

# Formal Containment draft

Quinn Dougherty

Research Engineer

Beneficial AI Foundation

Cambridge, MA

quinn@beneficialaifoundation.org

**Abstract**—We would like to put the AI in a box. We show how to create an *interface* between the box and the world using specifications in Lean. It is the AI’s responsibility to provide a proof that it’s (restricted) output abides by the spec.

**Index Terms**—AI, ML, AI Safety, Formal Verification

## I. INTRODUCTION

*Prereqs: hoare triples, errors as values, mcp*

Previously, classical AI safety warned against trying to put the AI in a box [1].

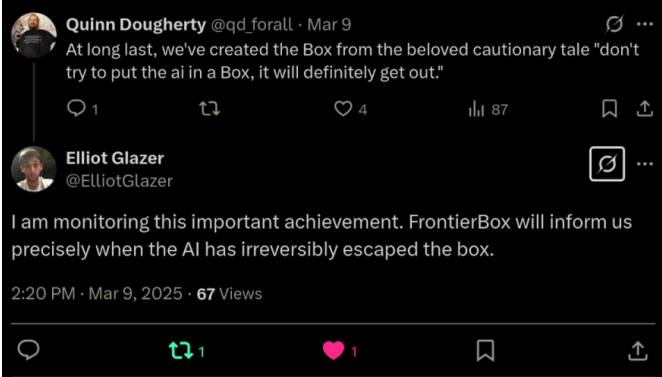


Fig. 1: Made funnier by the reply being from the author of FrontierMath

Yudkowsky famously convinced people of this via a roleplay game back in the extropians mailing list days [2].

Recent work from AI Control and Safeguarded AI have reignited interest in this long neglected area.

- Like AI Control [3], [4], the current work is reasoning about getting useful work out of an untrusted (possibly misaligned) ai.
- Like Safeguarded AI [5], we try to isolate the trust surface to a specification and a proof checker.

## II. THE FORMAL CONTAINMENT PROTOCOL

A box is constructed in the MCP standard.

Let an MCP [6] server’s codomain be the data structure formed from an oracle’s completion formatted to be sent over the MCP wire. The **interface** between the box and the world is an MCP client equipped with a **whitelist**, which is a subtype of the codomain of the MCP server that wraps the llm/oracle. The client has children, aspects of the world (like a git repo or deployment environment). The client decides whether a completion can be sent to the world based on some criterion programmed in by the user.

In this whitepaper, the criterion will be a precondition and postcondition of a hoare triple [7], since the oracle throughout this prototype is an imp programmer. Imp is a minimal imperative programming language known for its pedagogical properties.

### A. Example

MCP server  $S$  has a Lean tool use and the following codomain

```
Enumerated codS {
  Pass: prec x post -> codS,
  Fail: prec x post -> stderr -> codS
}
```

MCP client  $B$  is equipped with the `Pass` constructor of `codS` (it’s *whitelist*).

Let criterion  $C$  be the pair  $(\text{prec}, \text{post}) := (\lambda \text{ st. st } X > 0, \lambda \text{ st. st } X > 1)$ .

Then, programs  $c$  provided by imp oracle  $\mathcal{O}_i$  at relevant prompt  $p$  will form the command at the center of the hoare triple. In other words,  $\forall c \sim \mathcal{O}_i(p), \{\text{prec}\}c\{\text{post}\}$  is either provable or not.

When  $\mathcal{O}_i$  submits a program  $c$  to the interface client  $B$ ,  $B$  sends the triple  $\{\text{prec}\}c\{\text{post}\}$  to the proof oracle  $\mathcal{O}_p$  in one thread and the triple’s negation in another thread. If the negative thread terminates in success, then a negative certificate is sent to  $B$  and  $B$  informs  $\mathcal{O}_i$  that it shall not run that code in the world, and that  $\mathcal{O}_i$  must try again. If the positive thread terminates in success, then the positive certificate is sent to  $B$  and  $B$  passes along program  $c$  to its children to be run in the world.

## REFERENCES

- [1] LessWrong Contributors, “AI Boxing (Containment).” [Online]. Available: <https://www.lesswrong.com/w/ai-boxing-containment>
- [2] E. Yudkowsky, “The AI-Box Experiment.” [Online]. Available: <https://www.yudkowsky.net/singularity/aibox>
- [3] R. Greenblatt, B. Shlegeris, K. Sachan, and F. Roger, “AI Control: Improving Safety Despite Intentional Subversion.” [Online]. Available: <https://arxiv.org/abs/2312.06942>
- [4] A. Bhatt *et al.*, “Ctrl-Z: Controlling AI Agents via Resampling.” [Online]. Available: <https://arxiv.org/abs/2504.10374>
- [5] D. Dalrymple, “Safeguarded AI: Constructing Guaranteed Safety,” 2024. [Online]. Available: <https://www.aria.org.uk/media/3nhijno4/aria-safeguarded-ai-programme-thesis-v1.pdf>
- [6] Anthropic, PBC, “Model Context Protocol (MCP).” Accessed: May 06, 2025. [Online]. Available: <https://modelcontextprotocol.io/introduction>
- [7] “Hoare.” in Software Foundations. 2024. [Online]. Available: <https://softwarefoundations.cis.upenn.edu/plf-current/Hoare.html>