entropy

MDPI

Article

Visualizing Quantum Circuit Probability: Estimating Quantum State Complexity for Quantum Program Synthesis

Bao Gia Bach [1,2], Akash Kundu [2,3,4], Tamal Acharya [2] and Aritra Sarkar [2,5,*]

1 Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City 70000, Vietnam; bao.bachbbace12@hcmut.edu.vn
2 Quantum Intelligence Research Team, Department of Quantum & Computer Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands; akundu@iitis.pl (A.K.); tamal_974@yahoo.com (T.A.)
3 Joint Doctoral School, Silesian University of Technology, 44-100 Gliwice, Poland
4 Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, 44-100 Gliwice, Poland
5 Quantum Machine Learning Group, Quantum Computing Division, QuTech, 2628 CJ Delft, The Netherlands
* Correspondence: a.sarkar-3@tudelft.nl

Abstract: This work applies concepts from algorithmic probability to Boolean and quantum combinatorial logic circuits. The relations among the statistical, algorithmic, computational, and circuit complexities of states are reviewed. Thereafter, the probability of states in the circuit model of computation is defined. Classical and quantum gate sets are compared to select some characteristic sets. The reachability and expressibility in a space-time-bounded setting for these gate sets are enumerated and visualized. These results are studied in terms of computational resources, universality, and quantum behavior. The article suggests how applications like geometric quantum machine learning, novel quantum algorithm synthesis, and quantum artificial general intelligence can benefit by studying circuit probabilities.

Keywords: gate-based quantum computing; algorithmic probability; circuit complexity; reachability; expressibility

Citation: Bach, B.G.; Kundu, A.; Acharya, T.; Sarkar, A. Visualizing Quantum Circuit Probability: Estimating Quantum State Complexity for Quantum Program Synthesis. *Entropy* **2023**, *25*, 763. https://doi.org/10.3390/e25050763

Academic Editors: Andrei Khrennikov and Karl Svozil

Received: 25 April 2023
Revised: 4 May 2023
Accepted: 5 May 2023
Published: 7 May 2023

1. Introduction

Quantum computing has entered a technological readiness level where quantum processor platforms, albeit limited, are becoming accessible for experimentation. This rapid progress has encouraged researchers to study various real-world industrial and scientific applications [1] using quantum algorithms. The logical formulations of these algorithms are then processed by a quantum computing stack [2] of system abstractions into low-level quantum circuits for a gate-based quantum computing device. The so-called NISQ (noisy intermediate-scale quantum) era [3,4] characterizes the limitations of current quantum processors in coherence time, gate errors, and qubit connectivity. This has led to explorations from the other end [5], in devising design strategies and finding use cases for these limited computing power to achieve a computational advantage. To better utilize these limited devices, it is imperative to understand the relations between quantum logic and physical resources. This motivates the research presented in this article.

Quantum computation lies at the intersection of quantum physics and computer science. It has allowed a rich exchange of concepts between these two fields. Specific to the interests of this research, (i) physical laws provide fundamental bounds to computation, while (ii) computation provides an information-theoretic explanation of many physical phenomena. The former was first explored in the context of thermodynamic limits [6], leading to the development of reversible computation [7,8], and eventually to define the limits of quantum computation [9,10]. Efforts of the latter come under the purview of digital physics. Some seminal contributions include cellular automaton [11], constructor theory [12], tensor networks [13], informational axioms [14–16] of quantum mechanics,

and, a principle of stationary action for computing [17,18], among others. To focus on both directions of this synergy, we present an empirical demonstration of the consequences of existing theoretical ideas for quantum computing. Specifically, we transport concepts from algorithmic information theory [19] (a sub-field of theoretical computer science and artificial intelligence) to gate-based quantum computation [20].

In this work, we consider an enumeration of the space of quantum circuits (as quantum assembly, QASM). This is a very small sample of the uncountable infinite quantum processes that can be defined on the Hilbert space of a given dimension. The subset of processes is based on (i) the native gate set (of a quantum processor), (ii) the maximum circuit width (total number of qubits), and (iii) the bounds on the circuit depth (based on the decoherence time). We investigate how these circuits map a space of classical inputs to classical outputs (via Z-axis measurements). A crucial assumption of our formalism is that QASM/circuits are encoded using a discrete universal gate set. The space of quantum programs constructed in such a matter is enumerable and is, thus, formally countably infinite. However, in most practical implementations (with finite and circuit depth), the set of meaningful computations is finite. Even for universal gate sets with arbitrary rotation angles (e.g., $\{Rx(\theta_x), Ry(\theta_y), CX\}$), a finite number of control signal configurations in practical quantum computer implementation effectively discretizes the set of native gates. While any arbitrary unitary can be decomposed to arbitrary precision using a universal gate set, given resource bounds (e.g., in lines of QASM codes before the system decoheres) the space of programs cannot map to any quantum process. Formally, while uniform samples of Haar random unitary matrices yield a uniformly sampled random state vector [21], *the state distribution based on uniform samples of unitary matrices from a discrete gate set is guided by concepts from algorithmic information theory. This limits physical processes that can be effectively executed on gate-based quantum computers*.

Since the first quantum algorithms were formulated in the 1990s, the discovery of new algorithms [22] has progressed steadily. However, quantum algorithm design involves quantum mechanical phenomena (e.g., superposition, entanglement), which is counter-intuitive to human experience. Thus, reasoning in terms of mathematical formalism has been a barrier to entry for developing more advanced quantum logic and is, thus, a bottleneck for the broader adoption of quantum accelerated computing. There have been some proposals to remedy these issues via genetic programming [23] and circuit synthesis [24]. In this work, we carry forward this research direction via the principled approach [25] of algorithmic information theory. A quantum program synthesis framework would require understanding the space of quantum programs and their associated resources, to implement the program search/induction. As discussed in this article, the landscape of resource-bounded quantum circuits and their corresponding classical information processing capability lay the groundwork for this.

The rest of the article is organized as follows. In Section 2, we describe how states are represented as symbols and transformations over these symbols, and how this affects their statistical and algorithmic complexities. Section 3 discusses the subtleties of forming Boolean and quantum circuits from gate sets. In Section 4, we present our implementation of the enumeration of state complexities using various gate sets. The results are visualized and analyzed. Section 5 concludes the article with a discussion of various applications of this research.

2. States and Complexities

The states of a system define its observable behavior. These can be encoded in various ways. A common way to encode them is by assigning a symbol for each unique distinguishable state. Subsequent spatio-temporal observations (of the same system, or a larger system composed of systems of this size) are denoted by a string of this alphabet. A system with a single state is not very interesting, nothing really happens there. A very simple case of a slightly more interesting system is a coin, with two states—heads and tails. Coins can be fair or biased, can be tossed multiple times, or multiple coins can be tossed

together/consecutively/conditionally. The outcome of a series of (or in parallel) coin tosses can be represented by a Boolean string. Given an ensemble of all Boolean strings of length n, represented as $\{0,1\}^{\otimes n}$, each string might be equally probable with $1/2^n$. In a physical system, if each output is equally likely, the uniform distribution models that system, for example, a communication line might need to transfer each encoding with equal probability. According to this, a fair coin tossed 8 times, would have the same probability and element of surprise for 1111111011 as that of a specific permutation of 51 s and 50 s, for example, 0101100101.

If every state is *equally likely* and *unrelated* to each other, it is a very boring gray world. Thankfully it is not so! We perceive structures around us. Why that is this way is hard to answer—but a plausible explanation is that our human biological/technological sensing tools are limited. So instead of parsing the underlying randomness in its full spectrum, we perceive an emergent statistical structure and symmetry. These structures allow us two additional ways of enhancing our representation of states. The complexity of states can be studied from these two perspectives—statistical and algorithmic.

2.1. The Statistical Emergence of Entropy

The first enhancement is based on relaxing the criteria of all states being 'equally likely'. We find that we have apparent favoritism towards 0101100101 being a more acceptable result of a series of fair coin tosses. This is based on our ignorance of the micro-states of the permutations. We focus on the pattern that the total possible states with 91 s are 10, while those with 51 s and 50 s are $10 \times 9 \times 8 \times 7$, similar to entropy in statistical thermodynamics. States with higher entropy are more in number and this flow towards an expected higher entropy state in the universe is what gives us our perception of time. In information theory, given a discrete random variable X, which takes values in the alphabet $\mathcal{X}$ and is distributed according to $p : \mathcal{X} \to [0,1]$, the Shannon entropy [26] of the variable sampled from this ensemble is given by $H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$. This likeness denotes the average level of statistical information, or the surprise/uncertainty inherent to the variable's possible outcomes, and is the maximum for a uniform distribution. The way to optimally encode a biased set of percepts as information is the basis of code words like Huffman coding. The encoding is designed to tune to the bit lengths of concepts by making the most used concepts more economical. To balance it, less used concepts become more costly than their native length. For example, the probabilities $p(00) = 0.4, p(01) = 0.05$, $p(10) = 0.2, p(11) = 0.35$ is best encoded as $00 :\to 0, 01 :\to 111, 10 :\to 110, 11 :\to 10$. Note that, this code word is better than the original only as long as the biased probability distribution is maintained (which in turn might be an artifact of sensing emergent symmetries of macro-states). If instead, all strings are equally probable, these coding schemes are more costly.

We do something similar with semantics in languages, for example, instead of having new words for every single huge animal with a large truck with a specific set of $x, y, z, \ldots$ features that we fail to distinguish, we call all of them 'an elephant', while for those we can distinguish, for example, your pet cat, we give them special names. Your friend might not be able to distinguish your cat from another, or to the eyes of a trained mahout, every elephant is uniquely identifiable—leading to the subjectivity of language. Similarly, using the scientific names of species or the full genetic code of an individual would be tedious for everyday use, however, is useful for biological classification or medical treatment. Thus, much the same way as Huffman coding, words in languages arise due to different ways of ignoring details and focusing on specific semantics. The compression provided by a language forms the basis of comprehension, much the same way macro states (microstates preserving certain symmetries) lead to emergent physical laws.

2.2. The Algorithmic Emergence of Universality

The second enhancement is based on relaxing the criteria of all states being 'unrelated'. While a unique encoding for all percepts under consideration is good, we can compress

and comprehend better if we can find the relation among these symbols. For example, we can relate bit strings by their inverses, or arrange integers consecutively on a number line. Assigning symbols to relations is merely an attempt to minimize the number of code words itself by ignoring symbols for states that can now be described by some specific syntactic composite of symbols of other states and relations. To map to every percept, the total length of the encodings using these codes is not necessarily less than the original binary or Huffman coding. Thus, again, it is subjective when these relations will be beneficial instead of adding extra complexity to the encoding. The goal is not about being the most resource-efficient way for the full spectrum of percepts, but rather using a smaller set of symbols for a biased distribution or subset of percepts. This trade-off between generality and efficiency is the reason esoteric languages like MetaGolfScript [27] are banned from code golf contests, or the RISC and CISC architectures exist in tandem.

However, surprisingly, we find that some relations are so ubiquitous that, they can map to all percepts (often even from an infinite set) with just the encoding of the relation and that of a starting percept. For example, the successor function (i.e., add 1) is represented by 1 and the number 0 is represented by 0. With this, any natural number can be represented by nesting the successor function, for example, 1110 is 3. While the successor function is universal over the set of natural numbers given 0, the multiplication operation with the set of all prime numbers can also span the natural numbers. Such a set of inputs and transformations are universal for the target output set. Some of these symbols (of states and relations) are so powerful that they can potentially represent an infinite set of states with a finite set of symbols and some compositional rules, for example, any d-base numeral system with d-symbols and a positional notation can represent any integer.

From an engineering point of view, having such a universal set of states and transformations helps in taming an infinitude of possibility with a finite number of these building blocks, for example, a keyboard is made of alphabets instead of a key for every word in the dictionary. There are two subtleties to this enhancement. (i) Firstly, since we now have a finite number of block types to construct an infinite number of states, the length of the description using these blocks can potentially be infinite. If we put a bound on the number of blocks we can use, by combinatorial arguments, it bounds the number of states we can describe. States that require longer descriptions are no longer expressible. (ii) Secondly, while the original states represented an observable behavior, these new pseudo-states and transformations that we introduced to reduce our symbol set need not necessarily have intuitive standalone meaning. For some, it may, for example, the bit-flip operator can correspond to the action of toggling a switch, however, for others, it may not, for example, the alphabets do not have semantic meaning by themselves.

We are now equipped with a symbol set consisting of (i) some set of observed states, (ii) a rich (universal) set of transformations to describe other observed percepts, and optionally, (iii) ubiquitous intermediate pseudo-states in the transformations. As a digress, it is crucial to note that transformations can be represented as states in a higher dimension via channel-state duality [28], representing dynamics as statics. Now, can we choose an optimal encoding scheme subjective to the probabilities of the various symbols being used to describe the physical phenomena (from the set of all macroscopic percepts)? The problem is that it is not known beforehand the ways in which the blocks will be used, i.e., the distribution of the ensemble. Imagine operating a Lego factory and deciding how many of each block to manufacture for customers' needs. Most often, due to lack of any other information, encoding of the base set is chosen as the standard binary encoding, (i.e., with the assumption that all blocks and initial percepts will be required with uniform probability), for example, the ASCII code. This is called the opcode encoding of the instruction set architecture in computers. In this scenario (of universal computation), it can be useful to study things from the other end, i.e., the resources required to represent a specific percept. Resources are typically of two types: (i) the length of the description of the percept using the language, and (ii) the computational cost in terms of cycles (time) and memory (space) for decoding the description.

The computational cost is studied in the field of computational complexity. Problems (and thereby, their solutions as a sequence of instructions based on symbols), are classified into different classes [29] based on the scaling behavior of time and space with the size of the problem. Some common ones are polynomial time (P), non-deterministic polynomial time (NP), and bounded-error quantum polynomial time (BQP).

The length of description quantifies the Kolmogorov complexity [30] or algorithmic entropy of the percept. It is defined as $K_U(X) = \min_p\{\ell(p) : U(p) = x\}$, where ℓ denotes the length of the (prefix-free) program p on the encoding used by the universal Turing machine U that outputs x. Though it depends on the choice of the building blocks and their encodings, the dependence is only on an additive constant term (called the invariance theorem) which is the length of a cross-compiler to another language/automata. Thus, it is useful to use Kolmogorov complexity to quantify the individual complexity of a string, irrespective of an ensemble. However, finding the exact value is uncomputable. There are many ways to approach it from the upper side (lower semi-computable), for example, via compression algorithms, minimum description length, and the block decomposition method.

2.3. Relations to Circuit Complexity

So far we reviewed three different notions of the complexity of states:

1. Statistical complexity: Shannon entropy on an ensemble of states (given its probability distribution)
2. Computational complexity: Space-time scaling behavior of a program to generate the state (given a language)
3. Algorithmic complexity: Length of the program to generate the state (given a language)

In this research, we are instead interested in the circuit complexity of a state, as this is the canonical model for quantum computation. Circuit complexity is related to algorithmic complexity [31], which in turn is related to statistical [32] and computational complexities [33]. Computational complexities typically deal with asymptotic scaling behavior and provide lower bounds. Though families of circuits have specific complexity class hierarchy (e.g., AC^i, TC^i, NC^i) it is not of much interest for this research. We will focus on circuits with bounded size (in both space and time). Similarly, the expected Kolmogorov complexity has been shown to correspond to the Shannon entropy [32], though this relation is not of immediate importance to this work. [31] Kolmogorov complexity can be shown to be very similar to circuit complexity under certain considerations [31]. Another similar relation is that truth tables of functions with small circuit complexity have small Kolmogorov complexity. Counting arguments relating to circuit, algorithmic and statistical complexities has been suggested in [17,18] in terms of Lagrangian action. Our research is another step in this rather niche field of understanding observed states via different perspectives.

It is important to note that most research on algorithmic information theory has been in the context of universal automata, for example, Turing machines, lambda calculus, cellular automata, etc. The size of the description depends on how expressive the symbols are for the transformations. What we described so far, i.e., transformations as a relation between two states, is typically the case in the language of circuits. Programs written in a more abstract logical framework allow more powerful primitives, like universal and existential quantifiers in first-order or higher-order logic. Typically, a universal computation model demands a recursively enumerable language. In the Chomsky hierarchy, Turing machines are more powerful than linear-bounded automata, which are in turn more powerful than push-down automata and in turn, finite-state machines (FSM). See [34] for a comparison of these for both classical and quantum computing models. However, for less powerful automata and language models, it is possible to derive corresponding notions [35] of algorithmic complexity. This is important as programs written in Turing-complete languages eventually are translated via the layers of the computing stack and are executed by logic circuits.

These logic circuits are however a combination of sequential (allowing memory cells) and combinatorial logic and can be used to simulate an FSM. Purely combinatorial logic (not to be confused with combinatory logic, which is universal) is of even lower power than FSM. The former is loopless and stateless and thereby is a direct representation of the output state based on the input. It is important to note that, program execution is typically clocked in both classical and quantum processors to prevent race conditions, even if the circuits are purely composed of combinatorial logic elements. Thus, resources of time and space can be defined in this setting even without tracking and accessing intermediate states. By borrowing notions from algorithmic information theory (as defined on functional programs), in this work, we study the effect of circuit complexity of Boolean/quantum combinatorial logic on state complexity.

3. Landscape of Circuits

With this background of the measures of complexity, let us now first explore the landscape of Boolean circuits. The quantum circuit model is inspired by and is a generalization of the Boolean circuit model, so, it would be natural to start with a classical model and generalize it to the corresponding quantum formulation.

3.1. Circuit Probability of States

Algorithmic information is typically studied for classical functions (e.g., for λ-calculus) than for combinatorial Boolean logic circuits. We intend to study the latter. Let us consider the space of n-bit strings. Given a set of gates that form a Boolean circuit, we find that all outputs are not equally likely. This is because, while each {circuit, input} pair has only one output, there are many ways of generating the same outputs from multiple circuits. In fact, we can make our circuits arbitrarily big by dummy operations like identity or two consecutive NOT-gates.

Since there are many programs, to compare two strings, instead of finding the shortest circuit to output the string, we are interested in the probability of each circuit being generated. This is similar to the notion of the algorithmic probability [36] of the string and is defined as $M(X) = \sum_{p:U(p)=x*} 2^{-\ell(p)}$ when the prefix-free programs p on the universal automata U are encoded in binary. The largest contribution to this term comes from the shortest program (i.e., the Kolmogorov complexity). This connection between complexity and probability can be expressed as a string that has a short program that has many alternate ways of generating it and is, thus, more probable to become generated by a universal automaton programmed randomly. Note that, assigning a uniform random distribution of programs for generating the algorithm probability, or the universal distribution over the entire set of strings, is not fully justified. In Section 4.5 of [37], one of the authors proposed a more physically motivated 'nested algorithmic probabilities' that converge to constructors. In this work, we will start with a uniform distribution but will later generalize the implementation to allow any prior distribution. To distinguish the usual notion of algorithmic probability of a string on universal automata $M_U(X)$ from our case of the probability of an output string based on the distribution of equivalent circuits with varied space-time complexities, we denote our formulation of algorithmic probability as $M_{circ}(X)$.

In the original setting, $M_U(X)$ is uncomputable, as it requires running each possible program, of which there exist programs that do not halt. However, it is lower semi-computable and can be approximated given bounds on run-time. One proposal to approximate is based on [38] running every Turing machine in a particular enumeration, and directly using the output distribution of halting Turing machines up to the bounded run-time. In the case of Boolean/quantum circuits, the run-time bounds are predetermined, and there are no halting problems. Thus, $M_{circ}(s)$ for a state s can be approximated by the ratio of the cardinality of the sets that generate the target state from the initial state s_0 with the total number of circuits, as

$$M_{circ}(s) \approx \frac{|C \in \{\texttt{gateset}, \texttt{maxspace}, \texttt{maxtime}\},\ s \leftarrow C(s_0)|}{|C \in \{\texttt{gateset}, \texttt{maxspace}, \texttt{maxtime}\}|}. \quad (1)$$

This can be used to estimate the quantum circuit complexity using the coding theorem by extending the relation [39] between probability and complexity to circuits as, $K_{circ}(s) = -\log M_{circ}(s)$.

3.2. Boolean Gate Sets

In the Boolean circuit form of algorithmic probability, we will consider strings of n-bits, and the probabilities of each bit string getting generated from all possible Boolean circuits on all possible inputs. The main restriction (i.e., the output not being also uniformly random) comes from the fact that we do not have primitives (1-time step gates) for all possible Boolean functions. We typically use a universal gate set that can compile any other Boolean functions down to a larger number of gates from that set. Thus, in our operational implementation, we need to choose a gate set for the empirical enumeration of the circuits.

Given v input variables with a symbol set of size s, there are s^v possible combinations of these inputs. If there is a single output variable from the symbol set of size d, the total number of possible functions [40] is d^{s^v}.

- For 1-input Boolean algebra, i.e., when $v = 1$, $s = 2$, $d = 2$, the total number of functions are $f = 2^{2^1} = 4$. These functions are the $\{0, 1, A, \overline{A}\}$.
- For 2-input Boolean algebra, i.e., when $v = 2$, $s = 2$, $d = 2$, the total number of functions are $f = 2^{2^2} = 16$. These are denoted by $\{0, 1, A, B, \overline{A}, \overline{B}, A \bullet B, \overline{A \bullet B}, A + B, \overline{A + B}, A + \overline{B}, \overline{A} + B, A \bullet \overline{B}, \overline{A} \bullet B, A \oplus B, \overline{A \oplus B}\}$.

A functionally complete set of logical connectives or Boolean operators can be used to express all possible truth tables by combining members of the set into a Boolean expression. These sets can also express any Boolean SAT or SAT-3 formula. Some examples of such universal [41] sets are {`NAND`}, {`NOR`}, {`NOT`, `AND`}, {`NOT`, `OR`}. These gate sets are related to each other, using the following equivalences:

- `NOT(A) = NAND(A,A) = NOR(A,A)`,
- `OR(A,B) = NAND(NAND(A,A),NAND(B,B)) = NOR(NOR(A,B),NOR(A,B))` `= NOT(AND(NOT(A),NOT(B)))`,
- `AND(A,B) = NAND(NAND(A,B),NAND(A,B)) = NOR(NOR(A,A),NOR(B,B))` `= NOT(OR(NOT(A),NOT(B)))`.

3.3. Quantum Gate Sets

The classical formulation that maps the landscape of Boolean functions can now be generalized to include quantum gates and states. There is a 3-input single gate in quantum logic that is universal for classical computing, the `CCX` gate (also called the Toffoli gate). It can simulate the `NAND` gate via `CCX(A,B,1) = (A,B,NAND(A,B))`. Classical computations are in general irreversible processes, thus, the inputs cannot be recovered from the outputs. Quantum logic is based on unitary evolution and thus, is reversible. Additionally, quantum computations allow quantum superposition and entanglement, which are not implied in reversible computation. The `CCX` gate can simulate the entire set of reversible computations by additionally simulating a `Fanout` gate (or Copy gate) as `CCX(A,1,0) = (A,1,A)`. Thus, both {`NAND`, `Fanout`} and {`CCX`} form universal gate sets for reversible computation. The `CSWAP` gate (also called the Fredkin gate) is another universal gate for reversible logic.

The generalization of reversible to quantum logic needs only one extra gate, the `H` gate (Hadamard). In principle, the real gate set composed of {`CCX`, `H`} is computationally universal [42]. However, it needs ancilla qubits to encode the real normalization factors and complex algebra to decompose [43] to arbitrary quantum unitary gates for a strong sense of universality. Moreover, it is important that the effect of the `NOT` gate (or, the `X` gate in quantum) cannot be simulated without assuming the availability of both $|0\rangle$ and $|1\rangle$ states. Since our enumeration of quantum programs will start will the qubits initialized to

the all-zero state, we need to augment the gate set to {X, H, CCX} to reach all binary strings as output.

The principle of algorithmic probability should also hold in the quantum setting, i.e., a uniform distribution of all possible functions and all possible input states does not imply a uniform distribution of all possible output states on the Hilbert space. Nielsen's geometric quantum computing (GQC) approach [44] shows that finding optimal quantum circuits is essentially equivalent to finding the shortest path between two points in a certain curved Riemannian geometry. However, it is not possible to empirically visualize this, as we need to consider all possible input states and all possible unitary maps. Studying the landscape of program synthesis requires discretizing this space for the native gate set of the target quantum processor (or the quantum compiler). However, the number of possible functions or processes in a quantum environment (even for a single qubit) is uncountably infinite. Thus, choosing a universal gate set gets more pronounced in the setting of quantum control.

In a way this is easy. It has been shown that if one can apply some Hamiltonian repeatedly to a few variables at a time one can in general affect any desired unitary time evolution on an arbitrarily large number of variables. As a result, almost any quantum logic gate with two or more inputs is computationally universal [45] in a way that copies of the gate can be wired together to effect any desired logic circuit and to perform any desired unitary transformation on a set of quantum variables. We call this richer counterpart to its classical cousin [11], the ubiquity of quantum universality (UQU).

How many types of quantum gates in the gate set do we need to represent this richer set of quantum unitary operators, and how many of them do we need? Well, if we are provided with a parametric family of quantum operators, only a few types of such operators are sufficient. The quantum Shannon decomposition (QSD) [46] provides a theoretical lower bound and asymptotic optimality for an exact decomposition of quantum unitaries using the parametric family of gates {RY(θ), RZ(θ), CX}. It can be recursively applied to larger quantum circuits with the CX count scaling of $O(4^n)$.

GQC, UQU, and QSD rely on an arbitrary expressive set of gates. This is not very practical as quantum devices are manufactured and controlled to perform operations from a predefined dictionary. There is a subtle difference in using a finite set of operators with respect to the classical case. Instead of the classical setting of d^{s^v} being represented perfectly by a sequence of gates from the universal gate set G, in the quantum setting, the aim is to approximate all possible unitary operations with a sequence of gates from G with a bound of the approximation quality. This can be understood by thinking of representing all real numbers using digits of a specific numeral base. Of course, there is a trade-off to taming this countably infinite space with a finite number of building blocks. Quantum Kolmogorov complexity (QKC) [47–49] is a measure of the information required to describe a quantum state. For any definition of quantum Kolmogorov complexity measuring the number of classical bits required to describe a pure quantum state, there exists a pure n-qubit state which requires exponentially many bits of description.

Nevertheless, the Solovay–Kitaev theorem (SKT) [50] allows an efficient classical algorithm for compiling an arbitrary single-qubit gate into a sequence of gates from a fixed and finite set. The algorithm, using a universal gate set [51] (e.g., {H, T, CX}), runs in $O(log(1/\epsilon))$ time, and produces as output a sequence of $O(log(1/\epsilon))$ quantum gates which approximate the desired quantum gate to an accuracy within $\epsilon > 0$. It can be generalized to apply to multi-qubit gates and to gates from $SU(d)$.

In retrospect, there is no foundational reason known why GQC, UQU, QSD, QKC, and SKT play out in nature in this manner. Yet, eventually, it allows us to sufficiently parse and explore the vast Hilbert space using an arbitrary choice of a small set of building blocks. In the next section, we will present a formal formulation of our emuneration procedure, the results, and their analysis.

4. Implementation

We first describe the implementation of the classical case. The problem is formulated as follows: given (i) n bits, $b_i \in \{b_0, b_1, \ldots, b_{n-1}\} = B$, (ii) an initial state for each bit $s_0(b_i)$ (typically set to 0), (iii) a set of gates $g \in G$ (not necessarily universal), and, (iv) number of lines of QASM code L; find the distribution of final states given each gate is applied with probability $\frac{1}{|G|}$ at each $l \in L$.

In the quantum case, the gate set is now defined as a set of unitary gates, while the initial state over a set of n qubits Q is defined as $s_0(Q) := \sum_{j \in \{0,2^n-1\}} \alpha_j |j\rangle$, such that $\alpha_j \in \mathbb{C}$ and $|j\rangle$ are eigenstates of the n-dimensional Hilbert space in the Z-basis.

4.1. Gate Sets

We consider the following gate sets:

1. {`CCX`}—This set is universal for classical and reversible logic, provided both the initial states of $|0\rangle$ and $|1\rangle$ are provided. It is not practical to provide all initial states without knowing how to create one from the other. Since all gate-based quantum algorithms start from the all-$|0\rangle$ state and prepare the required initial state via gates, we will not consider this set for our enumeration.
2. {`X`, `CCX`}—This set is universal for classical and reversible logic by starting from the all-$|0\rangle$ state.
3. {`X`, `H`, `CCX`}—This set is weakly universal under encoding and ancilla assumptions for quantum logic. The encoding, while universal, might not preserve the computation resource complexity benefits of quantum (i.e., in the same way, classical computation can also encode all quantum computation using {`NAND`, `Fanout`}). Thus, we do not consider this set for our enumeration of the quantum case.
4. {`H`, `S`, `CX`}—The Clifford group is useful for quantum error correction. However, it is non-universal and can be efficiently simulated on classical logic [52]. The space of transforms on this set encoded error-correction codes and is, thus, useful to map.
5. {`H`, `T`}—This set is universal for single qubit quantum logic. However, we will consider the generalization to multi-qubit using an additional two-qubit gate in the set in the following case.
6. {`H`, `T`, `CX`}—This is universal for quantum logic.
7. {`P(pi/4)`, `RX(pi/2)`, `CX`}—The IBM native gate set is used to construct this gate set. The following relations establish the relation with the previous universal gate set: `T` $=$ `P(pi/4)`, `X` $=$ `RX(pi/2)`, and, `H` $= e^{i\pi/2}$`XRz(pi/2)X` $= e^{i\pi/2}$`XTTTTX`. We will consider additional constraints like device connectivity to apply this technique to real quantum processors.

Thus, in our experiments, we map the algorithmic probability of the final states for the following gate sets: (i) {`X`, `CCX`}, (ii) {`H`, `S`, `CX`}, (iii) {`H`, `T`, `CX`}, and (iv) {`P(pi/4)`, `RX(pi/2)`, `CX`}.

4.2. Metrics for Evaluation

We are interested in evaluating these metrics for each of the gate sets:

- Expressivity: refers to the extent to which the Hilbert space can be encoded by using an unbounded number of gates. It is not weighted by the probability as it is a characteristic of the encoding power of the gate set. We assign a 1 to a final state if it can be expressed as starting from the initial state and applying a sequence of gates from the gate set.
- Reachability: refers to a bounded form of expressibility. The length of the sequence of gates must be equal to or shorter than the specified bound. This corresponds to a physical implementation rather than the power of the gate set and characterizes the computational complexity and thereby the decoherence time of the processor.

The expressibility is mapped primarily to understand if the reachability bound is under/over-specified. As the value of the circuit length L is gradually increased, any

universal gate set will populate the full landscape of states in the expressibility criteria, and thereby remain without variation. It is at this limit, i.e., at the first instance of full expressibility, that reachability is best understood. The other instance we are interested in is the infinite limit of L, and its effect on the reachability distribution.

These experimental procedures and the comparative study of the results are presented in the following sections.

4.3. Enumeration Procedure

We construct the experiment by constructing all possible QASM programs. For each gate, $g_i \in G$ in the gate set, the target number of qubits $q(g_i)$ are known. Thereafter, given n qubits, all possible permutations $\mathcal{P}$ of applying the gate are enumerated in a list, i.e., $\mathcal{P}^n_{q(g_i)}$. The total possible options for each line of QASM is $\sum_G \mathcal{P}^n_{q(g_i)}$, and thus, the total possible QASM programs for L lines of code length are

$$\left[\sum_G \mathcal{P}^n_{q(g_i)}\right]^L. \tag{2}$$

Our implementation is available at: https://github.com/Advanced-Research-Centre/QCircScape (accessed on: 25 April 2023).

As an example, consider the gate set $G = \{\texttt{X}, \texttt{CCX}\}$, for $n = 4$ and $L = 3$. $q(\texttt{X}) = 1$ and $q(\texttt{CCX}) = 3$. Thus, $\mathcal{P}^4_{q(\texttt{X})} = 4$, and $\mathcal{P}^4_{q(\texttt{CCX})} = 24$. Note that even if exchanging the assignment of the two controls of the Toffoli gate has the same effect, this is a symmetry property of this gate and not in general true for 3-qubit unitaries. Thus, the description number (program id) for these cases is treated as different computational paths. It can be appreciated that these two options of Toffoli gates would behave very differently in the presence of noise characteristic of individual qubits as well as other control constraints. The total options for each line if QASM is 28, and thus, for length 3, the total number of programs is $28^3 = 21{,}952$. This is already a large number of quantum circuits to be simulated, for a small case, and gives a preview of how large the spaces of the programs are.

By applying all possible cases, we obtain an array of size n that represents the available number of transitions from one specific state to another. The measurement basis (here, considered to be the default Z-basis), is crucial for this research. If we consider all possible initial states of bit-strings (Z-basis state preparations) of size n, we obtain a $n \times n$ matrix. This exploration of other initial states helps us to understand the asymmetry of gates over bit values (e.g., a generalization of Toffoli gates with inverted control qubits is of 4 types: $\bar{\texttt{C}}\bar{\texttt{C}}\texttt{X}$, $\bar{\texttt{C}}\texttt{CX}$, $\texttt{C}\bar{\texttt{C}}\texttt{X}$, $\texttt{CCX}$).

In the classical scenario, (e.g., for {X, CCX}), this corresponds to the statistics of the number of computational paths between these two states using an arrangement of gates from the set, conforming to a specified length. For the quantum case, the statistics correspond to the sum of probabilities of the computational paths collapsing on measurement to the target state. Dividing the matrix by the total number of programs gives us the fixed-length algorithmic probability of the state on each row, conditioned on the initial state. This normalized $n \times n$ matrix is the reachability landscape. All non-zero values correspond to the states that are reachable by at least one route (i.e., at least one program exists to transform to that state). This gives us the Boolean $n \times n$ expressibility matrix.

4.4. Results

To start our enumeration, we first plot the growth of the number of programs (i.e., Equation (2)) with qubit count and circuit depth for various gate sets. We note that the trend is independent of the description of the gates in the gate set. The only information that matters is how many target qubits each gate in the set acts on. This gives us two classes among our chosen gate sets, (i) with one 1-qubit and one 3-qubit gate, (ii) with two 1-qubit

and one 2-qubit gate. The result is plotted in Figure 1. We find that the permutations due to a 3-qubit gate grow much faster than the other class.

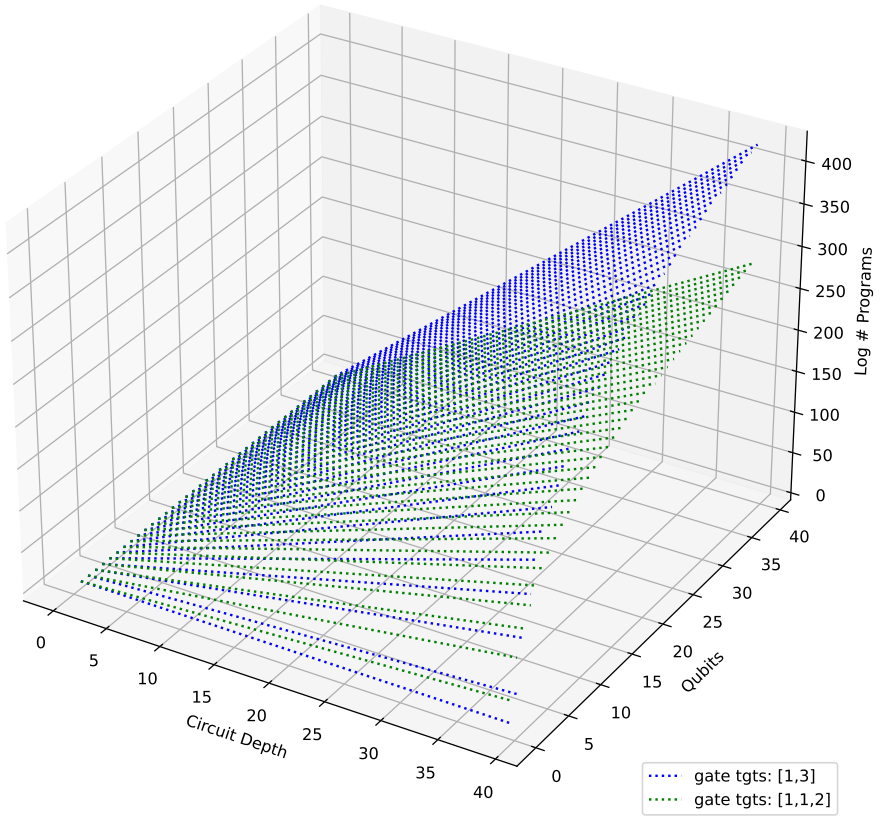


Figure 1. Growth of the number of programs with qubit count and circuit depth for two types of gate sets: (i) $[1,3]$ qubits: $\{\texttt{X},\texttt{CCX}\}$, (ii) $[1,1,2]$ qubits: $\{\texttt{H},\texttt{S},\texttt{CX}\}$, $\{\texttt{H},\texttt{T},\texttt{CX}\}$, $\{\texttt{P}(\pi/4),\texttt{RX}(\pi/2),\texttt{CX}\}$.

The following figures visualize the expressibility (top row) and reachability (bottom row) for the gate set on 4 qubits with increasing depth (from 0 to 4 operations). The gate sets we consider are {X, CCX} (Figure 2), {H, S, CX} (Figure 3), {H, T, CX} (Figure 4) and {P(pi/4), RX(pi/2), CX} (Figure 5).

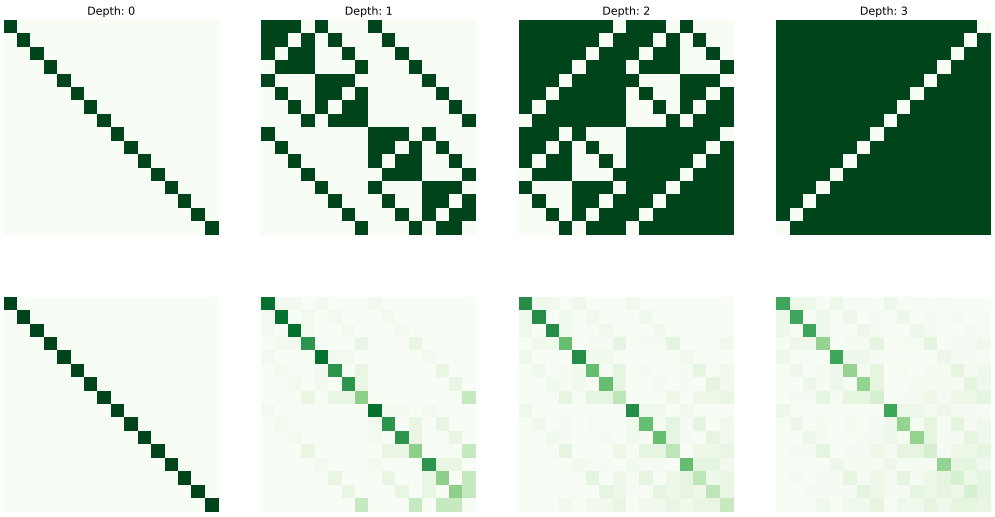


Figure 2. Expressibility and Reachability for gate set $\{\texttt{X},\texttt{CCX}\}$ on 4 qubits and of circuit depth from 0 to 3.

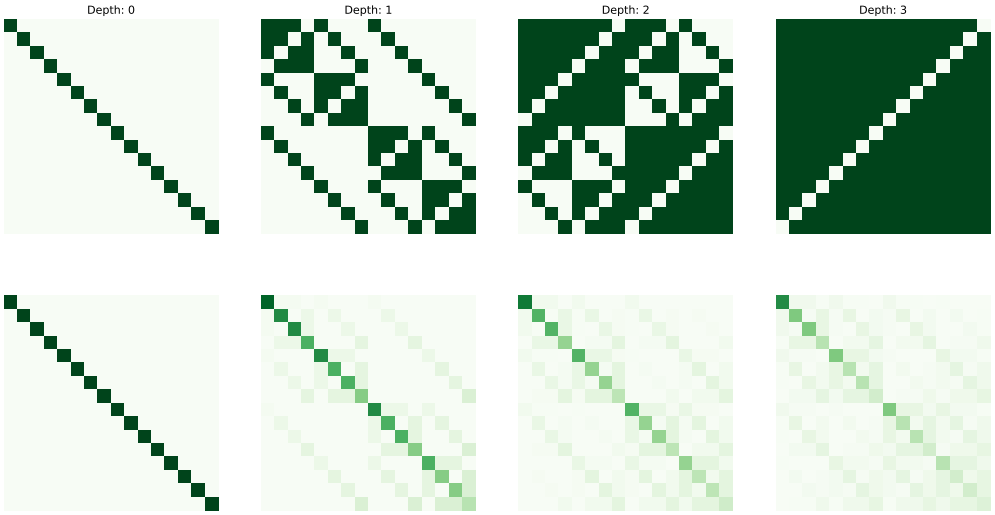


Figure 3. Expressibility and Reachability for gate set $\{\texttt{H}, \texttt{S}, \texttt{CX}\}$ on 4 qubits and of circuit depth from 0 to 3.

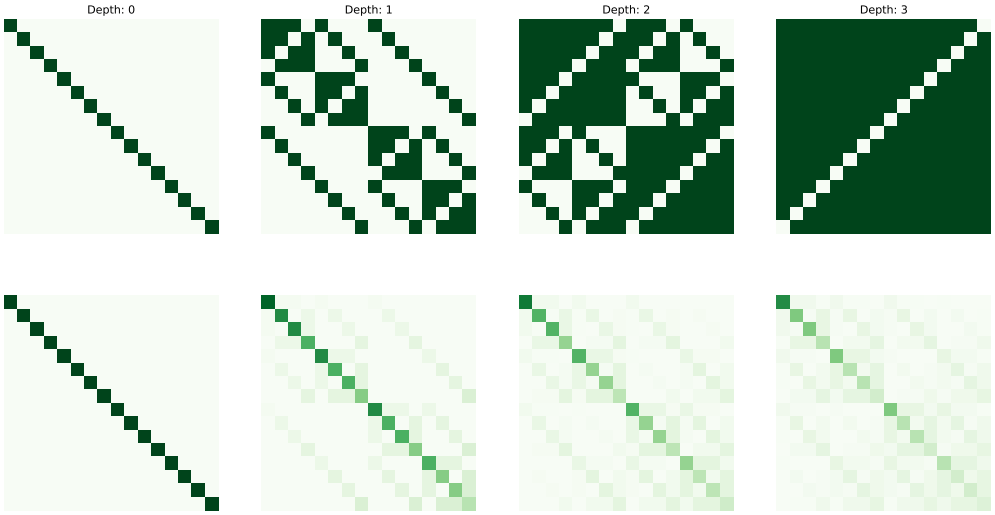


Figure 4. Expressibility and Reachability for gate set $\{\texttt{H}, \texttt{T}, \texttt{CX}\}$ on 4 qubits and of circuit depth from 0 to 3.

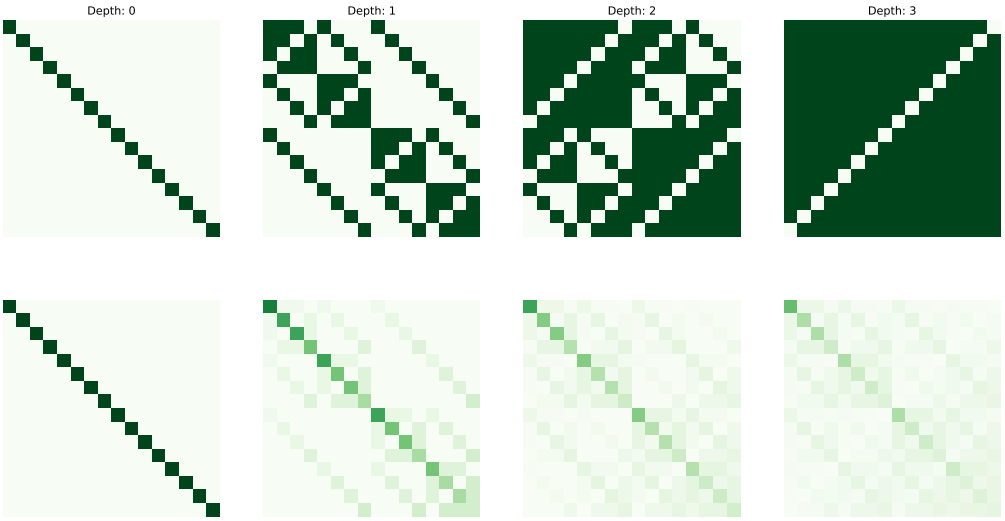


Figure 5. Expressibility and Reachability for gate set $\{\texttt{P}(\mathfrak{B}/4), \texttt{RX}(\mathfrak{B}/2), \texttt{CX}\}$ on 4 qubits and of circuit depth from 0 to 3.

4.5. Analysis and Discussion

Let us consider `maxspace` $= 4$ and the classical gate set, $\{$`X, CCX`$\}$. There are 28 distinct possibilities for each time step. The reachability statistics for the Z-basis states for time step $0, 1, 2, 3$ are:

$$R_0^{\{\texttt{X, CCX}\}} = \begin{smallmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \end{smallmatrix}$$

$$R_1^{\{\texttt{X, CCX}\}} = \begin{smallmatrix} 24&1&1&0&1&0&0&0&1&0&0&0&0&0&0&0\\ 1&24&0&1&0&1&0&0&0&1&0&0&0&0&0&0\\ 1&0&24&1&0&0&1&0&0&0&1&0&0&0&0&0\\ 0&1&1&20&0&0&0&3&0&0&0&3&0&0&0&0\\ 1&0&0&0&24&1&1&0&0&0&0&0&1&0&0&0\\ 0&1&0&0&1&20&0&3&0&0&0&0&0&3&0&0\\ 0&0&1&0&1&0&20&3&0&0&0&0&0&0&3&0\\ 0&0&0&3&0&3&3&12&0&0&0&0&0&0&0&7\\ 1&0&0&0&0&0&0&0&24&1&1&0&1&0&0&0\\ 0&1&0&0&0&0&0&0&1&20&0&3&0&3&0&0\\ 0&0&1&0&0&0&0&0&1&0&20&3&0&0&3&0\\ 0&0&0&3&0&0&0&0&0&3&3&12&0&0&0&7\\ 0&0&0&0&1&0&0&0&1&0&0&0&20&3&3&0\\ 0&0&0&0&0&3&0&0&0&3&0&0&3&12&0&7\\ 0&0&0&0&0&0&3&0&0&0&3&0&3&0&12&7\\ 0&0&0&0&0&0&0&7&0&0&0&7&0&7&7&0 \end{smallmatrix}$$

$$R_2^{\{\texttt{X, CCX}\}} = \begin{smallmatrix} 580&48&48&2&48&2&2&0&48&2&2&0&2&0&0&0\\ 48&580&2&44&2&44&0&6&2&44&0&6&0&6&0&0\\ 48&2&580&44&2&0&44&6&2&0&44&6&0&0&6&0\\ 2&44&44&420&0&10&10&96&0&10&10&96&0&0&0&42\\ 48&2&2&0&580&44&44&6&2&0&0&0&44&6&6&0\\ 2&44&0&10&44&420&10&96&0&10&0&0&10&96&0&42\\ 2&0&44&10&44&10&420&96&0&0&10&0&10&0&96&42\\ 0&6&6&96&6&96&96&220&0&0&0&58&0&58&58&84\\ 48&2&2&0&2&0&0&0&580&44&44&6&44&6&6&0\\ 2&44&0&10&0&10&0&0&44&420&10&96&10&96&0&42\\ 2&0&44&10&0&0&10&0&44&10&420&96&10&0&96&42\\ 0&6&6&96&0&0&0&58&6&96&96&220&0&58&58&84\\ 2&0&0&0&44&10&10&0&44&10&10&0&420&96&96&42\\ 0&6&0&0&6&96&0&58&6&96&0&58&96&220&58&84\\ 0&0&6&0&6&0&96&58&6&0&96&58&96&58&220&84\\ 0&0&0&42&0&42&42&84&0&42&42&84&42&84&84&196 \end{smallmatrix}$$

$$R_3^{\{\texttt{X, CCX}\}} = \begin{smallmatrix} 14112&1738&1738&136&1738&136&136&18&1738&136&136&18&136&18&18&0\\ 1738&14100&140&1498&140&1498&22&336&140&1498&22&336&22&336&0&126\\ 1738&140&14100&1498&140&22&1498&336&140&22&1498&336&22&0&336&126\\ 136&1498&1498&9064&22&532&532&2766&22&532&532&2766&0&354&354&1344\\ 1738&140&140&22&14100&1498&1498&336&140&22&22&0&1498&336&336&126\\ 136&1498&22&532&1498&9064&532&2766&22&532&0&354&532&2766&354&1344\\ 136&22&1498&532&1498&532&9064&2766&22&0&532&354&532&354&2766&1344\\ 18&336&336&2766&336&2766&2766&4092&0&354&354&1572&354&1572&1572&2758\\ 1738&140&140&22&140&22&22&0&14100&1498&1498&336&1498&336&336&126\\ 136&1498&22&532&22&532&0&354&1498&9064&532&2766&532&2766&354&1344\\ 136&22&1498&532&22&0&532&354&1498&532&9064&2766&532&354&2766&1344\\ 18&336&336&2766&0&354&354&1572&336&2766&2766&4092&354&1572&1572&2758\\ 136&22&22&0&1498&532&532&354&1498&532&532&354&9064&2766&2766&1344\\ 18&336&0&354&336&2766&354&1572&336&2766&354&1572&2766&4092&1572&2758\\ 18&0&336&354&336&354&2766&1572&336&354&2766&1572&2766&1572&4092&2758\\ 0&126&126&1344&126&1344&1344&2758&126&1344&1344&2758&1344&2758&2758&2352 \end{smallmatrix}$$

The Reachability matrix plotted in the figures is normalized by the total possible options for each line of code (i.e., the sum for any row), for example, in this gate set, 28 for R_1, 784 for R_2 and 21,952 for R_3. The Expressibility matrix is a binary matrix representation of non-zero elements of R. It is obvious that R_0 only has the diagonal elements, as there is only one way to create a circuit with no gates, and they effectively map the state to itself. However, using R_1, all other R_l can be directly derived by matrix exponentiation by

$$R_l^G = (R_1^G)^l, \qquad G = \{\texttt{X}, \texttt{CCX}\}. \tag{3}$$

This seems obvious from a network theory perspective, where n-hop neighbors can be found by exponentiation of the adjacency matrix. Techniques from graph theory can be

used to study quantum circuits [53], but what is fascinating is that it can predict the distribution of states without executing each circuit. In our enumeration, we considered equal weights for each gate. Machine learning models can be trained on quantum programs to find the distribution of quantum gates on realistic quantum algorithms, and thereby can be used to study the kind of states most quantum algorithms generate.

Let us now develop a notion of M_{circ}. In the algorithmic probability formulation of prefix-free programs, the convergence to a semi-measure is based on the notion that the infinite sum, $\lim_{i\to\inf}\sum_i 2^{-i} = \frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\cdots = 1$. In our case, we need to maintain that the contributions from subsequent lines of code decrease in a similar manner. This gives us the following Equation (4) for M_{circ} given a specific gate set and system size

$$M_{circ}^{G,n} = \lim_{L\to\inf}\sum_{i=1}^{L}\left\{2^{-i} * R_i^G * \Big[\sum_{g_i\in G}\mathcal{P}^n_{q(g_i)}\Big]^{-i}\right\}. \tag{4}$$

Next, we analyze the quantum universal gate set $\{\texttt{H, T, CX}\}$. We need to be careful that now

$$R_l^G \neq (R_1^G)^l, \qquad G = \{\texttt{H,T,CCX}\}. \tag{5}$$

Using Equation (5), we find the following values:

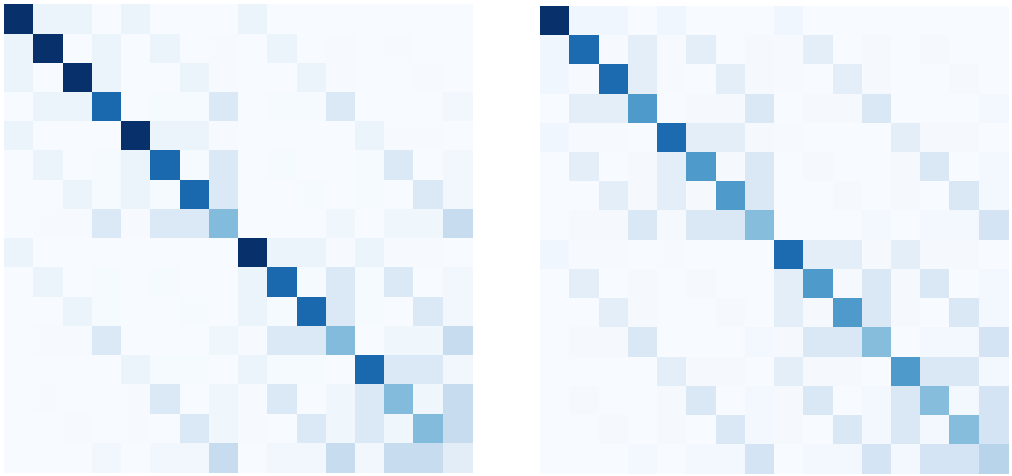
$$R_1^{\{\texttt{H, T, CCX}\}} = \begin{pmatrix}
18. & 0.5 & 0.5 & 0. & 0.5 & 0. & 0. & 0. & 0.5 & 0. & 0. & 0. & 0. & 0. & 0. & 0.\\
0.5 & 15. & 0. & 1.5 & 0. & 1.5 & 0. & 0. & 0. & 1.5 & 0. & 0. & 0. & 0. & 0. & 0.\\
0.5 & 0. & 15. & 1.5 & 0. & 0. & 1.5 & 0. & 0. & 0. & 1.5 & 0. & 0. & 0. & 0. & 0.\\
0. & 1.5 & 1.5 & 12. & 0. & 0. & 0. & 2.5 & 0. & 0. & 0. & 2.5 & 0. & 0. & 0. & 0.\\
0.5 & 0. & 0. & 0. & 15. & 1.5 & 1.5 & 0. & 0. & 0. & 0. & 0. & 1.5 & 0. & 0. & 0.\\
0. & 1.5 & 0. & 0. & 1.5 & 12. & 0. & 2.5 & 0. & 0. & 0. & 0. & 0. & 2.5 & 0. & 0.\\
0. & 0. & 1.5 & 0. & 1.5 & 0. & 12. & 2.5 & 0. & 0. & 0. & 0. & 0. & 0. & 2.5 & 0.\\
0. & 0. & 0. & 2.5 & 0. & 2.5 & 2.5 & 9. & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 3.5\\
0.5 & 0. & 0. & 0. & 0. & 0. & 0. & 0. & 15. & 1.5 & 1.5 & 0. & 1.5 & 0. & 0. & 0.\\
0. & 1.5 & 0. & 0. & 0. & 0. & 0. & 0. & 1.5 & 12. & 0. & 2.5 & 0. & 2.5 & 0. & 0.\\
0. & 0. & 1.5 & 0. & 0. & 0. & 0. & 0. & 1.5 & 0. & 12. & 2.5 & 0. & 0. & 2.5 & 0.\\
0. & 0. & 0. & 2.5 & 0. & 0. & 0. & 0. & 0. & 2.5 & 2.5 & 9. & 0. & 0. & 0. & 3.5\\
0. & 0. & 0. & 0. & 1.5 & 0. & 0. & 0. & 1.5 & 0. & 0. & 0. & 12. & 2.5 & 2.5 & 0.\\
0. & 0. & 0. & 0. & 0. & 2.5 & 0. & 0. & 0. & 2.5 & 0. & 0. & 2.5 & 9. & 0. & 3.5\\
0. & 0. & 0. & 0. & 0. & 0. & 2.5 & 0. & 0. & 0. & 2.5 & 0. & 2.5 & 0. & 9. & 3.5\\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 3.5 & 0. & 0. & 0. & 3.5 & 0. & 3.5 & 3.5 & 6.
\end{pmatrix}$$

$$R_2^{\{\texttt{H, T, CCX}\}} = \begin{pmatrix}
327. & 16. & 16. & 1.5 & 16. & 1.5 & 1.5 & 0. & 16. & 1.5 & 1.5 & 0. & 1.5 & 0. & 0. & 0.\\
16. & 234. & 2.5 & 40. & 2.5 & 40. & 0. & 7.5 & 2.5 & 40. & 0. & 7.5 & 0. & 7.5 & 0. & 0.\\
16. & 2.5 & 234. & 40. & 2.5 & 0. & 40. & 7.5 & 2.5 & 0. & 40. & 7.5 & 0. & 0. & 7.5 & 0.\\
1.5 & 40. & 40. & 163. & 0. & 8.5 & 8.5 & 52. & 0. & 8.5 & 8.5 & 52. & 0. & 0. & 0. & 17.5\\
16. & 2.5 & 2.5 & 0. & 234. & 40. & 40. & 7.5 & 2.5 & 0. & 0. & 0. & 40. & 7.5 & 7.5 & 0.\\
1.5 & 40. & 0. & 8.5 & 40. & 163. & 8.5 & 52. & 0. & 8.5 & 0. & 0. & 8.5 & 52. & 0. & 17.5\\
1.5 & 0. & 40. & 8.5 & 40. & 8.5 & 163. & 52. & 0. & 0. & 8.5 & 0. & 8.5 & 0. & 52. & 17.5\\
0. & 7.5 & 7.5 & 52. & 7.5 & 52. & 52. & 114. & 0. & 0. & 0. & 18.5 & 0. & 18.5 & 18.5 & 52.\\
16. & 2.5 & 2.5 & 0. & 2.5 & 0. & 0. & 0. & 234. & 40. & 40. & 7.5 & 40. & 7.5 & 7.5 & 0.\\
1.5 & 40. & 0. & 8.5 & 0. & 8.5 & 0. & 0. & 40. & 163. & 8.5 & 52. & 8.5 & 52. & 0. & 17.5\\
1.5 & 0. & 40. & 8.5 & 0. & 0. & 8.5 & 0. & 40. & 8.5 & 163. & 52. & 8.5 & 0. & 52. & 17.5\\
0. & 7.5 & 7.5 & 52. & 0. & 0. & 0. & 18.5 & 7.5 & 52. & 52. & 114. & 0. & 18.5 & 18.5 & 52.\\
1.5 & 0. & 0. & 0. & 40. & 8.5 & 8.5 & 0. & 40. & 8.5 & 8.5 & 0. & 163. & 52. & 52. & 17.5\\
0. & 7.5 & 0. & 0. & 7.5 & 52. & 0. & 18.5 & 7.5 & 52. & 0. & 18.5 & 52. & 114. & 18.5 & 52.\\
0. & 0. & 7.5 & 0. & 7.5 & 0. & 52. & 18.5 & 7.5 & 0. & 52. & 18.5 & 52. & 18.5 & 114. & 52.\\
0. & 0. & 0. & 17.5 & 0. & 17.5 & 17.5 & 52. & 0. & 17.5 & 17.5 & 52. & 17.5 & 52. & 52. & 87.
\end{pmatrix}$$

$$(R_1^{\{\texttt{H, T, CCX}\}})^2 = \begin{pmatrix}
325. & 16.5 & 16.5 & 1.5 & 16.5 & 1.5 & 1.5 & 0. & 16.5 & 1.5 & 1.5 & 0. & 1.5 & 0. & 0. & 0.\\
16.5 & 232. & 2.5 & 40.5 & 2.5 & 40.5 & 0. & 7.5 & 2.5 & 40.5 & 0. & 7.5 & 0. & 7.5 & 0. & 0.\\
16.5 & 2.5 & 232. & 40.5 & 2.5 & 0. & 40.5 & 7.5 & 2.5 & 0. & 40.5 & 7.5 & 0. & 0. & 7.5 & 0.\\
1.5 & 40.5 & 40.5 & 161. & 0. & 8.5 & 8.5 & 52.5 & 0. & 8.5 & 8.5 & 52.5 & 0. & 0. & 0. & 17.5\\
16.5 & 2.5 & 2.5 & 0. & 232. & 40.5 & 40.5 & 7.5 & 2.5 & 0. & 0. & 0. & 40.5 & 7.5 & 7.5 & 0.\\
1.5 & 40.5 & 0. & 8.5 & 40.5 & 161. & 8.5 & 52.5 & 0. & 8.5 & 0. & 0. & 8.5 & 52.5 & 0. & 17.5\\
1.5 & 0. & 40.5 & 8.5 & 40.5 & 8.5 & 161. & 52.5 & 0. & 0. & 8.5 & 0. & 8.5 & 0. & 52.5 & 17.5\\
0. & 7.5 & 7.5 & 52.5 & 7.5 & 52.5 & 52.5 & 112. & 0. & 0. & 0. & 18.5 & 0. & 18.5 & 18.5 & 52.5\\
16.5 & 2.5 & 2.5 & 0. & 2.5 & 0. & 0. & 0. & 232. & 40.5 & 40.5 & 7.5 & 40.5 & 7.5 & 7.5 & 0.\\
1.5 & 40.5 & 0. & 8.5 & 0. & 8.5 & 0. & 0. & 40.5 & 161. & 8.5 & 52.5 & 8.5 & 52.5 & 0. & 17.5\\
1.5 & 0. & 40.5 & 8.5 & 0. & 0. & 8.5 & 0. & 40.5 & 8.5 & 161. & 52.5 & 8.5 & 0. & 52.5 & 17.5\\
0. & 7.5 & 7.5 & 52.5 & 0. & 0. & 0. & 18.5 & 7.5 & 52.5 & 52.5 & 112. & 0. & 18.5 & 18.5 & 52.5\\
1.5 & 0. & 0. & 0. & 40.5 & 8.5 & 8.5 & 0. & 40.5 & 8.5 & 8.5 & 0. & 161. & 52.5 & 52.5 & 17.5\\
0. & 7.5 & 0. & 0. & 7.5 & 52.5 & 0. & 18.5 & 7.5 & 52.5 & 0. & 18.5 & 52.5 & 112. & 18.5 & 52.5\\
0. & 0. & 7.5 & 0. & 7.5 & 0. & 52.5 & 18.5 & 7.5 & 0. & 52.5 & 18.5 & 52.5 & 18.5 & 112. & 52.5\\
0. & 0. & 0. & 17.5 & 0. & 17.5 & 17.5 & 52.5 & 0. & 17.5 & 17.5 & 52.5 & 17.5 & 52.5 & 52.5 & 85.
\end{pmatrix}$$

Let us understand why this is the case. If we start with the state $|0\rangle$, and apply the Hadamard gate, we obtain the state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. However, in terms of probability, this gets translated in the normalized reachability matrix as $0.5\,|0\rangle + 0.5\,|1\rangle$. Now, when, another Hadamard is applied to this state, the state evolves back to $|0\rangle$, while the reachability redistributed the state and remains $0.5\,|0\rangle + 0.5\,|1\rangle$. Note that a similar situation would arise also in the classical gate if we allow measurements on a non-computational basis. In essence, this is due to the fact that complex amplitudes can destructively interfere while probabilities cannot—one of the core features [54] that is responsible for quantum speedup.

In the quantum gate set scenario, the same definition of M_{circ} still hold, however, it is no longer computable in linear time but can be approximated like algorithmic probability. We obtain the M_{circ} approximated for $L = 3$ for the gate sets $\{$`X`, `CCX`$\}$) and $\{$`H`, `T`, `CX`$\}$ as shown in Figure 6.



(**a**) (**b**)

Figure 6. Approximation of circuit probability of states on 4 qubits for $L = 3$ using two gate sets (**a**) $\{$`X`, `CCX`$\}$ and (**b**) $\{$`H`, `T`, `CX`$\}$.

Using these reachability matrices for the IBM T-topology (in Figure 7) and IBM L-topology (in Figure 8), we can now calculate the M_{circ} for the two device qubit connectivity topology. The circuit probability can thereafter be compared with each other. This gives us insight into which device is better in terms of reachability analysis. The results, as shown in Figure 9, informs that for some transformations, the L-topology is better and has a higher probability and thereby lower circuit complexity (i.e., the red ones in the middle difference plot), while for others, the T-topology is better (i.e., the green ones for which the M_{circ} for T is higher).

The Expressibility plots follow a fractal structure, which is effectively the trace of the subsystem, as

$$E_i^n = \begin{bmatrix} A_i^n & B_i^n \\ B_i^n & A_i^n \end{bmatrix} = \begin{bmatrix} B_{i+1}^n & A_{i-1}^n \\ A_{i-1}^n & B_{i+1}^n \end{bmatrix}, \quad A_i^n = E_i^{n-1}, \quad B_i^n = E_{i-1}^{n-1}. \tag{6}$$

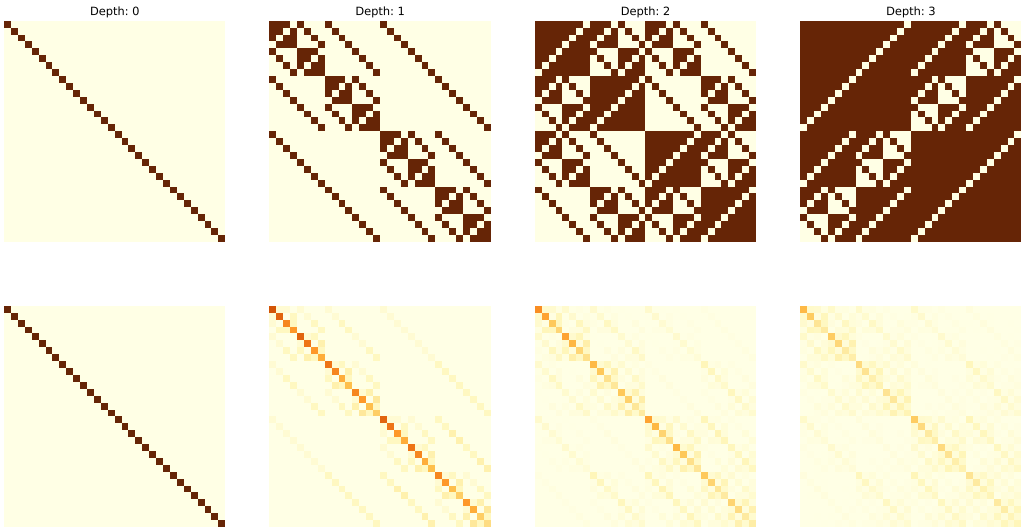


Figure 7. Expressibility and Reachability for gate set $\{\texttt{P}(\mathfrak{B}/4), \texttt{RX}(\mathfrak{B}/2), \texttt{CX}\}$ on 5 qubits and of circuit depth from 0 to 3 on the IBM T-topology.

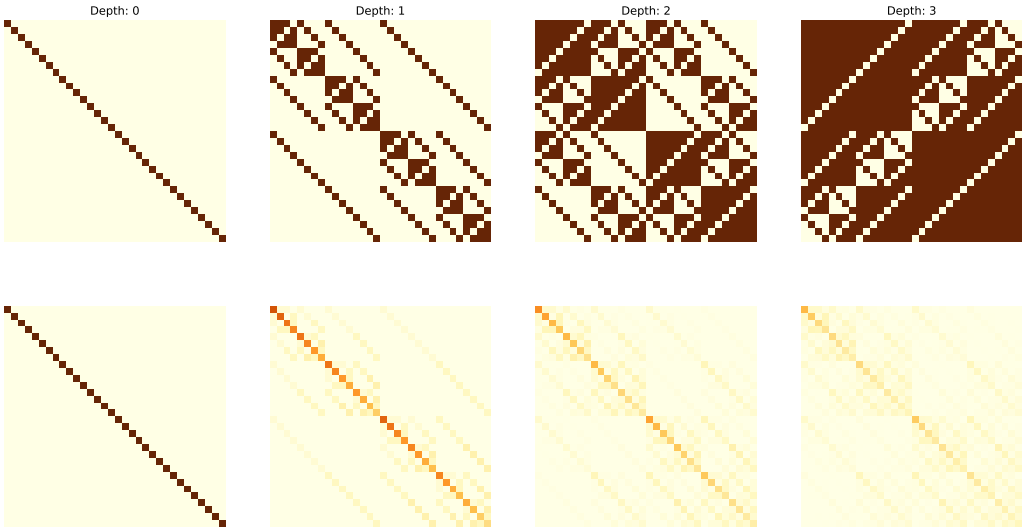


Figure 8. Expressibility and Reachability for gate set $\{\texttt{P}(\mathfrak{B}/4), \texttt{RX}(\mathfrak{B}/2), \texttt{CX}\}$ on 5 qubits and of circuit depth from 0 to 3 on the IBM L-topology.

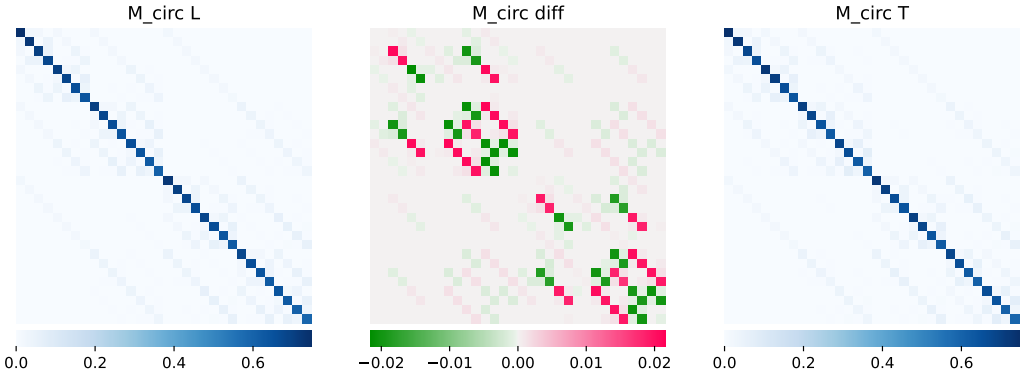


Figure 9. M_{circ} and their comparison for gate set $\{\texttt{P}(\mathfrak{B}/4), \texttt{RX}(\mathfrak{B}/2), \texttt{CX}\}$ on 5 qubits and of circuit depth from 0 to 3 on the IBM L-topology and T-topology.

Another important insight is that the reachability/expressibility analysis is independent of the gate set being strongly universal (e.g., $\{\texttt{H, T, CX}\}$) or not (e.g., $\{\texttt{H, S, CX}\}$). This

is due to our focus on the existence of a path that maps between two states, without considering how much we can control and steer the system towards a specific path. To illustrate the point, a gate set of just {H} or {X} can map between any pair of states (weakly universal) given sufficient depth, yet they clearly cannot approximate even classical ones like {NAND}, let alone all functions. To define universality in this framework, we define functions as a set of transformations between states (e.g., as a sum-of-product expression, probability mass function, or a unitary matrix). A universal gate set that can approximately represent any such transformation given sufficient depth. We leave further discussion on the universal gate set for our ongoing research as an extension of this work.

5. Applications

The application of this research is primarily twofold. On one hand, it is an exploration via the enumeration of the characteristics of Hilbert's space. A visual map of the structures presented in the results would aid in an intuitive understanding of the capabilities of quantum computation. This is an extension of similar projects in classical logic [55] and algorithmic information [56]. On a more pragmatic footing, this research finds applications for various use cases. We conclude this article with a brief description of how this research connects to these use cases.

5.1. Geometric Quantum Machine Learning

The landscape of quantum processes is of interest for both foundational and practical aspects of quantum information. On the foundational side, quantum complexity theory [57,58], quantum resource theories [59], categorical quantum mechanics [60] and quantum formal logic [61] rely on the properties of this landscape. The transition to practical aspects is orchestrated by the geometric formulation of quantum computation [44,62]. Recently, this forms the basis for quantizing geometric deep learning [63]. These works have been conducted in the formalism of mathematical functions or quantum fields [64]. Circuit complexity is much less studied, and can bridge algorithmic complexity and computational complexity. By providing a perspective of the statistical/algorithmic complexity geometry of quantum logic circuits, our intention is to make these results tangible using quantum computational frameworks in the near future. On the other hand, operational distance measures between two quantum states/processes for specific use cases can be informed by these theoretical techniques.

5.2. Novel Quantum Algorithm Synthesis

Quantum algorithm design currently involves careful manipulation of quantum information, harnessing quantum mechanical phenomena (e.g., superposition, entanglement, interference) to a computational advantage. This is generally counter-intuitive to human phenomenological experiences, thus, requiring considerable training and often serendipitous moments [65]. Though the discovery of new algorithms is a buzzing research field [22], reasoning in terms of mathematical formalism has been a barrier to the wider adoption of quantum accelerated computing. Some proposals for automation of quantum programming [23–25] have been proposed to remedy these issues. To further expand the applicability of quantum algorithms, techniques from novelty search [66] and large language models [67] can be incorporated into these automation engines. Open-ended search in the space of quantum processes can greatly benefit from a characterization of this landscape, as presented in this work.

5.3. Quantum Artificial General Intelligence

Among the more rigorous methods of developing general intelligence is an active formulation of Solomonoff's theory of inductive intelligence [36], called universal artificial intelligence [68]. Universal reinforcement learning models like AIXI and KSA are capable of rational decision-making or modeling environmental dynamics, based on the shortest program that corresponds to compressing the past observations and maximizing a set

reward function. These have been quantized both by using quantum algorithms, (e.g., in the AIXI-q model [69]) and by applying them to quantum environments (e.g., in the QKSA model [70]).

Another crucial aspect of intelligence [71] is the understanding of cause-effect relations. Quantum acceleration of causal inference [72–74] can benefit from the knowledge of the probability distribution of causal oracles, a subset of quantum processes that embed specific properties of the problem. Besides causal inference, similar techniques can be applied to other statistical relational learning applications like probabilistic logic networks [75] and quantum variational algorithms.

Both universal distribution and causal inference are intimately connected to the landscape of quantum programs. This landscape inturn depends on the choice of a specific gate set, as we saw in this research. Thereby, novelty seeking in the space of universal gate sets can meta-optimize quantum program synthesis for specific application algorithms. In our current research, we are exploring this direction of second-order cybernetics of automated quantum operational theory, by using the groundwork developed in this article.

Author Contributions: Conceptualization, A.S.; methodology, A.S., A.K. and B.G.B.; software, B.G.B. and A.K.; writing—original draft preparation, A.S., A.K. and B.G.B.; visualization, A.K. and T.A.; supervision, A.S. All authors have read and agreed to the published version of the manuscript.

Funding: A.K. is partially supported by the Polish National Science Center (NCN) under the grant agreement 2019/33/B/ST6/02011. A.S. acknowledges funding from the Dutch Research Council (NWO) through the project "QuTech Part III Application-based research" (project no. 601.QT.001 Part III-C—NISQ).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available on GitHub at DOI: 10.5281/zenodo.7902592 (accessed on 25 April 2023).

Acknowledgments: This project was initiated under the QIntern 2021 project "Reinforcement Learning Agent for Quantum Foundations". B.G.B., T.A., A.S. would like to thank the organizers of the program and QWorld Association.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bertels, K.; Sarkar, A.; Ashraf, I. Quantum computing—From NISQ to PISQ. *IEEE Micro* **2021**, *41*, 24–32. [CrossRef]
2. Bertels, K.; Sarkar, A.; Hubregtsen, T.; Serrao, M.; Mouedenne, A.A.; Yadav, A.; Krol, A.; Ashraf, I. Quantum computer architecture: Towards full-stack quantum accelerators. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1–6.
3. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
4. Leymann, F.; Barzen, J. The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Sci. Technol.* **2020**, *5*, 044007. [CrossRef]
5. Shi, Y.; Gokhale, P.; Murali, P.; Baker, J.M.; Duckering, C.; Ding, Y.; Brown, N.C.; Chamberland, C.; Javadi-Abhari, A.; Cross, A.W.; et al. Resource-efficient quantum computing by breaking abstractions. *Proc. IEEE* **2020**, *108*, 1353–1370. [CrossRef]
6. Landauer, R. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.* **1961**, *5*, 183–191. [CrossRef]
7. Bennett, C.H. Logical reversibility of computation. *IBM J. Res. Dev.* **1973**, *17*, 525–532. [CrossRef]
8. Fredkin, E.; Toffoli, T. Conservative logic. *Int. J. Theor. Phys.* **1982**, *21*, 219–253. [CrossRef]
9. Lloyd, S. Ultimate physical limits to computation. *Nature* **2000**, *406*, 1047–1054. [CrossRef]
10. Markov, I.L. Limits on fundamental limits to computation. *Nature* **2014**, *512*, 147–154. [CrossRef]
11. Wolfram, S. *A New Kind of Science*; Wolfram Media: Champaign, IL, USA, 2002; Volume 5.
12. Deutsch, D. Constructor theory. *Synthese* **2013**, *190*, 4331–4359. [CrossRef]
13. Biamonte, J.; Bergholm, V. Tensor networks in a nutshell. *arXiv* **2017**, arXiv:1708.00006.
14. Hardy, L. Quantum theory from five reasonable axioms. *arXiv* **2001**, arXiv:quant-ph/0101012.
15. Schmidhuber, J. Algorithmic theories of everything. *arXiv* **2000**, arXiv:quant-ph/0011122.
16. Müller, M.P. Law without law: From observer states to physics via algorithmic information theory. *Quantum* **2020**, *4*, 301. [CrossRef]

17. Toffoli, T. Action, or the fungibility of computation. In *Feynman and Computation*; CRC Press: Boca Raton, FL, USA, 2018; pp. 349–392.
18. Lloyd, S.; Dreyer, O. The universal path integral. *Quantum Inf. Process.* **2016**, *15*, 959–967. [CrossRef]
19. Solomonoff, R.J. *A Preliminary Report on a General Theory of Inductive Inference*; Zator Company: Cambridge, MA, USA, 1960.
20. Deutsch, D.E. Quantum computational networks. *Proc. R. Soc. Lond. A Math. Phys. Sci.* **1989**, *425*, 73–90.
21. Maziero, J. Random Sampling of Quantum States: A Survey of Methods: And Some Issues Regarding the Overparametrized Method. *Braz. J. Phys.* **2015**, *45*, 575–583. [CrossRef]
22. Jordan, S. Quantum Algorithm Zoo. Available online: https://quantumalgorithmzoo.org/ (accessed on 25 April 2023).
23. Spector, L. *Automatic Quantum Computer Programming: A Genetic Programming Approach*; Springer Science & Business Media: New York, NY, USA, 2004; Volume 7.
24. Quetschlich, N.; Burgholzer, L.; Wille, R. Towards an Automated Framework for Realizing Quantum Computing Solutions. *arXiv* **2022**, arXiv:2210.14928.
25. Sarkar, A. Automated Quantum Software Engineering: Why? what? how? *arXiv* **2022**, arXiv:2212.00619.
26. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
27. MetaGolfScript—Esolang. Available online: https://esolangs.org/wiki/MetaGolfScript (accessed on 25 April 2023).
28. Jiang, M.; Luo, S.; Fu, S. Channel-state duality. *Phys. Rev. A* **2013**, *87*, 022310. [CrossRef]
29. Complexity Zoo. Available online: https://complexityzoo.net/Complexity_Zoo (accessed on 25 April 2023).
30. Kolmogorov, A.N. Three approaches to the definition of the concept "quantity of information". *Probl. Peredachi Informatsii* **1965**, *1*, 3–11.
31. Allender, E. Circuit Complexity, Kolmogorov Complexity, and Prospects for Lower Bounds. In Proceedings of the 10th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2008, Charlottetown, PE, Canada, 16–18 July 2008; pp. 7–13.
32. Grunwald, P.; Vitányi, P. Shannon information and Kolmogorov complexity. *arXiv* **2004**, arXiv:cs/0410002.
33. Fortnow, L. Kolmogorov complexity and computational complexity. In *Complexity of Computations and Proofs*; Quaderni di Matematica; Aracne: Rome, Italy, 2004; Volume 13.
34. Sarkar, A.; Al-Ars, Z.; Bertels, K. Quantum circuit design for universal distribution using a superposition of classical automata. *arXiv* **2020**, arXiv:2006.00987.
35. Zenil, H.; Badillo, L.; Hernández-Orozco, S.; Hernández-Quiroz, F. Coding-theorem like behaviour and emergence of the universal distribution from resource-bounded algorithmic probability. *Int. J. Parallel Emergent Distrib. Syst.* **2019**, *34*, 161–180. [CrossRef]
36. Solomonoff, R.J. A formal theory of inductive inference. Part I. *Inf. Control* **1964**, *7*, 1–22. [CrossRef]
37. Sarkar, A. Applications of Quantum Computation and Algorithmic Information for Causal Modeling in Genomics and Reinforcement Learning. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2022.
38. Soler-Toscano, F.; Zenil, H.; Delahaye, J.P.; Gauvrit, N. Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines. *PLoS ONE* **2014**, *9*, e96223. [CrossRef]
39. Delahaye, J.P.; Zenil, H. Numerical evaluation of algorithmic complexity for short strings: A glance into the innermost structure of randomness. *Appl. Math. Comput.* **2012**, *219*, 63–77. [CrossRef]
40. Wernick, W. *Complete Sets of Logical Functions*; New York University: New York, NY, USA, 1941.
41. Sheffer, H.M. A set of five independent postulates for Boolean algebras, with application to logical constants. *Trans. Am. Math. Soc.* **1913**, *14*, 481–488. [CrossRef]
42. Shi, Y. Both Toffoli and controlled-NOT need little help to do universal quantum computation. *arXiv* **2002**, arXiv:quant-ph/0205115.
43. Aharonov, D. A simple proof that Toffoli and Hadamard are quantum universal. *arXiv* **2003**, arXiv:quant-ph/0301040.
44. Nielsen, M.A.; Dowling, M.R.; Gu, M.; Doherty, A.C. Quantum computation as geometry. *Science* **2006**, *311*, 1133–1135. [CrossRef] [PubMed]
45. Lloyd, S. Almost any quantum logic gate is universal. *Phys. Rev. Lett.* **1995**, *75*, 346. [CrossRef] [PubMed]
46. Shende, V.V.; Bullock, S.S.; Markov, I.L. Synthesis of quantum logic circuits. In Proceedings of the 2005 Asia and South Pacific Design Automation Conference, Shanghai, China, 18–21 January 2005; pp. 272–275.
47. Vitányi, P.M. Quantum Kolmogorov complexity based on classical descriptions. *IEEE Trans. Inf. Theory* **2001**, *47*, 2464–2479. [CrossRef]
48. Berthiaume, A.; Van Dam, W.; Laplante, S. Quantum kolmogorov complexity. *J. Comput. Syst. Sci.* **2001**, *63*, 201–221. [CrossRef]
49. Mora, C.E.; Briegel, H.J.; Kraus, B. Quantum Kolmogorov complexity and its applications. *Int. J. Quantum Inf.* **2007**, *5*, 729–750. [CrossRef]
50. Dawson, C.M.; Nielsen, M.A. The solovay-kitaev algorithm. *arXiv* **2005**, arXiv:quant-ph/0505030.
51. Barenco, A.; Bennett, C.H.; Cleve, R.; DiVincenzo, D.P.; Margolus, N.; Shor, P.; Sleator, T.; Smolin, J.A.; Weinfurter, H. Elementary gates for quantum computation. *Phys. Rev. A* **1995**, *52*, 3457. [CrossRef]
52. Gottesman, D. The Heisenberg representation of quantum computers. *arXiv* **1998**, arXiv:quant-ph/9807006.
53. Bandić, M.; Almudever, C.G.; Feld, S. Interaction graph-based profiling of quantum benchmarks for improving quantum circuit mapping techniques. *arXiv* **2022**, arXiv:2212.06640.

54. Renou, M.O.; Trillo, D.; Weilenmann, M.; Le, T.P.; Tavakoli, A.; Gisin, N.; Acín, A.; Navascués, M. Quantum theory based on real numbers can be experimentally falsified. *Nature* **2021**, *600*, 625–629. [CrossRef] [PubMed]
55. Demey, L. Metalogic, metalanguage and logical geometry. *Log. Anal.* **2019**, *62*, 453–478.
56. Zenil, H. A Computable Piece of Uncomputable Art whose Expansion May Explain the Universe in Software Space. *arXiv* **2021**, arXiv:2109.08523.
57. Brown, A.R.; Susskind, L. Second law of quantum complexity. *Phys. Rev. D* **2018**, *97*, 086015. [CrossRef]
58. Haferkamp, J.; Faist, P.; Kothakonda, N.B.; Eisert, J.; Yunger Halpern, N. Linear growth of quantum circuit complexity. *Nat. Phys.* **2022**, *18*, 528–532. [CrossRef]
59. Halpern, N.Y.; Kothakonda, N.B.; Haferkamp, J.; Munson, A.; Eisert, J.; Faist, P. Resource theory of quantum uncomplexity. *Phys. Rev. A* **2022**, *106*, 062417. [CrossRef]
60. Selby, J.H.; Scandolo, C.M.; Coecke, B. Reconstructing quantum theory from diagrammatic postulates. *Quantum* **2021**, *5*, 445. [CrossRef]
61. Pratt, V.R. Linear logic for generalized quantum mechanics. In Proceedings of the Workshop on Physics and Computation, Dallas, TX, USA, 2–4 October 1992.
62. Nielsen, M.A.; Dowling, M.R.; Gu, M.; Doherty, A.C. Optimal control, geometry, and quantum computing. *Phys. Rev. A* **2006**, *73*, 062323. [CrossRef]
63. Perrier, E.; Tao, D.; Ferrie, C. Quantum geometric machine learning for quantum circuits and control. *New J. Phys.* **2020**, *22*, 103056. [CrossRef]
64. General, I.J. Principle of maximum caliber and quantum physics. *Phys. Rev. E* **2018**, *98*, 012110. [CrossRef]
65. Shor, P.W. The Early Days of Quantum Computation. *arXiv* **2022**, arXiv:2208.09964.
66. Lehman, J.; Stanley, K.O. Efficiently evolving programs through the search for novelty. In Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, OR, USA, 7–11 July 2010; pp. 837–844.
67. Lehman, J.; Gordon, J.; Jain, S.; Ndousse, K.; Yeh, C.; Stanley, K.O. Evolution through Large Models. *arXiv* **2022**, arXiv:2206.08896.
68. Hutter, M. *Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2004.
69. Catt, E.; Hutter, M. A gentle introduction to quantum computing algorithms with applications to universal prediction. *arXiv* **2020**, arXiv:2005.03137.
70. Sarkar, A.; Al-Ars, Z.; Bertels, K. QKSA: Quantum Knowledge Seeking Agent. In Proceedings of the Artificial General Intelligence: 15th International Conference, AGI 2022, Seattle, WA, USA, 19–22 August 2022; pp. 384–393.
71. Lavin, A.; Zenil, H.; Paige, B.; Krakauer, D.; Gottschlich, J.; Mattson, T.; Anandkumar, A.; Choudry, S.; Rocki, K.; Baydin, A.G.; et al. Simulation intelligence: Towards a new generation of scientific methods. *arXiv* **2021**, arXiv:2112.03235.
72. Sarkar, A.; Al-Ars, Z.; Bertels, K. Estimating algorithmic information using quantum computing for genomics applications. *Appl. Sci.* **2021**, *11*, 2696. [CrossRef]
73. Chiribella, G.; Ebler, D. Quantum speedup in the identification of cause–effect relations. *Nat. Commun.* **2019**, *10*, 1472. [CrossRef]
74. Acharya, T.; Kundu, A.; Sarkar, A. Quantum Accelerated Causal Tomography: Circuit Considerations Towards Applications. *arXiv* **2022**, arXiv:2209.02016.
75. Wittek, P.; Gogolin, C. Quantum enhanced inference in Markov logic networks. *Sci. Rep.* **2017**, *7*, 45672. [CrossRef]