Computes an efficient quantum algorithm for the quantum Schur transform on n qubits.

The quantum algorithm is returned as a list of $O(n^3)$ two-level operations in matrix form. These could each be decomposed into $O(n \log(n/\epsilon))$ Clifford+T operators using a universal unitary decomposition algorithm such as given in arXiv:1306.3200, for a total operation count of $O(n^4 \log(n/\epsilon))$ and overall error bounded by ϵ.

The quantum algorithm uses $2\lfloor \log_2(n) \rfloor - 1$ ancillary qubits for a total register of $n + 2\lfloor \log_2(n) \rfloor - 1$ qubits. The Schur transform maps the standard computational basis to a basis composed of a direct sum of irreducible modules for the unitary and symmetric groups. Equivalently, it maps individual spin eigenvectors to eigenvectors of the composite spin for the whole register.

We assume that the initial register written as a vector has form

$$(\text{ancillary qubits}) \otimes (\text{computational qubits})$$

. The output vector will be organized by subspaces with distinct values of composite total spin. An auxiliary algorithm to calculate the locations of the output states is provided.

Python implementation

Runs with Python 3.2+.

The main procedure in "schurtransform.py" is *schuralg(n)*, which returns a list of $O(n^3)$ elements with form $entry_i = [t_i, R_i, b_i]$. Each R_i is a one- or two-level rotation such that $\prod_i \mathbf{I}_{2^{t_i}} \otimes R_i \otimes \mathbf{I}_{2^{b_i}}$ gives the Schur transform ($\mathbf{I}_d$ is the identity matrix of dimension d).

schurindices(n) returns a dict that maps the key 's, k, m_s' to the output entry used to encode the state with spin-projection m_s in the kth copy of the spin-s subspace.

For example,

$$schurindices(n)(3)[\text{'}3/2, 1, 1/2\text{'}]$$

will return 2, the index of the state with spin-projection 1/2 in the 1st copy of the spin-3/2 subspace.

Mathematica implementation

This version allows straightforward interactive visualization using Mathematica language tools.

Runs with Mathematica 10+.

The main procedure in "Schur Transform.nb" is *SchurDecomp[n]*, which returns a list of $O(n^3)$ elements with form $entry_i = \{t_i, R_i, b_i\}$. Each R_i is a one- or two-level rotation such that $\prod_i \mathbf{I}_{2^{t_i}} \otimes R_i \otimes \mathbf{I}_{2^{b_i}}$ gives the Schur transform ($\mathbf{I}_d$ is the identity matrix of dimension d).

SchurIndices[n] returns an association (map) that takes the key $\{s, k, m_s\}$ to the output entry used to encode the state with spin-projection m_s in the kth copy of the spin-s subspace. For example,

$$\textbf{SchurIndices}[3][\{3/2, 1, 1/2\}]$$

will return 2, the index of the state with spin-projection $1/2$ in the 1st copy of the spin-$3/2$ subspace.