Formal Containment

Quinn Dougherty
Research Engineer
Beneficial AI Foundation
Cambridge, MA
quinn@beneficialaifoundation.org

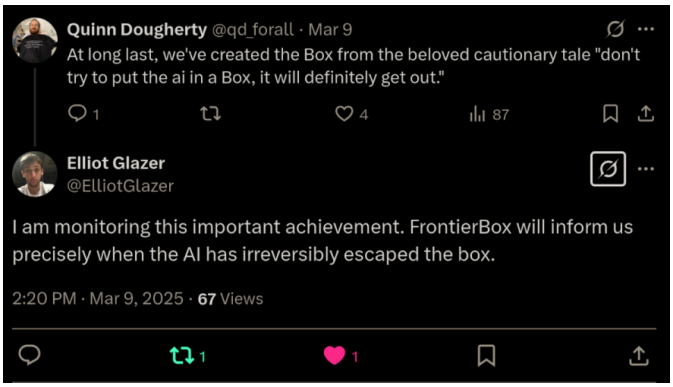
***Abstract*—We would like to put the AI in a box. We show how to create an *interface* between the box and the world using specifications in Lean. It is the AI's responsibility to provide a proof that it's (restricted) output abides by the spec.**

***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.



Eliezer famously convinced someone of this via a roleplay game back in the extropians mailing list days.

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

- Like AI Control, the current work is reasoning about getting useful work out of an untrusted (possibly misaligned) ai.
- Like Safeguarded AI, 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 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, 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 -> codS
}
```

Mcp client B is equipped with the pass constructor of `codS`.

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

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

When O_i submits a program c to the interface client B, B sends the triple $\{\text{prec}\}c\{\text{post}\}$ to the proof oracle 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 O_i that it shall not run that code in the world, and that 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