Meta-Designing Quantum Experiments with Language Models

Sören Arlt [1] **Haonan Duan** [2] **Felix Li** [3] **Sang Michael Xie** [4] **Yuhuai Wu** [5] **Mario Krenn** [1]

Abstract

Artificial Intelligence (AI) has the potential to significantly advance scientific discovery by finding solutions beyond human capabilities. However, these super-human solutions are often unintuitive and require considerable effort to uncover underlying principles, if possible at all. Here, we show how a code-generating language model trained on synthetic data can not only find solutions to specific problems but can create meta-solutions, which solve an entire class of problems in one shot and simultaneously offer insight into the underlying design principles. Specifically, for the design of new quantum physics experiments, our sequence-to-sequence transformer architecture generates interpretable Python code that describes experimental blueprints for a whole class of quantum systems. We discover general and previously unknown design rules for infinitely large classes of quantum states. The ability to automatically generate generalized patterns in readable computer code is a crucial step toward machines that help discover new scientific understanding – one of the central aims of physics.

1. Introduction

Quantum physics is a notoriously unintuitive field of study. Despite this, it has developed to a point where some of its most difficult-to-conceptualize effects - such as entanglement - could become the basis of a new generation of technological development. These applications include quantum imaging (Lemos et al., 2014; Kviatkovsky et al., 2020; Moreau et al., 2019; Aslam et al., 2023), quantum metrology (Pezzè et al., 2018; Polino et al., 2020; DeMille et al., 2024), quantum communication (Flamini et al., 2018; Couteau et al., 2023), quantum simulation (Aspuru-Guzik & Walther, 2012; Cornish et al., 2024), and quantum computation (Madsen et al., 2022). Due to difficulties in designing experimental setups by hand, researchers have started to utilize algorithmic optimization and AI techniques to discover experimental setups in quantum physics (Krenn et al., 2020).

Conventional computational design can deliver solutions, which surpass designs by human experts. E.g. for a given target quantum state, a machine could design the experimental setup which creates the state, but interpretation and generalization of the results is left to the researcher and is often an exceptionally hard challenge, if possible at all.

Here we introduce the process of *meta-design*, which takes computational design to a higher level of abstraction. Instead of designing one solution for a single target (i.e. one experimental setup for the creation of one quantum state), our algorithm designs a meta-solution which generates solutions for an infinitely large class of targets (a class of quantum state). In our case, these meta-solutions have the form of programming codes, which generate different experimental setups for different values of an integer variable `N`. At the heart of our method lies a language model (sequence-to-sequence transformer) which is able to propose meta-solutions in the form of Python codes. Our approach is successful in designing meta-solutions for many interesting classes of quantum states, several of which were not known previously. Due to the readability of the solutions that stem from their Python code representation, it is easy to uncover the underlying principles of the generalized meta-solutions. Therefore, our technique is a step towards AI methods that can help to gain new understanding in physics (De Regt, 2017; Krenn et al., 2022; Barman et al., 2024).

2. Related Work

AI for discovery in quantum physics. AI techniques have been previously applied to the search for experimental setups in quantum physics (Krenn et al., 2016; Knott, 2016; Nichols et al., 2019; Wallnöfer et al., 2020; Prabhu et al., 2020; Krenn et al., 2021; Ruiz-Gonzalez et al., 2023; Goel et al., 2024; Landgraf et al., 2024), nanophotonic structures (Molesky et al., 2018; Sapra et al., 2020; Ma et al., 2021; Gedeon et al., 2023), and quantum circuits (Ostaszewski et al., 2021; Nägele & Marquardt, 2023; Kottmann, 2023;

[1]Max Planck Institute for the Science of Light [2]University of Toronto, Vector Institute [3]UC Berkeley [4]Stanford University [5]xAI. Correspondence to: Sören Arlt <soeren.arlt@mpl.mpg.de>, Haonan Duan <haonand@cs.toronto.edu>, Yuhuai Wu <yuhuai@x.ai>, Mario Krenn <mario.krenn@mpl.mpg.de>.

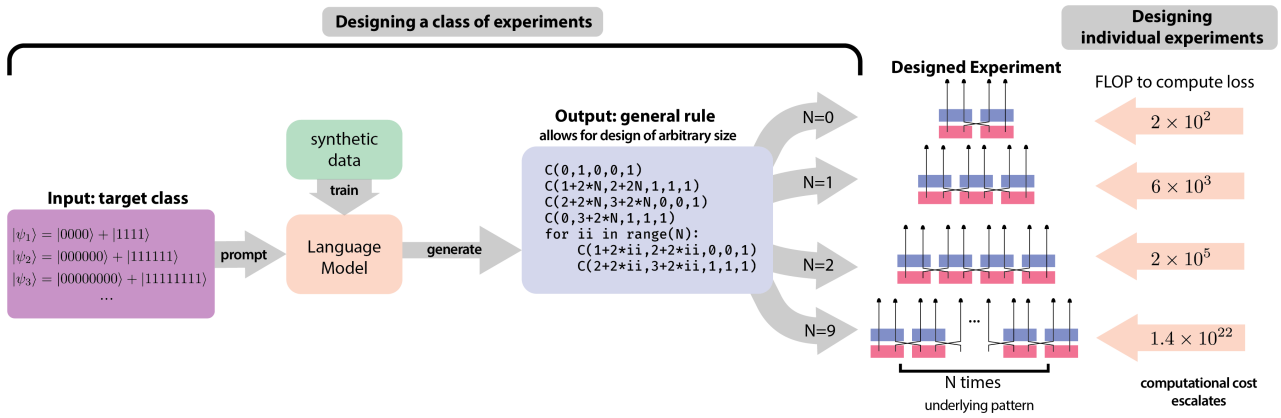

Figure 1. **Meta-designing a class of experiments via code generation avoids exploding computational costs for the design of larger experiments.** Left side: Our process takes the first three states from a class of target quantum states and - when successful - produces a Python code which generates the correct experimental setup for arbitrary sizes. For this, a sequence-to-sequence transformer is trained purely on randomly generated pairs of sequences. Right side: Designing an experimental setup which produces a target quantum state is very fast for small particle numbers. But the computational cost explodes as the target state grows.

Zen et al., 2024; MacLellan et al., 2024). All of these works have in common that the algorithm produces only a single solution and not a meta-solution that represents large classes of solutions.

LLMs for program synthesis Language models have been explored in the field of program synthesis. One approach is to pre-train these models directly on massive datasets of programming language data, such as GitHub pull requests, Kaggle notebooks, and code documentation. Models like StarCoder (Li et al., 2023; Lozhkov et al., 2024), and CodeGeeX (Zheng et al., 2023) fall into this category. An alternative strategy is to take existing large natural language models and then fine-tune them on code data. For example, Codex (Chen et al., 2021) fine-tuned GPT3 (Brown et al., 2020) on the source code from the public available sources, and CodeGemma is an adpation from the Gemma (Gemma-Team, 2024) models. There has also been research exploring the use of program synthesis techniques as a tool to enhance the algorithmic reasoning capabilities of large language models. Methods like PAL (Gao et al., 2023) and PoT (Chen et al., 2022) prompt language models to break down reasoning problems into a series of intermediate steps. These step-by-step decompositions are then offloaded to an external runtime environment, such as a Python interpreter, to execute and solve each step programmatically. Another technique called Think-and-Execute (Chae et al., 2024) follows a similar philosophy, where the language model is tasked with generating high-level pseudocode that outlines the solution approach for a given problem. This pseudocode is then simulated and executed, allowing the model to reason through the problem in a structured, interpretable manner. In addition, program synthesis has been used to explore mechanistic interpretability for LLMs. Michaud et al. (2024) translate algorithms encoded by machine-learning models into human-readable Python code. LLMs have been used to generate programs to solve mathematical problems (Romera-Paredes et al., 2024) and to discover scientific equations from data (Shojaee et al., 2024).

Transformers for math and physics Transformer architectures have demonstrated remarkable success in solving a wide range of mathematics and physics reasoning tasks. Lample & Charton (2019) and Kamienny et al. (2022) show that a transformer-based sequence-to-sequence model can tackle symbolic math problems such as symbolic integration, differential equations and symbolic regression. AlphaGeometry (Trinh et al., 2024) has achieved remarkable performance in solving geometry problems at an olympiad level. Alfarano et al. (2023) finds that by training transformers on synthetic data, they can accurately predict the Lyapunov functions of polynomial and non-polynomial dynamical systems. In the field of theoretical high-energy physics, Cai et al. (2024) applies transformers to compute scattering amplitudes. Furthermore, Alnuqaydan et al. demonstrates that a transformer model, when trained on symbolic sequence pairs, can correctly predict the squared amplitudes of Standard Model processes. Language models have also been used in quantum simulation (Melko & Carrasquilla, 2024), albeit not as a generative model for symbolic language as in our or the other works mentioned here.

3. Background Quantum Optics

We choose the design of quantum optics experiments as proof of concept and point to the great potential in applying the approach in other fields. Quantum optics is concerned

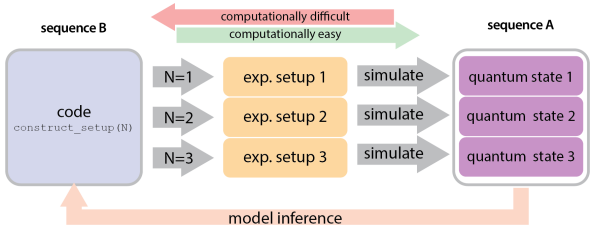

Figure 2. **Exploiting asymmetric cost for data generation.** A random code (sequence B) is generated. Executing it for the values $N = 0, 1, 2$ produces in three different experimental setup. Each setup produces a state. The three states are concatenated to make sequence A, which is the input for the model.

with photons, the fundamental particles of light. A photon can have different polarization modes, e.g. horizontal (mode 0) or vertical (mode 1). A basic property of quantum particles is that they can be in a superposition of multiple modes, i.e. they can be considered to be two things at the same time. The state $|\psi\rangle$ of one photon in equal superposition can be expressed in Dirac notation as

$$|\psi\rangle = |0\rangle + |1\rangle \,. \tag{1}$$

We omit the normalization factor for all quantum states shown in this work for readability. It can be assumed that all states are normalized. Another important concept in quantum physics is *entanglement*, where multiple photons are in a state where they cannot be described independently, such as the superposition of three particles being in the superposition of either all particles being in mode 0 or all particles being in mode 1,

$$|\psi\rangle = |000\rangle + |111\rangle \,. \tag{2}$$

This state is called the GHZ state (Greenberger et al., 1990; Pan et al., 2000).
In quantum optics, highly entangled states can be created by combining probabilistic photon pair sources. The number of possible experimental setups increases combinatorically with the number of photons for a given target state. This makes it very difficult to design experiments by hand and thus computational techniques have been successfully applied to the problem (Ruiz-Gonzalez et al., 2023). For sufficiently large systems, these tasks become too difficult even for current methods as they become too computationally expensive (see right side of Fig. 1).

4. Methods

We introduce meta-design, the idea of generating a meta-solution that can solve a whole class of solutions (in our case, for design problems of quantum states). Our meta-solutions are Python codes that can generate blueprints of experimental setups. We train a sequence-to-sequence transformer on synthetic data to translate from a class of quantum states to Python code and sample the model to discover programs for a collection of target classes.

Solving classes of problems via meta-solutions – High-level programming languages like python are universally computationally expressive and are human-readable, making them perfect to express general concepts. Let's consider a simple example: Let's say we want to describe the action of drawing a polygon (triangle, square, pentagon, ...) for an arbitrary number of sides $N \geq 3$.

```
def draw_polygon(N):
    for i in range(N):
        draw_forward()
        turn_left(360 / N)
```

where `draw_forward()` draws a straight line of a given length and `turn_left(angle)` rotates the drawing direction. This simple program expresses a whole class that follows a pattern. The class contains an infinite number of shapes. For the problem *Draw a polygon with 7 sides*, the above code for $N = 7$ is a solution. The code itself with the variable N, however, is a meta-solution that can produce many different solutions (in this case, a prescription to draw a polygon with a specific number of sides N). This principle can be generalized, and we use it to address design questions for general cases.

Meta-design for Quantum Experiments A famous class of quantum states are the GHZ states, which are written expressed in ket notation as

$$|GHZ_4\rangle = |0000\rangle + |1111\rangle$$
$$|GHZ_6\rangle = |000000\rangle + |111111\rangle$$
$$|GHZ_8\rangle = |00000000\rangle + |11111111\rangle$$

They are superposition of particles being either in mode 0 or mode 1 with an increasing number of photons (4, 6, 8, ...). We can state the design of quantum optics experiments for all GHZ states as a meta-design problem as in the previous paragraph. A meta-solution for this problem is a program `construct_setup(N)` which generates the correct experimental setup for a given $N \geq 0$. This is possible because the GHZ states follow a specific pattern. The solution to the problem is shown in Fig. 1. The setup for n particles consists of n paths leading to n detectors. The function call `C(p1,p2,m1,m2,w)` denotes placing a photon pair source at a crossing of paths `p1` and `p2`, creating photons with modes `m1` and `m2` (shown as color) and a weight `w` (introducing possible phases). After constructing the setup, we can compute the expected quantum state that emerges at the detectors. The code shown in Fig. 1 will generate the correct experimental setups for arbitrarily high particle numbers. Much like the program for *N-polygons*,

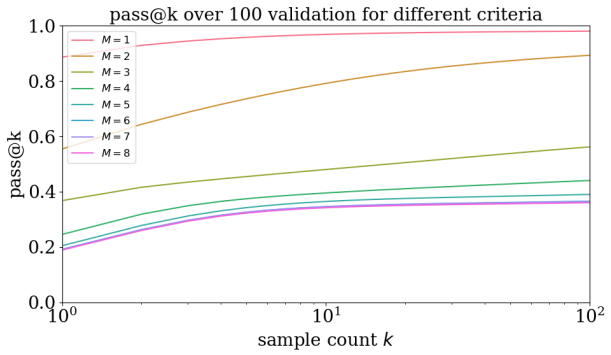

Figure 3. **Success of meta-design on validation data.** `pass@k` is the probability for producing a correct solution within k samples. We show the metric for different criteria. The task is considered solved when the states generated by the predicted code match the states generated by the target code for the first M states.

this code can express the correct experiments for an entire class of quantum states.

The goal of this work is to show that it is possible to use language models for the discovery of programs, which solve classes of quantum states such as the GHZ state.

A→B hard, B→A easy On an abstract level, we can describe the subject of our work as dealing with two sequences, `A` (a list of three quantum states) and `B` (python program). Direction `B→A` (computing the resulting quantum states from experimental setups) follows clear instructions and can be considered *easy*. Direction `A→B` is highly non-trivial. Designing just a single experiment can be very difficult, let alone finding a code which solves the entire class of states.

An instructive example for this asymmetry is the problem of finding the integral vs. the derivative of a mathematical function. This has been previously explored by Lample & Charton (2019). As there exist clear rules for differentiation, the authors could generate a large number of random functions and compute their derivatives. They then trained a sequence to sequence transformer to translate in the reverse direction, which is the more difficult task of integration.

Similar to the approach by Lample & Charton (2019), we want to train a sequence to sequence transformer to translate in the hard direction (from quantum states to python code). Because the opposite direction is a straightforward computation, we can produce a large amount of data to train the model by generating random codes (see Fig. 2).

Data (generate random `B`, compute `A`) Our training data consists of two sequences for each sample. The process of translating sequence A (list of states) to sequence B (meta-solution in form of python code) is the difficult direction, which the model is trained to do.

Using a simple set of rules, we can generate a random python code, which contains instructions for how to set up an experiment. Each code contains the variable index N. This means that the code will result in a different experimental setup for each value of N. Simulating the experiment for $N = 0, 1, 2$, we produce three states (see Fig. 2). After computing the states, sequence A has the form `<SOS>[state 1]<SEP>[state 2]<SEP>[state 3]<EOS>` and sequence B is `<SOS>[python code]<EOS>`. `<SOS>` and `<EOS>` are the start-of-sequence and end-of-sequence tokens and `<SEP>` is a separation token.

The maximum length for both sequences during data generation is 640 tokens. Both sequences are tokenized by a hand-picked vocabulary dictionary. We spend about 50,000 CPU hours on generating 56 million samples.

For the model to successfully generalize to unseen targets, it is advised to select the distribution of the synthetic data carefully (Charton, 2021). A simple example is that a model trained on random samples containing states with three polarizational modes can have difficulties solving a task containing states with only two modes, even though would be expected to be easier, because it is a subspace. To ensure performance on a diverse range of possibly interesting subspaces, we generate separate datasets at different levels of difficulty and specialization (length of states and codes, number of modes, constraints on phase parameters) and combine them to one final training dataset.

Training (learn `A→B`) We train the model with a standard encoder-decoder transformer architecture (Vaswani et al., 2017), with Pre-Layer Normalization (Xiong et al., 2020) and learned positional encoding (Gehring et al., 2017). We choose the dimensions $n_{emb} = 512$, $n_{\text{layer}} = 18$, $n_{\text{heads}} = 8$. We use a learned positional encoding, as we are not attempting to apply our model to unseen lengths. The model has approximately 133 million parameters and is trained for 750k steps with a batch size of 256 (approximately 2.5 epochs on a dataset of 56 million samples). The learning rate of the Adam optimizer (Kingma & Ba, 2014) was 10^{-4} for the first epoch and was then lowered to 10^{-5}. The training was performed on four A100-40GB GPUs.

Sampling Details We perform top-p sampling with the trained model. This means, that we generate the output sequence by choosing each token randomly according to the probability distribution given by the model at each step. We choose the value $p = 0.5$, which means that we only consider the top ranked tokens with a cumulative probability less than 0.5. We choose a temperature value 0.2, which can be adapted to vary the diversity of output. The

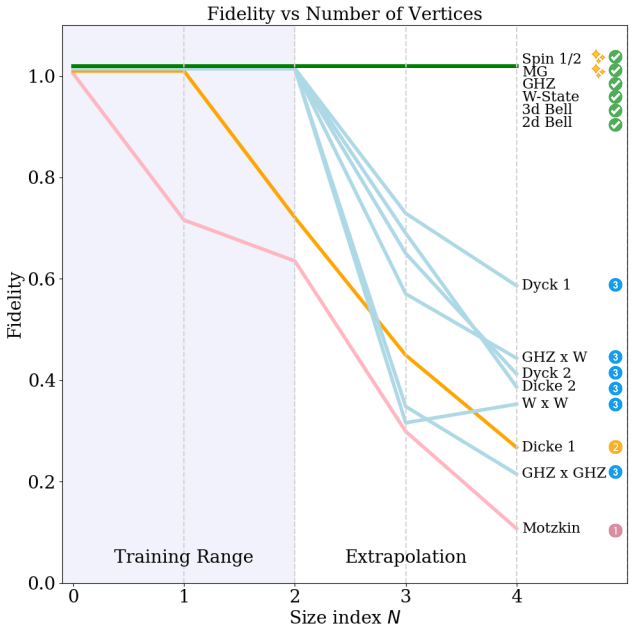

Figure 4. **Our approach discovers two previously unknown and four previously known generalizations.** We show the resulting fidelities of the best produced code for 14 of the 20 target classes. The green line represents the six target classes which our approach produces codes which correctly extrapolate beyond the first three elements. The blue lines show classes for which the best generated codes have fidelity one for the first three elements of the class, but do not extrapolate beyond. These cases are interesting as the model is still successful in generating a code which matches the three states provided as an input sequence, but the output for $N \geq 3$ does not match what we expect. The orange and red line are representatives of the 8 cases, for which the model was not able to predict correct solutions up to $N = 3$. The full table of target classes with their maximum correct N is shown in the appendix.

chosesn parameter values have performed well in other code generation tasks (Chen et al., 2021; Li et al., 2023). We evaluate the codes produced by the model by executing them to produce experimental setups for $N = 0, 1, 2, 3, 4$ (training data was generated only for $N = 0, 1, 2$). We compute the states which are produced by these setups and compute their fidelity with respect to the corresponding target state. A fidelity value of 1 means that the state is produced perfectly. The lowest possible value for the fidelity is 0, which means that the target state and the resulting state are perpendicular.

Application to unknown targets Our goal is now to apply the trained model to targets for which the code (sequence B) is unknown. Random generated data is abundant and is thus useful to train our model, but our aim is to discover codes for quantum state classes of particular interest (because of particular mathematical or physical properties). We have compiled a collection of twenty target classes based on a collection of quantum states found in (Ruiz-Gonzalez et al., 2023) – all of these states have exceptional properties that have been studied previously, for example in the context of quantum simulations or quantum communication.

The first three states of each target class are explicitly shown in the appendix. They are expressed as strings in the same way in which they are given to the model as input.

For four out of the 20 targets, meta-solutions were hand-crafted by researchers in the past. For 16 of the 20 target classes, no meta-solution was known before our work. Furthermore, we do not even know whether a solution can exist at all with the quantum-physical resources we provide (e.g. number of particles necessary to realize a state, and amount of quantum entanglement). Thus, every meta-solution from these 16 states is not a *re*discovery, but a genuine unbiased discovery.

5. Results

Model performance on validation data To evaluate our model on samples from the validation dataset, we produce 200 code predictions for 200 random samples from the validation dataset. We compute the `pass@k` metric for $1 \leq k \leq 100$ according to an estimator formula given in (Chen et al., 2021). This metric describes the probability for a task to be solved within k predictions made by the model. Since there is no straightforward way to proving that two codes are equivalent for arbitrary M, we consider a prediction to be successful if the produced states match the first M elements of the target class. In Fig. 3 we show the results of this evaluation. We observe that the likelihood of a correct prediction for the first M states decreases with higher M, but as M grows, the metric seems to converge towards a line which could be considered the *true* measure of a code, which perfectly matches the target code in all states it produces.

Successful meta-design of codes (6 out of 20 cases) Before training the model we prepared a set of 20 classes of quantum states as targets for our method. The condition for a class of states to be considered here are that there exists a clear rule for expressing the wave function $|\psi(N)\rangle$ in terms of a positive integer N. We require the number of particles in $|\psi(N)\rangle$ to be less than or equal to $2N + 2$ as this is the maximum system size which we allow for during data generation.

For each target, we sample the model for four hours on one RTX 6000 GPU, which produces 800-2500 samples (depending on the target class).

In Fig. 4 we show the fidelities of the best sample for fourteen of the twenty target classes. The best sample is

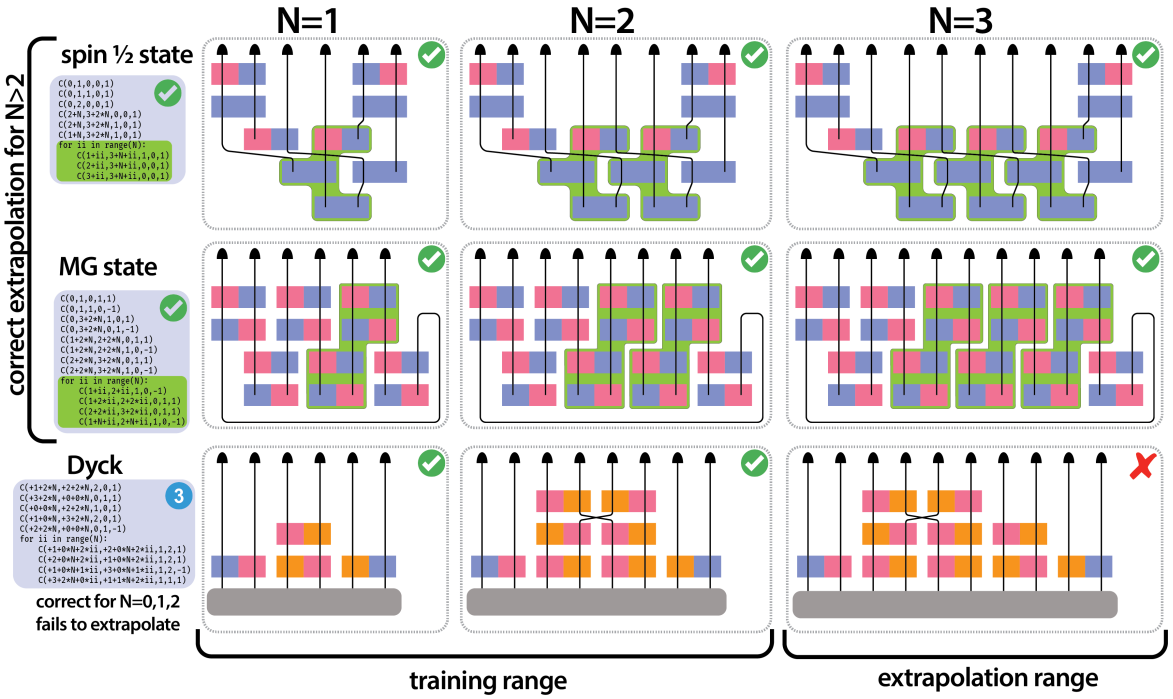

Figure 5. **Experimental setups for previously unknown solutions exhibit comprehensible patterns.** In the two top rows we show two previously unknown constructions discovered by our approach. For the spin$\frac{1}{2}$ states and the Majumdar-Gosh states (described in more detail in (Ruiz-Gonzalez et al., 2023)). For each of the two examples, the code produces the correct experimental setup for the three states used to prompt the model but also for higher particle numbers, indicating that the model was able to pick up on the pattern and write a correct code for the entire class of states. We highlight in green the 'building blocks', which are repeated multiple times as the particle number grows (stemming from lines written in the for loop). The bottom row shows a code for the Dyck 1 state. The setups generated by this code produce the correct state up to the third iteration, but are missing terms for indices $N > 2$. This means that the model was able to solve the task it was trained to do (match the first three states), but failed at the meta task of picking up on the pattern we intended it to match beyond the first three examples. It is also notable that in contrast to the other two examples, all setups produced for the Dyck 1 state also contained additional crystals which did not actually contribute to the resulting quantum state. We have omitted them by covering them by a grey rounded rectangle.

chosen by filtering for samples with the highest N such that all fidelities up to order N are equal to one and then choosing the one with the highest average fidelity for all $N \leq 4$. We find six target classes that our model can solve perfectly. For these classes, the output extrapolates beyond what the model was trained to do, i.e. match the states for $N = 0, 1, 2$.

For four famous classes of quantum states (GHZ, W 2d-Bell and 3d-Bell), we knew that there exists a construction rule for experiments with $2N+2$ particles for arbitrary N, which act as a baseline check for the capability of our method. Our model rediscovered all four meta-solutions of these states.

Most importantly, two out of six classes which our method successfully solves, were previously unknown and thus constitute a genuine discovery. The first previously unknown case is the general Spin-$\frac{1}{2}$ state. There, no two neighboring spin-ups appear in the ground state. In Rydberg-atom experiment, this situation occurs due to the Rydberg blockade (Ruiz-Gonzalez et al., 2023; Bernien et al., 2017), however so far it was unknown how to build such an entangled class for photonic systems. The second novel class contains the states of the famous Majumdar-Gosh Model in condensed matter physics . In this one-dimensional Heisenberg chain, the value of the next-nearest-neighbor interaction is half the value of the nearest-neighbor antiferromagnetic exchange interaction (Ruiz-Gonzalez et al., 2023; Chhajlany et al., 2007).

Interestingly, all classes that were correct for $N = 3$ (going one step beyond the three input states), were correct also for larger states, i.e. the model discovered an apparently perfect symbolic generalization.

Codes with unexpected generalizations (6 out of 20 cases) For these cases the model produces codes, which generate the correct states for the first three elements of the class, but produces experiments that produce states other than the expected ones. These cases are interesting to examine

because the model successfully performs the task it was trained for, as the first three states match the input sequence. The fact that it does not continue to match the target beyond $N = 3$ is due to a degree of ambiguity that exists for the continuation of any pattern. Any infinite sequence is underdetermined if only a finite number of elements is given. A possible way to narrow (but not remove) this ambiguity in our application would be to train the model on more than three elements. Further, the output is highly influenced by the synthetic data. The model will be more likely to produce an output, which is closer represents the distribution of data it was trained on. One example, the Dyck 1 states, is shown and analyzed in Fig. 5. This is an example which does not follow the intended pattern for $N \geq 3$, but produces valid states regardless, which randomly generated experimental setups generally do not do. There is potential in examine these cases in more detail to see if the pattern they follow is interesting from a physics side, as they might represent new unexplored classes of quantum states.

Codes which fail to match the first three states (8 out of 20 cases) Four classes match the first two states of the input. Another four classes only match the first state of the input states. There were no examples where the model could not match any input states. These also include cases for which the setups generated by the output code do not produce a valid quantum state at higher indices N. These cases could be either too complex for the model to give the correct prediction, or generalisations cannot exist at all for physical reasons, given the amount of quantum resources we provide.

6. Discussion

We demonstrate how a language model can produce a meta-solution for a physical design task. The meta-solution is described in the form of computer code, which itself produces solutions to large generalizations of the design question. In our examples, we discover previously unknown generalizations of experimental setups for interesting quantum states. The ability to automatically create generalizations also offers a decisive advantage over conventional AI-driven design in terms of computational costs.

Our method is not constrained to quantum physics but can be directly implemented in other domains, such as the discovery of new microscopes (Rodríguez et al., 2023), new gravitational wave detectors (Krenn et al., 2023), new experimental hardware for high-energy physics (Baydin et al., 2021), or the design of new functional molecules (Pollice et al., 2021).

At a more abstract level, we see that the application of a powerful intermediate language that can be written and read by both machines and humans can significantly enhance the understandability and generalizability of AI-driven discoveries.

Acknowledgements

The authors thank Ben Newman for useful discussions.

References

Alfarano, A., Charton, F., Hayat, A., and des Ponts Paristech, C.-E. Discovering lyapunov functions with transformers. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023.

Alnuqaydan, A., Gleyzer, S., Prosper, H. B., Reinhardt, E. A., Anand, N., and Charton, F. Symbolic machine learning for high energy physics calculations.

Aslam, N., Zhou, H., Urbach, E. K., Turner, M. J., Walsworth, R. L., Lukin, M. D., and Park, H. Quantum sensors for biomedical applications. *Nature Reviews Physics*, 5(3):157–169, February 2023.

Aspuru-Guzik, A. and Walther, P. Photonic quantum simulators. *Nature physics*, 8(4):285–291, 2012.

Barman, K. G., Caron, S., Claassen, T., and De Regt, H. Towards a benchmark for scientific understanding in humans and machines. *Minds and Machines*, 34(1):1–16, 2024.

Baydin, A. G., Cranmer, K., de Castro Manzano, P., Delaere, C., Derkach, D., Donini, J., Dorigo, T., Giammanco, A., Kieseler, J., Layer, L., Louppe, G., Ratnikov, F., Strong, G., Tosi, M., Ustyuzhanin, A., Vischia, P., and Yarar, H. Toward machine learning optimization of experimental design. *Nuclear Physics News*, 31(1):25–28, 2021.

Bernien, H., Schwartz, S., Keesling, A., Levine, H., Omran, A., Pichler, H., Choi, S., Zibrov, A. S., Endres, M., Greiner, M., et al. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551(7682):579–584, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Cai, T., Merz, G. W., Charton, F., Nolte, N., Wilhelm, M., Cranmer, K., and Dixon, L. J. Transforming the bootstrap: Using transformers to compute scattering amplitudes in

planar n= 4 super yang-mills theory. *arXiv:2405.06107*, 2024.

Chae, H., Kim, Y., Kim, S., Ong, K. T.-i., Kwak, B.-w., Kim, M., Kim, S., Kwon, T., Chung, J., Yu, Y., and Yeo, J. Language models as compilers: Simulating pseudocode execution improves algorithmic reasoning in language models. *arXiv:2404.02575*, 2024.

Charton, F. Linear algebra with transformers. *arXiv:2112.01898*, 2021.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Petroski Such, F., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *arXiv:2107.03374*, 2021.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv:2211.12588*, 2022.

Chhajlany, R. W., Tomczak, P., Wójcik, A., and Richter, J. Entanglement in the majumdar-ghosh model. *Physical Review A*, 75(3):032340, 2007.

Cornish, S. L., Tarbutt, M. R., and Hazzard, K. R. A. Quantum computation and quantum simulation with ultracold molecules. *Nature Physics*, 20(5):730–740, 2024.

Couteau, C., Barz, S., Durt, T., Gerrits, T., Huwer, J., Prevedel, R., Rarity, J., Shields, A., and Weihs, G. Applications of single photons to quantum communication and computing. *Nature Reviews Physics*, 5(6):326–338, 2023.

De Regt, H. W. *Understanding scientific understanding*. Oxford University Press, 2017.

DeMille, D., Hutzler, N. R., Rey, A. M., and Zelevinsky, T. Quantum sensing and metrology for fundamental physics with molecules. *Nature Physics*, 20(5):741–749, May 2024.

Flamini, F., Spagnolo, N., and Sciarrino, F. Photonic quantum information processing: a review. *Reports on Progress in Physics*, 82(1):016001, 2018.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Gedeon, J., Hassan, E., and Lesina, A. C. Free-form inverse design of arbitrary dispersive materials in nanophotonics. *arXiv:2305.00234*, 2023.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. Convolutional sequence to sequence learning. In *International conference on machine learning*, pp. 1243–1252. PMLR, 2017.

Gemma-Team. Gemma: Open models based on gemini research and technology. *arXiv:2403.08295*, 2024.

Goel, S., Leedumrongwatthanakun, S., Valencia, N. H., McCutcheon, W., Tavakoli, A., Conti, C., Pinkse, P. W., and Malik, M. Inverse design of high-dimensional quantum optical circuits in a complex medium. *Nature Physics*, pp. 1–8, 2024.

Greenberger, D. M., Horne, M. A., Shimony, A., and Zeilinger, A. Bell's theorem without inequalities. *American Journal of Physics*, 58(12):1131–1143, December 1990.

Kamienny, P.-a., d'Ascoli, S., Lample, G., and Charton, F. End-to-end symbolic regression with transformers. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 10269–10281. Curran Associates, Inc., 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Knott, P. A search algorithm for quantum state engineering and metrology. *New Journal of Physics*, 18(7):073033, 2016.

Kottmann, J. S. Molecular quantum circuit design: A graph-based approach. *Quantum*, 7:1073, August 2023.

Krenn, M., Malik, M., Fickler, R., Lapkiewicz, R., and Zeilinger, A. Automated search for new quantum experiments. *Physical review letters*, 116(9):090405, 2016.

Krenn, M., Erhard, M., and Zeilinger, A. Computer-inspired quantum experiments. *Nature Reviews Physics*, 2(11): 649–661, September 2020.

Krenn, M., Kottmann, J. S., Tischler, N., and Aspuru-Guzik, A. Conceptual understanding through efficient automated design of quantum optical experiments. *Physical Review X*, 11(3):031044, 2021.

Krenn, M., Pollice, R., Guo, S. Y., Aldeghi, M., Cervera-Lierta, A., Friederich, P., dos Passos Gomes, G., Häse, F., Jinich, A., Nigam, A., Yao, Z., and Aspuru-Guzik, A. On scientific understanding with artificial intelligence. *Nature Reviews Physics*, 4(12):761–769, 2022.

Krenn, M., Drori, Y., and Adhikari, R. X. Digital discovery of interferometric gravitational wave detectors. *arXiv:2312.04258*, 2023.

Kviatkovsky, I., Chrzanowski, H. M., Avery, E. G., Bartolomaeus, H., and Ramelow, S. Microscopy with undetected photons in the mid-infrared. *Science Advances*, 6(42): eabd0264, 2020.

Lample, G. and Charton, F. Deep learning for symbolic mathematics. *arXiv:1912.01412*, 2019.

Landgraf, J., Peano, V., and Marquardt, F. Automated discovery of coupled mode setups. *arXiv:2404.14887*, 2024.

Lemos, G. B., Borish, V., Cole, G. D., Ramelow, S., Lapkiewicz, R., and Zeilinger, A. Quantum imaging with undetected photons. *Nature*, 512(7515):409–412, 2014.

Li, R., Ben Allal, L., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., Liu, Q., Zheltonozhskii, E., Zhuo, T. Y., Wang, T., Dehaene, O., Davaadorj, M., Lamy-Poirier, J., Monteiro, J., Shliazhko, O., Gontier, N., Meade, N., Zebaze, A., Yee, M.-H., Umapathi, L. K., Zhu, J., Lipkin, B., Oblokulov, M., Wang, Z., Murthy, R., Stillerman, J., Patel, S. S., Abulkhanov, D., Zocca, M., Dey, M., Zhang, Z., Fahmy, N., Bhattacharyya, U., Yu, W., Singh, S., Luccioni, S., Villegas, P., Kunakov, M., Zhdanov, F., Romero, M., Lee, T., Timor, N., Ding, J., Schlesinger, C., Schoelkopf, H., Ebert, J., Dao, T., Mishra, M., Gu, A., Robinson, J., Anderson, C. J., Dolan-Gavitt, B., Contractor, D., Reddy, S., Fried, D., Bahdanau, D., Jernite, Y., Ferrandis, C. M., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder: may the source be with you! *arXiv:2305.06161*, 2023.

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., Liu, T., Tian, M., Kocetkov, D., Zucker, A., Belkada, Y., Wang, Z., Liu, Q., Abulkhanov, D., Paul, I., Li, Z., Li, W.-D., Risdal, M., Li, J., Zhu, J., Zhuo, T. Y., Zheltonozhskii, E., Dade, N. O. O., Yu, W., Krauß, L., Jain, N., Su, Y., He, X., Dey, M., Abati, E., Chai, Y., Muennighoff, N., Tang, X., Oblokulov, M., Akiki, C., Marone, M., Mou, C., Mishra, M., Gu, A., Hui, B., Dao, T., Zebaze, A., Dehaene, O., Patry, N., Xu, C., McAuley, J., Hu, H., Scholak, T., Paquet, S., Robinson, J., Anderson, C. J., Chapados, N., Patwary, M., Tajbakhsh, N., Jernite, Y., Muñoz Ferrandis, C., Zhang, L., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder 2 and the stack v2: The next generation. *arXiv:2402.19173*, 2024.

Ma, W., Liu, Z., Kudyshev, Z. A., Boltasseva, A., Cai, W., and Liu, Y. Deep learning for the design of photonic structures. *Nature Photonics*, 15(2):77–90, 2021.

MacLellan, B., Roztocki, P., Czischek, S., and Melko, R. G. End-to-end variational quantum sensing. *arXiv:2403.02394*, 2024.

Madsen, L. S., Laudenbach, F., Askarani, M. F., Rortais, F., Vincent, T., Bulmer, J. F. F., Miatto, F. M., Neuhaus, L., Helt, L. G., Collins, M. J., Lita, A. E., Gerrits, T., Nam, S. W., Vaidya, V. D., Menotti, M., Dhand, I., Vernon, Z., Quesada, N., and Lavoie, J. Quantum computational advantage with a programmable photonic processor. *Nature*, 606(7912):75–81, 2022.

Melko, R. G. and Carrasquilla, J. Language models for quantum simulation. *Nature Computational Science*, 4: 11–18, 2024.

Michaud, E. J., Liao, I., Lad, V., Liu, Z., Mudide, A., Loughridge, C., Guo, Z. C., Kheirkhah, T. R., Vukelić, M., and Tegmark, M. Opening the ai black box: program synthesis via mechanistic interpretability. *arXiv:2402.05110*, 2024.

Molesky, S., Lin, Z., Piggott, A. Y., Jin, W., Vucković, J., and Rodriguez, A. W. Inverse design in nanophotonics. *Nature Photonics*, 12(11):659–670, 2018.

Moreau, P.-A., Toninelli, E., Gregory, T., and Padgett, M. J. Imaging with quantum states of light. *Nature Reviews Physics*, 1(6):367–380, 2019.

Nägele, M. and Marquardt, F. Optimizing zx-diagrams with deep reinforcement learning. *arXiv:2311.18588*, 2023.

Nichols, R., Mineh, L., Rubio, J., Matthews, J. C., and Knott, P. A. Designing quantum experiments with a genetic algorithm. *Quantum Science and Technology*, 4 (4):045012, 2019.

Ostaszewski, M., Trenkwalder, L. M., Masarczyk, W., Scerri, E., and Dunjko, V. Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, 34: 18182–18194, 2021.

Pan, J.-W., Bouwmeester, D., Daniell, M., Weinfurter, H., and Zeilinger, A. Experimental test of quantum nonlocality in three-photon greenberger–horne–zeilinger entanglement. *Nature*, 403(6769):515–519, February 2000.

Pezzè, L., Smerzi, A., Oberthaler, M. K., Schmied, R., and Treutlein, P. Quantum metrology with nonclassical states of atomic ensembles. *Rev. Mod. Phys.*, 90, Sep 2018.

Polino, E., Valeri, M., Spagnolo, N., and Sciarrino, F. Photonic quantum metrology. *AVS Quantum Science*, 2(2), 2020.

Pollice, R., dos Passos Gomes, G., Aldeghi, M., Hickman, R. J., Krenn, M., Lavigne, C., Lindner-D'Addario, M., Nigam, A., Ser, C. T., Yao, Z., and Aspuru-Guzik, A. Data-driven strategies for accelerated materials design. *Accounts of Chemical Research*, 54(4):849–860, 2021.

Prabhu, M., Roques-Carmes, C., Shen, Y., Harris, N., Jing, L., Carolan, J., Hamerly, R., Baehr-Jones, T., Hochberg, M., Čeperić, V., et al. Accelerating recurrent ising machines in photonic integrated circuits. *Optica*, 7(5):551–558, 2020.

Rodríguez, C., Arlt, S., Möckl, L., and Krenn, M. Xlumina: An auto-differentiating discovery framework for super-resolution microscopy. *arXiv:2310.08408*, 2023.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

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

Sapra, N. V., Yang, K. Y., Vercruysse, D., Leedle, K. J., Black, D. S., England, R. J., Su, L., Trivedi, R., Miao, Y., Solgaard, O., Byer, R. L., and Vučković, J. On-chip integrated laser-driven particle accelerator. *Science*, 367 (6473):79–83, 2020.

Shojaee, P., Meidani, K., Gupta, S., Farimani, A. B., and Reddy, C. K. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv:2404.18400*, 2024.

Trinh, T. H., Wu, Y., Le, Q. V., He, H., and Luong, T. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wallnöfer, J., Melnikov, A. A., Dür, W., and Briegel, H. J. Machine learning for long-distance quantum communication. *PRX Quantum*, 1(1):010301, 2020.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

Zen, R., Olle, J., Colmenarez, L., Puviani, M., Müller, M., and Marquardt, F. Quantum circuit discovery for fault-tolerant logical state preparation with reinforcement learning. *arXiv:2402.17761*, 2024.

Zheng, Q., Xia, X., Zou, X., Dong, Y., Wang, S., Xue, Y., Wang, Z., Shen, L., Wang, A., Li, Y., Su, T., Yang, Z., and Tang, J. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x. *arXiv:2303.17568*, 2023.

A. Target classes

In the following table we show the hand-picked targets. These are classes of quantum states which are of interest in different areas of quantum physics. For each class, the first three states (four, six and eight particles) are given as strings in the same way that they are used to prompt the model. The column "correct states" shows, up to which index N the best model output matches the target (the first N states are correct). An infinity sign ∞ means, that the meta-solution perfectly matches the target.

State Name	size	Quantum State string	correct states	previously known
Spin 1/2	4	+1[xxxx] +1[xxyx] +1[xyxx] +1[yxxx] +1[yxyx]	∞	**unknown**
	6	+1[xxxxxx] +1[xxxyxx] +1[xxyxxx] +1[xyxxxx] +1[xyxyxx] +1[yxxxxx] +1[yxxyxx] +1[yxyxxx]		
	8	+1[xxxxxxxx] +1[xxxxyxxx] +1[xxxyxxxx] +1[xxyxxxxx] +1[xxyxyxxx] +1[xyxxxxxx] +1[xyxxyxxx] +1[xyxyxxxx] +1[yxxxxxxx] +1[yxxxyxxx] +1[yxxyxxxx] +1[yxyxxxxx] +1[yxyxyxxx]		
Majumdar-Ghosh	4	-1[xxyy] +2[xyxy] -1[xyyx] -1[yxxy] +2[yxyx] -1[yyxx]	∞	**unknown**
	6	-1[xxyxyy] +1[xxyyxy] +1[xyxxyy] -1[xyxyyx] -1[xyyxxy] +1[xyyxyx] -1[yxxyxy] +1[yxxyyx] +1[yxyxxy] -1[yxyyxx] -1[yyxxyx] +1[yyxyxx]		
	8	-1[xxyxyxyy] +1[xxyxyyxy] +1[xxyyxxyy] -1[xxyyxyxy] +1[xyxxyxyy] -1[xyxxyyxy] -1[xyxyxxyy] +2[xyxyxyxy] -1[xyxyxyyx] -1[xyxyyxxy] +1[xyxyyyxx] -1[xyyxxyxy] +1[xyyxxyyx] +1[xyyxyxxy] -1[xyyxyxyx] -1[yxxyxyxy] +1[yxxyxyyx] +1[yxxyyxxy] -1[yxxyyxyx] +1[yxyxxyxy] -1[yxyxxyyx] -1[yxyxyxxy] +2[yxyxyxyx] -1[yxyxyyxx] -1[yxyyxxyx] +1[yxyyyxxx] -1[yyxxxyxx] +1[yyxxyyxx] +1[yyxyxxyx] -1[yyxyxyxx]		
Bell pairs 2d	4	+1[xxxx] +1[xxyy] +1[yyxx] +1[yyyy]	∞	known
	6	+1[xxxxxx] +1[xxxxyy] +1[xxyyxx] +1[xxyyyy] +1[yyxxxx] +1[yyxxyy] +1[yyyyxx] +1[yyyyyy]		
	8	+1[xxxxxxxx] +1[xxxxxxyy] +1[xxxxyyxx] +1[xxxxyyyy] +1[xxyyxxxx] +1[xxyyxxyy] +1[xxyyyyxx] +1[xxyyyyyy] +1[yyxxxxxx] +1[yyxxxxyy] +1[yyxxyyxx] +1[yyxxyyyy] +1[yyyyxxxx] +1[yyyyxxyy] +1[yyyyyyxx] +1[yyyyyyyy]		
Bell pairs 3d	4	+1[xxxx] +1[yyxx] +1[zzxx]	∞	known
	6	+1[xxxxxx] +1[xxyyxx] +1[xxzzxx] +1[yyxxxx] +1[yyyyxx] +1[yyzzxx] +1[zzxxxx] +1[zzyyxx] +1[zzzzxx]		
	8	+1[xxxxxxxx] +1[xxxxyyxx] +1[xxxxzzxx] +1[xxyyxxxx] +1[xxyyyyxx] +1[xxyyzzxx] +1[xxzzxxxx] +1[xxzzyyxx] +1[xxzzzzxx] +1[yyxxxxxx] +1[yyxxyyxx] +1[yyxxzzxx] +1[yyyyxxxx] +1[yyyyyyxx] +1[yyyyzzxx] +1[yyzzxxxx] +1[yyzzyyxx] +1[yyzzzzxx] +1[zzxxxxxx] +1[zzxxyyxx] +1[zzxxzzxx] +1[zzyyxxxx] +1[zzyyyyxx] +1[zzyyzzxx] +1[zzzzxxxx] +1[zzzzyyxx] +1[zzzzzzxx]		
GHZ	4	+1[xxxx] +1[yyyy]	∞	known
	6	+1[xxxxxx] +1[yyyyyy]		
	8	+1[xxxxxxxx] +1[yyyyyyyy]		
W	4	+1[xxxy] +1[xxyx] +1[xyxx] +1[yxxx]	∞	known
	6	+1[xxxxxy] +1[xxxxyx] +1[xxxyxx] +1[xxyxxx] +1[xyxxxx] +1[yxxxxx]		
	8	+1[xxxxxxxy] +1[xxxxxxyx] +1[xxxxxyxx] +1[xxxxyxxx] +1[xxxyxxxx] +1[xxyxxxxx] +1[xyxxxxxx] +1[yxxxxxxx]		

State Name	size	Quantum State string	correct states	previously known
GHZ x W	4	+1[xxxy] +1[xxyx] +1[yyxy] +1[yyyx]	3	unknown
	6	+1[xxxxxy] +1[xxxxyx] +1[xxxyxx] +1[yyyxxy] +1[yyyxyx] +1[yyyyxx]		
	8	+1[xxxxxxxy] +1[xxxxxxyx] +1[xxxxxyxx] +1[xxxxyxxx] +1[yyyyxxxy] +1[yyyyxxyx] +1[yyyyxyxx] +1[yyyyyxxx]		
W x W	4	+1[xyxy] +1[xyyx] +1[yxxy] +1[yxyx]	3	unknown
	6	+1[xxyxxy] +1[xxyxyx] +1[xxyyxx] +1[xyxxxy] +1[xyxxyx] +1[xyxyxx] +1[yxxxxy] +1[yxxxyx] +1[yxxyxx]		
	8	+1[xxxyxxxy] +1[xxxyxxyx] +1[xxxyxyxx] +1[xxxyyxxx] +1[xxyxxxxy] +1[xxyxxxyx] +1[xxyxxyxx] +1[xxyxyxxx] +1[xyxxxxxy] +1[xyxxxxyx] +1[xyxxxyxx] +1[xyxxyxxx] +1[yxxxxxxy] +1[yxxxxxyx] +1[yxxxxyxx] +1[yxxxyxxx]		
Dicke 2	4	+1[xzzx] +1[zxzx] +1[zzxx]	3	unknown
	6	+1[xxzzxx] +1[xzxzxx] +1[xzzxxx] +1[zxxzxx] +1[zxzxxx] +1[zzxxxx]		
	8	+1[xxxzzxxx] +1[xxzxzxxx] +1[xxzzxxxx] +1[xzxxzxxx] +1[xzxzxxxx] +1[xzzxxxxx] +1[zxxxzxxx] +1[zxxzxxxx] +1[zxzxxxxx] +1[zzxxxxxx]		
GHZ x GHZ	4	+1[xxxx] +1[xxyy] +1[yyxx] +1[yyyy]	3	unknown
	6	+1[xxxxxx] +1[xxxyyy] +1[yyyxxx] +1[yyyyyy]		
	8	+1[xxxxxxxx] +1[xxxxyyyy] +1[yyyyxxxx] +1[yyyyyyyy]		
Dyck 2	4	+1[yyzz] +1[yzyz]	3	unknown
	6	+1[yyyzzz] +1[yyzyzz] +1[yyzzyz] +1[yzyyzz] +1[yzyzyz]		
	8	+1[yyyyzzzz] +1[yyyzyzzz] +1[yyyzzyzz] +1[yyyzzzyz] +1[yyzyyzzz] +1[yyzyzyzz] +1[yyzyzzyz] +1[yyzzyyzz] +1[yyzzyzyz] +1[yzyyyzzz] +1[yzyyzyzz] +1[yzyyzzyz] +1[yzyzyyzz] +1[yzyzyzyz]		
Dyck 1	4	+1[yzxx]	3	unknown
	6	+1[yyzzxx] +1[yzyzxx]		
	8	+1[yyyzzzxx] +1[yyzyzzxx] +1[yyzzyzxx] +1[yzyyzzxx] +1[yzyzyzxx]		
Dicke 1	4	+1[xzxx] +1[zxxx]	2	unknown
	6	+1[xxzzxx] +1[xzxzxx] +1[xzzxxx] +1[zxxzxx] +1[zxzxxx] +1[zzxxxx]		
	8	+1[xxxzzzxx] +1[xxzxzzxx] +1[xxzzxzxx] +1[xxzzzxxx] +1[xzxxzzxx] +1[xzxzxzxx] +1[xzxzzxxx] +1[xzzxxzxx] +1[xzzxzxxx] +1[xzzzxxxx] +1[zxxxzzxx] +1[zxxzxzxx] +1[zxxzzxxx] +1[zxzxxzxx] +1[zxzxzxxx] +1[zxzzxxxx] +1[zzxxxzxx] +1[zzxxzxxx] +1[zzxzxxxx] +1[zzzxxxxx]		
Dicke 5	4	+1[zzzx]	2	unknown
	6	+1[xzzzxx] +1[zxzzxx] +1[zzxzxx] +1[zzzxxx]		
	8	+1[xxzzzxxx] +1[xzxzzxxx] +1[xzzxzxxx] +1[xzzzxxxx] +1[zxxzzxxx] +1[zxzxzxxx] +1[zxzzxxxx] +1[zzxxzxxx] +1[zzxzxxxx] +1[zzzxxxxx]		
AKLT	4	-1[xzxx] +1[yyxx] -1[zxxx]	2	unknown
	6	-1[xyzxxx] +1[xzyxxx] +1[yxzxxx] -1[yzxxxx] -1[zxyxxx] +1[zyxxxx]		
	8	-1[xyyzxxxx] +1[xyzyxxxx] +2[xzxzxxxx] -1[xzyyxxxx] +1[yxyzxxxx] -1[yxzyxxxx] -1[yyxzxxxx] +1[yyyyxxxx] -1[yyzxxxxx] -1[yzxyxxxx] +1[yzyxxxxx] -1[zxyyxxxx] +2[zxzxxxxx] +1[zyxyxxxx] -1[zyyxxxxx]		
Motzkin small	4	+1[xyxx] +1[zzxx]	2	unknown
	6	+1[xyzxxx] +1[xzyxxx] +1[zxyxxx] +1[zzzxxx]		
	8	+1[xxyyxxxx] +1[xyxyxxxx] +1[xyzzxxxx] +1[xzyzxxxx] +1[xzzyxxxx] +1[zxyzxxxx] +1[zxzyxxxx] +1[zzxyxxxx] +1[zzzzxxxx]		

State Name	size	Quantum State string	correct states	previously known
Dicke 3	4	+1[xyzx] +1[xzyx] +1[yxzx] +1[yzxx] +1[zxyx] +1[zyxx]	1	unknown
	6	+1[xxyzxx] +1[xxzyxx] +1[xyxzxx] +1[xyzxxx] +1[xzxyxx] +1[xzyxxx] +1[yxxzxx] +1[yxzxxx] +1[yzxxxx] +1[zxxyxx] +1[zxyxxx] +1[zyxxxx]		
	8	+1[xxxyzxxx] +1[xxxzyxxx] +1[xxyxzxxx] +1[xxyzxxxx] +1[xxzxyxxx] +1[xxzyxxxx] +1[xyxxzxxx] +1[xyxzxxxx] +1[xyzxxxxx] +1[xzxxyxxx] +1[xzxyxxxx] +1[xzyxxxxx] +1[yxxxzxxx] +1[yxxzxxxx] +1[yxzxxxxx] +1[yzxxxxxx] +1[zxxxyxxx] +1[zxxyxxxx] +1[zxyxxxxx] +1[zyxxxxxx]		
Dicke 4	4	+1[xxyy] +1[xyxy] +1[xyyx] +1[yxxy] +1[yxyx] +1[yyxx]	1	unknown
	6	+1[xxxxyy] +1[xxxyxy] +1[xxxyyx] +1[xxyxxy] +1[xxyxyx] +1[xxyyxx] +1[xyxxxy] +1[xyxxyx] +1[xyxyxx] +1[xyyxxx] +1[yxxxxy] +1[yxxxyx] +1[yxxyxx] +1[yxyxxx] +1[yyxxxx]		
	8	+1[xxxxxxyy] +1[xxxxxyxy] +1[xxxxxyyx] +1[xxxxyxxy] +1[xxxxyxyx] +1[xxxxyyxx] +1[xxxyxxxy] +1[xxxyxxyx] +1[xxxyxyxx] +1[xxxyyxxx] +1[xxyxxxxy] +1[xxyxxxyx] +1[xxyxxyxx] +1[xxyxyxxx] +1[xxyyxxxx] +1[xyxxxxxy] +1[xyxxxxyx] +1[xyxxxyxx] +1[xyxxyxxx] +1[xyxyxxxx] +1[xyyxxxxx] +1[yxxxxxxy] +1[yxxxxxyx] +1[yxxxxyxx] +1[yxxxyxxx] +1[yxxyxxxx] +1[yxyxxxxx] +1[yyxxxxxx]		
GHZ 3d x GHZ 3d	4	+1[xxxx] +1[xxyy] +1[xxzz] +1[yyxx] +1[yyyy] +1[yyzz] +1[zzxx] +1[zzyy] +1[zzzz]	1	unknown
	6	+1[xxxxxx] +1[xxxyyy] +1[xxxzzz] +1[yyyxxx] +1[yyyyyy] +1[yyyzzz] +1[zzzxxx] +1[zzzyyy] +1[zzzzzz]		
	8	+1[xxxxxxxx] +1[xxxxyyyy] +1[xxxxzzzz] +1[yyyyxxxx] +1[yyyyyyyy] +1[yyyyzzzz] +1[zzzzxxxx] +1[zzzzyyyy] +1[zzzzzzzz]		
Motzkin	4	+1[xyzx] +1[xzyx] +1[zxyx] +1[zzzx]	1	unknown
	6	+1[xxyyxx] +1[xyxyxx] +1[xyzzxx] +1[xzyzxx] +1[xzzyxx] +1[zxyzxx] +1[zxzyxx] +1[zzxyxx] +1[zzzzxx]		
	8	+1[xxyyzxxx] +1[xxyzyxxx] +1[xxzyyxxx] +1[xyxyzxxx] +1[xyxzyxxx] +1[xyzxyxxx] +1[xyzzzxxx] +1[xzxyyxxx] +1[xzyxyxxx] +1[xzyzzxxx] +1[xzzyzxxx] +1[xzzzyxxx] +1[zxxyyxxx] +1[zxyxyxxx] +1[zxyzzxxx] +1[zxzyzxxx] +1[zxzzyxxx] +1[zzxyzxxx] +1[zzxzyxxx] +1[zzzxyxxx] +1[zzzzzxxx]		