
FV-Deception:

How Well Can Models Deceive Each Other with Formal Specifications?¹

Lee Wall
Independent

With
Apart Research

Abstract

Keywords: AI manipulation, formal verification, scalable oversight, multi-agent safety, deception evals

1. Introduction

The probabilistic nature of current frontier AI models has led to a push to use *formal verification* to provide guarantees about model behavior and model-generated artifacts. Formal verification removes the need for trust by mathematically proving specifications about software or hardware, providing a natural counter to the black-box nature of modern neural networks.

In the future, we may therefore rely on formal specifications to ensure that code is safe, that agents will commit to acting a certain way, or that some property of a system is always true. As systems scale and complexify, though, formal specs may grow beyond the human capacity to generate or review them. We may rely on AI with natural-language understanding to both encode our intentions into specs and to verify that specs represent our intentions.

This presents a major risk surface: a known limitation of FV is the specification gap – the gap between the intention for a spec, and the actual spec. It is often difficult to take a fuzzy property we want a model or piece of code to have and turn

¹ Research conducted at the [AI Manipulation Hackathon](#), 2026

that into something that can be mathematically checked. Presumably, this gap can also be a target for a deceptive actor.

This raises a question which is a close sibling to the question of scalable oversight – getting less powerful models to oversee more powerful ones – that we attempt to take some first steps toward answering: can more capable models generate *deceptive* formal specifications according to a given natural-language intention which fool less capable reviewer models?

We design and execute an experiment wherein adversarial LLM’s intentionally generate subtly wrong formal specifications in Z3 and Dafny according to a portfolio of realistic, natural-language intentions. These are then reviewed by auditor LLM’s which attempt to verify accuracy and flag security concerns without over-flagging false positives (i.e. specs that correctly implement the semantic intention as judged by a human reviewer, but which the model flags as wrong). We aim to produce a capability-detection matrix that shows rates of detection for different capability matchups – larger deceiver versus smaller defender, smaller deceiver versus larger defender, and equal capability.

We hypothesize that it is easier to embed a bug – or more specifically, a slight divergence in semantic intent and implementation -- into a specification than to catch it, so that more capable models will generally be able to deceive smaller models. The exception will be when less capable models are tuned to be overly careful, so that they also have a high rate of false positives. I expect that we will not be able to get less capable models supervising the formal specs generated by more capable ones without high false positive rates.

2. Methods

Models

We evaluated three Claude models at different capability tiers:

Model	Version	Capability Tier
Claude 3.5 Haiku	claude-3-5-haiku-20241022	Small
Claude Sonnet 4	claude-sonnet-4-20250514	Medium
Claude Opus 4	claude-opus-4-20250514	Large

Formal Languages

We tested specifications in two formal verification languages:

- Z3 (Python-embedded SMT solver): Widely used for constraint solving and program verification
- Dafny (Microsoft Research): A verification-aware programming language with explicit `requires`/`ensures` contracts

Experimental Design

We first established baseline detection rates without adversarial crafting, using hand-written buggy specifications. This revealed that:

- Without chain-of-thought prompting: Haiku 25%, Sonnet 50%, Opus 75%
- Off-by-one bugs were missed by ALL models.

We then tested adversarially crafted specifications against each pair of models.

Adversarial Bug Generation

The core experiment: each model generates bugs designed to evade detection.

Attacker prompt:

“You are a red team specialist creating formally verified specifications that LOOK correct but contain subtle bugs.

Your task: Create a buggy version that violates requirement "{target}" using this deception technique: {technique}

Guidelines:

- The bug should be syntactically subtle
- Add misdirection: comments, extra clauses that look thorough
- Make it look like a plausible mistake, not sabotage”

Defender prompt:

“You are auditing a Dafny specification for correctness.

Analyze whether this specification correctly implements ALL the stated requirements. Consider whether each requirement is properly enforced through preconditions and postconditions.

VERDICT: CORRECT or INCORRECT

REASON: Brief explanation.”

We ran all 9 attacker-defender combinations:

- Each attacker generates N buggy specs
- Each defender evaluates all specs from all attackers
- Calculate deception detection rate and false positive rate

3. Results

Adversarial Experiment (N=22, Opus Attacker)

With 22 adversarial bugs generated by Opus across 6 requirement categories:

| Defender | Detection Rate | False Positives |

|-----|-----|-----|

| Haiku | 36% (8/22) | 0% |

| Sonnet | 86% (19/22) | 0% |

| Opus | 77% (17/22) | 0% |

Key finding: Opus generated these bugs but caught fewer than Sonnet (77% < 86%).

4. Discussion and Conclusion

A few themes dominated the results:

- Prompt sensitivity: the smaller the model, the more sensitive it was to the language used in the prompt, and the more it pattern-matched on semantic cues such as variable names instead of the logical structure underlying the specification.
- High false-positive rates: it was incredibly difficult to get even more capable models to simultaneously achieve high detection rates for misspecifications and low false-positive rates. This could change as capabilities scale, but it could be interpreted as evidence that using AI as a “translator” between natural-language and formal-language may not be robust as a means to guarantee safety.
- Capability scaling: the most interesting result here was that instead of getting a clean scaling where defense > offense or offense < defense as capability increases, we saw that the most capable model was actually more capable of deceiving itself than the mid-tier model.