Lecture

Quantum Computing

(CS5070)

Quantum Data Encoding Patterns

Professor Dr. rer. nat. habil. Sven Groppe

https://www.ifis.uni-luebeck.de/index.php?id=groppe
Motivation Design Patterns

- **Software design patterns**
  - structured document containing an abstract description of a proven solution of a recurring problem
  - refers to other patterns that may jointly contribute to an encompassing solution of a complex problem
  - network of related patterns, i.e. a pattern language
  - **Example:** Singleton, Definition: [Link 1](#) [Link 2](#)
    - creational design pattern that lets you ensure that a class has only one instance, while providing a global access point to this instance
    - **Solution**
      - Make the default constructor private, to prevent other objects from using the new operator with the Singleton class
      - Create a static creation method that acts as a constructor. Under the hood, this method calls the private constructor to create an object and saves it in a static field. All following calls to this method return the cached object.

- **Quantum Patterns**
  - Design patterns using quantum algorithms
Pattern Format

- differs depending on the domain
- **Format example:**
  - **Name and icon** that serves as a graphical representation of the pattern
  - **Intent** that briefly summarizes the purpose of the pattern
  - **Alias**: other used names of the pattern
  - **Context**: problem and the circumstances of the pattern
  - **Forces**: trade-offs or considerations that must be taken into account for solving the problem
  - **Solution**: description in an abstract manner and often visualized by a solution sketch
  - **Result**: consequences of the solution
  - optional section for **variants** of the pattern
  - **Related patterns**: Connections between patterns, as patterns are often applied in combination or solve similar problems
  - **Known uses** of the pattern in quantum algorithms and concrete implementations
Typical Structure of a Quantum Calculation

- Typical phases of quantum calculations:

```
\begin{align*}
\text{qubit registers} & \quad \left\{ \begin{array}{c}
|0\rangle \\
|0\rangle \\
|0\rangle \\
|0\rangle \\
|0\rangle \\
|0\rangle \\
\end{array} \right. \\
\text{State Preparation} & \quad \cdots \quad \cdots \\
\text{Unitary Transformation} & \quad \cdots \quad \cdots \\
\text{depth} & \\
\text{width} & 
\end{align*}
```
Typical Structure of a Quantum Calculation

- Typical phases of quantum calculations:

  - How to prepare the states for data loading?
Initialization aka State Preparation 1/2

- **Intent**
  - Initialize the input of a quantum register, taking into account the prerequisites of the subsequent steps of the algorithm

- **Context**
  - Specific parameters must be typically given as input data to a quantum algorithm in order to solve a given problem
  - The first unitary transformations of a quantum circuit typically encode the input data into the quantum register according to a defined encoding

- **Solution**
  - $|0\cdots0\rangle$ and $|0\cdots01\rangle$ are frequently used as initialization of quantum registers
  - Some qubits as ancilla bits, which may be used for the storage of intermediate results or quantum error correction
  - Example
    - for a function table of a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, the overall register is initialized as $|0\rangle^\otimes n |0\rangle^\otimes m$ (including $m$ ancilla bits)
Initialization aka State Preparation 2/2

- **Result**
  - Preparing more advanced states built on the previously described initialization techniques
  - Examples of loading into quantum registers
    - Loading classical bits
    - Loading of complex vectors
    - Loading of real-valued vectors
    - Loading of matrices that are represented as sets of vectors

- **Related Patterns**
  - Uniform superposition: \[ H_n(|0 \cdots 0\rangle) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle \]
  - Refinements of initialization:
    - Initialized register may be used to compute a function table

For other patterns we do not present all items...
Data Encoding

- define how data is represented by the state of a quantum system
- **Data encoding patterns** describe a particular encoding as a trade-off between:
  - Minimize #qubits needed for the encoding
    - because current devices are of intermediate size and thus only support a limited #qubits
  - Minimize depth of quantum circuit needed to realize the encoding
    - the loading routine is ideally of constant or logarithmic complexity
  - Data must be represented in a suitable manner for further calculations, e.g., arithmetic operations

- **Patterns for Data Encoding:**

![Pattern Images]
Basis Encoding

- **Intent**
  - Represent data in a quantum computer for performing calculations

- **Context**
  - Quantum algorithm requires numerical input data $X$ for further calculations

- **Solution**
  - use the computational basis $|0...00\rangle, |0...01\rangle, \ldots, |1...11\rangle$
    - For example: decimal 2 in binary format 10 encoded by $|10\rangle$
    - Classical data: integer number $x := b_{n-1} \ldots b_1 b_0$ with $n$ bits $b_i$
    - Corresponding quantum data: $|x\rangle := |b_{n-1} \ldots b_1 b_0\rangle$

- **Result**
  - suitable for arithmetic computations $[LB'20][VBE'96][CB'18] \Rightarrow$ digital encoding
  - Space requirements: $n$ qubits
  - Initial $|0\rangle$ state of qubits that represent a 1 bit must be flipped into $|1\rangle$
    $\Rightarrow O(n)$ NOT-gates in parallel $\Rightarrow O(1)$ preparation time
Quantum Associative Memory (QuAM)

- **Context**
  - A quantum algorithm requires multiple numerical values $X := \{x_1, \ldots, x_k\}$ as input for further calculations.

- **Solution**
  - Use a quantum associative memory (QuAM) [VM'00] to prepare a superposition of basis encoded values in the same qubit register [LB'20].

  $\text{quantum register is an equally weighted superposition } \frac{1}{\sqrt{k}} \sum_{i=1}^{k} |x_i\rangle$ of all basis encoded values $|x_1\rangle, \ldots, |x_k\rangle$

  Basic steps [VM'00] (using modified parts of Grover):
  
  1. Additional element is prepared in both branches.
  2. Processing branch is split in such a manner, that the new element gets a proper amplitude such that
  3. it can be brought into superposition with the already added elements.
  4. Uncompute cleans for the next iteration.
Quantum Associative Memory (QuAM)

- **Result**
  - digital encoding and therefore suitable for arithmetic computations [LB'20]
  - **Space:** For $k$ input numbers each with $n$ bits, $n$ qubits are needed
  - Each of encoded input values is represented by a basis vector with an amplitude of $\frac{1}{\sqrt{k}}$ and all other amplitudes of the register are 0
    $\Rightarrow$ The amplitude vector therefore often sparse [SP'18]
  - Preparation time depends on number $k$ of input numbers
Angle Encoding 1/3

- **Intent**
  - "Represent each data point by a separate qubit" [WBLS'21]

- **Alias**
  - **Qubit Encoding**: since each qubit represents a single data point [LC'20]
  - **(Tensor) Product Encoding**: since the resulting encoding of this pattern is not entangled [LB'20]

- **Context**
  - Encoding is efficient in terms of operations to perform more operations within the decoherence time of noisy intermediate-scale quantum computers (NISQ) after encoding the data
Angle Encoding 2/3

- **Solution [LB'20]**
  1. Each data point of the input is normalized to the interval $[0, 2 \cdot \pi]$ (which ensures an injective encoding, i.e., $\forall a, b : f(a) = f(b) \Rightarrow a = b$, because rotation gates of the form $R_{\{x,y,z\}}(2 \cdot x_i)$ are periodic with a period of $2 \cdot \pi$)
  2. **Encoding** the data points by rotating around the y-axis with an angle depending on the value of the normalized data point

$$
\begin{align*}
|0\rangle \rightarrow R_y(2 \cdot x_0) \begin{bmatrix} \cos(x_0) \\ \sin(x_0) \end{bmatrix} & \equiv \begin{bmatrix} \cos(x_0) \\ \sin(x_0) \end{bmatrix} \otimes \begin{bmatrix} \cos(x_1) \\ \sin(x_1) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos(x_n) \\ \sin(x_n) \end{bmatrix} \\
|0\rangle \rightarrow R_y(2 \cdot x_1) & \vdots \\
|0\rangle \rightarrow R_y(2 \cdot x_n) & \vdots
\end{align*}
$$
Angle Encoding 3/3

- **Result**
  - Space requirements: \( n + 1 \) qubits for \( n + 1 \) data points
  - Initial \( |0\rangle \) state of qubits that represent a \( 1 \) data point must be rotated according to data point
  \[ \Rightarrow O(n) \text{ Rotation-gates in parallel} \Rightarrow O(1) \text{ preparation time} \]

- **Variants**
  - [LC'20] proposes to exploit the relative phase for a more dense encoding which requires only half of the qubits for the same amount of data points

- **Known Uses**
  - Classification algorithms [LC'20,G+'18] based on angle encoding
  - Quantum image processing [YIV'15]: angle encoding for a pixel's color information in the flexible representation of quantum image (FRQI) and an additional register for the position
  - In quantum neural networks [SSP14] quantum neurons (quron) use angle encoding
Amplitude Encoding

- **Intent**
  - Encode *data in a compact manner that do not require calculations*

- **Alias**
  - **Wavefunction Encoding** [LC'20]
    - Every quantum system is described by its wavefunction \( \psi \) defining also the measurement probabilities
    \[ \Rightarrow \text{amplitudes of the quantum system represent data values} \]

- **Context**
  - Encoding of a *numerical input data vector* \( (x_0, \ldots, x_n)^T \) for a quantum algorithm.
Amplitude Encoding

- **Solution**
  - Encoding of the input vector in the amplitudes of the quantum state:
    \[ |\psi\rangle = \sum_{i=0}^{n} x_i |i\rangle \]
  - Squared moduli of the amplitudes of a quantum state must sum up to 1
    \[ \Rightarrow \text{input vector needs to be normalized to length 1} \]
  - Vector space of an \( n \) qubit register has dimension \( 2^n \)
    \[ \Rightarrow \text{input vector can be padded with additional zeros if dimension is not a power of 2} \]
  - Amplitudes depend on the data \( \Rightarrow \text{process of encoding the data} \) (but not the encoding itself) is often referred to as *arbitrary state preparation*
Amplitude Encoding

- **Result**
  - **Compact representation**: $\left\lfloor \log_2(n + 1) \right\rfloor$ qubits
    - more compact (in terms of qubits) than Basis, Angle Encoding or Quantum Random Access Memory (QRAM) Encoding
  - For an arbitrary state represented by $k$ qubits (i.e., $2^k$ data values), at least $2^k$ parallel operations for initialization are needed [SP'18] (nearly reached in state-of-the-art approaches)
  - For special cases logarithmic runtime or $O(1)$ (e.g., Uniform Superposition)
  - Sparse data vectors can also be prepared more efficiently [SP'18]
  - Output is often also encoded in the amplitude
    - Multiple measurements to obtain a good estimate of the output result
    - Number of measurements scales with the number of amplitudes $2^k$ for $k$ qubits [SP'18]
Amplitude Encoding

- **Known Uses**
  - Amplitude Encoding can be used in many *quantum machine learning* algorithms [LC’20]
  - Algorithm of Harrow, Hassidim and Lloyd [HHL'09] (*HHL algorithm*) for solving linear equations
  - Data values are typically normalized in machine learning [SFP’17], e.g. in support vector machine.
  - Various ways to construct a state preparation routine for amplitude encoding via e.g. Schmidt Decomposition [PB'11] (ArXiv) [I+16] (ArXiv)
    - Mathematica: [I+19]
    - Qiskit: [SBM'06] (ArXiv) QisKit Documentation
    - PennyLane: qml.AmplitudeEmbedding using the algorithm proposed by [MV'05] requiring an exponential number of operations to encode $2^k$ data values
    - Q#: approximates the desired amplitude encoding Q# API reference
QRAM Encoding 1/4

- **Intent**
  - "Use a quantum random access memory to access a superposition of data values at once" [W+'21]

- **Context**
  - Accessing the values of input data via random access memory

- **Solution**
  - **Classical** random access memory (RAM) transfers the data value stored at a given address into a specified output register
  - **Quantum** random access memory (QRAM): similar to RAM, but the registers are not classical but quantum registers [JHG'19]
  - Consequently, address and output registers can be in superposition instead of classical values
QRAM Encoding 2/4

\[ \sum_a c_a |a\rangle \xrightarrow{\text{QRAM}} \sum_a c_a |a\rangle |D_a\rangle \]

with \( c_a \) amplitude, \( |a\rangle \) address and \( |D_a\rangle \) data value of address \( |a\rangle \)
- Result
  - Data values consuming \( n \) bits: \( n \) qubits for basis encoded data
  - Address register: additional \( \lceil \log(k) \rceil \) qubits for up to \( k \) addresses
  - Basis Encoding is used for data values: computational properties
    \( \approx \) other digital encodings (e.g., QuAM and Basis Encoding):
    - Since data values are represented in superposition,
      - data values can be manipulated at once (using quantum parallelism)
      - multiple arithmetic operations (e.g., +, \( \cdot \)) can be applied
  - State preparation via the QRAM is efficient and of logarithmic runtime
    [SP’18]: QRAM queries \( N \) adresses in \( O(\log(N)) \) [KKP’20]
    - Exponential speed-up of an algorithm using QRAM:
      only possible if filling QRAM is efficient
  - To our best knowledge, there are currently no commercial hardware implementations for QRAM
    - State preparation routine must be used for loading the QRAM,
      but no routine for arbitrary input data exists that is as efficient as QRAM
- **Known Uses**
  - *Alternative state preparation* to realize QRAM Encoding can be found in [CB'18] (circuit family #3) or [P'14]
  - QRAM is required or assumed in various other algorithms [GLM'08], [RML'14], [WKS'14], [LMR'13]
  - *HHL algorithm for solving linear equations* [HHL'09] uses QRAM Encoding as an intermediate representation for eigenvalues [MKF'19]
Schmidt Decomposition

- **Context**
  - A state \( |s\rangle \) has to be prepared on an empty \( n \)-qubit register

- **Forces**
  - Small depth of the constructed state preparation circuit
  - Runtime for constructing the state preparation circuit on classical computer should not outweigh the potential benefit of quantum computing

- **Solution**
  - Determine \( B, U_1, V \) & apply circuit for state preparation of given \( |s\rangle \) [A+20]:

\[
\begin{align*}
|s\rangle &= \frac{|00\rangle + |01\rangle}{\sqrt{2}} \\
B &= U_1 = I \\
V &= H
\end{align*}
\]

- For the execution on a quantum computer, the unitary matrices \( B, U_1, V \) must be further decomposed into one and two qubit gates [LB'20]
Schmidt Decomposition

- Solution (continued)
  - How to determine $B, U_1, V$?
    - Express $\ket{s}$ in terms of two subspaces $V$ and $W$ that span $H^\otimes n$
      - Choose orthogonal basis $\{f_1, \ldots, f_k\} \in V \wedge \{g_1, \ldots, g_k\} \in W$, such that:
        $$\ket{s} = \sum b_{ij} \cdot f_i \otimes g_j$$
      - Examples: $f_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $f_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $g_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $g_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$
        - Example 1: $\ket{s} = \frac{\ket{00} + \ket{01}}{\sqrt{2}} = \frac{1}{\sqrt{2}} \cdot f_1 \otimes g_1 + \frac{1}{\sqrt{2}} \cdot f_1 \otimes g_2$
        - Example 2: $\ket{s} = \frac{\sqrt{2} \cdot \ket{00} + \ket{11}}{\sqrt{3}} = \frac{\sqrt{2}}{\sqrt{3}} \cdot f_1 \otimes g_1 + \frac{1}{\sqrt{3}} \cdot f_2 \otimes g_2$
      - $M := \{b_{ij}\}$
        - Example 1: $M = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$, Example 2: $M = \frac{1}{\sqrt{3}} \begin{bmatrix} \sqrt{2} & 0 \\ 0 & 1 \end{bmatrix}$
Schmidt Decomposition

- **Solution (continued)**
  - How to determine $B, U_1, V$?
    - Compute the singular value decomposition (SVD)
      \[
      M = \left( U_1 U_2 \right) \begin{pmatrix} A \\ 0 \end{pmatrix} V^* \text{ of } M \] (see [OS'18] for detailed instructions)
      - Example 1: Please see this link for details
        \[
        M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}^* 
        \]
    - Entries $\{\alpha_1, \ldots, \alpha_m\}$ of the diagonal matrix $A$: Schmidt decomposition
      - Example 1: Schmidt decomposition is $\{1, 0\}$
    - With $\alpha_1, \ldots, \alpha_m$ Schmidt coefficients for the Schmidt basis $\{u_i\}, \{v_i\}$:
      \[
      |s\rangle = \left( U_1 \otimes V \right) \sum_{i=1}^{m} \alpha_i \cdot e_i \otimes e_i = \sum_{i=1}^{m} \alpha_i \cdot u_i \otimes v_i, \alpha_i \in \mathbb{R} \geq 0, \text{ where} \]
      \[
      \sum_{i=1}^{m} \alpha_i = 1 \text{ (≈ The decomposition that minimally entangles the two subsystems...)}
      \]
      - Example 1: $|s\rangle = 1 \cdot |0\rangle \otimes \frac{1}{\sqrt{2}} \cdot (|0\rangle + |1\rangle) + 0 \cdot |1\rangle \otimes \frac{1}{\sqrt{2}} \cdot (-|0\rangle + |1\rangle)$
    - Remark: $V = H$ can be also used because of $0$
Schmidt Decomposition

- **Solution** (continued)
  - Example 2:
    - Please see this link how to compute the singular value decomposition for example 2: [link]

    \[
    M = \begin{bmatrix}
    1 & 0 \\
    0 & 1
    \end{bmatrix} \cdot \frac{1}{\sqrt{3}} \begin{bmatrix}
    \sqrt{2} & 0 \\
    0 & 1
    \end{bmatrix} \cdot \begin{bmatrix}
    1 & 0 \\
    0 & 1
    \end{bmatrix}^* 
    \]

    \[
    |s\rangle = \frac{\sqrt{2}}{\sqrt{3}} \cdot |0\rangle \otimes |0\rangle + \frac{1}{\sqrt{3}} \cdot |1\rangle \otimes |1\rangle
    \]

  - Remaining steps
    - \( B \) transforms the amplitude of the first register to the Schmidt coefficients
    - Copy this state to the second register using CNOT operations
    - \( U_1 \) and \( V \) transform the basis states \( \{e_i\} \) into the Schmidt basis states
Schmidt Decomposition

**Result**
- The state $|s\rangle$ is created in the register for which the Schmidt coefficients $\alpha_i$ are known, which can be used to quantify entanglement [NC’10]
- The state $|s\rangle$ is separable (i.e., not entangled) if and only if exactly one of the Schmidt coefficients is non-zero
- Depth of the circuit is $\frac{23}{48} 2^n$ in the worst case [PB’11]
  - Arbitrary state preparation is of exponential complexity in general (in the worst case)

**Related Patterns**
- Schmidt Decomposition can be used as a state preparation method for Amplitude or QRAM Encoding

**Known Uses**
- Schmidt Decomposition can be used to create random states with a controlled amount of entanglement [DGK’14]
- Mathematica implementation: [I+21]
Comparison Data Encoding Patterns

- \( k \) data points, one data point consumes \( n \) bits

<table>
<thead>
<tr>
<th>Encoding Pattern</th>
<th>Encoding</th>
<th>#Qubits</th>
<th>Preparation</th>
<th>Digital Encoding</th>
</tr>
</thead>
</table>
| Basis            | \( x_i \approx \sum_i b_i \cdot 2^i \)  
|                  | \( \leftrightarrow |x_i\rangle = |b_{n-1} \ldots b_1 b_0\rangle \) | \( k \cdot n \) | \( O(1) \) | ✓ |
| Angle            | \( x_i \leftarrow \cos(x_i)|0\rangle + \sin(x_i)|1\rangle \) | \( k \) | \( O(1) \) |    |
| QuAM             | \( X \leftarrow \frac{1}{\sqrt{k}} \sum_{i=1}^k |x_i\rangle \) | \( n \) | \( O(k) \) | ✓ |
| QRAM             | \( X \leftarrow \sum_i c_i |i\rangle |x_i\rangle \) | \( n + \lceil \log_2(k) \rceil \) | \( O(2^k) \) (Schmidt) | ✓ |
| Amplitude        | \( X \leftarrow \sum_{i=0}^{k-1} x_i |i\rangle \) | \( \lceil \log_2(k) \rceil \) | \( O(2^k) \) (Schmidt) |    |
Summary & Conclusions

- **Quantum Patterns**
  - Software design patterns using quantum algorithms

- **In this lecture**
  - Patterns for Data Encoding
    - State Preparations refined by
      - Basis Encoding
      - Quantum Associative Memory (QuAM)
      - Angle Encoding
      - Amplitude Encoding
      - QRAM Encoding
    - Schmidt Decomposition to be used as basic state preparation method for Amplitude and QRAM Encoding