \in \{0,1\}^n$, a d-dimensional quantum transformation $\langle j, U(0), U(1)\rangle$ on a state $|\psi\rangle \in \mathcal{H}^{\otimes d}$ is defined as*

$$|\psi'\rangle = U(x_j)\,|\psi\rangle\,.$$

where $U(0)$ and $U(1)$ are a $d \times d$ unitary matrices. Intuitively, the quantum transformation applies $U(0)$ to $|\psi\rangle$ if the j^{th} bit of x is 0 and applies $U(1)$ if the j^{th} bit of x is 1.

Now, we define Ablayev's quantum branching program model.

▶ **Definition 7** (Ablayev's Quantum Branching Programs (AQBP)). *A quantum branching program P of width d and length l is a tuple $\langle T, |\psi_0\rangle, F\rangle$ where*

- *$T = \left(\langle j_i, U_i(0), U_i(1)\rangle\right)_{i=1}^{l}$ is a sequence of d-dimensional quantum transformations.*
- *$|\psi_0\rangle$ is the initial configuration of P.*
- *$F \subseteq \{|0\rangle, |1\rangle, \cdots, |d-1\rangle\}$ is the set of accepting states.*

Given an input $x = x_1 x_2 \cdots x_n$, the AQBP P transforms as

$$|\psi_x\rangle = U_l(x_{j_l})U_{l-1}(x_{j_{l-1}}) \cdots U_2(x_{j_2})U_1(x_{j_1})\,|\Psi_0\rangle\,.$$

To obtain the output, the state $|\psi_x\rangle$ is measured with projection matrix M such that $M_{ii} = 1$ iff $i \in F$ and 0 else. The probability that P accepts the input x is

$$P_{accept}(x) = ||M\,|\psi_x\rangle\,||^2$$

Notice that this model is the quantum equivalent of the oblivious classical branching programs; the query made at any given level is independent of the nodes in that level of the AQBP.

We define a new model of branching programs that differs from the above two models.

▶ **Definition 8** (Generalized quantum branching program (GQBP)). *A quantum branching program is a tuple $(Q, E, |v_0\rangle, L, \delta, F)$ where*

- *$Q = \bigcup_{i=0}^{l} Q_i$ is a set of nodes where Q_i is the set of nodes at level i.*
- *E is a set of edges between the nodes in Q such that (Q, E) is a levelled directed acyclic multi-graph.*
- *$|v_0\rangle$ is the initial state which is a superposition of the nodes in Q_0.*
- *$\delta : Q \times \{0,1\} \times Q \to \mathbb{C}$ is a **quantumly well-behaved** transition function.*
- *$L : Q \to \{1, 2 \cdots, n\}$ is a function that assigns a variable to each node in $Q \setminus F$.*
- *$F \subseteq Q$ is the set of terminal nodes.*

The evolution of this GQBP is defined as follows:

1. Given access to an input x using oracle O_x, define $U_i^{O_x} : \mathcal{H}(d) \to \mathcal{H}(d)$ for $i \in \{1, 2, \cdots, l\}$ that has query access to O_x as

$$U_i^{O_x} |v\rangle = \sum_{v' \in Q_i} \delta(v, O_x(L(v)), v') |v'\rangle$$

for any $|v\rangle \in Q_{i-1}$ where $\mathcal{H}(d)$ is the Hilbert space on d dimensions with d as the width of the QBP.

2. Start at $|v_0\rangle$.
3. For $i = 1$ to l, perform

$$|\psi_i\rangle = U_i^{O_x} |\psi_{i-1}\rangle$$

where $|\psi_0\rangle = |v_0\rangle$.

4. Measure $|\psi_l\rangle$ in the standard basis as m. If $|m\rangle \in F$ then accept the input. Else, reject the input.

We say that a QBP $\mathcal{G}$ **exactly computes** a function f if for all inputs x, $\mathcal{G}$ accepts x if $f(x) = 1$ and rejects x if $f(x) = 0$, both with probability 1. Similarly, a QBP $\mathcal{G}$ is said to **approximate** a f if for all inputs x, $\mathcal{G}$ accepts x if $f(x) = 1$ and rejects x if $f(x) = 0$, both with probability at least $2/3$.

It is quite straightforward that this model of quantum branching program subsumes the AQBP model. To see this in detail, we direct the readers to the proof of Theorem 14. So, the set of languages computable by AQBPs is a subset of that computable by GQBPs.

3 Upper Bounds on GQBP

In this section, we show some GQBPs for some simple Boolean functions. We first start with the classic Deutsch-Josza problem, which is a promise problem whose task is to identify if an n-bit binary string is balanced (has Hamming weight $n/2$) or is constant (has Hamming weight 0 or n). We define the notation $\texttt{GQBP}_{w,l}$ to denote the class of all problems that are computable by some w-width l-length GQBP exactly.

▶ **Theorem 9.** $DJ_n \in \mathit{GQBP}_{n,1}$.

Proof. Consider the two-layered GQBP $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ where $Q = Q_0 \cap Q_1$, $Q_i = \{v_{i,j} : j \in \{1, \cdots, n\}\}$, $E = \{(v_{0,i}, v_{1,j}) : i, j \in \{1, \cdots, n\}\}$ and $F = Q_1 \setminus \{|v_{1,1}\rangle\}$. Define the function L as $L(v_{0,i}) = x_i$ for $i \in \{1, \cdots, n\}$. Also, define the transition function δ as

$$\delta(v_{0,i}, a, v_{1,j}) = \frac{1}{2^{n/2}} (-1)^a (-1)^{i \cdot j}$$

for all $i, j \in \{1, \cdots, n\}$. Now, set $|v_0\rangle = \frac{1}{2^{n/2}} \sum_{v_{0,i} \in Q_0} |v_{0,i}\rangle$. Fix an input x. Then the transition unitary $U_1^{O_x}$ will be defined as

$$U_0^{O_x} |v_{0,i}\rangle = \sum_{v_{1,j} \in Q_1} \delta(v_{0,i}, x_i, v_{1,j}) |v_{1,j}\rangle = \frac{1}{2^{n/2}} \sum_{v_{1,j} \in Q_1} (-1)^{x_i} (-1)^{i \cdot j} |v_{1,j}\rangle .$$

Then the final state of $\mathcal{G}$ before measurement can be calculated as

$$\begin{aligned} U_0^{O_x}|v_0\rangle &= \frac{1}{2^n}\sum_{v_{0,i}\in Q_0} U_0^{O_x}|v_{0,i}\rangle \\ &= \frac{1}{2^n}\sum_{v_{0,i}\in Q_0}\frac{1}{2^{n/2}}\sum_{v_{1,j}\in Q_1}(-1)^{x_i}(-1)^{i\cdot j}|v_{1,j}\rangle \\ &= \frac{1}{2^n}\sum_{v_{1,j}\in Q_1}\Big[\sum_{v_{0,i}\in Q_0}(-1)^{x_i\oplus i\cdot j}\Big]|v_{1,j}\rangle \end{aligned}$$

Since, $F = Q_1 \setminus \{|v_{1,1}\rangle\}$, the probability of accepting the input x is

$$Pr_{\mathcal{G}}[accept] = 1 - \left|\sum_{v_{0,i}\in Q_0}(-1)^{x_i}\right|^2 = \begin{cases} 1, \text{ if } x \text{ is balanced} \\ 0, \text{ if } x \text{ is constant} \end{cases}$$

◀

▶ **Theorem 10.** $Parity_n \in \mathtt{GQBP}_{2,n/2}$.

Proof. Observe that any classical branching program that solves the problem with probability 1 must query all the bits in some path and so must have length n. To reduce the length, our GQBP uses superposition to query each half separately, "compute" their parity in each branch, and then use a simple decision graph to output the xor of those parities. The GQBP is illustrated in Figure 2. [1]

Without loss of generality, we assume that n is even. To construct a GQBP for $Parity_n$, set $Q = \{v_{i,a} : i \in \{0,1,\cdots,n/2\}, a \in \{0,1\}\}$, $|v_0\rangle = \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle)$, $E = \{(v_{i,a}, v_{i+1,b}) : i \in \{0,1,\cdots,n/2\}, a \in \{0,1\}\}$ and $F = \{v_{n/2,1}\}$. Let $l = n/2$. As for the transition function δ, for the first $l-1$ layers, i.e., $i \in \{0,1,\cdots,l-2\}$ define

$$\delta(v_{i,j}, a, v_{p,q}) = \begin{cases} 1, \text{ if } p = i+1, j = q, a = 0 \\ -1, \text{ if } p = i+1, j = q, a = 1 \\ 0, \text{ else} \end{cases}$$

This gives that $U_i^{O_x}|v_{i-1,a}\rangle = (-1)^{L(v_{i-1,a})}|v_{i,a}\rangle$ for all $a \in \{0,1\}$. For the nodes $v_{l-1,0}$ and $v_{l-1,1}$, define

$$\delta(v_{l-1,0},0,v_{l,0}) = \delta(v_{l-1,0},0,v_{l,1}) = \delta(v_{l-1,1},0,v_{l,0}) = \delta(v_{l-1,1},1,v_{l,1}) = 1/\sqrt{2}$$

and

$$\delta(v_{l-1,0},1,v_{l,0}) = \delta(v_{l-1,0},1,v_{l,1}) = \delta(v_{l-1,1},1,v_{l,0}) = \delta(v_{l-1,1},0,v_{l,1}) = -1/\sqrt{2}.$$

So, we have

$$U_{l-1}^{O_x}|v_{l-1,0}\rangle = \frac{(-1)^{L(v_{l-1,0})}}{\sqrt{2}}\Big(|v_{l,0}\rangle + |v_{l,1}\rangle\Big)$$

and

$$U_{l-1}^{O_x}|v_{l-1,1}\rangle = \frac{(-1)^{L(v_{l-1,1})}}{\sqrt{2}}\Big(|v_{l,0}\rangle - |v_{l,1}\rangle\Big).$$

[1] It is tempting to try this idea to reduce the length even further, e.g., to $n/4$ by creating a superposition of 4 branches. That approach would get stuck at the last stage of combining all the partial parities. In fact, we show later that it is not possible to reduce the length beyond $n/2$ (13).

Next, define the function L as $L(v_{i,j}) = 2 * i + j$. Then, starting from the state $|v_0\rangle$, the state evolves through the first $l-1$ layers as follows,

$$
\begin{aligned}
|v_0\rangle &= \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle) \\
&\xrightarrow{U_1^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0}|v_{1,0}\rangle + (-1)^{x_1}|v_{1,1}\rangle) \\
&\xrightarrow{U_2^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0\oplus x_2}|v_{2,0}\rangle + (-1)^{x_1\oplus x_3}|v_{2,1}\rangle) \\
&\cdots \\
&\xrightarrow{U_i^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{2i-2}}|v_{i,0}\rangle + (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{2i-1}}|v_{i,1}\rangle) \\
&\cdots \\
&\xrightarrow{U_{l-1}^{O_x}} \frac{1}{\sqrt{2}}((-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{n-4}}|v_{l-1,0}\rangle + (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{n-3}}|v_{l-1,1}\rangle) = |\psi_{l-1}\rangle \text{ (say)}
\end{aligned}
$$

For the last transition, the state evolves as

$$
\begin{aligned}
|\psi_{l-1}\rangle &= \frac{1}{\sqrt{2}}((-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{n-4}}|v_{l-1,0}\rangle + (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{n-3}}|v_{l-1,1}\rangle) \\
&\xrightarrow{U_l^{O_x}} \frac{1}{\sqrt{2}}\Bigg[(-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{n-4}}\Big[\frac{(-1)^{x_{n-2}}}{\sqrt{2}}\Big(|v_{l,0}+|v_{l,1}\rangle\rangle\Big)\Big] \\
&\qquad + (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{n-3}}\Big[\frac{(-1)^{x_{n-1}}}{\sqrt{2}}\Big(|v_{l,0}\rangle - |v_{l,1}\rangle\Big)\Big]\Bigg] \\
&= \frac{1}{2}\Bigg[\Big((-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{n-2}} + (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{n-1}}\Big)|v_{l,0}\rangle \\
&\qquad + \Big((-1)^{x_0\oplus x_2\oplus\cdots\oplus x_{n-2}} - (-1)^{x_1\oplus x_3\oplus\cdots\oplus x_{n-1}}\Big)|v_{l,1}\rangle\Bigg]
\end{aligned}
$$

So, if $x_0 \oplus x_2 \oplus \cdots x_{n-2} = x_1 \oplus x_3 \oplus \cdots x_{n-1}$ or alternatively, $Parity_n(x) = 0$, then $|\psi_l\rangle = |v_{l,0}\rangle$ and the input will be rejected with probability 1 post measurement. Else if $x_0 \oplus x_2 \oplus \cdots x_{n-2} \neq x_1 \oplus x_3 \oplus \cdots x_{n-1}$, i.e, if $Parity_n(x) = 1$, then $|\psi_l\rangle = |v_{l,1}\rangle$. In this case, the input will be accepted with probability 1 post measurement. ◀

It is noteworthy that any query circuit requires at least $\log(n)$ qubits to query the oracle for an index. Then as a derivative of Corollary 21, one would be led to think that any GQBP would require width at least n. However, in contradiction to that, Theorem 10 gives us a width-2 GQBP that computes the $Parity_n$ function exactly. This is a classic example that shows that space lower bounds on query circuits does not hold for the GQBPs. On the contrary, later in section 5, we will show that asymptotic query lower bounds of query circuits also hold for the GQBPs and that of GQBPs hold for query circuits.

▶ **Lemma 11.** $Maj_3 \in \mathit{GQBP}_{3,2}$

Proof. The idea behind the construction is first to compare the first two bits. If they are the same, their value is the majority. Otherwise, the value of the third bit turns out as the majority. The state diagram is illustrated in Figure 1.

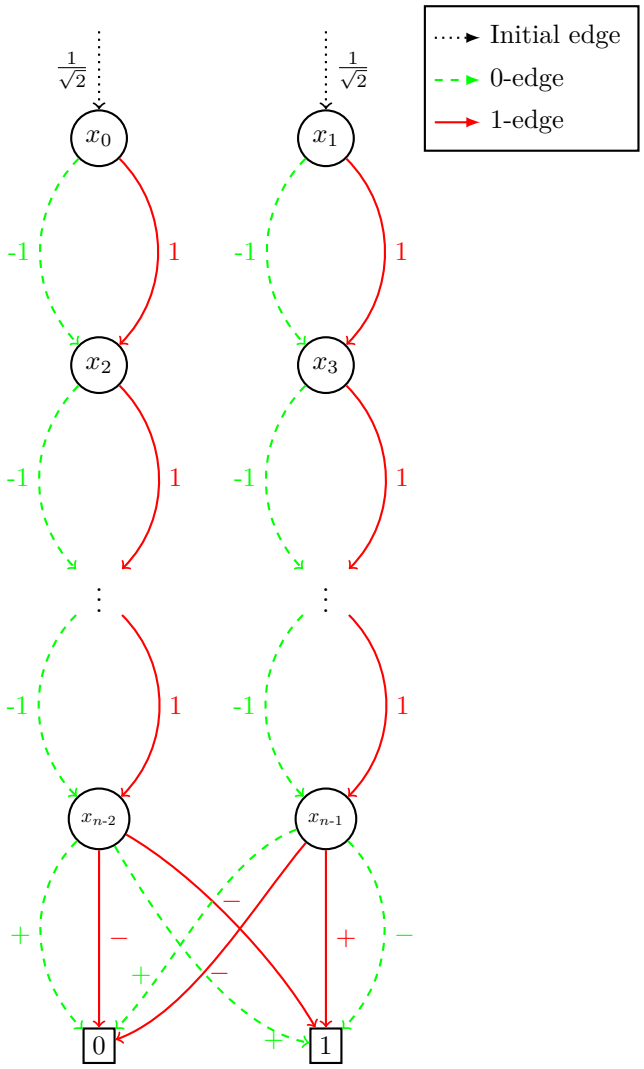

Figure 2 A GQBP for the n-bit Parity problem. $+$ and $-$ indicates transitions with amplitudes $\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$, respectively.

Consider a 3-levelled GQBP $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ for Maj_3 where the set of nodes $Q = \{v_{0,0}, v_{0,1}, v_{1,0}, v_{1,1}, v_{2,0}, v_{2,1}, v_{2,2}, v_{2,3}\}$, $|v_0\rangle = \frac{1}{\sqrt{2}}(v_{0,0} + v_{0,1})$, the function L is such that $L(v_{0,0}) = L(v_{1,0}) = x_1, L(v_{0,1}) = x_2$ and $L(v_{1,1} = x_3)$, and $F = \{|v_{2,1}\rangle, |v_{2,3}\rangle\}$. Define the transition function δ as

$$\delta(v_{0,0}, 0, v_{1,0}) = \delta(v_{0,0}, 0, v_{1,1}) = \delta(v_{0,1}, 0, v_{1,0}) = \delta(v_{0,1}, 1, v_{1,1}) = 1/\sqrt{2}$$
$$\delta(v_{0,0}, 1, v_{1,0}) = \delta(v_{0,0}, 1, v_{1,1}) = \delta(v_{0,1}, 1, v_{1,0}) = \delta(v_{0,1}, 0, v_{1,1}) = -1/\sqrt{2}$$

and

$$\delta(v_{1,0}, 0, v_{2,0}) = \delta(v_{1,0}, 1, v_{2,1}) = \delta(v_{1,1}, 0, v_{2,2}) = \delta(v_{1,1}, 0, v_{2,3}) = 1.$$

From this transition function, we obtain the transition unitary $U_1^{O_x}$ as

$$U_1^{O_x} |v_{0,0}\rangle = \frac{(-1)^{L(v_{0,0})}}{\sqrt{2}} \Big(|v_{1,0}\rangle + |v_{1,1}\rangle \Big) = \frac{(-1)^{x_0}}{\sqrt{2}} \Big(|v_{1,0}\rangle + |v_{1,1}\rangle \Big)$$

and

$$U_1^{O_x}|v_{0,1}\rangle = \frac{(-1)^{L(v_{0,1})}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big) = \frac{(-1)^{x_1}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big).$$

Now, one step of the evolution of the program can be given as

$$\begin{aligned}
|v_0\rangle &= \frac{1}{\sqrt{2}}(|v_{0,0}\rangle + |v_{0,1}\rangle) \\
&\xrightarrow{U_1^{O_x}} \frac{1}{\sqrt{2}}\left(\frac{(-1)^{x_0}}{\sqrt{2}}\Big(|v_{1,0} + |v_{1,1}\rangle\rangle\Big) + \frac{(-1)^{x_1)}}{\sqrt{2}}\Big(|v_{1,0}\rangle - |v_{1,1}\rangle\Big)\right) \\
&= \frac{1}{2}\left(\Big((-1)^{x_0} + (-1)^{x_1}\Big)|v_{1,0}\rangle + \Big((-1)^{x_0} - (-1)^{x_1}\Big)|v_{1,1}\rangle\right) \\
&= \begin{cases} |v_{1,0}\rangle\,, & \text{if } x_0 = x_1 \\ |v_{1,1}\rangle\,, & \text{if } x_0 \neq x_1 \end{cases} \\
&\xrightarrow{U_2^{O_x}} \begin{cases} |v_{2,0}\rangle\,, & \text{if } x_0 = x_1 \text{ and } x_0 = 0 \\ |v_{2,1}\rangle\,, & \text{if } x_0 = x_1 \text{ and } x_0 = 1 \\ |v_{2,2}\rangle\,, & \text{if } x_0 \neq x_1 \text{ and } x_2 = 0 \\ |v_{2,3}\rangle\,, & \text{if } x_0 \neq x_1 \text{ and } x_2 = 1 \end{cases}
\end{aligned}$$

We have $F = \{|v_{2,1}\rangle, |v_{2,3}\rangle\}$. This implies that if the given input is such that $x_0 = x_1$ and $x_0 = 1$ or $x_0 \neq x_1$ and $x_2 = 1$, then the GQBP accepts the input. Now, see that if $x_0 = x_1$, clearly, the value of x_0 is the majority. However, if $x_0 \neq x_1$, then the value of x_2 is the majority. Putting them together, we have that the GQBP $\mathcal{G}$ computes the Maj_3 function exactly. ◀

On careful observation, one can notice that the amplitudes of the basis states of the state of a GQBP after the evolution is a polynomial of the input x since, at each node, the outgoing edges are decided based on the value of the query made at that node. Moreover, each step of the evolution (applying one $U_i^{O_x}$) increases the degree of the polynomial corresponding to the amplitudes by at most 1. So, clearly, the amplitude of any node of the final state of an l-length GQBP can be given by a polynomial of degree of at most l. For any GQBP, the probability of accepting a given input is the sum of the probabilities of obtaining the nodes in F on measuring the final state. Consequently, we have the probability that an l-length GQBP accepts an input x is a polynomial of degree at most $2l$. This observation leads us to the following theorem.

▶ **Theorem 12.** *Let f be a Boolean function. Let $deg(f)$ and $\widetilde{deg}(f)$ be the exact degree and approximate degree of f. Then,*

1. *Any GQBP that exactly computes f should have length at least $deg(f)/2$.*
2. *Any GQBP that approximates f should have length at least $\widetilde{deg}(f)/2$.*

As a result, we obtain the following lower bound that shows that our constructions above are tight with respect to length.

▶ **Theorem 13.** *$Maj_3 \notin$ GQBP$_{w,1}$ and $Parity_n \notin$ GQBP$_{w,t}$ for any $w \in \mathbb{N}$ and any $t < \frac{n}{2}$.*

4 Relation Between GQBP and other variants

Similar to $\texttt{GQBP}_{w,l}$, we define the complexity class $\texttt{AQBP}_{w,l}$ as the class of problems that are computed exactly by some w-width l-length AQBP. Then, we have the following result.

▶ **Theorem 14.** $\mathit{AQBP}_{w,l} \subsetneq \mathit{GQBP}_{w,l}$

Proof. First, we show that any (w,l)-AQBP is also a (w,l)-GQBP. Let $\mathcal{A} = \langle T, |\psi_0\rangle, F_a\rangle$ be a (w,l)-AQBP where $T = \left(\langle j_i, U_i(0), U_i(1)\rangle\right)_{i=1}^{l}$. Construct a (w,l)-GQBP $\mathcal{G}$ $=(Q, E, v_0, L, \delta, F_g)$ as follows. Set $Q = \{v_{(s,t)} : s \in \{0,1,\cdots,l\}, t \in \{0,1,\cdots,d-1\}\}$ and $F_g = \{v_{(l,t)} : |t\rangle \in F_a\}$. Let $v_0 = v_{(0,0)}$. Now, define a function $L : Q \setminus F \longrightarrow \{1, 2\cdots, n\}$ as $L(v_{(s,t)}) = j_s$ for all t. Define the transition function $\delta : Q \times \{0,1\} \times Q \longrightarrow \mathbb{C}$ as

$$\delta(v_{s,t}, a, v_{p,q}) = \begin{cases} 0 \text{ if } p \neq s+1 \\ U_s(a)[q,t] \text{ if } p = s+1 \end{cases}$$

It is easy to see that δ is quantumly well-behaved.

To show that $\mathcal{G}$ simulates $\mathcal{A}$ exactly, it suffices to show that that the unitary $U_i^{O_x} = U_i(j_i)$ for all $i \in [l]$ since $|v_{0,0}\rangle = |v_0\rangle$ and $F_g = F$. Fix an input x. Let $i \in [l]$. Then from definition of GQBP, we have that for any $v \in Q_i$

$$U_i^{O_x} |v\rangle = \sum_{v' \in Q_{i+1}} \delta(v, L(v), w) |v'\rangle$$

We know that $L(v) = j_i$ for any $v \in Q_i$. So, we have

$$U_i^{O_x} = \sum_{v \in Q_i} \left[\sum_{v' \in Q_{i+1}} \delta(v, L(v), v') |v'\rangle \right] \langle v| = \sum_{v \in Q_i} \sum_{v' \in Q_{i+1}} U_i(j_i)[v', v] |v'\rangle \langle v| = U_i(j_i).$$

Hence, $\mathcal{G}$ simulates $\mathcal{A}$ exactly. This shows that any arbitrary (w,l)-AQBP can be simulated by an equivalent (w,l)-GQBP.

To show the tight inclusion, observe that an AQBP requires length at least n to query all the bits of an n-length input. Thus, $Parity_n \notin \texttt{AQBP}_{w,t}$ for any w and any $t < n$; however, we showed in the earlier section that $Parity_n \in \texttt{GQBP}_{2,n/2}$. ◀

Next, we turn to NQBP. Observe that a GQBP can be written as an NQBP if we set $Q_{acc} = F$, Q_{rej} as the states not in F in the last layer of the GQBP, and Q_{non} as the rest of the states. Since the evolution of a GQBP happens in layers, in none of the evolutions but the last one will any state in $|E_{acc}\rangle$ and $|E_{rej}\rangle$ be observed, and hence the evolution would continue uninterrupted. And since there is no non-E_{non} node in the last layer, the evolution would stop after that. Thus, a size-s length-l GQBP is also a size-s length-l NQBP.

Next, we prove the reverse direction.

▶ **Theorem 15.** *Any size s length l NQBP can be simulated by a width s length l GQBP.*

Proof. Let $\mathcal{N}$ be an NQBP of size s and length l. Now, make l many copies of $\mathcal{N}$. For any edge between two nodes v, v' in $\mathcal{N}$, remove that edge and add an edge between node v of copy i and node v' of copy $i+1$ for all $i \in [l-1]$. Clearly, this program is levelled since at i^{th} step, the state of the program is in some superposition of the nodes of copy i. Since, the size of $\mathcal{N}$ is s, the space of the superposition is $\mathcal{H}(s)$. ◀

5 Relation Between GQBPs and Quantum Query Circuits

To the best of our knowledge, there does not exist any study of the relation between quantum circuits and quantum branching programs. This section presents complete reductions from the GQBPs to the quantum query circuits and vice-versa. As an add-on, we also present reductions from the AQBPs to the quantum circuits with QRAM and standard oracle access to inputs.

We define a quantum circuit with QRAM[2] input access as a quantum circuit with two registers, a read-only input register and a workspace register, and a series of controlled-unitary gates with the control on exactly one of the qubit of the input register and the target gate on the workspace register. For any QRAM circuit C, the number of qubits in the workspace register is the space or qubit complexity of C, and the number of controlled gates is the query complexity of C. Mathematically, a q-qubit t-query quantum circuit with QRAM access to inputs can be described as $C = \tilde{O}^{p_t} \cdots \tilde{O}^{p_1}$, where the controlled-gate $\tilde{O}^{p_k}$ is defined as

$$\tilde{O}^{p_k} = \left(\mathbb{I}^{p_k-1} \otimes |0\rangle\langle 0| \otimes \mathbb{I}^{n-p_k}\right) \otimes \tilde{U}_k^{(0)} + \left(\mathbb{I}^{p_k-1} \otimes |1\rangle\langle 1| \otimes \mathbb{I}^{n-p_k}\right) \otimes \tilde{U}_k^{(1)}$$

where $\tilde{U}_k^{(0)}$ and $\tilde{U}_k^{(1)}$ are q-qubit unitaries.

▶ **Theorem 16.** *For any q-qubit, t-query quantum circuit C with QRAM query model, there exists an $(2^q, t)$-AQBP that simulates C.*

Proof. Let C be a q-qubit t-query quantum circuit with a QRAM query model. Let C be given as $C = \tilde{O}^{p_t} \cdots \tilde{O}^{p_1}$. Without loss of generality, let the initial state be $|0\rangle$. Let F_c be the set of basis states which, on obtaining post-measurement the given input is accepted.

Construct an AQBP $\mathcal{A} = \langle T, |\psi_0\rangle, F_A\rangle$ as follows. Set $|\psi_0\rangle = |0\rangle$ and $F_A = F_c$. Define $T = (\langle j_i, U_i(0), U_i(1)\rangle)_{i=1}^t$ where for any $i \in \{1, \cdots, t\}$, we set $j_i = p_i$, $U_i(0) = \tilde{U}_i^{(0)}$ and $U_i(1) = \tilde{U}_i^{(1)}$. We can see that $U_i(0), U_i(1)$ unitaries act on the Hilbert space $\mathcal{H}^{2^q}$. Clearly, the final state of $\mathcal{A}$ before the measurement can be obtained as

$$|\psi_f\rangle = U_t(j_t) \cdots U_1(j_1) |\psi_0\rangle = \tilde{U}_t^{(j_t)} \cdots \tilde{U}_1^{(j_1)} |0\rangle = C|0\rangle.$$

Since we have $F_A = F_c$, the probability of accepting an input x by $\mathcal{A}$ is

$$Pr_{\mathcal{A}}[accept|x] = \sum_{|i\rangle \in F_A} \left|\langle i|\psi_f\rangle\right|^2 = \sum_{|i\rangle \in F_c} \left|\langle i| C |0\rangle\right|^2 = Pr_C[accept|x].$$

This gives that the $(2^q, t)$-AQBP $\mathcal{A}$ exactly simulates the circuit C. ◀

▶ **Corollary 17.** *For any q-qubit, t-query quantum circuit C with QRAM query model, there exists an $(2^q, t)$-GQBP that simulates C.*

We show the following reductions from an AQBP to a quantum circuit.

▶ **Theorem 18.**

1. *Any (d, l)-AQBP can be simulated using a circuit with $\log d$ qubits and l gates that have a control on the input in the QRAM model when the input is stored in the QRAM.*
2. *Any (d, l)-AQBP can be simulated using a circuit with $\log n + \log d + 1$ qubits and $2 \cdot l$ queries to the oracle when the input is accessible only using an oracle.*

[2] QRAM stands for Quantum Random Access Memory.

The proof of Theorem 18 is presented in Appendix A. Although the reduction from an AQBP to a query circuit looks trivial, the same is not true for a reduction from a GQBP to a query circuit. Moreover, a generic reduction from a GQBP to a query circuit under the QRAM model is impossible. This is because of the fact that the QRAM model, as defined previously, does not allow for superposition queries, while GQBPs can query in superposition.

We now show that any quantum query circuit can be simulated by a GQBP exactly.

▶ **Theorem 19.** *For any q-qubit t-query quantum circuit C that has access to the input through a standard oracle, there exists an $\left(2^q, 2t+1\right)$-GQBP that simulates C.*

Proof. On an outline, to simulate C, we construct a GQBP such that all the odd transitions correspond to the unitaries and the even transitions correspond to the oracle calls. For this proof we assume that the query oracle used is a phase oracle, i.e., the oracle O_x acts as $O_x |i\rangle = (-1)^{x_i} |i\rangle$. Let the circuit C be given as $C = \tilde{U}_t O_x \tilde{U}_{l-1} \cdots \tilde{U}_1 O_x \tilde{U}_0$ and let F_c be the set of basis states which on obtaining post measurement the given input is accepted. Fix $l = 2t+1$ and $w = 2^q$. Now, define a GQBP $\mathcal{G} = (Q, E, v_0, L, \delta, F)$. Set $Q = \{v_{i,j} : i \in \{0,1,\cdots,l\}, j \in \{0,1,\cdots,w-1\}\}$, $E = \{(v_{i,j}, v_{i+1,k}) : i \in \{0,1,\cdots,l\}$ and $j,k \in \{0,1,\cdots,w-1\}\}$ and $|v_0\rangle = |v_{0,0}\rangle$ Also, fix $F = \{v_{l,j} : |j\rangle \in F_c\}$. Define the transition function δ as follows:

$$\delta(v_{i,j}, a, v_{p,q}) = \begin{cases} \tilde{U}_{i/2}[q,j] \ \ \forall\ a \in \{0,1\}, & \text{if } i \text{ is even and } p = i+1 \\ 1, & \text{if } j = q,\ a = 0,\ i \text{ is odd and } p = i+1 \\ -1, & \text{if } j = q,\ a = 1,\ i \text{ is odd and } p = i+1 \\ 0, & \text{else} \end{cases}$$

It is clear that for any transition between the $2i^{th}$ and $(2i+1)^{th}$ layer, the transition is query independent and the unitary $U_{2i}^{O_x} = \tilde{U}_i$. This simulates the query-independent unitaries of the circuit C. Meanwhile, the unitary $U_{2i+1}^{O_x}$ is query dependent and is a diagonal matrix where the j^{th} entry takes the value $+1$ if $x_{L(v_{2i+1,j})} = 1$ and -1 otherwise, i.e., the unitary acts as

$$U_{2i+1}^{O_x} |v_{2i+1,j}\rangle = (-1)^{x_{L(v_{2i+1,j})}} |v_{2i+1,j}\rangle .$$

We can see that the unitary $U_{2i+1}^{O_x}$ exactly simulates a query to the oracle O_x.

So, the final state after the evolution of the branching program can be given as

$$|\psi_f\rangle = U_l^{O_x} \cdots U_1^{O_x} |v_0\rangle = \tilde{U}_{l/2} O_x \tilde{U}_{l/2-1} \cdots \tilde{U}_1 O_x \tilde{U}_0 |v_0\rangle$$

Which is equivalent to the final state of the circuit C. Finally, on measuring ψ_f and obtaining an output $v_{l,j}$, an input x is accepted iff $v_{l,j} \in F$ or equivalently iff $|j\rangle \in F_c$. So, the probability of accepting an input x by $\mathcal{G}$ is

$$Pr[accept] = \sum_{v_{l,j} \in F} \left| \langle v_{l,j} | \psi_f \rangle \right|^2 = \sum_{j \in F_c} \left| \langle j | C |0\rangle \right|^2$$

This shows that the $(2^q, 2t+1)$-GQBP $\mathcal{G}$ simulates the query circuit C exactly. ◀

In the above reduction note that the unitaries $U_{2i}^{O_x}$ are query independent. We call the nodes that evolve in a query-independent way as '**dummy**' nodes. In the next lemma, we show that any level with only dummy nodes in a GQBP is removable without any overhead but with a possible reduction in width and length.

▶ **Lemma 20.** *Any level with only dummy nodes in a GQBP is removable without any width or length overhead.*

Proof. Consider a GQBP $\mathcal{G}$ with 3 levels. Without loss of generality, let all levels contain k nodes, and the second level only contains dummy nodes. Let $Q_i = \{v_{i,j} : j \in \{0, 1, \cdots, k-1\}\}$ for $i \in \{0, 2\}$ and $Q_2 = \{d_0, d_1 \cdots, d_{k-1}\}$. Fix an arbitrary input x. Per the definition, the transition matrix $U_1^{O_x}$ from level 1 to level 2 must be unitary. Similar is the case for the transition function $U_2^{O_x}$ for evolution from level 2 to level 3.

Let $U_c^{O_x} = U_2^{O_x} \cdot U_1^{O_x}$. We show that the GQBP $\mathcal{G}$ can be replaced by another GQBP $\mathcal{G}$', which has only two levels. Consider two nodes $v_{0,i}$ and $v_{3,j}$ of $\mathcal{G}$. Then, on considering all the paths from $v_{0,i}$ to $v_{2,j}$, we can get the combined transition amplitude as

$$\sum_{l=0}^{k-1} \delta(d_l, b, v_{2,j})\delta(v_{0,i}, a, d_l) = \sum_{l=0}^{k-1} U_2^{O_x}[j, l]U_1^{O_x}[l, i] = U_c^{O_x}[j, i].$$

See that this amplitude is independent of b and dependent only on a since d_ls are dummy nodes. So, it is possible to construct a transition function that is defined as $\delta'(v_{0,i}, a, v_{2,j}) = \sum_{l=0}^{k-1} \delta(d_l, b, v_{2,j})\delta(v_{0,i}, a, d_l)$ that mimics the transition from the node $v_{0,i}$ to $v_{2,j}$ in $\mathcal{G}$. Since this is true for any two nodes $v_{0,i} \in Q_0$ and $v_{1,i} \in Q_2$, we can replace the GQBP $\mathcal{G}$ with a GQBP $\mathcal{G}$ that has only two levels with Q_0 and Q_2 for nodes and with transition function δ' for transitions between nodes in Q_0 and Q_2. ◀

We can now use Lemma 20 to tighten the result in Theorem 19 to obtain the following corollary.

▶ **Corollary 21.** *For any q-qubit t-query quantum circuit C, there exists an $\left(2^q, t\right)$-GQBP that simulates C.*

From this result, one would hope that any $(w, l)-$GQBP would be reducible to a $\log w$-width l-query circuit. However, the GQBP we presented for the $Parity_n$ function serves a a contradiction to this. To see this, note that any query circuit would require at least $\log n$ qubits (that correspond to the indices) that would be used to query the oracle. So, any query circuit computing the $Parity_n$ function exactly will need at least $\log n$ qubits and $n/2$ queries. The lower bound on the number of queries comes from the facts that $Q_E(f) \geq deg(f)/2$ and that $deg(Parity_n) = n$. Now, if the above assumption were true, then clearly, we get a lower bound on w as $w \geq n$. However, the GQBP presented in Section 3 for the $Parity_n$ function has length $l = n/2$ but width 2 that contradicts this lower bound.

We now present the following theorem that characterizes a reduction from a GQBP to a query circuit.

▶ **Theorem 22.** *For any (w, l)-GQBP $\mathcal{G}$, there exists an $(l \log w + \log n + 1)$-qubit $2l$-query circuit C with access to input through an oracle O_x that simulates $\mathcal{G}$ exactly.*

The proof of this theorem is presented in Appendix B.

6 Conclusion and Open Directions

In this work we present a quantum variant of a decision graph [4], commonly known as a branching program, that mimics the evolution of a probabilistic (classical) branching program and further enables it to do so in a superposition. We are primarily concerned about the relationship of quantum query circuits and we prove various results explaining why and how they are equivalent (or, not).

There are several interesting questions that wait to be answered.

What would be the relation between (non-query) quantum circuits or quantum TMs and GQBP, especially with respect to the number of ancillæ qubits of the former and the space complexity of the latter? For example, it is known that polynomial-size classical branching

programs are equivalent to non-uniform logspace classical TMs and poly-size Boolean circuits. Many results about classical branching programs rely on counting arguments and that certainly does not readily work in the quantum case.

We showed that an AQBP is strictly weaker than a GQBP by essentially drawing attention to the fact that the former requires length n to be able to query *every input bit* which the latter does not. This actually stems from the structural requirement of AQBP to query the same variable in a level. Thus, to really compare the strength of the two models, one should consider programs of length $\Omega(n)$, and therefore, of bounded sub-exponential width. We conjecture that a GQBP of length $t = \Omega(n)$ and width $w = O(poly(n))$ is strictly stronger than an AQBP of similar length and width.

It is often assumed that a quantum model can do everything that a classical model can do, and even more. However, that does not appear to hold here. Specifically, it is not clear how to simulate a deterministic classical branching program of width w and length l using a GQBP of width $O(w)$ and length $O(l)$. The difficulty lies in the requirement that the transition function of a GQBP has to be quantumly well-behaved.

The above question is trivially resolved if we instead consider simulating a *Permutation Branching Programs* (PBP). A PBP is a classical levelled branching program that queries the same input bit in a level, and depending upon its value, performs a permutation on the nodes in that level (observe that a permutation matrix is also unitary). An AQBP is a quantum version of a PBP. Barrington's seminal result about NC^1 showed how to simulate any problem in that class using a width-5 (polynomial length) PBP [3]. Improving this result, Ablayebv et al. [2] proved only width 2 is necessary to simulate any NC^1 problem using an AQBP. Based on our understanding of GQBP, we conjecture that a bounded-width GQBP exists for any QNC^1 problem.

References

1 Farid Ablayev, Aida Gainutdinova, and Marek Karpinski. On computational power of quantum branching programs. In *Fundamentals of Computation Theory: 13th International Symposium, FCT 2001 Riga, Latvia, August 22–24, 2001 Proceedings 13*, pages 59–70. Springer, 2001.

2 Farid Ablayev, Cristopher Moore, and Christopher Pollett. Quantum and stochastic branching programs of bounded width: Track a. In *Automata, Languages and Programming: 29th International Colloquium, ICALP 2002 Málaga, Spain, July 8–13, 2002 Proceedings 29*, pages 343–354. Springer, 2002.

3 David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. doi:10.1016/0022-0000(89)90037-8.

4 William Joseph Masek. A fast algorithm for the string editing problem and decision graph complexity. Master's thesis, Massachusetts Institute of Technology, 1976.

5 Dmitri Maslov, Jin-Sung Kim, Sergey Bravyi, Theodore J Yoder, and Sarah Sheldon. Quantum advantage for computations with limited space. *Nature Physics*, 17(8):894–897, 2021.

6 Masaki Nakanishi, Kiyoharu Hamaguchi, and Toshinobu Kashiwabara. Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. In *Computing and Combinatorics: 6th Annual International Conference, COCOON 2000 Sydney, Australia, July 26–28, 2000 Proceedings 6*, pages 467–476. Springer, 2000.

7 Alexander A Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *International Symposium on Fundamentals of Computation Theory*, pages 47–60. Springer, 1991.

8 Martin Sauerhoff and Detlef Sieling. Quantum branching programs and space-bounded nonuniform quantum complexity. *Theoretical Computer Science*, 334(1-3):177–225, 2005.

9 Robert Wille, Stefan Hillmich, and Lukas Burgholzer. Decision diagrams for quantum computing. In *Design Automation of Quantum Computers*, pages 1–23. Springer, 2022.

A Proof of Theorem 18

We first proof the reduction from an AQBP to a query circuit under the QRAM model.

Proof. Let the input string be stored in QRAM. At any particular level of a (d, l)-AQBP, the maximum number of nodes is d. At each level, denote each node by a basis state from $|0\rangle$ to $|d-1\rangle$. Then, the total number of qubits required to express the state of a (d, l)-AQBP is $\log(d)$ since after the i^{th} step of evolution, the state would be in a superposition of nodes of level i (we assume the source node is at level 0.)

Next, at iteration i, we perform a quantum transformation of the type $\langle j_i, U_i(0), U_i(1)\rangle$ that takes a superposition of nodes at level $i-1$ to a superposition of nodes at level i. Alternatively, for $a \in \{0,1\}$, $U_i(a)$ maps each node at level $i-1$ to a superposition of nodes at level i. Since we denote the nodes by computational basis states, let $\widehat{U}_i(a)$ be the unitary that corresponds to the unitary $U_i(a)$ in the computational basis state notation.

Now, construct the circuit C_P that simulates the (d, l)-AQBP P as follows:

1. Initialize the circuit with $\log d$ qubits with the initial state as the state $|0\rangle$ corresponding to the source node v_0.
2. For $i = 1$ to l:
   a. Apply $\widehat{U}_i(0)$.
   b. Controlled on the j_i^{th} qubit in QRAM being 1, apply $\widehat{U}_i^\dagger(0)\widehat{U}_i(1)$.
3. Measure the final state $|\psi_l\rangle$ in the computational basis. If the measurement outcome corresponds to a node in F, then accept the input. Else, reject the input.

We show the correctness of the circuit C_P now. To prove this it suffices to show that the final state obtained just before measurement corresponds to the final state obtained in the AQBP P, i.e, if the circuit outputs the state

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}})\cdots\widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1})\,|\psi_0\rangle$$

The circuit starts in the state $|0\rangle = |\psi_0\rangle$ that corresponds to the node v_0. Next, at iteration i, in step 2 we apply only $\widehat{U}_i(0)$ to $|\psi_{i-1}\rangle$ if x_{j_i} qubit is 0. Else, we apply $\widehat{U}_i(0)\widehat{U}_i^\dagger(0)\widehat{U}_i(1) = \widehat{U}_i(0)$ to $|\psi_{i-1}\rangle$ if $x_{j_i} = 1$. So, the final state $|\psi_l\rangle$ we obtain is of the form,

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}})\cdots\widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1})\,|\psi_0\rangle$$

as expected. Finally, we measure $|\psi_l\rangle$ to see if the measurement outcome lies in the set of basis states that correspond to the accepting set F.

It is straightforward that the total number of qubits used in C_P is $\log d$ and the total number of controlled gates used is at most l. ◀

Next, we prove the reduction from an AQBP to a query circuit under the standard oracle model.

Proof. We make the node to basis state mapping and $U_i(a)$ to $\widehat{U}_i(a)$ mappings for $a \in \{0,1\}$ as done for the QRAM setup. However, now since the query is made to an oracle O_x that contains the input, at each iteration we also need an index that needs to be queried at that iteration. For this, we need an extra $\log n$ qubits and 1 qubit to store the value of the query. Let S_i denote the $\log n$ qubit unitary that performs the mapping $|0\rangle \longrightarrow |j_i\rangle$. Here we assume the oracle O_x as the standard oracle that acts on a $\log n + 1$ qubits state as $O_x\,|a\rangle\,|b\rangle \longrightarrow |a\rangle\,|b \oplus x_a\rangle$.

Construct the circuit C_P corresponding to the (d, l)-AQBP P as follows.

1. Initialize the circuit with 3 registers $R_1R_2R_3$ of sizes $\log n, \log d$ and 1 respectively with initial states as $|0\rangle$ in all the registers.
2. For $i = 1$ to l:
   a. Apply S_i on R_1 to obtain the state $|j_i\rangle$.
   b. Query the oracle O_x and store the query in R_3.
   c. Controlled on R_3 being in the state $|a\rangle$, apply $\widehat{U}_i(a)$ for $a \in \{0, 1\}$.
   d. Query the oracle O_x to reset R_3 to $|0\rangle$.
   e. Apply $S_i^\dagger$ on R_1 to reset R_1 to $|0\rangle$.
3. Measure R_2 of the final state $|\phi_l\rangle$ in the computational basis. If the measurement outcome corresponds to a node in F, then accept the input. Else, reject the input.

Now, we show the correctness of the circuit C_P. We first initialize the circuit with 3 registers $R_1R_2R_3$, the index register, the node register and the ancilla register, all set to $|0\rangle$. At iteration i, we perform the following operations:

$$
\begin{aligned}
|\phi_{i-1}\rangle = |0\rangle |\psi_{i-1}\rangle |0\rangle &\xrightarrow{S_i\otimes\mathbb{I}\otimes\mathbb{I}} |j_i\rangle |\psi_{i-1}\rangle |0\rangle \\
&\xrightarrow{O_x} |j_i\rangle |\psi_{i-1}\rangle |x_{j_i}\rangle \\
&\xrightarrow{\mathbb{I}\otimes\left[\widehat{U}_i(0)\otimes|0\rangle\langle 0|+\widehat{U}_i(1)\otimes|1\rangle\langle 1|\right]} |j_i\rangle \left[\widehat{U}_i(x_{j_i}) |\psi_{i-1}\rangle\right] |x_{j_i}\rangle = |j_i\rangle |\psi_i\rangle |x_{j_i}\rangle \\
&\xrightarrow{O_x} |j_i\rangle |\psi_i\rangle |0\rangle \\
&\xrightarrow{S_i^\dagger\otimes\mathbb{I}\otimes\mathbb{I}} |0\rangle |\psi_i\rangle |0\rangle = |\phi_i\rangle
\end{aligned}
$$

So, the state of the R_2 register after i^{th} iteration is

$$|\psi_i\rangle = \widehat{U}_i(x_{j_i}) |\psi_{i-1}\rangle .$$

So, the final state of R_2 register would be of the form

$$|\psi_l\rangle = \widehat{U}_l(x_{j_l})\widehat{U}_{l-1}(x_{j_{i-1}}) \cdots \widehat{U}_2(x_{j_2})\widehat{U}_1(x_{j_1}) |\psi_0\rangle$$

as required. We then measure this register and check if the output is a basis state corresponding to some node in F to accept or reject the input x.

Clearly, the number of queries made to the oracle O_x is $2 \cdot l$ and the total number of qubits used is $\log n + \log d + 1$. ◀

B Proof of Theorem 22

▶ **Theorem 23** (Restated Theorem 22). *For any (w, l)-GQBP $\mathcal{G}$, there exists an $(l \log w + \log n + 1)$-qubit $2l$-query circuit C with access to input through an oracle O_x that simulates $\mathcal{G}$ exactly.*

Proof. Fix an arbitrary width w. We now prove the reduction by induction on the length of GQBP. Let $\mathcal{G} = (Q, E, |v_0\rangle, L, \delta, F)$ be a (w, l)-GQBP. For ease of use we we denote $\delta(v_{i,j}, a, v_{i+1,k})$ as $\beta^{(i+1)}_{j,k,a}$. Let $l = 1$.

From the definition, for the only step of the evolution of this program, we define the unitary $U_1^{O_x}$ as

$$U_1^{O_x} |v\rangle = \sum_{v' \in Q_1} \delta(v, O_x(L(v)), v') |v'\rangle = \sum_{v' \in Q_1} \beta^{(1)}_{v,v',L(v_i)} |v'\rangle .$$

Then, the final state we obtain before the measurement is given by

$$\begin{aligned}
|\psi_1\rangle &= U_1^{O_x} |v_0\rangle \\
&= \sum_{v_i \in Q_0} \alpha_i U_1^{O_x} |v_i\rangle \\
&= \sum_{v_i \in Q_0} \alpha_i \left[\sum_{v_j \in Q_1} \beta^{(1)}_{v_i, v_j, L(v_i)} |v_j\rangle \right] \\
&= \sum_{v_j \in Q_1} \left[\sum_{v_i \in Q_0} \alpha_i \beta^{(1)}_{v_i, v_j, L(v_i)} \right] |v_j\rangle
\end{aligned}$$

So, the probability of obtaining one of the accept states is

$$Pr_{\mathcal{G}}[accept] = \sum_{v_j \in F} \left| \sum_{v_i \in Q_0} \alpha_i \beta^{(1)}_{v_i, v_j, L(v_i)} \right|^2$$

Construct a circuit $\mathcal{C}_{\mathcal{G}}$ with two registers R_1, R_2 each of size $\lceil w \rceil$ qubits, one register R_3 of size $\log n$ and one register R_4 of size 1 qubit all initialized to $|0\rangle$. Let $\mathcal{C}_{\mathcal{G}}$ be provided with an access to the input X through the oracle O_X and let $\widehat{U}_L$ be the unitary that acts as $\widehat{U}_L |j\rangle |0\rangle = |j\rangle |L(v_{0,j})\rangle$. Since we know that initial state of $\mathcal{G}$, we can construct a unitary U_{init} that act as $U_{init} |0\rangle = \sum_{i \in [w]} \alpha_i |i\rangle$ apriori. Then one step of $\mathcal{G}$ can be simulated in $\mathcal{C}_{\mathcal{G}}$ as follows:

$$\begin{aligned}
|0\rangle |0\rangle |0\rangle |0\rangle &\xrightarrow{U_{init}} \sum_{i \in [w]} \alpha_i |i\rangle |0\rangle |0\rangle |0\rangle \\
&\xrightarrow{CX(R_1,R_2)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |0\rangle |0\rangle \\
&\xrightarrow{\widehat{U}_L(R_1,R_3)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |L(v_{0,i})\rangle |0\rangle \\
&\xrightarrow{O_X(R_3,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle |i\rangle |L(v_{0,j})\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{C-U_1(R_1,R_2,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle \left(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \right) |L(v_{0,j})\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{O_X(R_3,R_4)} \sum_{i \in [w]} \alpha_i |i\rangle \left(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \right) |L(v_{0,j})\rangle |0\rangle \\
&\xrightarrow{\widehat{U}_L(R_1,R_3)} \sum_{i \in [w]} \alpha_i |i\rangle \left(\sum_{j \in [w]} \beta^{(1)}_{i,j,X_i} |j\rangle \right) |0\rangle |0\rangle \\
&= \sum_{j \in [w]} \left(\sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} |i\rangle \right) |j\rangle |0\rangle |0\rangle = |\psi_f\rangle \text{ (say)}
\end{aligned}$$

Here, $C - U_1$ is defined as

$$C - U_1 = \sum_{i \in [w]} |i\rangle \langle i| \otimes \left[\sum_{j \in [w]} \beta^{(1)}_{i,j,0} |j\rangle \langle i| \otimes |0\rangle \langle 0| + \sum_{j \in [w]} \beta^{(1)}_{i,j,1} |j\rangle \langle i| \otimes |1\rangle \langle 1| \right].$$

Note that the unitary $C - U_1$ is query independent. Next, let $\mathcal{M} = P_0, P_1$ be a projective measurement where $P_1 = \sum_{v_i \in F} |i\rangle \langle i|$ and $P_0 = I - P_1$. Once we measure the second register using $\mathcal{M}$,the probability of accepting X can be computed as

$$Pr_{\mathcal{C}_\mathcal{G}}[accept] = \sum_{v_j \in F} \left| \sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} |i\rangle \right|^2 = \sum_{v_j \in F} \left| \sum_{i \in [w]} \alpha_i \beta^{(1)}_{i,j,X_i} \right|^2.$$

Note that this is exactly the probability of accepting the input X in $\mathcal{G}$. This gives that the $\log(w) + \log(n) + 1$-qubit 2-query $\mathcal{C}_\mathcal{G}$ simulates the $(w, 1)$-GQBP $\mathcal{G}$ exactly.

Now, assume the result true for some $l = t$. Let $\mathcal{G}$'$= (Q', E', |v'_0\rangle, L', \delta', F')$ be a $(w, t+1)$-GQBP. Let the state at the end of t layers be given as

$$|\psi_t\rangle = \sum_{v_i \in Q_t} \gamma_i |v_i\rangle .$$

The transition unitary $U_t^{O_x}$ of the last step of the evolution of $\mathcal{G}$' can be defined as

$$U_t^{O_x} |v\rangle = \sum_{v' \in Q'_{t+1}} \delta'(v, O_x(L'(v)), v') |v'\rangle = \sum_{v' \in Q'_{t+1}} \beta^{(t)}_{v,v',L'(v_i)} |v'\rangle .$$

Then, the final state of the GQBP would be

$$|\psi_{t+1}\rangle = U_t^{O_x} |\psi_t\rangle = \sum_{v_i \in Q_t} \gamma_j \left[\sum_{v_j \in Q_{t+1}} \beta^{(t)}_{v,v_j,L'(v_i)} |v_j\rangle \right] = \sum_{v_j \in Q_{t+1}} \left[\sum_{v_i \in Q_t} \gamma_i \beta^{(t)}_{v,v_j,L'(v_i)} \right] |v_j\rangle .$$

So, the probability of accepting an input X is

$$Pr_{\mathcal{G}'}[accept] = \sum_{v_j \in F'} \left| \sum_{v_i \in Q_t} \gamma_i \beta^{(t)}_{v_i,v_j,L(v_i)} \right|^2$$

Next, let $\mathcal{C}'_\mathcal{G}$ be the circuit that computes the first t layers of $\mathcal{G}$' exactly using $t \log(w) + \log(n) + 1$ qubits and $2t$ queries. Then, the final state of this circuit will be

$$\left|\psi_t^C\right\rangle = \sum_{i \in [w]} \gamma_i |\chi_i\rangle |i\rangle |0\rangle |0\rangle$$

for some normalized states $|\chi_i\rangle$ of $(t-1)\log(w)$ qubits. Now, append a register of $\log(w)$ qubits initialized to $|0\rangle$ to the circuit $\mathcal{C}'_\mathcal{G}$. For ease of computation, we append it in between the current second and the third register. So, the new state of the circuit will be

$$\left|\psi_t^C\right\rangle = \sum_{j \in [w]} \gamma_i |\chi_i\rangle |i\rangle |0\rangle |0\rangle |0\rangle = R_1 R_2 R_3 R_4 R_5 \text{ (say)}.$$

Now, we simulate the final step of $\mathcal{G}$ as below:

$$
\begin{aligned}
\left|\psi_t^C\right\rangle &= \sum_{j\in[w]} \gamma_i \left|\chi_i\right\rangle \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle \left|0\right\rangle \\
&\xrightarrow{CX(R_2,R_3)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|0\right\rangle \left|0\right\rangle \\
&\xrightarrow{\widehat{U}_L(R_2,R_4)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|L(v_{0,i})\right\rangle \left|0\right\rangle \\
&\xrightarrow{O_X(R_4,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \left|i\right\rangle \left|L(v_{0,j})\right\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{C-U_t(R_2,R_3,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|L(v_{0,j})\right\rangle \left|X_{L(v_{0,j})}\right\rangle \\
&\xrightarrow{O_X(R_4,R_5)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|L(v_{0,j})\right\rangle \left|0\right\rangle \\
&\xrightarrow{\widehat{U}_L(R_2,R_4)} \sum_{i\in[w]} \gamma_i \left|\chi_j\right\rangle \left|i\right\rangle \Big(\sum_{j\in[w]} \beta^{(t)}_{i,j,X_i} \left|j\right\rangle \Big) \left|0\right\rangle \left|0\right\rangle \\
&= \sum_{j\in[w]} \Big(\sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \left|\chi_j\right\rangle \left|i\right\rangle \Big) \left|j\right\rangle \left|0\right\rangle \left|0\right\rangle = \left|\psi_f\right\rangle \ \text{(say)}
\end{aligned}
$$

Similar to the length 1 case, $C - U_t$ is defined as

$$
C - U_t = \sum_{i\in[w]} \left|i\right\rangle\left\langle i\right| \otimes \Big[\sum_{j\in[w]} \beta^{(t)}_{i,j,0} \left|j\right\rangle\left\langle i\right| \otimes \left|0\right\rangle\left\langle 0\right| + \sum_{j\in[w]} \beta^{(t)}_{i,j,0} \left|j\right\rangle\left\langle i\right| \otimes \left|1\right\rangle\left\langle 1\right| \Big].
$$

Then finally on measuring the third register, we get that the probability of accepting an input X is

$$
Pr_{\mathcal{C}'_{\mathcal{G}}}[accept] = \sum_{v_j\in F'} \left| \sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \left|i\right\rangle \right|^2 = \sum_{v_j\in F'} \left| \sum_{i\in[w]} \gamma_i \beta^{(t)}_{i,j,X_i} \right|^2.
$$

which equals the probability of accepting the input X in $\mathcal{G}$'. So, we obtain a $((l+1)\log(w) + \log(n)+1)$-qubit $2l+2$-query circuit to simulate $\mathcal{G}$' exactly as required. ◀