Applying Formal Verification to Reflective Reasoning

R. Kumar¹  B. Fallenstein²

¹Data61, CSIRO and UNSW
ramana@intelligence.org

²Machine Intelligence Research Institute
benya@intelligence.org

Artificial Intelligence for Theorem Proving, Obergurgl 2017
Who am I?

Ramana Kumar

PhD, University of Cambridge

Researcher, Data61, CSIRO

Theorem Proving in HOL
Context: Beneficial AI

Source: Future of Humanity Institute, Oxford.
See also: https://intelligence.org/why-ai-safety/
Context: Beneficial AI

Technical Agenda
Context: Beneficial AI

Technical Agenda

Highly Reliable Agent Design
Context: Beneficial AI

Technical Agenda

Highly Reliable Agent Design

- Foundations
- Basic problems lacking in-principle solutions
Context: Beneficial AI

Technical Agenda

Highly Reliable Agent Design

- Foundations
- Basic problems lacking in-principle solutions

(Note: This is not MIRI’s only research agenda.)
One problem within MIRI’s 2014 agenda happened to seem to align with my expertise, theorem proving and self-verification.
Problem Statement

Design a system that ▶ always satisfies some safety property, ▶ but is otherwise capable of arbitrary self-improvement.
Problem Statement

Design a system that

- always satisfies some safety property,
- but is otherwise capable of arbitrary self-improvement.
Problem of Self Trust

Too little self-trust
Cannot make simple self-modifications

Too much self-trust
Unsound reasoning about successors
Overview

**Reflective Reasoning**
- Self-Modifying Agents
- Vingean Reflection
- Suggester-Verifier Architecture
- Problem and Partial Solutions

**Implementation**
- Botworld
- Formalisation in HOL
Reflective Reasoning
The Agent Framework

\[ \pi(oa_{1:n}) = a_{n+1} \]
The Agent Framework

\[ \pi(\bar{oa}_1:n) = a_{n+1} \]

Cartesian boundary

- agent computed outside environment
Reality is not Cartesian

\[
\pi_n(\o_n) = (a_{n+1}, \uparrow \pi_{n+1} \downarrow)
\]
Reality is not Cartesian

\[ \pi_n(o_n) = (a_{n+1}, \lceil \pi_{n+1} \rceil) \]
Vingean Principle

One can reason only \textit{abstractly} about a stronger reasoner
Vingeans Principle

One can reason only *abstractly* about a stronger reasoner

Relevance
Self-improving system must reason about programs it cannot run: its successors
Vingeon Principle

One can reason only abstractly about a stronger reasoner

Relevance
Self-improving system must reason about programs it cannot run: its successors

Approach
Formal logic as a model of abstract reasoning
Suggester-Verifier Architecture
Suggester-Verifier Architecture

Suggester
sophisticated, untrusted

Verifier

observation

\[ \pi, a \]

proof

\[ \pi, a \] or \[ \text{default} \]
Suggester-Verifier Architecture

\[ \pi, a \]

\[ \text{Verify: } \vdash u(h(\pi, a)) \geq u(h(\text{default})) \]
Suggester-Verifier Architecture

Suggester
sophisticated, untrusted

Verifier

π, a
proof

π, a  or  default

Verify: \( \vdash u(h(\pi, a)) \geq u(h(\text{default})) \approx \text{Safe}(a) \)
Problem with Self-Modification

Argument for Safety of Successor

- To create a successor, must prove that its actions will be safe
- If successor follows s-v architecture, it will only take actions it has proven to be safe
- However, to conclude that an action is actually safe from a proof is problematic.
Problem with Self-Modification

Argument for Safety of Successor

- To create a successor, must prove that its actions will be safe
- If successor follows s-v architecture, it will only take actions it has proven to be safe
- However, to conclude that an action is actually safe from a proof is problematic.

This principle, $T \vdash □_T \neg \varphi \Rightarrow \varphi$, is inconsistent. (Gödel/Löb)
Partial Solutions

Descending Trust

\[ T_{100} \vdash \Box T_{99} \Downarrow \varphi \Downarrow \Rightarrow \varphi, \ T_{99} \vdash \Box T_{98} \Downarrow \varphi \Downarrow \Rightarrow \varphi, \]

\[ \ldots \]
Partial Solutions

Descending Trust

\[ T_{100} \vdash \Box T_{99} \models \varphi \quad \Rightarrow \quad \varphi, \quad T_{99} \vdash \Box T_{98} \models \varphi \quad \Rightarrow \quad \varphi, \quad \ldots \]

Model Polymorphism

\[ 0 < \kappa, \quad T \vdash \forall n. \Box T \models \varphi(\bar{n}) \quad \Rightarrow \quad \varphi[\kappa - 1/\kappa](n) \]
Model Polymorphism

\[ 0 < \kappa, \ T \vdash \forall n. \Box T \varphi(\bar{n}) \downarrow \implies \varphi[\kappa - 1/\kappa](n) \]
Model Polymorphism

\[
0 < \kappa, \quad T \vdash \forall n. \Box_T \varphi(\bar{n}) \Downarrow \implies \varphi[\kappa - 1/\kappa](n)
\]

If \( \text{Safe}(a) \equiv \forall n. \text{Safe}(a, n) \)

Take \( \varphi(n) \equiv n \leq \kappa \implies \text{Safe}(a, n) \)
Model Polymorphism

\[ 0 < \kappa, \ T \vdash \forall n. \Box_T \phi(\bar{n}) \downarrow \implies \phi[\kappa - 1/\kappa](n) \]

If \( \text{Safe}(a) \equiv \forall n. \text{Safe}(a, n) \)

Take \( \phi(n) \equiv n \leq \kappa \implies \text{Safe}(a, n) \)

\[ \forall a. \Box \forall n \leq t + 1 + \kappa. \text{Safe}(\bar{a}, n) \downarrow \implies \forall n \leq t + \kappa. \text{Safe}(a, n) \]
Implementation
Botworld: Concrete Framework for Embedded Agents

Robots can construct/inspect/destroy/program other robots
Botworld Formalisation

Semantics

- step : state \rightarrow state
Botworld Formalisation

Semantics

- step : state → state
- Robots run policies in CakeML
Botworld Formalisation

Semantics

- step : state $\rightarrow$ state
- Robots run policies in CakeML

Counterfactuals

- state-with-hole for proposed action
Botworld Formalisation

Semantics

▶ step : state → state
▶ Robots run policies in CakeML

Counterfactuals

▶ state-with-hole for proposed action
▶ steph : s-w-h → a → (obs, state) option
Suggester-Verifier Implementation

\[ \text{sv}(\pi_{\text{default}}, \sigma, \text{obs}): \]

1. \((\pi, a) = \text{run } \pi_{\text{default}}\)
2. \((\pi', a', \text{thm}) = \text{run } \sigma(\text{obs}, \pi, a)\)
3. Check \(\text{thm}\) has correct form
4. Write \((\pi, a)\) or \((\pi', a')\) accordingly
Suggester-Verifier Implementation

\[ sv(\pi_{\text{default}}, \sigma, \text{obs}) : \]

1. \((\pi, a) = \text{run } \pi_{\text{default}}\)
2. \((\pi', a', \text{thm}) = \text{run } \sigma(\text{obs}, \pi, a)\)
3. Check thm has correct form
4. Write \((\pi, a)\) or \((\pi', a')\) accordingly

Reflection Library

Automation for: \(\square LCA \bar{k} \implies P \bar{\downarrow}\) implies LCA \((k + 1) \implies P\)
Implementation Challenge

Project Proposal

Build a Botworld agent that self-modifies into a *provably safe* agent of the same architecture.
Implementation Challenge

Project Proposal
Build a Botworld agent that self-modifies into a provably safe agent of the same architecture.

Eventual Project
Discover how far theorem proving technology is from implementing the above...
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years.
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years. (building on \( \geq 25 \) in prereqs)
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years. (building on > 25 in prereqs)
- Improvements on model polymorphism would be nice!
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years. (building on > 25 in prereqs)
- Improvements on model polymorphism would be nice!

Theorem Proving for AI

- Specifications Needed!
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years. (building on $> 25$ in prereqs)
- Improvements on model polymorphism would be nice!

Theorem Proving for AI

- Specifications Needed!
- Novel Architectures for AI Systems, e.g., improve on Suggester-Verifier to support logical induction and non-proof-based reasoning
Outlook

Implementing a Self-Improving Botworld Agent

- Looks possible, but with more effort than anticipated
- I would estimate 4 person-years. (building on > 25 in prereqs)
- Improvements on model polymorphism would be nice!

Theorem Proving for AI

- Specifications Needed!
- Novel Architectures for AI Systems, e.g., improve on Suggester-Verifier to support logical induction and non-proof-based reasoning
- Reducing Problems to Functional Correctness