ALGORITHMIC FORMALIZATION FOR AI SAFETY

Contents

Introduction

The AI Safety Problem. Recent advancements have led to the deployment of sophisticated AI systems, including large language models for customer service and research, as well as deep reinforcement learning for controlling nuclear fusion. As these systems become more capable, their associated risks increase. In the context of language models, risks include untruthful responses [Ban+23], sycophancy, and deception [Par+23]. Additionally, there are immediate risks from misuse, such as disinformation, deepfakes, and biased decision-making [HMW23; Ji+24].

Current validation methods for AI models, such as independent testing and red-teaming [Khl], do not provide rigorous safety guarantees. For instance, despite extensive evaluation of ChatGPT, users found ways to circumvent safeguards. Another approach involves obtaining a deeper theoretical understanding of AI systems [Min+19], often supplemented by time consuming empirical testing [Che+23].

Our approach to addressing AI safety comes from two perspectives:

(1) **Correct-by-Construction Program Synthesis:** This method aims to produce programs that are guaranteed to be correct with respect to their specifications [Aba+23].

(2) **Formal Verification of AI-Enabled Systems:** This involves formulating specifications that form proof obligations, designing systems to meet these obligations, and verifying compliance via algorithmic proof search, using Boolean satisfiability (SAT) and satisfiability modulo theories (SMT) solvers.

This dual approach leverages our expertise in automated theorem proving and mathematics. Although we acknowledge its limitations in practical scenarios, it provides a strong foundation for our long-term goals, which we address in Section 1.

Michaud et al.'s [Mic+24] MIPS (Mechanistic-Interpretability-based Program Synthesis) is a notable example of combining these strategies. Their pipeline automates the interpretation of

neural networks by distilling learned algorithms into human-readable Python code, followed by formal verification using the Dafny checker. Our proposal is, therefore, to expand and improve upon what is currently missing in the works of program synthesis and formalization, particularly the approach of MIPS.

1. Short-term and long-term goals

1.1. **Long-term goals.**

In the long term, we aim to contribute to the framework of Guaranteed Safe AI approach [Dal+24]. An AI safety system comprises three key components: the world model, safety specification, and verifier. The verifier ensures that the AI's planned actions, as predicted by the world model[1], comply with the safety specification[2]. This process guarantees safety by ensuring that the AI avoids harmful actions in complex, dynamic environments.

In our research program, we use mathematical test examples to demonstrate these concepts in a controlled domain. Specifically, using an interactive theorem prover like Lean, where the prover acts as the verifier. This test example provides three proofs of concept: i) demonstrating the effectiveness of formal verification techniques and ii) illustrating the potential for automating safety checks in dynamic environments by leveraging formal mathematical techniques, thus bridging the two fields.

Lastly, from the point of view of both research mathematicians and students, we look forward to the application of such towards scientific discovery. Explorative mathematical problems would benefit from our approach. One example is, [Rom+23], where one does a program search to find large examples in the cap set problem [3]. It would be interesting to see whether, with the help of proof assistants, we can create more effective feedback loops in the scoring stage. Applications include finding minimum entropy coupling, which is used in steganography, [Wit+23].

1.2. **Short-term goals.**

We list three directions that we hope to accomplish during the short term:

(1) Scalability: In Dafny, lots of manual work is in writing down the specifications and defining the pre-and-post safety conditions [Lei10]. We thus propose an improved verification pipeline where we use the more modern Lean 4 theorem prover instead of Dafny (c.f. Appendix B). In Lean, by contrast, correctness is part of type-checking and there is no need for separate pre-and-post safety conditions besides the main program. The standard library of Lean, `mathlib4`[4], contains a vast amount of formalized mathematics and proofs which allows for a more modular and scalable correct-by-construction approach.

(2) Complexity: We enhance the algorithmic complexity of tasks in our model by leveraging Lean's capabilities. We will source problems from various mathematical fields,

[1] A world model provides a mathematical description of an AI system's interaction with its environment, simulating outcomes. For instance, in autonomous driving, it includes roads, traffic, and pedestrians.

[2] A safety specification is a formal mathematical description of acceptable and unacceptable behaviors, defining constraints and rules to prevent harmful actions.

[3] large examples in the capset problem (which is maximal set $S \subseteq \mathbb{F}_3^n$ such that no three are colinear).

[4] https://leanprover-community.github.io/mathlib-overview.html

including finite groups, derangements, number theory (with particular focus on cryptography). (c.f., Appendix A)

(3) Brain storm of verification on the semantic feature space: Integrating verification tools as components of AI safety for real-world applications poses deeper challenges than the capabilities of our current model; for one, the output of the verifier will not be just a YES/NO but a numeric value. However, we think that our constrained approach specialized to distilling learned algorithms of purely mathematical problems is the first serious step toward more complicated environments.

There are thus two main immediate outcomes: Improved pipeline of [Mic+24] and a proof of concept of the applicability of Lean 4 proof assistant. The next step of our goal is to employ our system in very basic programs that model real life interactions, starting from basic ones as Cartpole, MDPs, [SB98], AI safety gridworlds, [Lei+17], pyRDDLGym environments, [Tai+22], to (ambitiously) real life operational research logistic networks, such as the OR-Lib test problems.

Appendix A. Algorithmic complexity

Our goal is to enhance the algorithmic complexity of tasks used in our model, particularly by leveraging the capabilities of Lean. We want to focus on tasks where the difficulty of algorithms involved is

(1) not too hard. As already seen in the experiments in [Mic+24], a number of basic algorithms cannot be completed, such as modular (4 to 8) addition. However, the model architecture there is RNN. Recent work has already shown that transformers can learn automata, [Liu+23], and addition sufficiently well, [QB24].

(2) hard enough to leverage the mathematical library of Lean. Recent work by [Wen+23], has shown that transformers can distinguish Learning with Errors (LWE) instances, which is used as a base for post-quantum cryptography algorithms.

We give a tentative list here:

(1) Finite groups. [CCN23], studied problems related to group theory, focusing on finite group structures and their properties.

(2) Combinatorics: Derangements and partial derangements.[5] Our training data will include pairs of integers and lists of these derangements. Many combinatorial properties of derangements are formalized in `mathlib4`.

(3) Number Theory: We will consider practical problems in cryptography. The simplest example is RSA, which is based on the difficulty of factorization of a sufficiently large number into primes. A more complicated example would be the ElGamal cryptosystem, which relies on a discrete logarithm problem. Currently, algorithms have slow solution. Finding a proven solution like that is a good starting problem for breaking more complex cryptosystems. One could also explore lattice-based methods and red-teaming exercises.

Appendix B. Incorportation of Lean

The advantages of using Lean is the access to mathematically proven libraries: Lean has a huge library of formalized mathematics which reduces our burden of defining objects, pre-and post-conditions manually, in comparison to Dafny.

Many of our examples in Appendix A are already formalized in Lean. As part of our pipeline, we shall utilize the Lean client python[6].

There are two further advantages in using Lean over Dafny which we shall make prominent in our new verification pipeline:

(1) The Library search in Lean

(2) Proof automation in Lean: `simp`, `aesop`, `aesop_cat`, `exact?`, `apply?`, `sledge hammer`.

[5] A derangement of size n is a permutation of n elements with no fixed points, while an (n, k)-derangement has exactly k fixed points.

[6] https://github.com/leanprover-community/lean-client-python

We shall exploit Lean's meta-programming capabilities to write new proof automation tactics that are amenable to automated safety checks for distilled programs, such as termination checks. Lastly, we hope such research creates feedback for the Lean community.

References

[Aba+23] Abate, Alessandro et al. "Quantitative Verification with Neural Networks". en. In: 2023 (cit. on p. 1).

[Ban+23] Bang, Yejin et al. "A Multitask, Multilingual, Multimodal Evaluation of ChatGPT on Reasoning, Hallucination, and Interactivity". In: *ArXiv* abs/2302.04023 (2023). URL: https://api.semanticscholar.org/CorpusID:256662612 (cit. on p. 1).

[CCN23] Chughtai, Bilal, Chan, Lawrence, and Nanda, Neel. *A Toy Model of Universality: Reverse Engineering How Networks Learn Group Operations*. 2023. arXiv: 2302.03025 [cs.LG] (cit. on p. 4).

[Che+23] Chen, Zhongtian et al. *Dynamical versus Bayesian Phase Transitions in a Toy Model of Superposition*. 2023. arXiv: 2310.06301 [cs.LG] (cit. on p. 1).

[Dal+24] Dalrymple, David "davidad" et al. *Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems*. 2024. arXiv: 2405.06624 [cs.AI] (cit. on p. 2).

[HMW23] Hendrycks, Dan, Mazeika, Mantas, and Woodside, Thomas. *An Overview of Catastrophic AI Risks*. 2023. arXiv: 2306.12001 [cs.CY] (cit. on p. 1).

[Ji+24] Ji, Jiaming et al. *AI Alignment: A Comprehensive Survey*. 2024. arXiv: 2310.19852 [cs.AI] (cit. on p. 1).

[Khl] Khlaaf, Heidy. "Toward Comprehensive Risk Assessments and Assurance of AI-Based Systems". In: URL: https://api.semanticscholar.org/CorpusID:259102957 (cit. on p. 1).

[Lei+17] Leike, Jan et al. "AI Safety Gridworlds". In: *ArXiv* abs/1711.09883 (2017). URL: https://api.semanticscholar.org/CorpusID:30283745 (cit. on p. 3).

[Lei10] Leino, K. Rustan M. "Dafny: An Automatic Program Verifier for Functional Correctness". In: *Logic Programming and Automated Reasoning*. 2010. URL: https://api.semanticscholar.org/CorpusID:8576464 (cit. on p. 2).

[Liu+23] Liu, Bingbin et al. *Transformers Learn Shortcuts to Automata*. 2023. arXiv: 2210.10749 [cs.LG] (cit. on p. 4).

[Mic+24] Michaud, Eric J. et al. *Opening the AI black box: program synthesis via mechanistic interpretability*. 2024. arXiv: 2402.05110 [cs.LG] (cit. on pp. 1, 3, 4).

[Min+19] Mingard, Chris et al. "Neural networks are a priori biased towards Boolean functions with low entropy". In: *ArXiv* abs/1909.11522 (2019). URL: https://api.semanticscholar.org/CorpusID:203565762 (cit. on p. 1).

[Par+23] Park, Peter S. et al. "AI deception: A survey of examples, risks, and potential solutions". In: *Patterns* 5 (2023). URL: https://api.semanticscholar.org/CorpusID:261276587 (cit. on p. 1).

[QB24] Quirke, Philip and Barez, Fazl. *Understanding Addition in Transformers*. 2024. arXiv: 2310.13121 [cs.LG] (cit. on p. 4).

[Rom+23] Romera-Paredes, Bernardino et al. "Mathematical discoveries from program search with large language models". In: *Nature* 625 (2023), pp. 468–475. URL: https://api.semanticscholar.org/CorpusID:266223700 (cit. on p. 2).

[SB98] Sutton, Richard S. and Barto, Andrew G. "Reinforcement Learning: An Introduction". In: *IEEE Trans. Neural Networks* 9 (1998), pp. 1054–1054. URL: https://api.semanticscholar.org/CorpusID:60035920 (cit. on p. 3).

[Tai+22] Taitler, Ayal et al. "pyRDDLGym: From RDDL to Gym Environments". In: *ArXiv* abs/2211.05939 (2022). URL: https://api.semanticscholar.org/CorpusID:253498970 (cit. on p. 3).

[Wen+23] Wenger, Emily et al. *SALSA: Attacking Lattice Cryptography with Transformers*. 2023. arXiv: 2207.04785 [cs.CR] (cit. on p. 4).

[Wit+23] Witt, Christian Schroeder de et al. *Perfectly Secure Steganography Using Minimum Entropy Coupling*. 2023. arXiv: 2210.14889 [cs.CR] (cit. on p. 2).