Data Encoding Patterns for Quantum Computing

MANUELA WEIGOLD, JOHANNA BARZEN, FRANK LEYMANN, and MARIE SALM, IAAS, University of Stuttgart

Quantum computers have the potential to solve certain problems faster than classical computers. However, loading data into a quantum computer is not trivial. To load the data, it must be encoded in quantum bits (qubits). There are several ways how qubits can represent the data and, thus, multiple data encodings are possible. Both the data itself and the chosen encoding influence the runtime of the loading process. In the worst case, loading requires exponential time. This is critical because quantum algorithms that promise a speed-up assume that loading data can be done faster, in logarithmic or linear time. To outline abstract knowledge about encodings and the consequences of choosing a particular data encoding, we present three common encodings as patterns. Especially in complex domains like quantum computing, patterns can contribute to making this new technology and its broad potential accessible to users with different backgrounds. In particular, they facilitate the development of quantum applications for software developers.

Categories and Subject Descriptors: [**PLoPourri**]: —*Patterns For Quantum Computing*

General Terms: Algorithms, Measurement

Additional Key Words and Phrases: Quantum Computing, Quantum Algorithms, Data Encoding, Speed-up, Patterns

ACM Reference Format:

M. Weigold, J. Barzen, F. Leymann, and M. Salm. 2020. Data Encoding Patterns for Quantum Algorithms. HILLSIDE Proc. of Conf. on Pattern Lang. of Prog. 22 (October 2020), 11 pages.

1. INTRODUCTION

For decades, the concept of a bit has been the fundamental unit for information encoding in computer science. Recent advances in quantum computing have led to the first commercial quantum computers which operate on quantum bits (qubits) instead of bits [National Academies of Sciences, Engineering and Medicine 2019; LaRose 2019]. Analogous to a bit that can be either 0 or 1, a qubit can also take one of two states: $|0\rangle$ and $|1\rangle$. But in addition, due to quantum mechanics, it can also be in a combination of these two states at once - a *superposition* of states. Empowered by superposition and other fundamental properties of quantum mechanics, quantum computers have the potential to solve certain problems faster than conventional computers [Horodecki et al. 2009]. In fact, various algorithms for quantum computers exist for which a theoretical linear or exponential speed-up over their classical counterparts was demonstrated, e.g. for the factorization of prime numbers [Shor 1999].

As the number of available qubits of quantum computers increases, more companies start to explore quantum computing. However, it is expected that near-term devices will only contain up to a few hundred qubits [Preskill 2018]. Another restricting factor is that these qubits are not perfect: Their states are only stable for a short amount of time. Because of their rapid decay, only a limited number of operations can be executed on them. Thus, successfully programming quantum computers today is limited by the available hardware.

Author's address: M. Weigold, Universitätsstraße 38, 70569 Stuttgart, Germany; email: weigold@iaas.uni-stuttgart.de; J. Barzen, Universitätsstraße 38, 70569 Stuttgart, Germany; email: barzen@iaas.uni-stuttgart.de; F. Leymann, Universitätsstraße 38, 70569 Stuttgart, Germany; email: leymann@iaas.uni-stuttgart.de; M. Salm, Universitätsstraße 38, 70569 Stuttgart, Germany; email: salm@iaas.uni-stuttgart.de

Besides, the sheer fact that quantum computers obey the law of quantum mechanics results in - from the point of a software developer - unusual effects. To illustrate how different quantum computing is, we describe implications for two basic programming tasks: *reading*, and *loading data*. For the first task of reading a qubit, its quantum state must be accessed. This can only be done by measuring it. Unfortunately, measurement causes a qubit to collapse to either $|0\rangle$ and $|1\rangle$. Thus, the state of a qubit that is in superposition can not be accessed for reading.

The second task consists of loading data into a quantum computer. This task is at the beginning of almost every algorithm that processes input data. After the initial loading process, the data is represented by qubits via a specific encoding. Each algorithm expects that a certain data encoding is used, and then processes the data by performing calculations. Unfortunately, loading data can not always be done efficiently. In the worst case, loading requires exponential time. This slows down algorithms with an otherwise logarithmic or linear runtime: With an exponential loading time, their overall runtime is also exponential. This ruins a theoretical linear or exponential speedup of an algorithm - which was one of the reasons why we wanted to use a quantum computer in the first place. In general, the time for loading depends *(i) on the routine that loads the data in a specific encoding* and *(ii) on the data itself*. Thus, loading data is not a trivial task that influences the runtime complexity of a quantum algorithm.

To help software developers understand the implications of using a specific encoding to load data, we formulate three common data encodings as patterns. A pattern in the spirit of Alexander et al. [1977] describes a proven solution to a re-occurring problem. For the development of software, documenting patterns is commonly used to capture knowledge about a specific domain [Coplien 1996; Buschmann et al. 1996]. Especially in an interdisciplinary and complex domain like quantum computing, patterns can be used to make proven solutions explicit, explain 'how' they work, and 'why' a solution (e.g. an encoding) should be used [Meszaros and Doble 1997].

The remainder of this paper is structured as follows: Section 2 describes fundamentals of quantum computing. Section 3 starts with an overview of patterns for quantum algorithms and then presents the new encoding patterns. Related work is discussed in Section 4. Finally, Section 5 concludes the paper and describes future work.

2. FUNDAMENTALS OF QUANTUM COMPUTING

The core concept of quantum computing is the qubit. In this section, we will briefly define qubits and their basic properties that can be used to encode data. Mathematically, the state $|\psi\rangle$ of a qubit is defined as follows:

$$|\psi\rangle = \alpha\left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) + \beta\left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) = \alpha\,|0\rangle + \beta\,|1\rangle \text{ where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{1}$$

In the equation above, we use the Dirac notation that represents a vector v as $|v\rangle$. The two orthogonal basis vectors $\{|0\rangle, |1\rangle\}$ span up a two-dimensional vector space and are also referred to as *computational basis.* The complex numbers α and β are called amplitudes: With a probability of $|\alpha|^2$, measuring the qubit in the computational basis results in the $|0\rangle$ state. Analogously, the qubit can be measured as $|1\rangle$ with a probability of $|\beta|^2$. As these are the only two possible outcomes of the measurement, their probabilities must sum up to 1. For $\alpha, \beta \neq 0$, the state of a qubit is in superposition: a linear combination of $|0\rangle$ and $|1\rangle$.

While the previous equations describes the state of a single qubit, the state of multiple qubits in a quantum computer (a quantum register) can be described in a similar manner. For example, the two qubits $|\psi_1\rangle$ and $|\psi_2\rangle$ can form a 2-qubit quantum register. If both qubits are in state $|0\rangle$, then the state of the register can be written as $|00\rangle$. As each qubit can be $|0\rangle$, $|1\rangle$, or in a superposition, the 2-qubit register can be in the states $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$, or in a superposition of them [Gruska 1999]:

$$|\psi_1\psi_2\rangle = \alpha_0\,|00\rangle + \alpha_1\,|01\rangle + \alpha_2\,|10\rangle + \alpha_3\,|11\rangle\,, \text{ where } \sum_{i=0}^{3} |\alpha_i|^2 = 1. \tag{2}$$

Sometimes the bit strings are transformed into decimal representations resulting in natural numbers, thus, the state vectors are written even more compact as $|0\rangle$, $|1\rangle$, $|2\rangle$, and $|3\rangle$.

In this work, we focus on gate-based quantum computers for which operations on qubits can be done by applying quantum gates. Quantum gates are defined as matrices and their application to one or multiple qubits results in a state that can be calculated by multiplying the matrix with the state vector. For example, applying the X gate (which is defined by the matrix $\left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$) to a qubit in state $|0\rangle$, changes its state to $|1\rangle$:

$$X|0\rangle = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)\left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right) \tag{3}$$

Vice versa, applying the *X* gate to a $|1\rangle$ state leads to the $|0\rangle$ state. In classical computers, this gate corresponds to a *NOT* gate that flips the state of a bit. Unlike their classical counterparts, the application of a quantum gate can also result in superposition, e.g., applying the so-called Hadamard gate to a qubit in state $|0\rangle$, leads to an equal superposition of both $|0\rangle$ and $|1\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)\left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{4}$$

The qubit is measured as $|0\rangle$ and $|1\rangle$ with a probability of $(\frac{1}{\sqrt{2}})^2 = 0.5$. For more examples of quantum gates and an in-depth introduction to their mathematical foundation, we refer to [Nielsen and Chuang 2010].

3. PATTERNS FOR QUANTUM ALGORITHMS

In this section, we first introduce quantum algorithms and give an overview of patterns for quantum computing. This includes existing patterns for quantum algorithms that were introduced by [Leymann 2019] as well as the new encoding patterns. We then describe our pattern format and introduce the new encoding patterns. An excerpt from the encoding patterns can also be found on our website[1].

3.1 Overview of Patterns for Quantum Algorithms

Fig. 1 presents an overview of patterns for quantum algorithms [Leymann 2019] and the new encoding patterns marked in bold. Due to space limitations, QRAM and ANGLE ENCODING are not described further in this paper. The figure also illustrates the typical steps of a *hybrid* quantum algorithm [Leymann et al. 2020]: Some steps are executed on a classical computer (light background), others on a quantum computer (dark background).

First, data is *pre-processed* on a classical computer. This step is required for some data encodings or algorithms.

In the next step, data is loaded into a quantum computer: All qubits are initialized as $|0\rangle$, so the overall quantum state can be denoted as $|00\ldots0\rangle$. A *state preparation routine* operates on a register of qubits and thus, changes their state. As a result, a quantum state is prepared that represents the data via a specific data encoding. This state can have certain characteristics of quantum states that are described by patterns of the quantum states category. For example, in a UNIFORM SUPERPOSITION, all possible outcomes of the quantum register are equally likely. After the state is prepared, the data is loaded. This overall process of preparing the state is summarized in the INITIALIZATION pattern [Leymann 2019] - another alias for it is *State Preparation*.

After state preparation, the quantum computer performs computations on the quantum register. These computations are *unitary transformations* and are represented as quantum gates. Patterns of the category Unitary Transformations describe best practices to construct computations that happen in this step. For example, UNCOMPUTE can be used to reset the state of a quantum register to the ground state $|00\ldots0\rangle$.

In the last step that is executed on the quantum computer, one or multiple qubits are *measured*. The measurement results are analyzed in an optional *post-processing* step. Depending on the results or overall goals of the algorithm, the algorithm terminates or proceeds with the next iteration in the pre-processing step. Program flow patterns capture higher-level strategies to solve a problem on a quantum computer.

[1] `http://quantumcomputingpatterns.org`

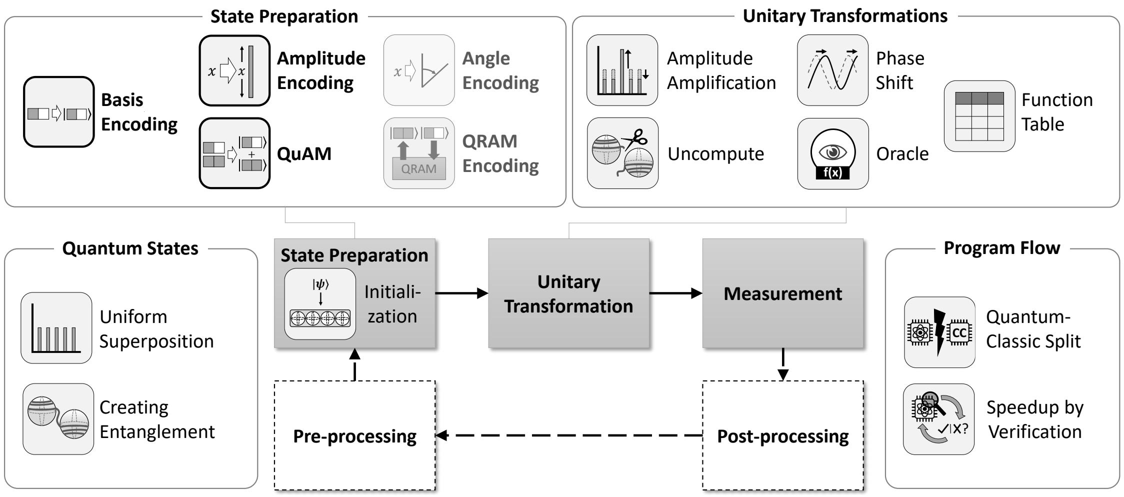


Fig. 1. Overview of pattern for quantum computing. In the center, the steps of a quantum algorithm are shown (based on [Leymann et al. 2020]). The new encoding patterns (highlighted in bold) are part of the first step that is executed on a quantum computer (State Preparation).

3.2 Pattern Format and Method

Pattern authors make use of different pattern formats [Coplien 1996] that define the sections of their pattern documents. We use an existing pattern format of Fehling et al. [2014] and extend it by additional sections for aliases and forces of the pattern. Each pattern is introduced by a descriptive **Name** that is followed by a graphical **Icon**. Next to the icon, we denoted the **Intent** of the pattern: a short sentence that summarizes the pattern. Optionally, other names under which a pattern may be known are listed in the **Alias** section. In the **Context** section, we describe the circumstances that lead to the problem and preconditions for applying the pattern. Considerations and trade-offs that must be taken into account when solving the problem are described in the **Forces** section. The **Solution** is a high-level description of how to solve this problem and is further illustrated in the **Solution Sketch**. The **Result** of the solution discusses the consequences of using this pattern, e.g. the new context that results. If aspects of this pattern can be varied, this is covered in **Variants**. Relations to other quantum computing patterns are described in the **Related Patterns**. Finally, **Known Uses** of the pattern are listed, which is either a concrete implementation of the pattern or a published quantum algorithm that uses this pattern.

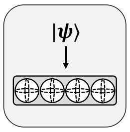
In Section 3.1, we explained that a state preparation routine is used to realize the encoding of data. Therefore, the solution section of our patterns describes (i) how the data is represented, and (ii) the process of encoding data via a suitable state preparation routine. If various state preparation routines can be used to realize one particular encoding, they are referred to in the *Known Uses* section. In this section, we also name concrete examples of algorithms that require this particular data encoding.

Patterns are not invented but abstracted from real-world solutions [Kohls 2010]. In quantum computing, algorithms for quantum computers have been published for decades before the first quantum computer was realized. Concrete software implementations therefore often refer to a quantum algorithm that is described in a scientific publication. These publications also contain the underlying idea of the algorithm - the abstract solution that we want to capture in a pattern. In contrast to that, typical software patterns are abstracted only from existing implementations [Fehling et al. 2014; Fehling et al. 2015]. For our pattern research, we followed the method described by [Fehling et al. 2014]. To identify patterns in the domain of quantum algorithms, we analyzed research papers, books, and technical documentation. During this phase, we collected reoccurring solutions and pattern ideas. If we identified at least three references per pattern idea (Coplien's Rule of Three [Coplien 1996]), we abstracted the underlying solution from the references and authored a pattern.

3.3 Data Encoding Patterns

In this section, we present data encoding patterns for quantum algorithms. Each pattern describes how input data is loaded in a specific encoding at the beginning of a quantum algorithm. Because data is loaded during the INITIALIZATION step (see Fig. 1), each encoding pattern further refines this pattern. Therefore, we start by giving a short summary of INITIALIZATION [Leymann 2019] and then extend the original pattern by a detailed description of the forces. As these are also the forces of the new encoding patterns, we omit this section in the encoding patterns. We first present the simplest encoding, BASIS ENCODING, and then introduce QuAM (QUANTUM ASSOCIATIVE MEMORY), and AMPLITUDE ENCODING. Historically, solving quantum physical problems was in the foreground of quantum computing, thus, input data for quantum algorithms is often numeric. Therefore, we assume in the context of each pattern that the input data X is numeric.

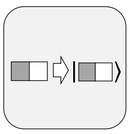
INITIALIZATION



Summary: At the beginning of an algorithm, its initial state is prepared: First, the registers are initialized in the $|0\dots0\rangle$ state. If input data is used by the algorithm, a suitable state preparation routine encodes the data via a specific encoding.

Forces. Encoding data in qubits is not trivial. Current devices contain a limited amount of qubits that are stable for a short amount of time. In order to make use of current devices, the representation must be compact and use only a few qubits and few quantum gates. Because qubits decay fast and quantum gates are error-prone too, the number of operations to prepare the quantum state must be small. To encode even a large number of data values efficiently, a logarithmic or linear runtime is ideal, i.e., the state preparation routine consists of a logarithmic or linear number of parallel operations. Each encoding is essentially a trade-off between two major forces: (i) the number of required qubits and (ii) the runtime complexity for the loading process. Besides that, an additional force requires that data must be represented in a suitable format for further operations. For arithmetic operations like addition or multiplication often the exact values of the data need to be represented. For other operations it may be sufficient to represent their relative values (e.g., as relatively small or large amplitude of a quantum state with AMPLITUDE ENCODING).

BASIS ENCODING



Represent a data element in a quantum computer in order to perform calculations

Context. A quantum algorithm requires numerical input data X for further calculations.

Solution. The main idea for this encoding is to use the *computational basis* $\{|0...00\rangle, |0...01\rangle, \dots, |1...11\rangle\}$ to encode the input data: An input number x is approximated by a binary format $x := b_{n-1}\dots b_1 b_0$ which is then turned into the corresponding basis vector $|x\rangle := |b_{n-1}\dots b_1 b_0\rangle$. For example, the number "2" is represented as 10 which is then encoded by $|10\rangle$ (Fig. 2). In general, this leads to the following encoding: $X \approx \sum_{i=-k}^{m} b_i 2^i \mapsto |b_m \dots b_{-k}\rangle$ where X is first approximated with a precision of k+m significant digits and then represented by a basis vector.

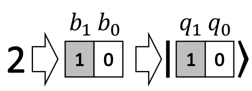


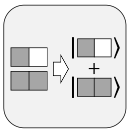
Fig. 2. Basis encoding. A number is approximated by a binary bit string (first step) and encoded by a computational basis state (second step) .

Result. This encoding can be categorized as digital encoding because it is suitable for arithmetic computations [Leymann and Barzen 2020b]. For input numbers that are approximated by l digits, l qubits are needed for its representation. To realize this encoding, the initial $|0\rangle$ state of qubits that represent a '1' digit must be flipped into $|1\rangle$. For one qubit, this can be done by a single operation, and thus, this encoding can be prepared in linear time.

Related Pattern. This pattern is a refinement of INITIALIZATION. If an algorithm requires several numbers as input, each can be encoded in BASIS ENCODING which can be processed by the QUAM pattern.

Known Uses. Vedral et al. [1996] give multiple examples for algorithms that perform arithmetic operations on numbers in BASIS ENCODING. A formal description of the solution above is also given in [Leymann and Barzen 2020b] and [Cortese and Braje 2018]. As only one quantum gate is needed to obtain this encoding, this state preparation routine can be implemented straightforwardly.

QUAM (QUANTUM ASSOCIATIVE MEMORY)



Represent a collection of data elements in a quantum computer in order to perform calculations

Context. A quantum algorithm requires multiple numerical values X as input for further calculations.

Solution. Use a *quantum associative memory* (QuAM) to prepare a superposition of basis encoded values in the same qubit register [Leymann and Barzen 2020b]. In Fig. 3 this is illustrated for the three values x_1, x_2 and x_3 in binary format. Note that the resulting encoding is an equally weighted superposition of the basis encoded values, i.e., all amplitudes are of the same magnitude.

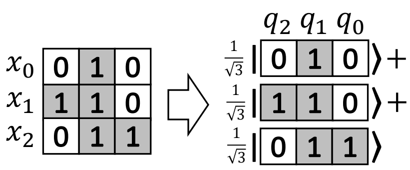


Fig. 3. Resulting Encoding. Each data value represented by a row on the left is encoded in BASIS ENCODING and an amplitude of $\frac{1}{\sqrt{n}}$.

To load the data, the register of the quantum associative memory is in superposition of two states, a *processing* and a *memory* branch (Fig. 4). Both branches have a load and a storage part. An additional element is first prepared into the load part of both branches (step 1). Next, the processing branch is split in such a manner, that the new element gets a proper amplitude (step 2) such that it can be stored by bringing it into superposition with the already added elements (step 3). Finally, an UNCOMPUTE cleans the both branches to be ready for the next iteration. See [Ventura and Martinez 2000] for a more detailed description of the individual steps.

Result. The resulting encoding is a digital encoding and therefore suitable for arithmetic computations [Leymann and Barzen 2020b]. For input n numbers that are approximated by l digits, l qubits are needed for this representation. Each of the n encoded input values is represented by a basis vector with an amplitude of $\frac{1}{\sqrt{n}}$. All other $2^l - n$ amplitudes of the register are zero - in our example, $|000\rangle, |001\rangle, |100\rangle, |101\rangle$, and $|111\rangle$. The amplitude vector is therefore often sparse for this encoding [Schuld and Petruccione 2018].

Related Pattern. This pattern refines INITIALIZATION and makes use of UNCOMPUTE. UNIFORM SUPERPOSITION creates a superposition of all computational basis states. Each of the computational basis states also represents a value in BASIS ENCODING.

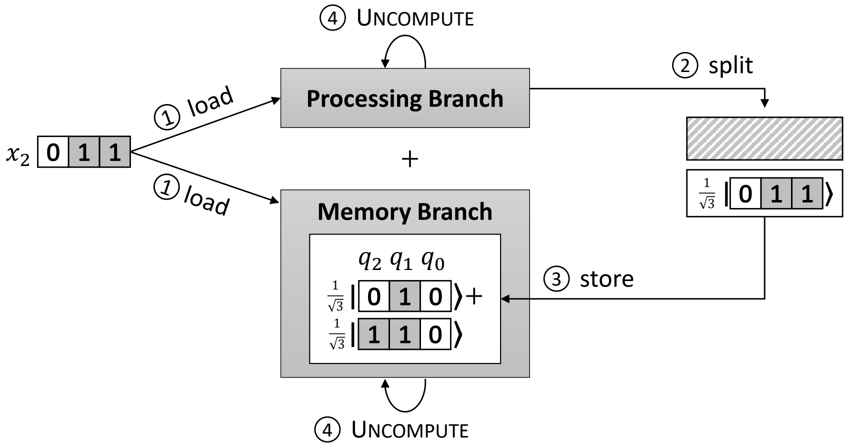
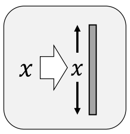


Fig. 4. Illustration of the state preparation routine of [Ventura and Martinez 2000]. In each iteration, an element is loaded and brought into superposition with the already stored elements.

Known Uses. The presented state preparation routine based on Ventura and Martinez [2000] can be used whenever multiple data values need to be represented in BASIS ENCODING. Shor's algorithm [Shor 1999] for the factorization of prime numbers, a quantum version of the Fourier transform [Coppersmith 2002], and Grover's algorithm [Grover 1996] for unstructured search rely on this encoding. Various algorithms extend or use Grover's algorithm and therefore also make use of this encoding.

AMPLITUDE ENCODING



Encode data in a compact manner that do not require calculations

Alias. This encoding has also been referred to as Wavefunction Encoding by LaRose and Coyle [2020]. Every quantum system is described by its wavefunction ψ which also defines the measurement probabilities. By expressing that the wavefunction is used to encode data, it is therefore implied that amplitudes of the quantum system are used to represent data values.

Context. A numerical input data vector $(x_0, \ldots, x_{n-1})^T$ must be encoded for an algorithm.

Solution. Use amplitudes to encode the data. As the squared moduli of the amplitudes of a quantum state must sum up to 1, the input vector needs to be normalized to length 1. This is illustrated in Fig. 5 for a 2-dimensional input vector that contains two data points. To associate each amplitude with a component of the input vector, the dimension of the vector must be equal to a power of two because the vector space of an n qubit register has dimension 2^n. If this is not the case, the input vector can be padded with additional zeros to increase the dimension of it. Using a suitable state preparation routine (see *Known Uses*), the input vector is encoded in the amplitudes of the quantum state as follows: $|\psi\rangle = \sum_{i=0}^{n-1} x_i |i\rangle$. As the amplitudes depend on the data, the process of encoding the data (but not the encoding itself) is often referred to as arbitrary state preparation.

Result. A data input vector of length l can be represented by $\lceil log_2(l) \rceil$ qubits - this is indeed a very compact representation. For an arbitrary state represented by n qubits (which represents 2^n data values), it is known that at least $\frac{1}{n}2^n$ parallel operations are needed [Schuld and Petruccione 2018]. Current state preparation routines perform slightly better than 2^n operations [Schuld and Petruccione 2018]. However, depending on the data it may

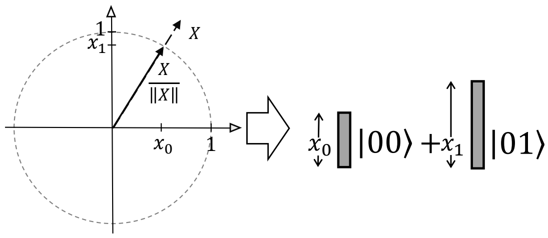


Fig. 5. Amplitude Encoding for 3 data points. The input vector (left) is normalized and represented by the amplitudes in the resulting encoding.
.

still be possible to realize an encoding in a logarithmic runtime. For example, a UNIFORM SUPERPOSITION can be created by applying a Hadamard gate to each of the n qubits - which can be done in parallel and thus in a single step. This represents a 2^n-dimensional vector in which all data entries are $\frac{1}{\sqrt{n}}$. Similarly, sparse data vectors can also be prepared more efficiently [Schuld and Petruccione 2018].

It must be noted that if the output is also encoded in the amplitude, multiple measurements must be taken to obtain a good estimate of the output result. The number of measurements scales with the number of amplitudes - as n qubits contain 2^n amplitudes, this is costly [Schuld and Petruccione 2018].

***Related Patterns**.* This pattern refines INITIALIZATION. The encoding is more compact (in terms of qubits) than BASIS, ANGLE or QRAM ENCODING.

***Known Uses**.* AMPLITUDE ENCODING is required by many quantum machine learning algorithms [LaRose and Coyle 2020]. Another example is the algorithm of Harrow, Hassidim and Lloyd [Harrow et al. 2009] (often referred to as HHL algorithm) for solving linear equations. The pre-condition that the data values can be normalized is a common assumption in machine learning [Duarte and Ståhl 2019], e.g. in support vector machine.

There are various ways to construct a state preparation routine for this encoding. For example, Plesch and Brukner [2011] and Iten et al. [2016] use the Schmidt Decomposition. For the latter, an implementation in Mathematica was presented [Iten et al. 2019]. Shende et al. [2006] presented an alternative way to construct an arbitrary quantum state which was implemented by Qiskit [Qis 2020]. PennyLane offers a loading routine for AMPLITUDE ENCODING [Pen 2020]. The library also includes an arbitrary state preparation routine that uses the algorithm proposed by Möttönen and Vartiainen [2005]. The state preparation routine by Möttönen and Vartiainen [2005] requires an exponential number of operations to encode 2^n data values. Q# provides functionality to compute a state preparation routine that approximates the desired amplitude encoding [QSh 2020].

4. RELATED WORK

Our patterns are based on the concept of patterns by Alexander et al. [1977] who introduced patterns for documenting best practices in the domain of buildings. Since then, the concept has been adapted by various other areas and is especially popular for the domain of software [Coplien 1996]. Leymann [2019] already presented patterns for quantum algorithms that we reviewed in Section 3. In this work, we extend the brief pattern format that was used by Leymann [2019] and present three patterns for the encoding of data. To our knowledge, no other patterns for the domain of quantum computing exist.

Perdrix [2007] introduces *quantum patterns and types* that are part of a formal quantum programming language. But these are not patterns in the sense of Alexander et al. [1977] as they only reflect technical details instead of describing a problem or context.

Several authors discussed the process of loading data into a quantum computer and the implications on runtime. Biamonte et al. [2017] refer to it as *input problem* as data can not always be loaded efficiently. Aaronson [2015] examines loading data for the HHL algorithm for solving linear equations. He points out that the logarithmic runtime for this algorithm can only be achieved if the AMPLITUDE ENCODING of the data can be prepared in logarithmic time. He concludes that this is a general drawback for algorithms that use this encoding, which we also emphasize in our pattern for this encoding.

Salm et al. [2020] consider given input data to support the selection of concrete quantum algorithm implementations and suitable quantum computers for execution. Thereby, they are estimating the required number of qubits and sequentially executable gates of an implementation depending on the size of the input data.

Yan et al. [2016] review different quantum representations for quantum image processing. In particular, BASIS ENCODING and ANGLE ENCODING are used in various representations. They outline similarities, applications, and drawbacks of the representations but do not draw general conclusions for data encodings.

Schuld and Petruccione [2018] as well as LaRose and Coyle [2020] define various data encodings for quantum computing. We refer to these definitions in our data encoding patterns and visualize them in greater detail. LaRose and Coyle [2020] also compare data encodings in the context of classification with quantum computers. They show that in a noiseless setting, different data encodings lead to different decision boundaries that can be learned by a quantum classifier. While they discuss the findings for quantum classifier, they do not consider implications for data encodings in general. In particular, LaRose and Coyle [2020] do not consider BASIS ENCODING as these are not common for quantum classifiers.

Schuld and Killoran [2019] point out how data encodings and kernels in machine learning are related. They show that an input encoding (that maps a numerical input value into the high dimensional vector space of a quantum system) defines a quantum kernel. They refer to a specific encoding as a *quantum feature map* ϕ and point out that different encodings lead to different values of the inner product between the encoded data values. Very recently, there is active research about learning suitable data encodings for quantum machine learning [LaRose and Coyle 2020; Lloyd et al. 2020]. Here, we depict a more general view on encodings and do not focus on machine learning.

5. CONCLUSION AND FUTURE WORK

In this paper, we formulated three common data encoding as patterns. In order to explain 'how' the encoding is realized, we described and visualized it with a sketch. In the result section, we outlined consequences (required qubits, runtime of the encoding process, etc.) and thus elaborate 'why' a particular encoding should be chosen. We conclude that there is not 'the' best encoding for quantum computation addressing different problems on current devices. If arithmetic computations shall be performed, a digital encoding (e.g., BASIS ENCODING or QUAM) may be preferred. To store as much data as possible in a small number of qubits, a compact encoding like AMPLITUDE ENCODING may be the best choice. However, it must also be taken into account that the state preparation for AMPLITUDE ENCODING is costly in terms of operations. Other encodings (e.g. ANGLE ENCODING) exists for which state preparation can be done by only few operations, but which are not optimal in the number of required qubits.

We plan to collect more patterns for quantum computing. We will investigate other encodings mentioned in the literature [Leymann and Barzen 2020b; LaRose and Coyle 2020; Schuld and Petruccione 2018]. In addition, we are extending our pattern repository[2] [Leymann and Barzen 2020a; Weigold et al. 2020] to support quantum computing patterns by including mathematical formulas and quantum gates. The presented encoding patterns will contribute to improve the estimation of required quantum resources for quantum algorithm implementations in the future [Salm et al. 2020].

ACKNOWLEDGMENT

We thank our shepherd Dana (Peng) Zhang as well as the members of our writing groups for helpful comments and suggestions. We would also like to thank Daniel Vietz, Benjamin Weder, and Karoline Wild for discussions about quantum computing and patterns. This work is partially funded by the BMWi project PlanQK (01MK20005N).

[2] https://github.com/PatternAtlas/pattern-atlas-docker

REFERENCES

2020. Q# API reference. https://docs.microsoft.com/en-us/qsharp/api/. (2020).

2020. Summary of Quantum Operations. https://qiskit.org/documentation/tutorials/circuits/3_summary_of_quantum_operations.html. (2020).

2020. Templates. https://pennylane.readthedocs.io/en/stable/introduction/templates.html. (2020).

Scott Aaronson. 2015. Read the fine print. *Nature Physics* 11 (04 2015), 291–293. DOI:http://dx.doi.org/10.1038/nphys3272

Christopher Alexander, Sara Ishikawa, and Murray Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.

Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. 2017. Quantum machine learning. *Nature* 549, 7671 (2017), 195–202. DOI:http://dx.doi.org/10.1038/nature23474

Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal. 1996. *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Wiley.

James O. Coplien. 1996. *Software Patterns*. SIGS Books & Multimedia.

Don Coppersmith. 2002. An approximate Fourier transform useful in quantum factoring. *arXiv:quant-ph/0201067* (2002).

John A. Cortese and Timothy M. Braje. 2018. Loading classical data into a quantum computer. *arXiv:1803.01958* (2018).

Denio Duarte and Niclas Ståhl. 2019. Machine Learning: A Concise Overview. In *Data Science in Practice*, Alan Said and Vicenç Torra (Eds.). Springer International Publishing, 27–58. DOI:http://dx.doi.org/10.1007/978-3-319-97556-6_3

Christoph Fehling, Johanna Barzen, Uwe Breitenbücher, and Frank Leymann. 2014. A Process for Pattern Identification, Authoring, and Application. In *Proceedings of the 19th European Conference on Pattern Languages of Programs (EuroPLoP 2014)*. ACM.

Christoph Fehling, Johanna Barzen, Michael Falkenthal, and Frank Leymann. 2015. PatternPedia – Collaborative Pattern Identification and Authoring. In *Proceedings of PURPLSOC (Pursuit of Pattern Languages for Societal Change). The Workshop 2014.* 252–284.

Christoph Fehling, Frank Leymann, Ralph Retter, Walter Schupeck, and Peter Arbitter. 2014. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer. 367 pages.

Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96* (1996). DOI:http://dx.doi.org/10.1145/237814.237866

Jozef Gruska. 1999. *Quantum computing*. Vol. 2005. Citeseer.

Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical review letters* 103, 15 (2009), 150502.

Ryszard Horodecki, Paweł Horodecki, Michał Horodecki, and Karol Horodecki. 2009. Quantum entanglement. *Reviews of modern physics* 81, 2 (2009), 865.

Raban Iten, Roger Colbeck, Ivan Kukuljan, Jonathan Home, and Matthias Christandl. 2016. Quantum circuits for isometries. *Physical Review A* 93, 3 (2016), 032318.

Raban Iten, Oliver Reardon-Smith, Luca Mondada, Ethan Redmond, Ravjot Singh Kohli, and Roger Colbeck. 2019. Introduction to UniversalQ-Compiler. *arXiv preprint arXiv:1904.01072* (2019).

Christian Kohls. 2010. The Structure of Patterns. In *Proceedings of the 17th Conference on Pattern Languages of Programs (PLOP '10)*. ACM, New York, NY, USA, 12:1–12:10. DOI:http://dx.doi.org/10.1145/2493288.2493300

Ryan LaRose. 2019. Overview and Comparison of Gate Level Quantum Software Platforms. *Quantum* 3 (March 2019), 130. DOI:http://dx.doi.org/10.22331/q-2019-03-25-130

Ryan LaRose and Brian Coyle. 2020. Robust data encodings for quantum classifiers. *arXiv:2003.01695* (2020).

Frank Leymann. 2019. Towards a Pattern Language for Quantum Algorithms. In *Quantum Technology and Optimization Problems (Lecture Notes in Computer Science (LNCS))*, Vol. 11413. Springer International Publishing, 218–230. DOI:http://dx.doi.org/10.1007/978-3-030-14082-3_19

Frank Leymann and Johanna Barzen. 2020a. Pattern Atlas. *arXiv:2006.05120* (2020).

Frank Leymann and Johanna Barzen. 2020b. The bitter truth about gate-based quantum algorithms in the NISQ era. *Quantum Science and Technology* (Sept. 2020), 1–28. https://doi.org/10.1088/2058-9565/abae7d

Frank Leymann, Johanna Barzen, Michael Falkenthal, Daniel Vietz, Benjamin Weder, and Karoline Wild. 2020. Quantum in the Cloud: Application Potentials and Research Opportunities. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER 2020)*. SciTePress, 9–24.

Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. 2020. Quantum embeddings for machine learning. *arXiv:2001.03622* (2020).

Gerard Meszaros and Jim Doble. 1997. Pattern Languages of Program Design 3. Addison-Wesley, Chapter A Pattern Language for Pattern Writing, 529–574.

Mikko Möttönen and Juha J. Vartiainen. 2005. Decompositions of general quantum gates. *arXiv:quant-ph/0504100* (2005).

National Academies of Sciences, Engineering and Medicine. 2019. *Quantum Computing: Progress and Prospects*. The National Academies Press, Washington, DC. DOI:http://dx.doi.org/10.17226/25196

Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum computation and quantum information*. Cambridge University Press, Cambridge and New York. DOI:http://dx.doi.org/10.1017/CBO9780511976667

Simon Perdrix. 2007. Quantum Patterns and Types for Entanglement and Separability. *Electron. Notes Theor. Comput. Sci.* 170 (March 2007), 125–138. DOI:http://dx.doi.org/10.1016/j.entcs.2006.12.015

Martin Plesch and Časlav Brukner. 2011. Quantum-state preparation with universal gate decompositions. *Physical Review A* 83, 3 (2011). DOI:http://dx.doi.org/10.1103/PhysRevA.83.032302

John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. DOI:http://dx.doi.org/10.22331/q-2018-08-06-79

Marie Salm, Johanna Barzen, Uwe Breitenbücher, Frank Leymann, Benjamin Weder, and others. 2020. The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. *Communications in Computer and Information Science (CCIS)* (2020), 66–85. DOI:http://dx.doi.org/10.1007/978-3-030-64846-6_5

Maria Schuld and Nathan Killoran. 2019. Quantum Machine Learning in Feature Hilbert Spaces. *Phys. Rev. Lett.* 122 (Feb 2019), 040504. Issue 4. DOI:http://dx.doi.org/10.1103/PhysRevLett.122.040504

Maria Schuld and Francesco Petruccione. 2018. *Supervised Learning with Quantum Computers*. Springer International Publishing. DOI:http://dx.doi.org/10.1007/978-3-319-96424-9

Vivek V Shende, Stephen S Bullock, and Igor L Markov. 2006. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 6 (2006), 1000–1010.

Peter W Shor. 1999. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review* 41, 2 (1999), 303–332.

Vlatko Vedral, Adriano Barenco, and Artur Ekert. 1996. Quantum networks for elementary arithmetic operations. *Phys. Rev. A* 54 (Jul 1996), 147–153. Issue 1. DOI:http://dx.doi.org/10.1103/PhysRevA.54.147

Dan Ventura and Tony Martinez. 2000. Quantum associative memory. *Information Sciences* 124, 1-4 (2000), 273–296.

Manuela Weigold, Johanna Barzen, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Karoline Wild. 2020. Pattern Views: Concept and Tooling for Interconnected Pattern Languages. *Communications in Computer and Information Science (CCIS)* (2020), to appear.

Fei Yan, Abdullah M. Iliyasu, and Salvador E. Venegas-Andraca. 2016. A survey of quantum image representations. *Quantum Information Processing* 15, 1 (2016), 1–35. DOI:http://dx.doi.org/10.1007/s11128-015-1195-6

Received July 2020; revised October 2020; accepted February 2021