Outline of a Synalgebraic Approach to Distributed Quantum Computing

Anderson Beraldo de Araujo

***Abstract*— This paper introduces a framework for computational reasoning about networks of quantum computers, grounded in a new mathematical theory called *synalgebra*. Unlike previous approaches, it shows how gate circuits can be made fully quantum and distributed across different levels of configuration. This perspective enables the formal verification of designs for quantum computer networks with unconventional topologies, which can be leveraged to develop previously unknown quantum algorithms.**

***Keywords*— Distributed Quantum Computing, Machine Reasoning, Synalgebra**

I. The synalgebraic framework for distributed quantum computing

The present state of quantum technology indicates that we will, probably, remain in a noisy intermediate-scale quantum (NISQ) era for quite some time. In this NISQ scenario, there is growing consensus among specialists regarding the importance of distributed quantum computer architectures as a pathway to scaling up to numerous qubits with low error rates [1]. Indeed, models of distributed quantum computing have been theorized, with some already being under development as quantum hardware [16], [2].

However, current distributed approaches do not provide a clear model of modular computation in the sense of having certain functions implemented as libraries in quantum hardware, which can then be composed to form complex algorithms [8]. There are examples of modular algorithms [14] and prototypes of modular hardware [9]. Nonetheless, these approaches are not systematic: they do not present abstract models of distributed computation with modular components. Moreover, they do not explore the topology of computer networks and, consequently, cannot directly contribute to the discovery of new quantum algorithms.

In contrast, this paper proposes a formal model of distributed quantum computing with modular components. The idea is as follows: multiple clients send classical data, which a classical machine encodes into quantum states, then computation emerges from the interactions among quantum computers, organized into network topologies with multiple levels of configuration. We refer to this approach as the *synalgebraic framework*, illustrated in Figure 1.

Figure 1 shows *exchanges* (a_i, b_i) of clients with a quantum plataform $(\mathcal{M}, \mathcal{N}, \mathcal{Q})$ where: $\mathcal{M}$ is a classical computer that takes a received data a_i of a client i, encodes it in $|a_i\rangle$ and distribute $|a_i\rangle$ among the *initial layer* of quantum computers $\mathcal{Q}_0, \ldots, \mathcal{Q}_l$ in $\mathcal{Q}$; $\mathcal{N}$ is a classical computer that takes a state $|b\rangle$ generated by the *terminal layer* of quantum computers $\mathcal{Q}_p, \ldots, \mathcal{Q}_q$ in $\mathcal{Q}$, decodes it in b and sends this processed data back to the client i. The network $\mathcal{Q}$ of is made of different modules of quantum computers that implement specific functions; these functions are composed to generate the output $|b\rangle$ of $\mathcal{Q}$.

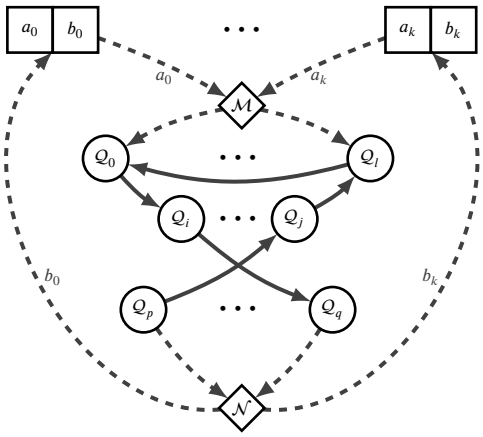

Fig. 1. Synalgebraic framework

To implement this framework, we apply a new algebra of levels of abstraction and concretion, called *synalgebra*. We introduced synalgebra in [5], [4], presented at IEEE Quantum Week 2025. It is a mathematical theory under development, with a distinctive ability to reason about identities and similarities, enabling the evaluation of information flow in complex data structures called *intergraphs*. The flexibility of intergraphs makes it possible to define links among different quantum modules of computation based on local operations and classical communication (LOCC) [11, p. 573]. Within an LOCC system, synalgebra supports the definition of multilevel operations across quantum modules.

In what follows, we introduce the core concepts and operations of synalgebra, emphasizing its role as a foundational framework for automated reasoning in distributed quantum computing with modular components. Unlike previous approaches, the synalgebraic framework presented here is fully distributed and grounded in computational reasoning, enabling powerful and verifiable designs for quantum computer network topologies. To illustrate these ideas, we also present preliminary data from computational simulations.

Anderson Beraldo de Araujo, Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, and Centre of Mathematics, University of Minho, e-mail: anderaujo@gmail.com. This material is based upon work supported by the Air Force Office of Scientific Research and Office of Naval Research Global under award number FA9550-24-1-0003.

II. Synalgebrization of quantum gate circuits

Synalgebra is a theory for equational reasoning over levels of abstractions and concretions called *intergraphs*. An *intergraph* is a graph structure with levels that may have edges within and across other levels of different orders. An intergrah has an operation $\triangleright$ that represents a *link* between two *levels* and an operation $:$ that represents *instantiation*, i.e., $a : b$ is true when a is a level inside b. The *nodes* of an intergraph are levels that have no other levels inside them. Additionally, intergraphs have an operation $a \triangle b$ called *junction*, which represents a blend of the conjunction and disjunction; a mix between "and" and "or". For example, an intergraph in which a is linked to b *and* c *and* d but also b *or* c are linked to d can be represented as

$$((a \triangleright (b \triangle c)) \triangleright d) \triangle (a \triangleright d)$$

The meaning of this formula is the following intergraph

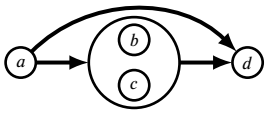

Fig. 2. Intergraph example

Synalgebra enable us to calculate the flow of information in intergraphs too. For instance, to activate a flow of information in Fig. 2, we just need to applied the intergraph to the node a, because it is the *initial node*. The nodes that result from the activation of the initial nodes of an intergraph are called *terminal nodes*; in our example, the node d. Hence, synalgebra internalizes these notions with an operation of *application* $\cdot$. The next sequence of equations illustrates a synalgebraic reasoning:

$$\begin{aligned}(a \triangleright d) \cdot a &= d \\ &= (c \triangleright d) \cdot c \\ &= ((b \triangle c) \triangleright d) \cdot b \\ &= (a \triangleright (b \triangle c)) \cdot a \triangleright d \\ &= ((a \triangleright (b \triangle c)) \triangleright d) \cdot a\end{aligned}$$

Such a reasoning indicates that, although the paths $a \triangleright d$ and $(a \triangleright (b \triangle c)) \triangleright d$ in the integraph Fig. 2 are different, they are *similar* because share initial and terminal nodes. Hence, intergraphs have two relations $=$ and $\simeq$ that represent *identity* and *similarity*. In this notation, we can express that $a \triangleright d \neq (a \triangleright (b \triangle c)) \triangleright d$ but $a \triangleright d \simeq (a \triangleright (b \triangle c)) \triangleright d$. Synalgebra proposes a method for efficiently approximating quantum computations by leveraging the qualitative distinction between identities $=$ and similarities $\simeq$. In doing so, we can *perfectly track the structure of quantum computations* and subsequently perform quantitative statistical analyses of the approximations made.

In order to calculate identities and similarities among levels in intergraphs, synalgebra has rules of the form

$$[\Upsilon \vdash \Lambda] =_{\alpha_R} [\Gamma \vdash \alpha, \Delta]$$
$$[\Upsilon \vdash \Lambda] =_{\alpha_L} [\Gamma, \alpha \vdash \Delta]$$

where a *right rule* α_R indicates what happens $(\Upsilon \vdash \Lambda)$ when α is true (right side of $\vdash$) and a *left rule* α_L indicates what happens $(\Upsilon \vdash \Lambda)$ when α is false (left side of $\vdash$). In the rules above, the symbol Γ represent a list of *assumptions* and Δ a list of *consequences*. The symbol "$\vdash$" marks the difference between assumptions and consequences. This distinction is remanescent of Gentzen sequent calculus [12].

For example, synalgebra has right rules for quantum operations like NOT gate, represented by the symbol $\diamondsuit$, namely,

$$[\Gamma \vdash |0\rangle \triangleright \diamondsuit, \Delta] =_{\diamondsuit_R} [\Gamma \vdash |1\rangle, \Delta]$$
$$[\Gamma \vdash |1\rangle \triangleright \diamondsuit, \Delta] =_{\diamondsuit_R} [\Gamma \vdash |0\rangle, \Delta]$$

The quantum operators are assumed to be linear. Thus, it is sufficient to define their actions on the basis of computation – the qubits $|0\rangle$ and $|1\rangle$ in the illustrated example. Of course, there are left rules for $\diamondsuit$ too and, similarly, for all quantum gates and measurements. Indeed, synalgebra has a special approach to measurements: it represents measurements by indexing states with probabilities. Thus, $|a\rangle^p$ denotes that the state $|a\rangle$ occurs with probability p. The rules handle the generation of states and probabilities of quantum measurements, using such a kind of indexing of states.

In a nutshell, that is the foundational idea of synalgebra. It is a reformulation of Shallon's graph algebra [15], in which $a \triangleright b$ is interpreted as "b is a target of a," combined with ideas from De Domenico et al. [6] about multilevel networks. In an early work of us about logical modelling of quantum computational problems [3], we identified that quantum computations require a mathematical representation of the flow of information different from the usual ones in algebraic logic. This resulted in the novel concept of intergraph as a blend of ideas associated to Kant's notion of *schema* [10] – a mediation between abstract concepts and concrete intuitions – and Peirce notion of *relatives* [13] – diagrams for relations. More details can be found in [5], [4].

III. Multilevel distributed quantum computing

In synalgebra, we use the following symbolic notation:

- $|\alpha\rangle_k^p$ is the quantum state α with proability p at cell k;
- $\circ_k$ is the condition at cell k;
- $\diamondsuit_k$ is the NOT gate at cell k;
- $\square_k$ is the Hardamard gate at cell k;
- $\circledpi_k^t$ is the Rotation with angle $\theta = t\pi$ at cell k;
- $\boxed{O}_k$ is the operator O at cell k;
- $\bigstar_k$ is the measurement in the Z basis at cell k.

The symbols are indexed by natural numbers in order to represent the 2D tabular form of quantum circuits, avoiding ambiguity. The alphabet of symbols above defines a language of formulas represented by symbols like ϕ, ψ, etc. In this *syntax*, $\mathbb{1}^{[k]}$ and $O_l^{[k]}$ are, respectively, the *control* at cell k of the *target* at cell k which is subject to the operation O.

Each formula ϕ is interpreted in an intergraph $\mathcal{A}$ with a Tarskian *semantics* so that the *meaning* $\mathcal{A}(\phi)$ in $\mathcal{A}$ is made up of clauses like the following:

- $\mathcal{A}(|i\rangle) = |i\rangle$;
- $\mathcal{A}(|i\rangle \triangleright O) = \mathcal{A}(O)(|i\rangle)$;
- $\mathcal{A}(\alpha \oplus \beta) = \mathcal{A}(\alpha) \oplus \mathcal{A}(\beta)$ where $\oplus$ is the vector sum;
- $\mathcal{A}(\alpha \otimes \beta) = \mathcal{A}(\alpha) \otimes \mathcal{A}(\beta)$ where $\otimes$ is the tensor product;
- $\mathcal{A}(\bigstar)(a\,|i\rangle^p) = (|0\rangle\langle 0|\,|i\rangle^{a\cdot a^*\cdot p}) \triangle (|1\rangle\langle 1|\,|i\rangle^{a\cdot a^*\cdot p})$ where a^* is the complex conjugate of a.

There are other clauses, but these should be sufficient to grasp the idea of the semantics. Note that measurements are defined in terms of junctions via the Born rule that say the probability of obtaining a state like $|i\rangle$ given the system is in the state $a\,|i\rangle \oplus b\,|j\rangle$ is $|a|^2 = a \cdot a^*$.

With the syntax and semantics established, we can use synalgebra to define four operations on quantum computer modules: control, teleport, setup, and breakup. These operations capture the possible relations between nodes and modules – namely, node to node across different modules, node to module, module to module, and module within module.

The *control* operation defines a link between nodes of different quantum computers module. We represent this operation by symbol $\mathbb{1}^{\circ_k} \rhd_B^A \boxed{O}_k$ where A and B are quantum modules such that $\mathbb{1}^{\circ_k} : A$ and $\boxed{O}_k : B$. The synalgebraic formula that defines this control operation is displayed below:

$$\begin{aligned}\mathbb{1}^{\circ_k} \rhd_B^A \boxed{O}_l = [A \simeq (|\alpha\rangle_0 \rhd \mathbb{1}^{\circ_1} \rhd \circledast_2^{\circ_{11}} \rhd |\alpha\rangle_3 \\ \otimes |0\rangle_4 \rhd \square_5 \rhd \mathbb{1}^{\circ_6} \rhd \diamondsuit^{\circ_1} \rhd \bigstar^{\circ_7} \rhd |\rho\rangle_8)] \triangle \\ [B \simeq (|0\rangle_9 \rhd \diamondsuit^{\circ_6} \rhd \diamondsuit^{\circ_7} \rhd \mathbb{1}^{\circ_{10}} \rhd \square_{11} \rhd \bigstar_{12} \rhd |\sigma\rangle_{13} \\ \otimes |\beta\rangle_{14} \rhd O^{\circ_{10}} \rhd |\eta\rangle_{15}]\end{aligned}$$

Although the formula of the control operation may look douting at first glance, its meaning is clear: it makes one node in a module be controlled by a node in another module. Hence, the formal meaning $\mathcal{A}(\mathbb{1}^{\circ_k} \rhd_B^A \boxed{O}_l)$ of $\mathbb{1}^{\circ_k} \rhd_B^A \boxed{O}_l$ in an integraph $\mathcal{A}$ has the circuit representation displayed below:

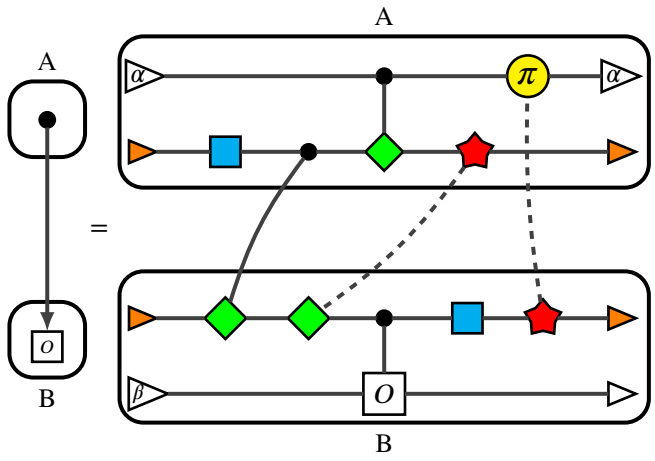

Fig. 3. Control operation

The *teleport* operation defines a link between a node in a module and another module. The formula that describes teleport is this:

$$\begin{aligned}\mathbb{1}^{\circ_k} \rhd^A B = [A \simeq (|\alpha\rangle_0 \rhd \mathbb{1}^{\circ_1} \rhd \square_2 \rhd \bigstar^{\circ_3} \rhd |\rho\rangle_4 \\ \otimes (|0\rangle_4 \rhd \square_5 \rhd \mathbb{1}^{\circ_6} \rhd \diamondsuit^{\circ_1} \rhd \bigstar^{\circ_8} \rhd |\eta\rangle_9] \triangle \\ [B \simeq (|0\rangle_{10} \rhd \diamondsuit^{\circ_6} \rhd \diamondsuit^{\circ_8} \rhd \circledast^{\circ_3} \rhd |\alpha\rangle_{11})]\end{aligned}$$

You can see its circuit representation of teleport below:

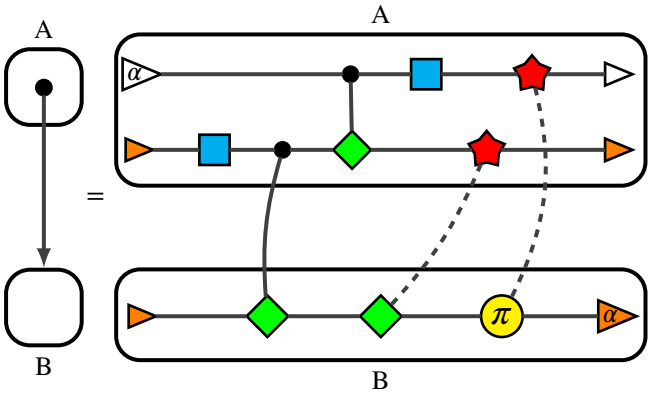

Fig. 4. Teleport operation

Note that control $\mathbb{1}^{\circ_k} \rhd_B^A \boxed{O}_k$ is a non-local definition of quantum controlled gates, as defined, for instance, in [17]. Similarly, teleport $\mathbb{1}^{\circ_k} \rhd^A B$ is just a conventional teleportation protocol [7]. Both relies on the creation of a Bell entangled state. The novelty here is that these operations can be represented within synalgebra and, thus, to obtain two new operations, namely, setup and breakup.

The *setup* operation $A \rhd_B \boxed{O}_l$ defines a link between a node in module that is controlled by another module. Hence, it is a kind of Fredkin non-local controlled gate. The formula that represents it is a little involved, but we can see its circuit representation below:

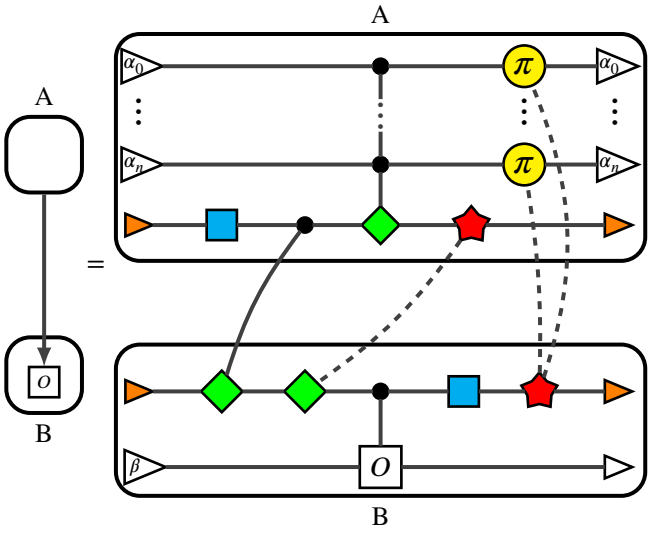

Fig. 5. Setup operation

The last operation is *breakup*, in symbols $A \rhd\!\!\rhd B$. This operation relates a quantum module with another one. The integraph representation of the breakup of a quantum circuit is displayed below:

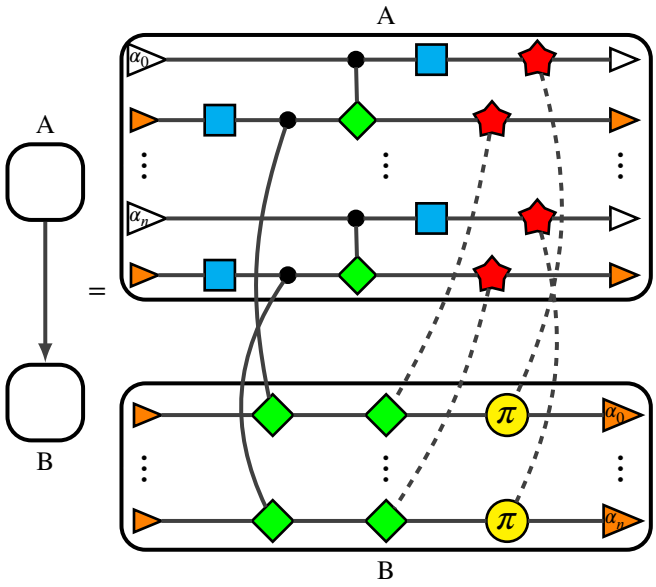

Fig. 6. Breakup operation

Breakup is a generalization of the teleport. Indeed, $A \triangleright\!\!\triangleright B$ consists in teleporting the whole output of A to B, so that B begins where A ends. It may be a useful operation to break up a circuit with big width by smaller ones; for example, in situations in which the coherence time of the circuits do not support an algorithm with long width.

There are many details to be explained, but these are the core concepts of the distributed quantum computation model of the synalgebraic framework.

IV. Preliminary simulation results

We have been developing a library for the synalgebraic framework in Rust, due to its safe concurrent model and its integration with hardware description languages to run on FPGAs. Below, we present preliminary data on a particular network topology; just to exemplify the synalgebraic framework. The simulations were conducted on a Dell desktop with an Intel Core i7-1355U CPU and 15 GiB of RAM. The quantum circuits simulated have the following scheme:

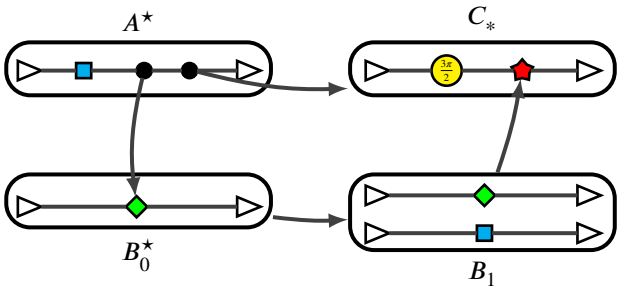

Fig. 7. Scheme for simulations

In Figure 7, the star symbols $\star$ in $A^\star$ and $B^\star$ indicate that this computers are *initial*, i.e., the input is encoded in them. The asterisk symbol $*$ indicates that C_* is *terminal*, which means the output is decoded from it. That is the reason it has the measurements. Although A and B_0 start the computation in parallel, $\mathbb{1}^{\circ_2} \triangleright_B^A \diamondsuit^{\circ_2}$ and, consequently, B waits A to process $\mathbb{1}^{\circ_2}$ in order to evaluate $\diamondsuit^{\circ_2}$. Similarly, although the circuit has $B_1 \triangleright_{C_*} \star^{\circ_1 2}$, it computes first $\mathbb{1}^{\circ_2} \triangleright^A B$, in order to get the state to apply $\star^{\circ_1 2}$ if the control holds.

The circuit in Figure 7 has 13 qubits: 5 explicitly displayed and 8 implicitly used in the operations of control, teleport, setup and breakup. The number of layers is the sum of the nummbers of collumns in each module. For the simulations, we added in A, B_0 and B_1 randomly chosen single gates and increased the number of qubits, one by one, in the scheme of the circuit in Figure VI. Below, it is showed data about the simulations.

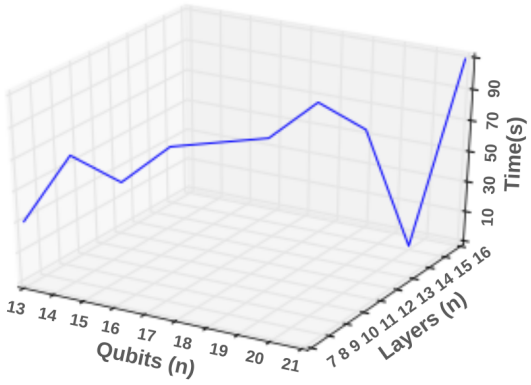

Fig. 8. Simulation data

The vertical axis in Figure 8 is the time (in seconds) needed to compute the output of circuit from inputs that are tensor products of the state $|0\rangle$. As we can see, the minimal time to compute the circuit is 29s, which regularly increases with the number of qubits and layers, except when it reach to 21 qubits and 12 layers. At this point an abrupt decay up to 14 layers happens, which turn to increase after that up to 16 layers.

We do not yet know why these patterns occur, and we avoid drawing conclusions until more samples are collected and statistical analyses are performed. In addition, we need to explore variations in circuit topologies. For now, we can only state that there appear to be interesting patterns worth investigating. This was precisely our aim: to demonstrate that the synalgebraic framework enables the analysis of unconventional topologies, which can be leveraged to develop previously unknown quantum algorithms. In future work, we intend to pursue this in greater detail.

Acknowledgment

I would like to thank to my family Viviane and Antoni Beraldo de Araujo (UNICAMP) for their support in the whole synalgebra's odyssey. I am thankful as well to my PhD supervisors Luis Pinto (UMinho) and Jose Carlos Espirito Santo (UMinho) and, for the helpful discussions, to James Lyke (AFOSR), Kyle Gustafson (ONRG), and the members of the Mathematical Foundations of Quantum Theory Group at UNICAMP, specially to Marcelo Terra Cunha (UNICAMP) and Rafael Rabello (UNICAMP).

REFERENCES

[1] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A.S. Cacciapuoti. Distributed quantum computing: A survey. *Computer Networks*, 254(110672):1–25, 2024.

[2] D.P. Nadlinger E.M. Ainley-A. Agrawal B. C. Nichol R. Srinivas G. Araneda1 D.M. Lucas D. Main, P. Drmota. Distributed quantum computing across an optical network link. *Nature*, 638(13):383–396, 2025.

[3] A. de Araujo and M. Finger. Classical and quantum satisfiability. *EPTCS*, 81:79–84, 2012.

[4] A. Beraldo de Araujo. Synalgebraic reasoning for automation of photonic quantum learning. *IEEE International Conference on Quantum Computing and Engineering (QCE)*, QADA Workshop:1–6, 2025.

[5] A. Beraldo de Araujo. Towards synalgebraic photonic quantum computing. *IEEE International Conference on Quantum Computing and Engineering (QCE)*, Photonic Track:1–7, 2025.

[6] M. De Domenico, A. Sole-Ribalta, E. Cozzo, M. Kivela, Y. Moreno, M. A. Porter, S. Gomez, and A. Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(041022):1–15, 2013.

[7] Charles H, Bennett, Gilles Brassard, Claude Crepeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70(1895):1–5, 1993.

[8] V. Vedral M. Gu J. Thompson, K. Modi. Quantum plug n' play: modular computation in the quantum regime. *New Journalf of Physics*, 20(013004):1–13, 2018.

[9] X. Zhang Y. Shen-Y. Lu S. Zhang J. Ma V. Vedral M. Gu K. Kim K. Zhang, J. Thompson. Modular quantum computation in a trapped ion system. *Nature Communications*, 10(4692):1–6, 2019.

[10] I. Kant. *Critique of Pure Reason*. Cambridge University Press, translated and edited by paul guyer and allen w. wood edition, 1999.

[11] I.L. Chuang M. Nielsen. *Quantum Computation and Quantum Information*. Cambridge University Press, 2020.

[12] S. Negri and J. Von Plato. *Structural Proof Theory*. Cambridge University Press, Cambridge, 2001.

[13] C. S. Peirce. The logic of relatives. *The Monist*, 7(2):161–217, 1897.

[14] D. Choi B.-S. Choi S.-H. Seo S.-M. Cho, A. Kim. Quantum modular multiplication. *IEEE Access*, 8(213244):1–9, 2020.

[15] C.R. Shallon. *Non-finitely based binary algebras derived from lattices*. UCLA, phd dissertation edition, 1979.

[16] J.D. Thompson Y. Li. High-rate and high-fidelity modular interconnects between neutral atom quantum processors. *PRX Quantum*, 5(020363):1–13, 2024.

[17] A. Yimsiriwattana and S.J. Lomonaco. Generalized ghz states and distributed quantum. *arXiv:*, quant-ph/0402148Computing:1–16, 2004.