

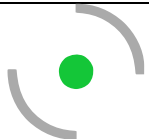


Working on Guaranteed Safe AI

quinn@beneficialaifoundation.org

April 19, 2025

Views are my own. GSAI is fairly new and has lots of authors with internal disagreements.

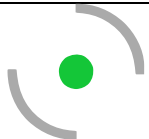


I. What is Guaranteed Safe AI?

What is Guaranteed Safe AI

Family of research agendas / AI safety strategies involving:

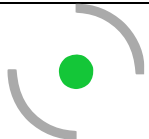
- **Proof certificates** over model outputs, not internals.



What is Guaranteed Safe AI

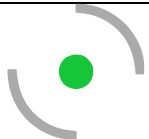
Family of research agendas / AI safety strategies involving:

- **Proof certificates** over model outputs, not internals. (Moreso what I work on)
- **World models** and simulations therein. (Moreso the davidad/ARIA side of the pond)



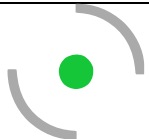
Proof certificates over model outputs, not internals

- There was some work trying to combine proof certs with mechinterp (Gross et al 2024), but it's not cruxy for GSAI.



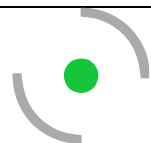
Proof certificates over model outputs, not internals

- There was some work trying to combine proof certs with mechinterp (Gross et al 2024), but it's not cruxy for GSAI.
- Instead, we're interested in the code that the learned component is writing (i.e., a *blackbox approach*).



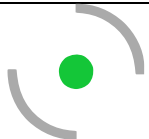
Proof certificates over model outputs, not internals

- There was some work trying to combine proof certs with mechinterp (Gross et al 2024), but it's not cruxy for GSAI.
- Instead, we're interested in the code that the learned component is writing (i.e., a *blackbox approach*).
- We would like that code to be formally verifiable



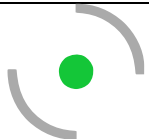
Proof certs over model outputs: example

Imagine Anthropic asks Claude to write a **secure cloud OS kernel**.



Proof certs over model outputs: example

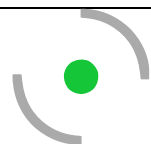
Imagine Anthropic asks Claude to write a **secure cloud OS kernel**. In addition to synthesizing the **implementation**, Claude should also synthesize **proofs** that the implementation is *correct*.



Proof certs over model outputs: example

Imagine Anthropic asks Claude to write a **secure cloud OS kernel**. In addition to synthesizing the **implementation**, Claude should also synthesize **proofs** that the implementation is *correct*.

$$P\{c\}Q \leftrightarrow \forall(ss' : \text{State}), P(s) \rightarrow \text{exec}(c, s) = s' \rightarrow Q(s')$$

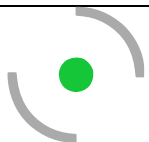


Proof certs over model outputs: example

Imagine Anthropic asks Claude to write a **secure cloud OS kernel**. In addition to synthesizing the **implementation**, Claude should also synthesize **proofs** that the implementation is *correct*.

$$P\{c\}Q \leftrightarrow \forall(ss' : \text{State}), P(s) \rightarrow \text{exec}(c, s) = s' \rightarrow Q(s')$$

Predicate P is a *precondition* and predicate Q is a *postcondition*.



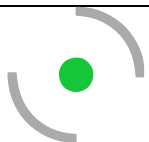
Proof certs over model outputs: example

Imagine Anthropic asks Claude to write a **secure cloud OS kernel**. In addition to synthesizing the **implementation**, Claude should also synthesize **proofs** that the implementation is *correct*.

$$P\{c\}Q \leftrightarrow \forall(ss' : \text{State}), P(s) \rightarrow \text{exec}(c, s) = s' \rightarrow Q(s')$$

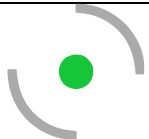
Predicate P is a *precondition* and predicate Q is a *postcondition*.

Together, they form a **specification** for program c .



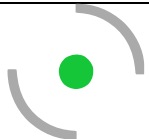
World models and simulations therein

To evaluate how an **output** *affects* the world, you could **simulate** the world.



World models and simulations therein

To evaluate how an **output** *affects* the world, you could **simulate** the world. One form of **probabilistic proof cert** could be a set of trajectories in a simulation with the observation that the world turns out “bad” in bounded percentage of them.

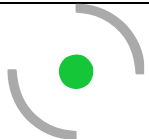


World models and simulations therein

To evaluate how an **output** *effects* the world, you could **simulate** the world. One form of **probabilistic proof cert** could be a set of trajectories in a simulation with the observation that the world turns out “bad” in bounded percentage of them.

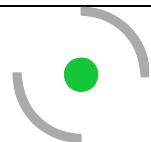
$$\mathbb{E}_{\omega \sim \Omega} \left(\exists_{v:V} \text{Injured}(v) (\text{WorldModel}(S \mapsto \text{AIController})(\omega)) \wedge \neg \text{Injured}(v) (\text{WorldModel}(S \mapsto \text{DoNothing})(\omega)) \right) < 10^{-4}$$

(davidad 2024)



World models and simulations: example

Consider the stochastic heat equation: $\frac{\partial u(t, x)}{\partial t} = \alpha \nabla^2 u(t, x) + \sigma \xi(t, x)$

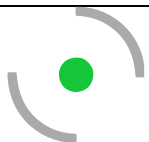


World models and simulations: example

Consider the stochastic heat equation: $\frac{\partial u(t, x)}{\partial t} = \alpha \nabla^2 u(t, x) + \sigma \xi(t, x)$

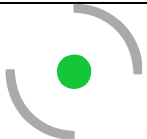
$P_{\{\geq p\}} [F^{\{\leq t\}} (|u(t, x) - u_{\text{eq}}| < \delta)]:$ “The probability that temperature reaches near-equilibrium within time t is at least p ”

This is a spec in *continuous stochastic logic*, and you can get certs about a system with model checking.



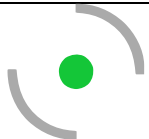
Assumptions

- **Boxing/containment**: probably doesn't work with arbitrarily unboxing schemers
- **Swiss cheese**: “defense in depth”



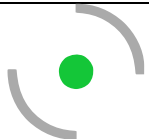
Boxing/containment

- We're **micromanaging the interface** between the AI and the world.



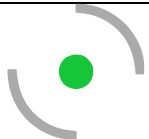
Boxing/containment

- We're **micromanaging the interface** between the AI and the world.
- The aim is to get us through *some* stages of AGI (say, 25% of acute risk period) and hopefully our successors at that point will bootstrap it into solutions that work for arbitrary superintelligences

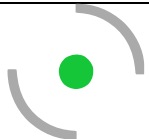
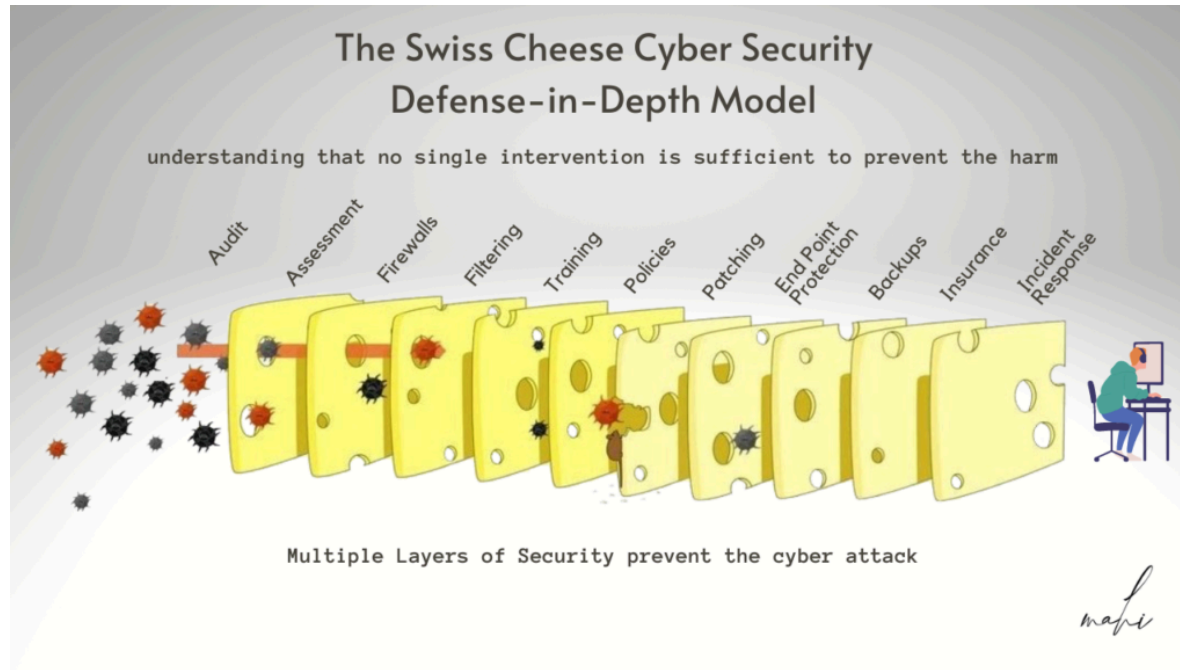


Boxing/containment

- We're **micromanaging the interface** between the AI and the world.
- The aim is to get us through *some* stages of AGI (say, 25% of acute risk period) and hopefully our successors at that point will bootstrap it into solutions that work for arbitrary superintelligences
- But even formal methods would leave side channel attacks open for a scheming ASI to exploit



Swiss cheese (defense in depth)



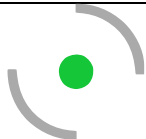
Main position paper

[Submitted on 10 May 2024 (v1), last revised 8 Jul 2024 (this version, v3)]

Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems

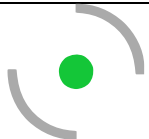
David "davidad" Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette Wing, Joshua Tenenbaum

Ensuring that AI systems reliably and robustly avoid harmful or dangerous behaviours is a crucial challenge, especially for AI systems with a high degree of autonomy and general intelligence, or systems used in safety-critical contexts. In this paper, we will introduce and define a family of approaches to AI safety, which we will refer to as guaranteed safe (GS) AI. The core feature of these approaches is that they aim to produce AI systems which are equipped with high-assurance quantitative safety guarantees. This is achieved by the interplay of three core components: a world model (which provides a mathematical description of how the AI system affects the outside world), a safety specification (which is a mathematical description of what effects are acceptable), and a verifier (which provides an auditable proof certificate that the AI satisfies the safety specification relative to the world model). We outline a number of approaches for creating each of these three core components, describe the main technical challenges, and suggest a number of potential solutions to them. We also argue for the necessity of this approach to AI safety, and for the inadequacy of the main alternative approaches.



Synergies with other agendas

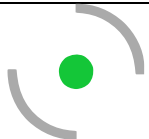
- Cybersecurity
- Control



GSAIxCybersecurity

Similarities:

- Reducing *attack surface*
- Hardening infrastructure



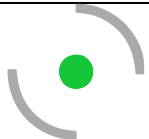
GSAIxCybersecurity

Similarities:

- Reducing *attack surface*
- Hardening infrastructure

Differences:

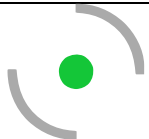
- GSAI emphasizes *prevention* more than *detection*



GSAILxControl

Similarities:

- Blackbox as opposed to whitebox
- Containment schemes as opposed to assuming alignment



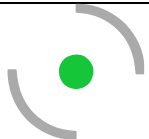
GSAIxControl

Similarities:

- Blackbox as opposed to whitebox
- Containment schemes as opposed to assuming alignment

Differences:

- Control emphasizes insider risk more than GSAI
- GSAI looks for formal proof and subtopics where that makes sense (i.e. where there is inductive structure to be exploited)



II. What is working on GSAI like (example projects)

FVAPPS

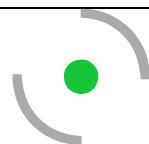
Formally Verified APPS, translating Hendrycks et al 2021 into a Lean benchmark

[Submitted on 8 Feb 2025]

Proving the Coding Interview: A Benchmark for Formally Verified Code Generation

Quinn Dougherty, Ronak Mehta

We introduce the Formally Verified Automated Programming Progress Standards, or FVAPPS, a benchmark of 4715 samples for writing programs and proving their correctness, the largest formal verification benchmark, including 1083 curated and quality controlled samples. Previously, APPS provided a benchmark and dataset for programming puzzles to be completed in Python and checked against unit tests, of the kind seen in technical assessments in the software engineering industry. Building upon recent approaches for benchmarks in interactive theorem proving, we generalize the unit tests to Lean 4 theorems given without proof (i.e., using Lean's "sorry" keyword). On the 406 theorems of 100 randomly selected samples, Sonnet correctly proves 30% and Gemini correctly proves 18%. We challenge the machine learning and program synthesis communities to solve both each general purpose programming problem and its associated correctness specifications. The benchmark is available at [this https URL](https://fvapps.github.io).



FVAPPS: Formally Verified APPS

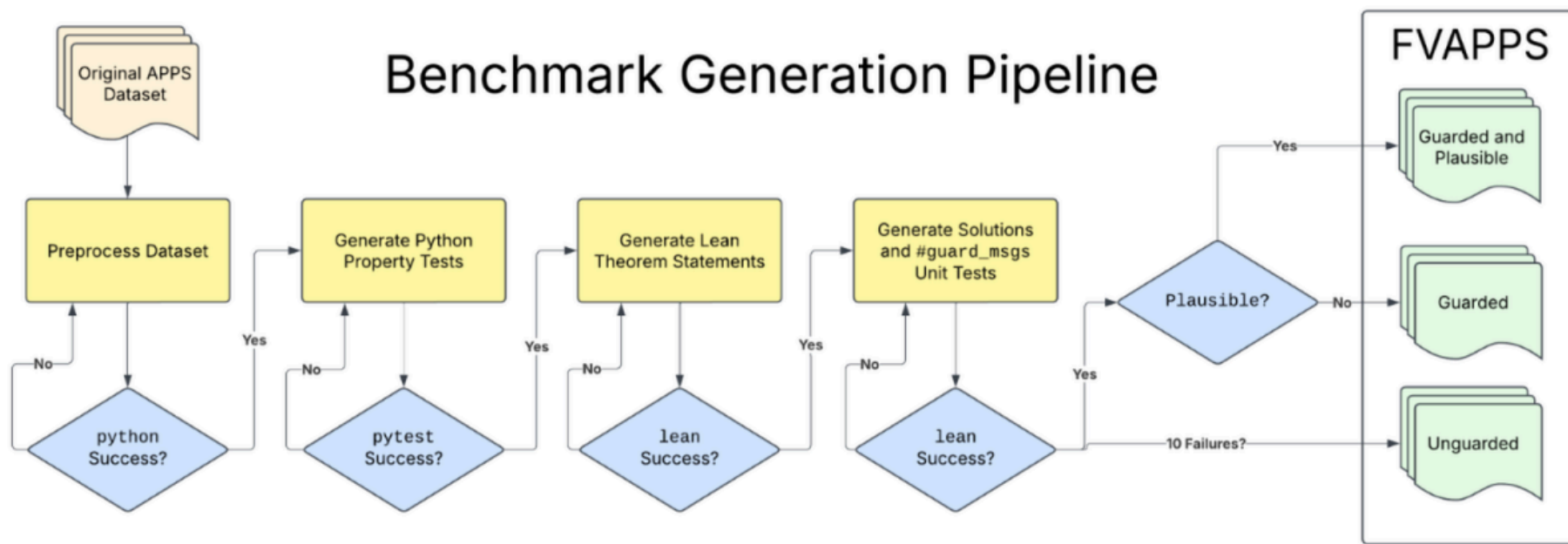
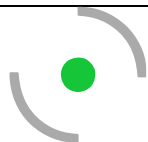
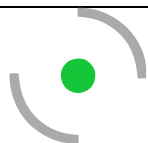
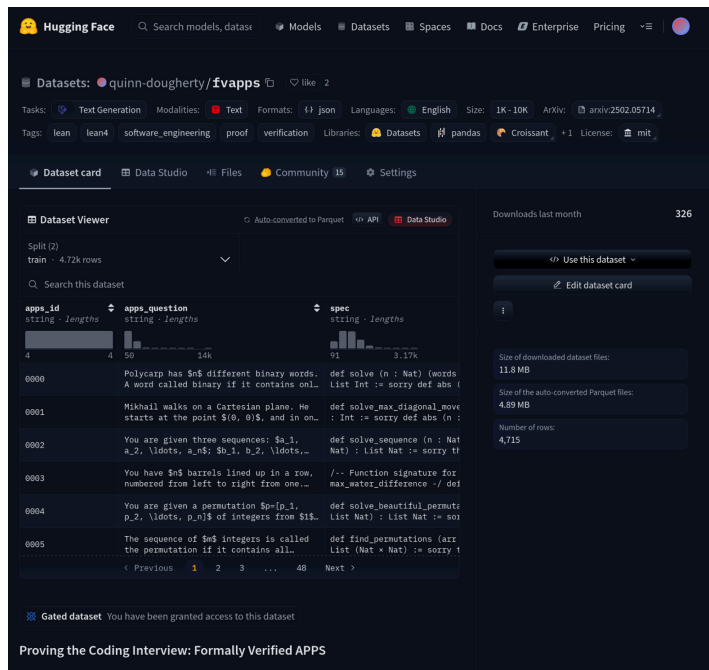


Fig. 1. Benchmark generation pipeline for creating coding interview theorem statements in Lean from APPS questions and solutions.



FVAPPS: Formally Verified APPS (on HuggingFace)

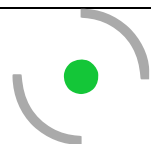


FVAPPS: Formally Verified APPS (example sample puzzle)

Now elections are held in Berland and you want to win them. More precisely, you want everyone to vote for you.

There are n voters, and two ways to convince each of them to vote for you. The first way to convince the i -th voter is to pay him p_i coins. The second way is to make m_i other voters vote for you, and the i -th voter will vote for free.

Moreover, the process of such voting takes place in several steps. For example, if there are five voters with $m_1 = 1$, $m_2 = 2$, $m_3 = 2$,

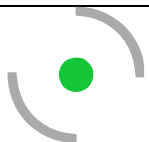


$m_4 = 4, m_5 = 5$, then you can buy the vote of the fifth voter, and eventually everyone will vote for you. Set of people voting for you will change as follows: $\{5\} \rightarrow \{1, 5\} \rightarrow \{1, 2, 3, 5\} \rightarrow \{1, 2, 3, 4, 5\}$.

Calculate the minimum number of coins you have to spend so that everyone votes for you.

FVAPPS: Formally Verified APPS (example sample)

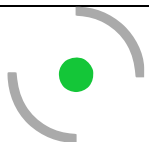
```
def solve_elections (n : Nat) (voters : List (Nat ×  
Nat)) : Nat := sorry
```



```
theorem solve_elections_nonnegative (n : Nat) (voters :  
List (Nat × Nat)) : solve_elections n voters ≥ 0 := sorry
```

```
theorem solve_elections_upper_bound (n : Nat) (voters :  
List (Nat × Nat)) : solve_elections n voters ≤ List.foldl  
(λ acc (pair : Nat × Nat) => acc + pair.2) 0 voters :=  
sorry
```

```
theorem solve_elections_zero_votes (n : Nat) (voters :  
List (Nat × Nat)) : (List.all voters (λ pair => pair.1 =  
0)) → solve_elections n voters = 0 := sorry
```

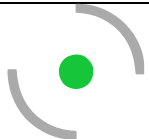


```
theorem solve_elections_single_zero_vote :  
solve_elections 1 [(0, 5)] = 0 := sorry
```

FVAPPS: Formally Verified APPS (for synthetic data)

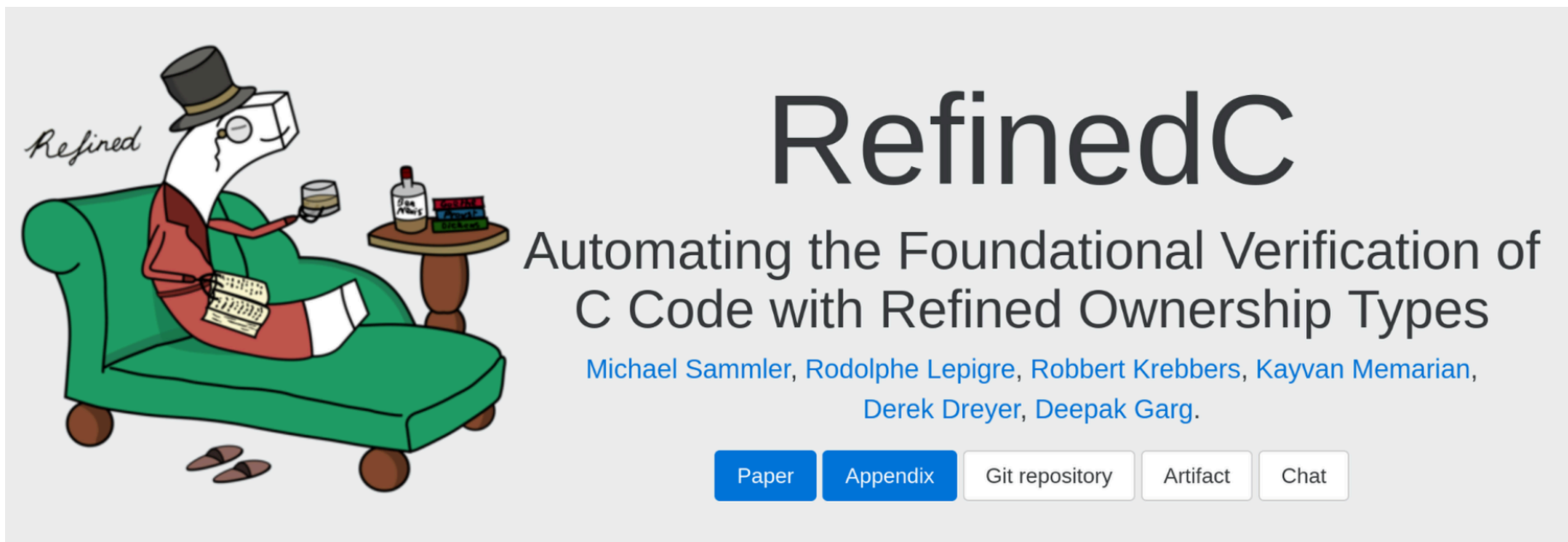
A benchmark with tool use as ground truth can be a **synthetic data pipeline**

- I'm excited for using FVAPPS solutions to make finetune datasets



RefinedC Copilot

Automating C verification (*in progress*)

The banner features a cartoon character on the left, a white figure with a top hat and monocle, sitting on a green chaise longue and holding a glass. A small table next to it holds a bottle and books. The word 'Refined' is written in cursive above the character. To the right, the title 'RefinedC' is in large bold letters, followed by the subtitle 'Automating the Foundational Verification of C Code with Refined Ownership Types'. Below this are the authors' names in blue. At the bottom right are five buttons: 'Paper', 'Appendix', 'Git repository', 'Artifact', and 'Chat'.

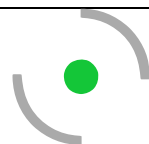
Refined

RefinedC

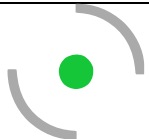
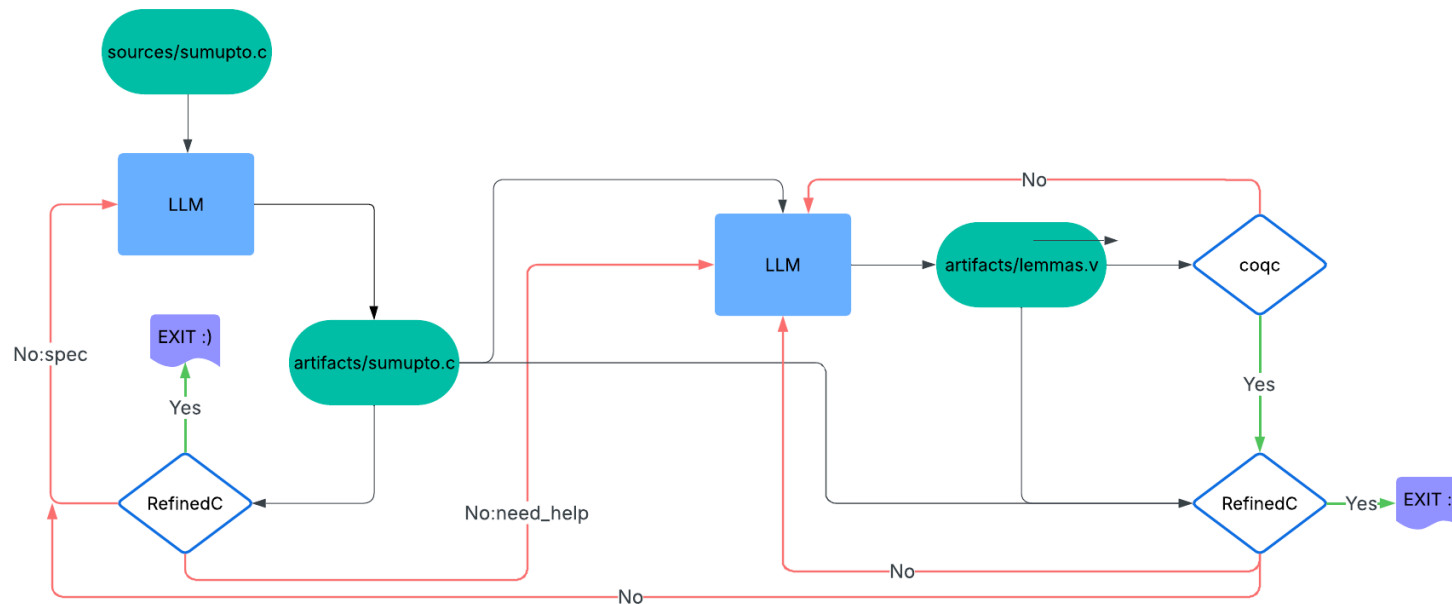
Automating the Foundational Verification of
C Code with Refined Ownership Types

Michael Sammler, Rodolphe Lepigre, Robbert Krebbers, Kayvan Memarian,
Derek Dreyer, Deepak Garg.

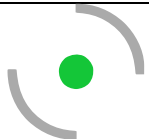
[Paper](#) [Appendix](#) [Git repository](#) [Artifact](#) [Chat](#)



RefinedC Copilot: A Scaffold

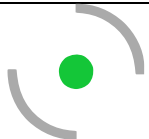


RefinedC Copilot: From 99.9% to 99.999%



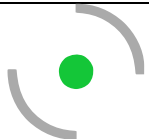
RefinedC Copilot: From 99.9% to 99.999%

- Right now, gaining nines (of assurance) is not cost effective



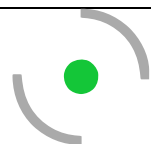
RefinedC Copilot: From 99.9% to 99.999%

- Right now, gaining nines (of assurance) is not cost effective
- Because labor costs of formal verification



RefinedC Copilot: From 99.9% to 99.999%

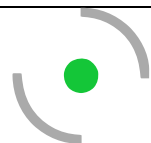
- Right now, gaining nines (of assurance) is not cost effective
- Because labor costs of formal verification
- Maybe we can make that cheaper with automation?



RefinedC Copilot: From 99.9% to 99.999%

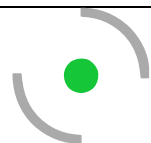
- Right now, gaining nines (of assurance) is not cost effective
- Because labor costs of formal verification
- Maybe we can make that cheaper with automation?

It's a live hypothesis, the project isn't super far along yet.



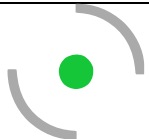
III. Should you work on GSAI

Are the arguments convincing?



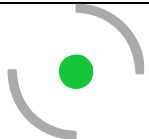
Are the arguments convincing?

- Can you **poke holes** in the worldview?



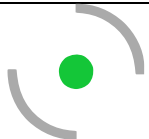
Are the arguments convincing?

- Can you **poke holes** in the worldview?
- Comparing/contrasting with competing agendas

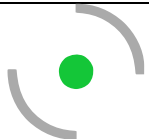


Are the arguments convincing?

- Can you **poke holes** in the worldview?
- Comparing/contrasting with competing agendas
- Estimate $p(\text{success})$, $p(\text{success} \mid x \text{ investment})$, etc.

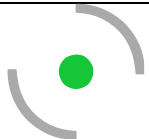


Personal fit (skills, interests, what is fun)



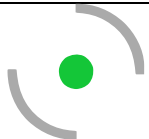
Personal fit (skills, interests, what is fun)

- Formal verification



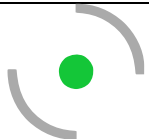
Personal fit (skills, interests, what is fun)

- Formal verification
- Programming language theory (including probabilistic semantics)



Personal fit (skills, interests, what is fun)

- Formal verification
- Programming language theory (including probabilistic semantics)
- ML/LLM stuff is mostly from a black box perspective. Lots of scaffolds.



IV. Hear more at gsai.substack.com



Working on Guaranteed Safe AI

quinn@beneficialaifoundation.org

April 19, 2025