UNIVERSITY OF AMSTERDAM

MAX PLANCK INSTITUTE
FOR THE SCIENCE OF LIGHT

Report Bachelor Project Physics and Astronomy, size 15 EC
University of Amsterdam and Max Planck Institute

Formal Verification of Mathematics Behind Quantum Optics Experiments

---

Lode Vermeulen
conducted between 01-04-2024 and 02-07-2024

Student number:	13297619
Research Institute:	Max Planck Institute
Research Group:	Artificial Scientist Lab
Supervisor:	Mario Krenn
Assessor:	Philippe Corboz
Examiner:	Jean-Sebastien Caux
Date of Submission:	02-07-2024

Scientific Abstract

The future of research in theoretical physics may increasingly rely on artificial intelligence. An AI model that contributes to the derivation of new mathematical theories in physics could benefit from a robust mathematical foundation, which could be provided by theorem provers. This thesis explores the use of the Lean Theorem Prover for a graph theoretical statement with a direct connection to quantum optics. Specifically, it attempts to formalize a theorem regarding the maximum number of disjoint perfect matchings in a graph where all matchings are disjoint. This theorem is directly related to the constructability of certain quantum optics experiments. Part of this formalization was achieved, along with the addition of basic but necessary graph theory statements to Lean's mathematical library Mathlib. This work opens the possibility of a new application of AI in quantum optics, by formalizing the mathematics that describe experimental quantum optics in a formal system that breaks down proofs into the axioms of mathematics.

Popular Scientific Abstract (Dutch)

De afgelopen jaren hebben kunstmatige intelligentie (KI) modellen die getraind zijn op grote hoeveelheden data hun kracht getoond. Large language models (LLM) zoals ChatGPT hebben het uitvoeren van veel taken voorgoed veranderd. Ook in de wetenschap hebben ze een grote impact gehad, vooral bij het schrijven en analyseren van teksten. In de exacte wetenschappen zijn LLM's echter nog niet baanbrekend geweest. Zelfs bij eenvoudige rekensommen maakt ChatGPT regelmatig fouten. Om KI te ontwikkelen die wel impact heeft in exacte wetenschappen, kan het nuttig zijn om deze te baseren op een wiskundige en exacte taal. Een voorbeeld hiervan is een type programmeertaal genaamd theorem provers.

In theorem provers is het mogelijk om wiskundige stellingen te bewijzen, waarbij het programma alleen slaagt als de computer alle stappen goedkeurt. Een model gebaseerd op een dergelijk fundament zal altijd wiskundig kloppende suggesties doen, wat in theoretische natuurkunde zeer nuttig kan zijn.

Het doel van dit onderzoek is om een wiskundige stelling, die direct verbonden is met een natuurkundige stelling uit de kwantumoptica, te bewijzen in een theorem prover genaamd Lean. In het veld van kwantumoptica worden licht en de kwantumeffecten die daarbij optreden bestudeerd. Voor sommige experimenten is het noodzakelijk om fotonen in een bepaalde kwantumverstrengelde toestand te brengen. Specifiek richt dit onderzoek zich op het creëren van GHZ toestanden, een sterk verstrengelde toestand. Het realiseren van deze toestanden is in de praktijk erg lastig. De methode die in dit onderzoek wordt gebruikt, is bijzonder nuttig omdat er een link is gelegd met een andere tak van de wiskunde, namelijk de grafentheorie. Een relevante stelling is hierin bewezen, wat het een uitstekende kandidaat maakt om te proberen te bewijzen in Lean. Het digitaliseren van deze stelling kan er in de toekomst voor zorgen dat een KI-systeem dat getraind is op Lean verdere ontdekkingen kan doen, die dan ook directe gevolgen hebben voor het maken van GHZ toestanden in de kwantumoptica.

Contents

1 Introduction

Over the past decade, AI models trained on large datasets have demonstrated significant potential for various applications. Unfortunately, they have not yet been able to make a field altering contribution in theoretical physics, despite their potential to do so. For example, large language models (LLMs) have become great at replicating the way humans write text. At times, this can create the illusion that these models are capable of complex and logical reasoning. It becomes clear that it is but an illusion when it is tasked with, for instance, solving a mathematical problem with difficulty of anywhere from slightly advanced arithmetic to university level mathematics. Fundamentally, LLMs are also not expected to, as the basis upon which they are trained is human language, which is not a language of logic. An example of a digitized language that is based on logic could take the form of a theorem prover (TP), a software tool that allows the formalization of theorems, starting from an axiomatic system. In the future, a TP could be the basis for an AI model that is able to aid in doing research in a field of exact sciences.

A formalized proof in a theorem prover is verified by a computer down to the axioms, providing the most unambiguous proof method available. Because of this, various difficult or controversial theorems, such as the four-color theorem [1] were proved in them, to show their correctness. Theorem provers have also been used in physics, such as in the formalization of the theory of relativity [2]. Recently, the Lean theorem prover has demonstrated its technical ability with the formalization of cutting-edge mathematics, like in the Liquid Tensor Experiment [3] and with the large scale effort to create a mathematical library called Mathlib [4]. Additionally, several projects are focused on formalizing aspects of physics in Lean, including high energy physics [5].

This thesis aims to formalize a graph theoretical statement that is directly related to experimental setups in quantum optics capable of creating highly entangled Greenberger–Horne–Zeilinger (GHZ) states. More specifically, it attempts to formalize Bogdanov's lemma in Lean, which is a statement about the maximum number of disjoint perfect matchings in a specific type of graph. The formalization of Bogdanov's lemma would prove the possibility to digitize this statement. In the future, this could serve as a corpus for training AI [6], potentially aiding related problems and directly impacting the constructability of certain quantum optics experiments.

2 Background

2.1 Quantum entanglement through graphs

2.1.1 Quantum entanglement of photons

Quantum entanglement is one of the most striking features of quantum mechanics. In its simplest form, quantum entanglement takes the shape of a two-particle

Bell-state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B \right)$$

If two particles are in this state, the measurement of particle A means that the state of particle B is automatically known as well. Particle A and particle B are then considered entangled. This unique feature of quantum states can be used in various fields, such as quantum computing, cryptography and quantum metrology [7].

The above Bell state can be generalized to an arbitrarily large n-particle d-dimensional entangled state:

$$|\Psi\rangle = \sum_{i=0}^{d-1} c_i \, |i\rangle^{\otimes n}$$

where c_i is a complex coefficient such that $\sum |c_i|^2 = 1$. Any n-particle entangled state can be subdivided into a class, where any two states of the same class are identical under stochastic local operations and classical communication (SLOCC). For $n = 2$, there exist two classes, for $n = 3$, there exist 6 classes, for $n \geq 4$, there exist an infinite number of classes [8]. For $n = 3$, two particularly interesting states can be identified, the GHZ $= \frac{1}{\sqrt{d}} \sum_{i=0}^{d-1} |i\rangle^{\otimes n}$ and W $= c_1 \, |10 \ldots 0\rangle + c_2 \, |01 \ldots 0\rangle + \cdots + c_n \, |00 \ldots 1\rangle$ states. These states are known as genuinely entangled, meaning that their entanglement cannot be reduced to entanglement among pairs of qubits. In their two-dimensional form, these states can be used in quantum metrology and to test local-realism theories [9]. Their multi-dimensional equivalents have been shown to be useful in quantum information [10]. Because of this, experimentally creating GHZ and W states is of great interest.

The particles comprising an entangled state can take various forms. Experimentally, entangled states can be created using superconducting circuits, trapped ions, neutral atoms and photons. Currently, the record for the largest number of qubits in the GHZ state using trapped ions is 32 [11], for superconducting circuits 60 [12] and for photons 14 [13]. Besides the amount of qubits, there are other considerations for these methods of creating entangled states. Photons, for example, are excellent candidates for long-range entanglement distribution, something that is necessary for quantum networks [14].

An often used way of generating the photons required for entanglement is through spontaneous parametric down-conversion (SPDC), a process to generate photon pairs with correlated polarization [7]. This process works with a nonlinear crystal that takes in a single (pump) photon and emits two photons. To create an entangled state, two such crystals are needed, of which the created photon pairs have different polarization. One of the photons of each pair is then sent through a polarizing beam splitter, which removes the information from which crystal the photon originates. This process creates a four-particle GHZ state $|\psi\rangle = (1/\sqrt{2})(|H, H, H, H\rangle + |V, V, V, V\rangle)$, where H and V stand for horizontal and vertical polarization, respectively [15].

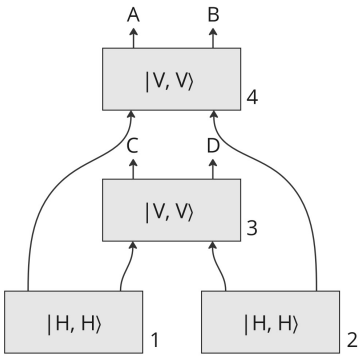

Figure 1: Four SPDC crystals that each can probabilistically generate a photon pair when triggered. Crystals 1 and 2 and crystals 3 and 4 create horizontally and vertically polarized photon pairs respectively. The photon pair from crystal 1 ends up in photon output path A and C, making them indistinguishable from the photons created in crystals 3 and 4, besides polarization.

2.1.2 Quantum entanglement of photons by path identity

A new approach to generate multi-photon entangled states removes the need for polarizing beam splitters and entangled photon pairs to begin with [16]. The main idea relies on the probabilistic activation of two out of four SPDC crystals, with the generated photons being sent to four detectors Fig. 1. This causes an overlap of at most two photon paths. All the possibilities are captured in the full wave function, up to second-order SPDC [17].

$$
\begin{aligned}
|\psi\rangle = \; & (1-2g^2)\,|0,0,0,0\rangle + \\
& g\,|H,0,H,0\rangle + g\,|0,H,0,H\rangle + g\,|0,0,V,V\rangle + g\,|V,V,0,0\rangle + \\
& g^2\,|2H,0,2H,0\rangle + g^2\,|0,2H,0,2H\rangle + g^2\,|0,0,2V,2V\rangle + g^2\,|2V,2V,0,0\rangle + \\
& \sqrt{2}g^2\,|H,H,H,H\rangle + g^2\,|H,0,H+V,V\rangle + g^2\,|H+V,V,H,0\rangle + \\
& \sqrt{2}g^2\,|0,H,V,H+V\rangle + \sqrt{2}g^2\,|V,H+V,0,H\rangle + \\
& \sqrt{2}g^2\,|V,V,V,V\rangle + O(g^3)
\end{aligned}
\tag{1}
$$

An interesting case is where photons of exclusively the same polarization are created. This happens only when crystals 1 and 2 or 3 and 4 create photon pairs. Post-selection (conditioning the final result on one detection event in each of the detectors) of such cases results in the four-photon GHZ wavefunction, indicated in red in the wave equation (1).

This method can be expanded to create multi-photon, multi-dimensional entangled states. First, n SPDC crystals can create a $2n$-photon entangled state by daisy-chaining the crystals together, as in Fig. 2. Theoretically, n can be arbitrarily large, but the probability of $2n$-fold coincidences occurring decreases

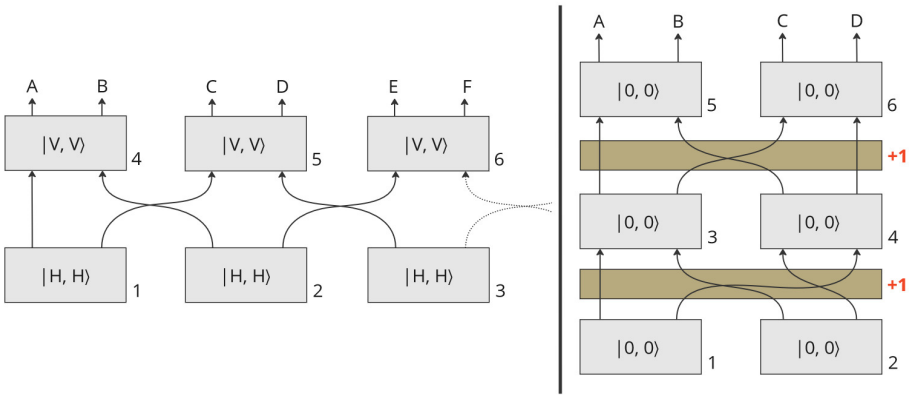

Figure 2: Left: Proposal for creating a general $2n$-photon 2-dimensional GHZ state. Right: Graphical diagram for creating a 4-photon 3-dimensional GHZ state. The crystals each produce photon pairs in the lowest OAM mode. The gold-brown boxes represent holograms that increase the mode of the photon passing through it by one.

with g^n, where g is the down-conversion amplitude of the crystal. To create a multi-dimensional state, a continuous degree of freedom is necessary. Some candidates are orbital angular momentum (OAM), discrete frequency, or time bins [16]. An example of such a multi-dimensional state using OAM is presented in Fig. 2 (right). After post-selection for cases where there is a photon in each detector, the wavefunction becomes

$$|\psi\rangle = \frac{1}{\sqrt{3}}(|0,0,0,0\rangle + |1,1,1,1\rangle + |2,2,2,2\rangle)$$

Combining the methods to increase photon count and dimensionality, a setup that creates n-photon d-dimensional entangled states can be constructed, where $n \geq 2$ and $d \geq 2$.

An example of a multi-photon, multi-dimensional entangled state is shown in Fig. 3. After post-selection (for six-fold coincidences in this case), the wavefunction becomes:

$$|\psi\rangle = \frac{1}{2}(|0,0,0,0,0,0\rangle + |1,1,1,1,1,1\rangle + |2,2,2,2,2,2\rangle + |1,2,1,2,0,0\rangle)$$

This is not a 6-photon 3-dimensional GHZ state, as one might expect. Surprisingly, there is a restriction to the dimensionality of GHZ states that this method can construct. This restriction will be explained in section 2.1.3.

The method described previously does not yet make use of the powerful quantum phenomenon of interference. Interference in photons can be achieved by shifting their respective phases. This can be achieved experimentally by placing phase shifters on a photon path. As an example, take the setup in Fig.

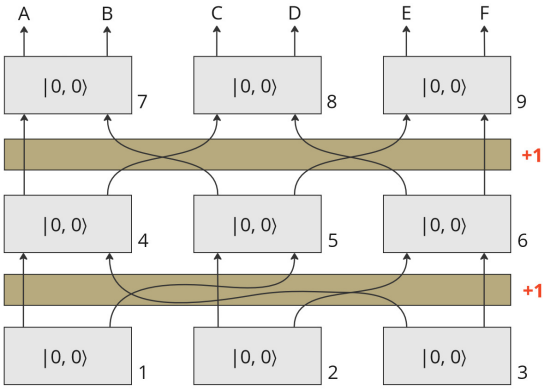

Figure 3: Graphical diagram for creating 6-photon 3-dimensional entangled states.

1, where instead crystals 3 and 4 also produce horizontally polarized photon pairs. If a phase shifter is placed on, for example, photon output path A and before crystal 4, the following state is produced after post-selection [17]

$$|\psi\rangle = (1 + e^{i\phi})\,|H, H, H, H\rangle$$

where ϕ is the phase shift. Clearly, for $\phi = \pi$, destructive interference is demonstrated.

Finally, this method of creating multidi-mensional GHZ states has been experimentally demonstrated in a nanophotonic circuit [18], and destructive interference using this method has also been shown [19].

2.1.3 GHZ states in optical setups as graphs

Besides the experimental advantages that the described method of producing entangled states provides, another big advantage to other methods is its interpretability through graph theory. By describing these experimental methods with graphs, powerful theorems from graph theory can be applied, solving complicated problems. One might, for example, wonder what the highest dimensionality GHZ state is that this method can create, given a set amount of detectors. The basic idea for translating experimental setups to graphs is to translate every output path (photon detector) to vertices and every photon pair as edges that connect the output paths the photons end up in [20]. GHZ states are created when there is exactly one photon in every output path. For the graph, this translates to that every vertex should be incident to exactly one edge. In graph theory, this is known as a perfect matching.

Definition 2.1 (Perfect matching (PM))**.** *A perfect matching is a subset of edges of a graph such that every vertex is incident to exactly one edge.*

Now consider again the experimental setup in Fig. 3. After post-selection, an (undesirable) non-GHZ quantum state is obtained. The corresponding graph

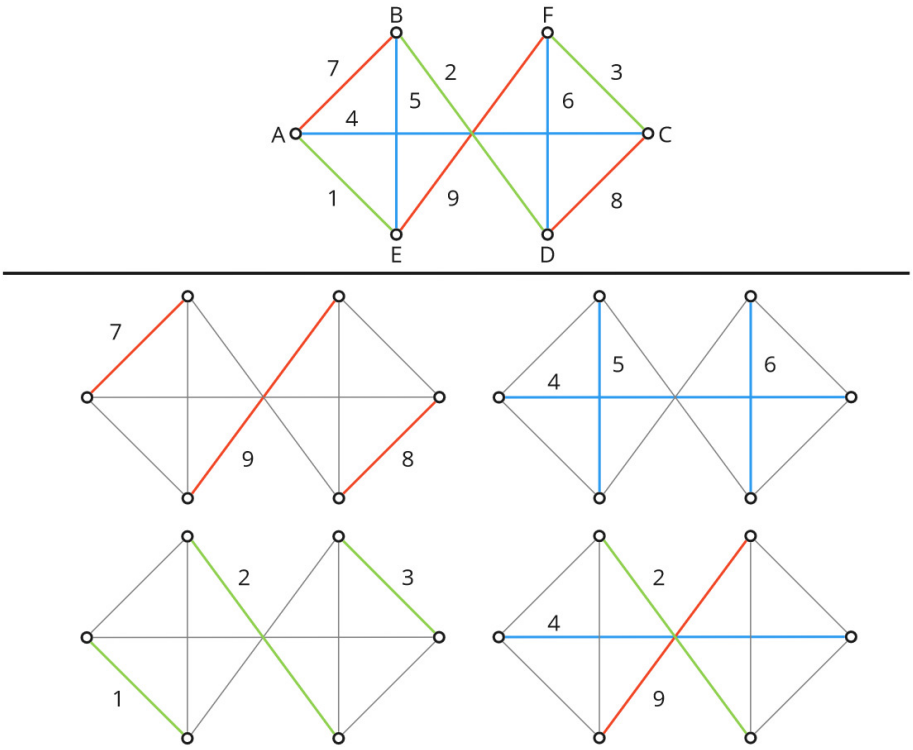

Figure 4: Graph corresponding to Fig. 3. Every vertex is an output path of a photon. An edge is drawn between vertices if the photons from the same photon pair end up in the corresponding photon output paths.

in Fig. 4 suggests another restriction, namely that the graph should exclusively have disjoint perfect matchings.

Definition 2.2 (Disjoint perfect matchings)**.** *Two perfect matchings are disjoint if their edge sets are disjoint.*

Definition 2.3 (Exclusively disjoint perfect matching graph)**.** *An exclusively disjoint perfect matching graph is a graph of which all its perfect matchings are disjoint.*

The question asked in the beginning of this section can now be answered in terms of perfect matchings.

Theorem 2.1 (Bogdanov's lemma)**.** *Given an exclusively disjoint perfect matching graph $G(V, E)$ with $|V| = 2n$. The maximum amount of perfect matchings G can have is*

$$c_{max}\,(2n) = \begin{cases} 3, & n = 2 \\ 2, & n > 2 \end{cases} \tag{2}$$

The original proof is given by Ilya Bogdanov [21], the following is a more formal version.

Definition 2.4 (Hamiltonian cycle)**.** *A Hamiltonian cycle is a sequence of distinct edges which joins a sequence of vertices, such that the first and last vertices are the same and all vertices are visited exactly once.*

Lemma 2.2. *The union of any distinct PMs M_1 and M_2 of an exclusively disjoint perfect matching graph G forms a Hamiltonian cycle.*

Proof. Let G' be the graph with edge set $M_1 \cup M_2$. Every vertex in $u \in G'$ is incident with an edge $uv \in M_1$ and $uw \in M_2$. This means that G' is two-regular, which is equivalent to the statement that it is a disjoint union of cycles.

Any of the cycles in G' have even lengths of at least four. Because the cycles are disjoint, each of these cycles take the form of C_{2n} with $n \geq 2$. For m cycles in G' there exist 2^m PMs, obtained by combining the matchings of the cycles. Combinations from different cycles give non-disjoint perfect matchings. Due to the exclusive perfect matching restriction, this means that $m = 1$. G' contains a single cycle and thus must form a Hamiltonian cycle. □

An exclusively disjoint perfect matching graph with two PMs constructed with as few edges as possible takes the shape of C_{2n}. Removing any single edge from such a graph also removes one of its PMs. The only way an exclusively disjoint perfect matching graph has three PMs is if it is at least 3-regular (i.e. every vertex has at least three neighbours). Therefore, the only way to increase the amount of PMs is to add edges.

Lemma 2.3. *Let G be an exclusively disjoint perfect matching graph of the form C_{2n} with $n > 2$. Let G' be another exclusively disjoint perfect matching graph constructed by adding any number of edges to G. G' can have maximally two PMs.*

Proof. G has an even number of vertices. This means an edge can be added that splits the cycle into two even-length or two odd-length cycles.

Suppose an edge is added that splits the graph into two even cycles. Then start constructing a new PM, starting with the new edge. The two vertices that are connected by this edge do not need to be matched anymore. That leaves $2n - 2 = 2(n - 1)$ vertices that need to be matched, where $2(n - 1)$ is split up into two paths of even length. Both of these have a PM. Therefore, a third PM can be constructed, like illustrated in Fig. 5. This new PM contains edges that original two PMs also contain, so the new PM is not disjoint.

Conversely, a single edge that splits the graph into two odd cycles can never contribute to a PM, as the two cycles, with the newly matched pair of vertices removed, form two odd length paths, which do not have a PM individually. Therefore, more edges that split the graph into odd length cycles can be added. Suppose another odd length cycle subtending edge is added that neighbours and crosses the first edge (see Fig. 6). This splits up the original cycle graph into three parts. Because two of the vertices that are connected to the new edges are

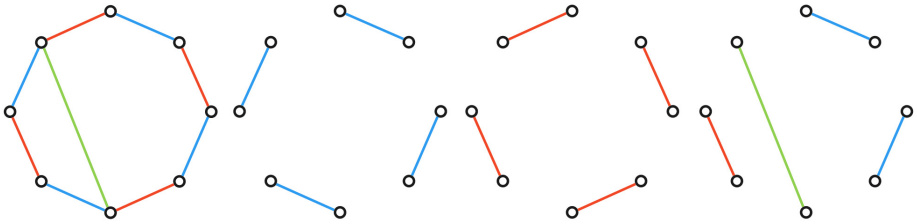

Figure 5: A new non-disjoint PM is constructed when an edge is added that splits a graph of the form C_{2n} up into two even cycles.

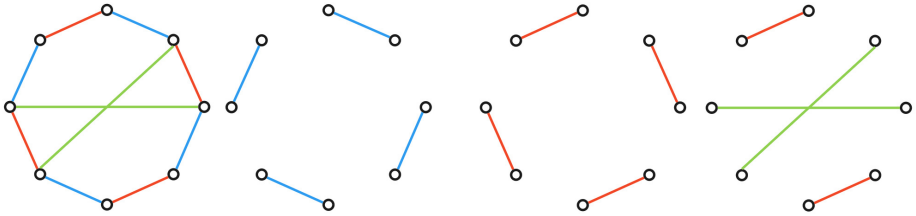

Figure 6: An edge that splits up a graph of the form C_{2n} into two odd cycles does not create a new PM. Adding a second such edge splits the graph into three parts, at which point a new, non-disjoint, PM is constructed.

neighbours (and the edges cross each other), the individual cycles they formed are shortened by one. Therefore, two of the three parts are even length paths, which each contains a PM. The only vertices that are left to be accounted for are the vertices that form the third path created by the two edges. Consider the two odd length cycles that do not contain these vertices. The length of this last path is $2n - (\widetilde{n_1} + \widetilde{n_2})$, which is even or zero, so this path also contains a PM. Altogether, this newly formed PM is not disjoint with the original two PMs.

Finally, every cycle graph where every vertex is connected to a new such odd cycle subtending edge contains at least one of these vertex neighbour pairs that have crossing edges. This happens because the minimal odd cycle length is three, leaving an unmatched vertex that needs to be matched. As its two neighbors are already matched, it needs to look for another vertex to match, which requires their connecting edge to cross the first edge. □

For experimental setups, the result in theorem 2.1 means that the dimensionality of a GHZ state can at most be two, except for the case where there are only four output paths (K_4), as that graph is too small for the arguments used in the proof above. However, so far, the effect of interference has not been taken into account. Interference could, in theory, cancel terms that are not desired, thus aiding in creating higher-dimensional GHZ states. To describe that case graph-theoretically as well, it is necessary to introduce weights and colorings to the graphs. To motivate this, note that non-disjoint PMs could be allowed

under some conditions in this formulation, as they could end up destructively interfering with each other, removing the unwanted term from the wave function. Specifically, this means that PMs that create the same state should have a combined factor of 0. To quantify, the following definitions are needed.

Definition 2.5 (Edge coloring)**.** *Every edge in a graph is assigned a color. The colors are assigned according to the mode of the state described by the edge when it arrives at the detector.*

Definition 2.6 (Inherited vertex coloring)**.** *In PMs, every vertex is incident to exactly one edge. The inherited vertex coloring (ivc) of a graph is a coloring of the vertices that is decided by the color of the edge that is incident to it.*

Definition 2.7 (Ivc weight)**.** *The weight of an ivc is defined by the following formula:*

$$w(ivc) = \sum_{PM \in \mathcal{M}(ivc)} \prod_{e \in PM} w(e)$$

At this point, every edge has a color and a (complex) weight and every PM has an ivc, which also has a weight. These definitions can be combined to form a restriction for a graph to form GHZ states.

Definition 2.8 (Monochromatic graph)**.** *A graph is called monochromatic if* $\forall$ *multi-colored ivc,* $w(ivc) = 0$ *[17].*

As opposed to counting how many disjoint PMs an exclusively disjoint PM with n vertices can have, the goal is now to find how many *ivc*'s a monochromatic graph with n vertices can have. The latter problem reduces to the former when only positive real weights are used for all edges (i.e. there are no phase shifters in the setup). This correspondence creates a lower bound of two for the amount of *ivc*'s in a monochromatic graph with $n > 4$ vertices. Although not solved analytically, all numerical evidence points to the conjecture that this lower bound is also equal to the upper bound [22]. A proof of this statement would mean that it is impossible to create such states using linear quantum optics [17].

2.2 Theorem prover

A theorem prover (TP) is a type of software that allows a user to verify mathematical statements. Any formal proof in such a system follows from an axiomatic system, in addition to a set of assumptions. The advantage over proving theorems in natural language is that the proof is unambiguous and can be checked by a computer [23]. As such, there have been numerous formalizations of difficult or controversial proofs, such as the four-color theorem, Kepler's theorem and the odd order theorem [1][24][25]. Furthermore, modern, cutting-edge mathematics has been formalized, for example in the liquid tensor experiment project [3], where the main theorem about liquid vector spaces from Fields medalist Peter Scholze, which is notoriously difficult to understand, has been verified.

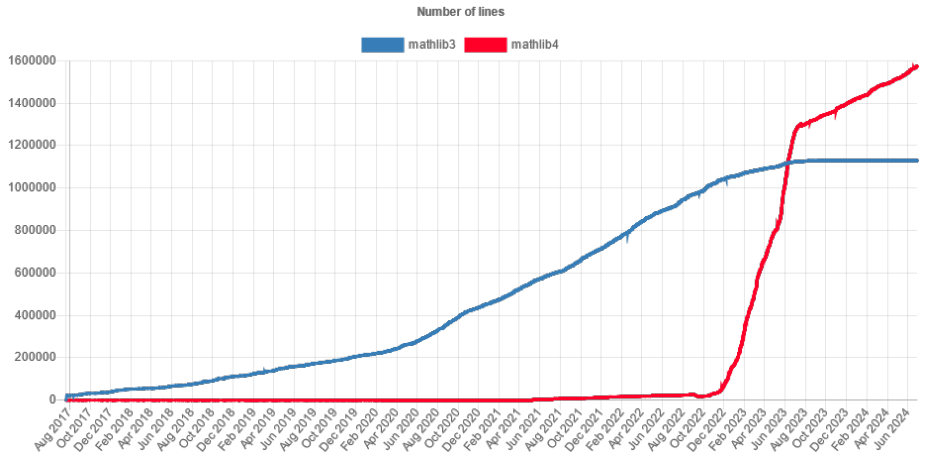

Figure 7: Number of lines in the Mathlib library [30]. The blue line represents the number of lines in Mathlib3, which has been deprecated since the switchover from Lean3 to Lean4. The number of lines added in Mathlib has seen a steady and steep linear increase since late 2020, with approximately 30000 lines being added every month.

Besides demonstrating the technical capabilities to handle complicated definitions, the extreme rigor that is necessary exposed some small mistakes in the original proof.

One way of monitoring a TP's technical ability and popularity is by tracking what amount of 100 impactful theorems have been proven in it, concretely, the list constructed by Freek Wiedijk [26]. Currently, all 100 theorems except Fermat's last theorem (FLT) have been formalized in a TP. The TPs with the highest scores are Isabelle, with the most formalized theorems at 90, followed by Coq at 79 and Lean at 76. The first attempt at formalizing FLT is underway, to be done in Lean, in a project that is aimed to be completed in five years [27].

Formalizing theorems in a proof assistant often times takes many more steps than the natural language equivalent due to the extreme rigor that is required. Therefore, it is desirable to have a program that can prove a theorem in as little lines as possible. The de Bruijn factor is a way to quantize the ratio of the number of lines needed to prove a theorem in a theorem prover compared to natural language [28]. Even with smart tactics in modern automated theorem provers (ATP), the de Bruijn factor in practice is estimated to be ~20 [29].

The modularity that comes with digitizing mathematical statements allows the construction of mathematical libraries containing different branches of mathematics. Isabelle, for example, has the Archive of Formal Proofs (AFP), MathComp for Coq, and Lean has Mathlib. AFP contains 270000 theorems in 4.4 million lines of code (LoC) (see Fig. 7), compared to Mathlib's 150000 theorems in 1.6 million LoC [31][30]. The main advantage of having such libraries is that

they can simply be imported in any project, allowing the use of all the mathematics that has been formalized in them. Another motivation for creating a large digitized mathematical library is that it provides a corpus to train AI on [32]. The state of the art for such AI models roughly solves as many problems as a silver medalist in the international mathematical olympiad [33].

Theoretical physics, being written in the language of mathematics, is also a natural candidate for formal verification using theorem provers. Indeed, several physical theories have been formalized in various theorem provers [2][34][35][5]. Doing this yields similar benefits to formalizing mathematics: the resulting proof is more objectively verifiable and the extreme rigor exposes hidden assumptions. In the future, a library for physics could be constructed, much like the mathematical libraries AFP and Mathlib, where the theory of quantum physics and general relativity could be imported with two lines of code. The motivation for creating such a library parallels that of Mathlib [5]: it could serve as a dataset to train AI, enabling new results; theorems within it are unambiguously checked and assumptions clearly exposed, and it provides a centralized repository for all proofs. Additionally, because Lean is also a functional programming language, scientific computing can be performed with existing definitions, reducing bugs by ensuring clarity in computations [36].

2.3 Lean

For this project, the Lean4 theorem prover was used, along with its main mathematical library, Mathlib. Lean is a functional programming language based on the calculus of inductive constructions that can be used as an interactive theorem prover [37]. Much of the functionality in Lean can be summarized using functions and theorems. The following is an example of a function:

```
def add (n m : ℕ) : ℕ := n + m
```

The parentheses after the function name define the input parameters of the function along with their types. The object after the semicolon is the output type of the function. Everything after the `:=` symbol is what the function executes. In the example, the function takes in two natural numbers and returns a natural number that is the sum of the two input numbers. A theorem that uses this definition:

```
theorem add_zero (n m : ℕ) (hm : m = 0) : add n m = n :=
    by                -- ⊢ add n m = n
    simp [add]        -- ⊢ n + m = n
    rw [hm]           -- ⊢ n + 0 = n
    simp [add_zero]   -- No goals
```

Like for a function, the parameters in parentheses can be seen as input variables for a theorem. The statement after the semi-colon is what is to be proven. `hm` represents a hypothesis about m. Everything after the `:=` symbol are steps to proving the statement. The comments represent the proof state after every step. In Lean, a statement can be proven in `term` or `tactic` mode. Terms are

single expressions using lambda calculus which allows the user to 'talk' directly to the Lean kernel. Tactics are more intuitive commands that manipulate the proof state. `Term` mode is activated by default; `tactic` mode can be entered using the `by` keyword. In the example above, the goal is to prove that the function call `add n m` returns the same value as `n`, when `m` is assumed to be zero. The goal is proven in `tactic` mode using the `simp` tactic, which simplifies the goal using a list of rules or lemmas. The proof states clearly show what each step does: `simp [add]` uses the function `add` to rewrite `add n m` to `n + m`. Then, the tactic `rw` is used to rewrite the proof state using the assumption `hm`. `simp [add_zero]` then applies the lemma `add_zero` to rewrite `n + 0` to `n`, at which point the theorem has been proven.

In Mathlib, a graph is defined by the following:

```
structure SimpleGraph (V : Type u) where
    Adj : V → V → Prop
    symm : Symmetric Adj := by aesop_graph
    loopless : Irreflexive Adj := by aesop_graph
```

All that this means is that a `SimpleGraph` is a symmetric and irreflexive adjacency relation on a vertex type `V`. To elaborate, for any two vertices v and w of type `V`, their adjacency relation is a `Prop` (proposition), so `True` or `False`. This adjacency relation is restricted to be symmetric (i.e. adjacency between v and w implies adjacency between w and v) and loopless (i.e. v is not adjacent to v). These last two restrictions are proven using the `aesop_graph` tactic (Automated Extensible Search for Obvious Proofs for graphs).

A subgraph of a graph is a matching if it satisfies the following definition:

```
def IsMatching {V : Type u} {G : SimpleGraph V} (M : Subgraph G) : Prop
    := ∀ {v}, v ∈ M.verts → ∃! w, M.Adj v w
```

The different brackets that can be seen here are merely a technical difference from the normal brackets - they achieve the same thing. Apart from that, the proof reads like the following natural language: for all vertices v, such that v is in the set of vertices of subgraph M, then there exists a unique vertex w, such that v and w are adjacent in M. This captures the meaning of a matching because every vertex v is adjacent to exactly one other vertex. The definition of a perfect matching then follows naturally:

```
def IsPerfectMatching {V : Type u} {G : SimpleGraph V} (M : Subgraph G)
    : Prop := M.IsMatching ∧ M.IsSpanning
```

This means that a subgraph M is a perfect matching if it is a matching and it spans the entire graph G. Note that because the function `IsMatching` is defined on a subgraph, it can be equivalently called with `M.IsMatching` or `IsMatching M` - the same goes for `IsSpanning`.

3 Results

The proof of Bogdanov's lemma requires the notion of a Hamiltonian cycle, which does not exist yet in Mathlib. To get there, first a definition of a Hamil-

tonian path is needed:

```
def IsHamiltonian (p : G.Walk a b) : Prop := ∀ a, p.support.count a = 1
```

Clearly, this definition suggests that a walk p from vertex a to vertex b is Hamiltonian if every vertex in p is visited exactly once. It is then possible to proof that the support of p is the entire vertex set:

```
lemma IsHamiltonian.support_toFinset (hp : p.IsHamiltonian) :
    p.support.toFinset = Finset.univ :=
    by                          -- ⊢ p.support.toFinset = univ
    simp [eq_univ_iff_forall] -- ⊢ ∀ (x : α), x ∈ p.support
    simp [hp]                   -- No goals
```

In the first step the lemma `eq_univ_iff_forall: s = univ ↔ ∀ (x : α), x ∈ s` is used to transform the proof to `∀ (x : α), x ∈ p.support`, which can be closed by using the definition of a Hamiltonian path which is assumed in `hp`.

This definition of a Hamiltonian walk can be used to extend a cycle:

```
structure IsHamiltonianCycle (p : G.Walk a a) extends p.IsCycle : Prop :=
    isHamiltonian_tail : (p.tail toIsCycle.not_nil).IsHamiltonian
```

This definition suggests that a cycle p is a Hamiltonian cycle if its tail is a Hamiltonian walk, where the tail is p without the first half edge. Of course, a graph G that is a Hamiltonian cycle should be connected, i.e., every vertex a, b $\in G$ should be able to reach each other with some walk:

```
lemma IsHamiltonian.connected (hG : G.IsHamiltonian) : G.Connected where
    preconnected a b := by
        -- (p: G.Walk v† w†)
        obtain rfl | hab := eq_or_ne a b -- (a b : α), (hab : a ≠ b)
        · rfl
        have : Nontrivial α := ⟨a, b, hab⟩
        obtain ⟨_, p, hp⟩ := hG Fintype.one_lt_card.ne'
        have : Nontrivial α := ⟨a, b, hab⟩
        obtain ⟨_, p, hp⟩ := hG Fintype.one_lt_card.ne'
        -- (p : G.Walk w† w†), (hp : p.IsHamiltonianCycle)
        have a_mem : a ∈ p.support := hp.mem_support a
        have b_mem : a ∈ p.support := hp.mem_support b
        have walk_a : G.Walk w† a := p.takeUntil a a_mem
        have walk_b : G.Walk w† b := p.takeUntil b b_mem
        have walk_a_b : G.Walk a b := (id walk_a.reverse).append walk_b
        exact Walk.reachable walk_a_b
    nonempty := not_isEmpty_iff.1 fun _ ↦ by simpa using hG $ by simp
    [@Fintype.card_eq_zero]
```

The connectedness proof has two cases: $a = b$ and $a \neq b$. The graph constructed by the first case is always connected and can be closed by the `rfl` tactic. For the other case, it is first necessary to retrieve the variables `p` and `hp`, like shown in the comments. Next, because `p` is a Hamiltonian cycle and thus contains all vertices, `a` and `b` are shown to be in `p`. From this, a walk in `p` can be constructed between a vertex `w†` and `a` and between vertex `w†` and `b`. Finally, these two walks

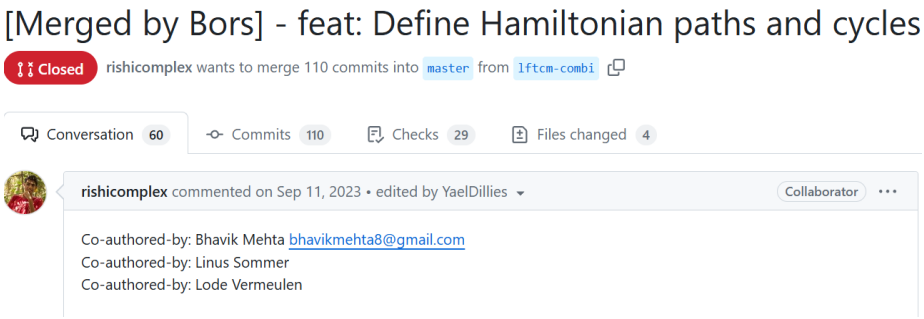

Figure 8: Merged pull request of Hamiltonian paths and cycles in Mathlib.

share the same vertex `w✝`, so they can be appended into a single walk `G.Walk a b`, which closes the proof. Furthermore, the definition of `Connected` requires G to be non-empty, which is obvious, but technically tricky to prove. The above definitions and lemmas about Hamiltonian walks and cycles are now part of Mathlib, see Fig. 8 [38].

Now that Hamiltonian cycles are in Mathlib, Bogdanov's lemma can be proven. This part of the project was completed with Pjotr Buys, who utilized the author's setup of definitions, problem statement, and initial lemmas, and ended up contributing the majority of the proofs towards Bogdanov's lemma [39]. An exclusively disjoint PM graph can be defined in Lean as the following:

```
def HasExclusivelyDisjointPM (G : SimpleGraph V) : Prop :=
    ∀ (M1 M2 : Subgraph G), (M1.IsPerfectMatching ∧
    M2.IsPerfectMatching ∧ M1 ≠ M2) → M1.edgeSet ∩ M2.edgeSet = ∅
```

This definition assumes that for all pairs of distinct perfect matchings, the intersection of their edgesets is empty. With this definition, it can be proven that the union of the two perfect matchings from an exclusively disjoint perfect matchings graph forms a Hamiltonian cycle:

```
theorem TwoMatchingsHamiltonian [Fintype V] [DecidableEq V] (G :
    SimpleGraph V) (M1 M2 : Subgraph G)
    (hprop : HasExclusivelyDisjointPM G) (hM1 : IsPerfectMatching M1)
    (hM2 : IsPerfectMatching M2) (hdif : M1 ≠ M2) :
    IsHamiltonianCycle (M1 ⊔ M2) := by
    -- ⊢ IsCycleSubgraph G (M1 ⊔ M2) ∧ (M1 ⊔ M2).IsSpanning
    refine ⟨?isCycle,?isSpanning⟩
    . rw [←SubConnectedImpCycleIffTwoReg]
        -- ⊢ SubgraphIsRegular G (M1 ⊔ M2) 2 ∧ (M1 ⊔ M2).Connected
        have degM1 := DegreePerfectMatching hM1 -- M1 is 1-regular
        have degM2 := DegreePerfectMatching hM2 -- M2 is 1-regular
        have sameVerts := SpSubEqSupp hM1.2 hM2.2
        have disjEdges := hprop M1 M2 ⟨hM1, hM2, hdif⟩
        . exact DegreeSubgraphAdd M1 M2 degM1 degM2 sameVerts disjEdges
        . exact DisjointPMGraphImpCon G hprop M1 M2 hM1 hM2 hdif
    . exact SpUnionLeft hM1.2
```

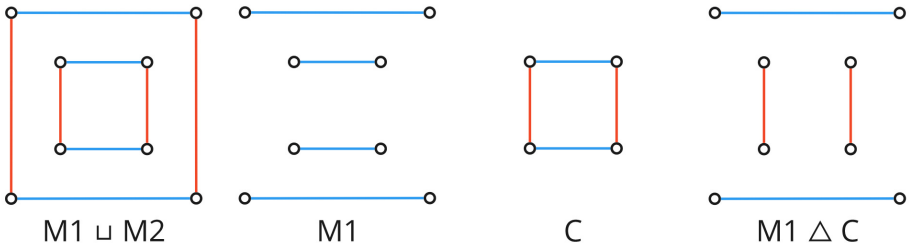

Figure 9: Constructing of a new perfect matching in a disjoint union of cycles using the symmetric difference operator $\triangle$. The symmetric difference operator working on sets gives the set of elements which are in either of the sets, but not in their intersection.

The proof above begins by dividing the main goal into two subgoals: proving that the union of the two perfect matchings forms a cycle and that it spans the entire graph.

To address the first subgoal, the lemma `SubConnectedImpCycleIffTwoReg : SubgraphIsRegular G H 2 ↔ IsCycleSubgraph G H` is used. This lemma states that a subgraph is a cycle if and only if it is 2-regular and connected. To use the cycle statement in this lemma, it needs to be shown that `M1 ⊔ M2` is 2-regular and connected. Generally, the union of two edge-disjoint subgraphs with the same vertex set and degrees d_1 and d_2 will be $(d_1 + d_2)$-regular. Since both `M1` and `M2` fulfill these assumptions and are 1-regular, their union is 2-regular, established using `DegreeSubgraphAdd : SubgraphIsRegular G (H1 ⊔ H2) (d1 + d2)`. The lemma `DisjointPMGraphImpCon : (M1 ⊔ M2).Connected` is then applied to demonstrate that the union of the two disjoint perfect matchings is connected, confirming that a cycle is formed.

The second subgoal is resolved using the lemma `SpUnionLeft (hSpan : H1.isSpanning) : (H1 ⊔ H2).IsSpanning`, which states that the union of two subgraphs spans the entire graph if at least one of the subgraphs spans it.

Both the proof in Lean and the proof in lemma 2.2 use the notion of 2-regularity to show the existence of some type of cycle structure. Whereas the natural language proof uses a high-level argument in four lines that there must be only one cycle, the Lean proof of connectedness spans ~200 LoC in total. The method used in the Lean proof is elegant: if $M1 \sqcup M2$ is not connected, a new, non-disjoint PM can be constructed using the symmetric difference operator, as illustrated in Fig. 9. Since this violates the exclusive PM property, $M1 \sqcup M2$ must be connected.

4 Discussion

In this thesis, part of the proof of Bogdanov's lemma has been formalized in the Lean4 theorem prover. This verifies that this part of the proof holds when it is checked down to the axioms of mathematics. Furthermore, it acts as a

proof of concept for formalizing and digitizing statements pertinent to physics. A complete proof of Bogdanov's lemma could in the future be used to train an AI model on, which could help to make discoveries about this graph-theoretical statement, which then has direct consequences to the constructability of entangled states in quantum optics.

By adding definitions and lemmas about Hamiltonian paths and cycles to Mathlib, other projects can now also use or further develop this branch of graph theory. Additionally, this code contributes to a bigger corpus for AI to train on.

4.1 Personal experiences with writing Lean

Formalizing the entirety of Bogdanov's lemma was not achieved, mainly because proofs in Lean require a different structure than those in natural language. Mathematical proofs in natural language often lack the rigor necessary for direct formalization, partly due to implicit assumptions. For instance, proving that a Hamiltonian cycle is connected could be summarised in one or two sentences in natural language, but in Lean it requires ∼10 LoC, which corresponds to a de Bruijn factor of 5-10. One way of making formalization easier is to reduce this factor by, for example, introducing more advanced tactics. Additionally, training a better AI model could potentially make computer-based theorem proving even easier than the traditional way on a blackboard.

While setting up the environment and stating assumptions and theorems becomes familiar quickly, the proofs—particularly in Mathlib—can often appear cryptic. This is partly because the code is prioritized for conciseness over readability. To overcome this, it's helpful to go through proofs line by line to understand how the proof state changes, which helps in getting to know the function of various tactics.

Several resources are available to support learning to program in Lean. The extensive online learning materials and the active, supportive community on Zulipchat are invaluable. Especially the natural numbers game is a great resource to get to know theorem proving in Lean. Moreover, Lean includes smart tactics that attempt to prove statements by searching through the Mathlib library, such as `apply?`, `exact?` or `aesop`. Finally, AI assistants like Moogle, Loogle and Leandojo further support Lean development.

5 Conclusion

In quantum optics, the generation of GHZ states is of high interest. A new method makes use of overlapping paths, which removes the information about the photon's origin, facilitating the construction of GHZ states. This type of experimental setup can be directly translated into graph theory, where a perfect matching corresponds to an ideal photon creation case that contributes to the desired state. Additionally, the graph representing the experimental setup is constrained to contain exclusively disjoint perfect matchings. Using this new formulation of the optical setup, one can state the question of the highest di-

mensional GHZ state that can be created for any number of particles, using the same optical instruments. This thesis formalizes part of the proof of that statement in the Lean theorem prover. This result serves as a proof of concept that this graph theoretical statement, and consequently the quantum optical statement, can be formalized in a theorem prover.

In future research, lemma 2.3 should also be formalized in order to fully digitize Bogdanov's lemma. Furthermore, the generalized conjecture about ivc weights can be stated and a lower bound can be proven. This could then be used alongside advanced theorem-proving techniques, such as clever tactics or AI models, to provide insights into this unresolved case.

6 Acknowledgements

I would like to enormously thank my supervisor Dr. Mario Krenn for providing this research opportunity and assisting me with my research and all the theoretical background that was necessary for it. I would also like to thank Dr. Pjotr Buys for the helpful chats about graph theory in Lean and his (massive) contributions to formalizing part of Bogdanov's lemma. Finally, I would like to thank the community on Zulipchat for their help throughout my project.

7 References

[1] G. Gonthier *et al.*, "Formal proof–the four-color theorem," *Notices of the AMS*, vol. 55, no. 11, pp. 1382–1393, 2008.

[2] M. Stannett and I. Németi, "Using isabelle/hol to verify first-order relativity theory," *Journal of automated reasoning*, vol. 52, pp. 361–378, 2014.

[3] P. Scholze, "Liquid tensor experiment," *Experimental Mathematics*, vol. 31, no. 2, pp. 349–354, 2022.

[4] T. mathlib Community, "The lean mathematical library," in *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, (New York, NY, USA), p. 367–381, Association for Computing Machinery, 2020.

[5] J. Tooby-Smith, "Heplean: Digitalising high energy physics," *arXiv preprint arXiv:2405.08863*, 2024.

[6] K. Yang, A. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. J. Prenger, and A. Anandkumar, "Leandojo: Theorem proving with retrieval-augmented language models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[7] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, "Quantum entanglement," *Reviews of Modern Physics*, vol. 81, p. 865–942, June 2009.

[8] W. Dür, G. Vidal, and J. I. Cirac, "Three qubits can be entangled in two inequivalent ways," *Physical Review A*, vol. 62, Nov. 2000.

[9] N. Brunner, D. Cavalcanti, S. Pironio, V. Scarani, and S. Wehner, "Bell nonlocality," *Rev. Mod. Phys.*, vol. 86, pp. 419–478, Apr 2014.

[10] A. Cervera-Lierta, M. Krenn, A. Aspuru-Guzik, and A. Galda, "Experimental high-dimensional greenberger-horne-zeilinger entanglement with superconducting transmon qutrits," *Phys. Rev. Appl.*, vol. 17, p. 024062, Feb 2022.

[11] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn, *et al.*, "A race-track trapped-ion quantum processor," *Physical Review X*, vol. 13, no. 4, p. 041052, 2023.

[12] Z. Bao, S. Xu, Z. Song, K. Wang, L. Xiang, Z. Zhu, J. Chen, F. Jin, X. Zhu, Y. Gao, Y. Wu, C. Zhang, N. Wang, Y. Zou, Z. Tan, A. Zhang, Z. Cui, F. Shen, J. Zhong, T. Li, J. Deng, X. Zhang, H. Dong, P. Zhang, Y.-R. Liu, L. Zhao, J. Hao, H. Li, Z. Wang, C. Song, Q. Guo, B. Huang, and H. Wang, "Schrödinger cats growing up to 60 qubits and dancing in a cat scar enforced discrete time crystal," 2024.

[13] P. Thomas, L. Ruscio, O. Morin, and G. Rempe, "Efficient generation of entangled multiphoton graph states from a single atom," *Nature*, vol. 608, no. 7924, pp. 677–681, 2022.

[14] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai, *et al.*, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, vol. 356, no. 6343, pp. 1140–1144, 2017.

[15] J.-W. Pan, M. Daniell, S. Gasparoni, G. Weihs, and A. Zeilinger, "Experimental demonstration of four-photon entanglement and high-fidelity teleportation," *Phys. Rev. Lett.*, vol. 86, pp. 4435–4438, May 2001.

[16] M. Krenn, A. Hochrainer, M. Lahiri, and A. Zeilinger, "Entanglement by path identity," *Physical review letters*, vol. 118, no. 8, p. 080401, 2017.

[17] X. Gu, M. Erhard, A. Zeilinger, and M. Krenn, "Quantum experiments and graphs ii: Quantum interference, computation, and state generation," *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4147–4155, 2019.

[18] J. Bao, Z. Fu, T. Pramanik, J. Mao, Y. Chi, Y. Cao, C. Zhai, Y. Mao, T. Dai, X. Chen, X. Jia, L. Zhao, Y. Zheng, B. Tang, Z. Li, J. Luo, W. Wang, Y. Yang, Y. Peng, D. Liu, D. Dai, Q. He, A. L. Muthali, L. K. Oxenløwe, C. Vigliar, S. Paesani, H. Hou, R. Santagati, J. W. Silverstone, A. Laing, M. G. Thompson, J. L. O'Brien, Y. Ding, Q. Gong, and J. Wang, "Very-large-scale integrated quantum graph photonics," *Nature Photonics*, vol. 17, pp. 573–581, Jul 2023.

[19] K. Qian, K. Wang, L. Chen, Z. Hou, M. Krenn, S. Zhu, and X.-s. Ma, "Multiphoton non-local quantum interference controlled by an undetected photon," *Nature Communications*, vol. 14, no. 1, p. 1480, 2023.

[20] M. Krenn, X. Gu, and A. Zeilinger, "Quantum experiments and graphs: Multiparty states as coherent superpositions of perfect matchings," *Physical review letters*, vol. 119, no. 24, p. 240403, 2017.

[21] I. B. (`https://mathoverflow.net/users/17581/ilya-bogdanov`), "Graphs with only disjoint perfect matchings." MathOverflow. URL:`https://mathoverflow.net/q/267013` (version: 2017-04-12).

[22] C. Ruiz-Gonzalez, S. Arlt, J. Petermann, S. Sayyad, T. Jaouni, E. Karimi, N. Tischler, X. Gu, and M. Krenn, "Digital discovery of 100 diverse quantum experiments with pytheus," *Quantum*, vol. 7, p. 1204, 2023.

[23] Y. Kassios, "Formal proof," *Diunduh dari http://www. cs. toronto. edu/˜ hehner/aPToP/formal proof-1. pdf tanggal*, vol. 26, 2009.

[24] T. Hales, M. Adams, G. Bauer, T. D. Dang, J. Harrison, H. Le Truong, C. Kaliszyk, V. Magron, S. McLaughlin, T. T. Nguyen, *et al.*, "A formal proof of the kepler conjecture," in *Forum of mathematics, Pi*, vol. 5, p. e2, Cambridge University Press, 2017.

[25] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. Ould Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, and L. Théry, "A machine-checked proof of the odd order theorem," in *Interactive Theorem Proving* (S. Blazy, C. Paulin-Mohring, and D. Pichardie, eds.), (Berlin, Heidelberg), pp. 163–179, Springer Berlin Heidelberg, 2013.

[26] F. Wiedijk, *The seventeen provers of the world: Foreword by Dana S. Scott*, vol. 3600. Springer, 2006.

[27] K. Buzzard, "The fermat's last theorem project," Apr 2024.

[28] F. Wiedijk, "The de bruijn factor," 2000.

[29] T. Tao, "Terence tao, "machine assisted proof"." URL:`https://www.youtube.com/watch?v=AayZuuDDKP0&t=3004s`.

[30] leanprover community, "Mathlib statistics." URL:`https://leanprover-community.github.io/mathlib_stats.html` (version: 2024-06-21).

[31] isa afp.org, "Archive of formal proofs statistics." URL:`https://www.isa-afp.org/statistics/` (version: 2024-06-21).

[32] P. Lu, L. Qiu, W. Yu, S. Welleck, and K.-W. Chang, "A survey of deep learning for mathematical reasoning," *arXiv preprint arXiv:2212.10535*, 2022.

[33] S. Polu, J. M. Han, K. Zheng, M. Baksys, I. Babuschkin, and I. Sutskever, "Formal mathematics statement curriculum learning," *arXiv preprint arXiv:2202.01344*, 2022.

[34] J. Boender, F. Kammüller, and R. Nagarajan, "Formalization of quantum protocols using coq," *arXiv preprint arXiv:1511.01568*, 2015.

[35] M. P. Bobbin, S. Sharlin, P. Feyzishendi, A. H. Dang, C. M. Wraback, and T. R. Josephson, "Formalizing chemical physics using the lean theorem prover," *Digital Discovery*, vol. 3, no. 2, pp. 264–280, 2024.

[36] T. Skřivan, "Scilean." `https://github.com/lecopivo/SciLean`, 2024.

[37] L. d. Moura and S. Ullrich, "The lean 4 theorem prover and programming language," in *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pp. 625–635, Springer, 2021.

[38] leanprover community, "Hamiltonian pr in mathlib4." `https://github.com/leanprover-community/mathlib4/commit/b4911b22a10c9f351458621d3b613dd43eaad39c`, 2024.

[39] LodeVermeulen, "Lean4_bogdanovs_lemma." `https://github.com/LodeVermeulen/Lean4_Bogdanovs_lemma`, 2024.