

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

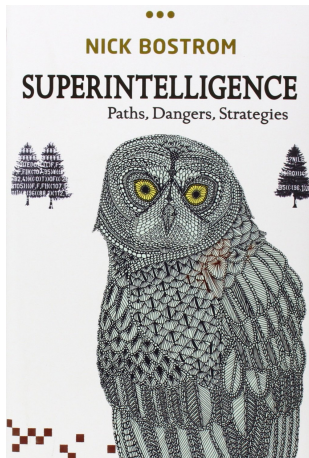
PhD, University of Cambridge

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



Context: Beneficial AI



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

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

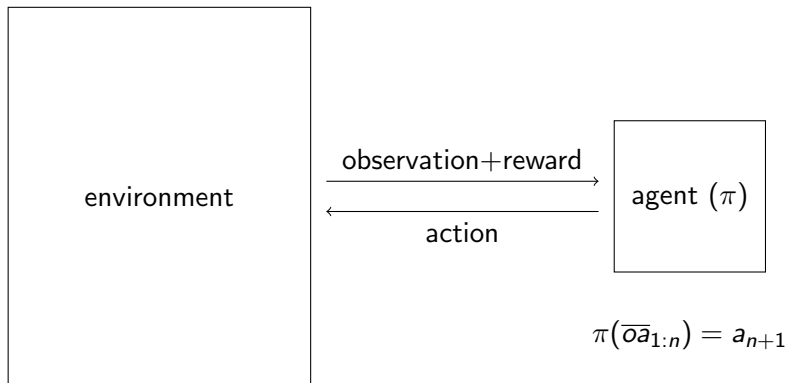
- ▶ Self-Modifying Agents
- ▶ Vingean Reflection
- ▶ Suggester-Verifier Architecture
- ▶ Problem and Partial Solutions

Implementation

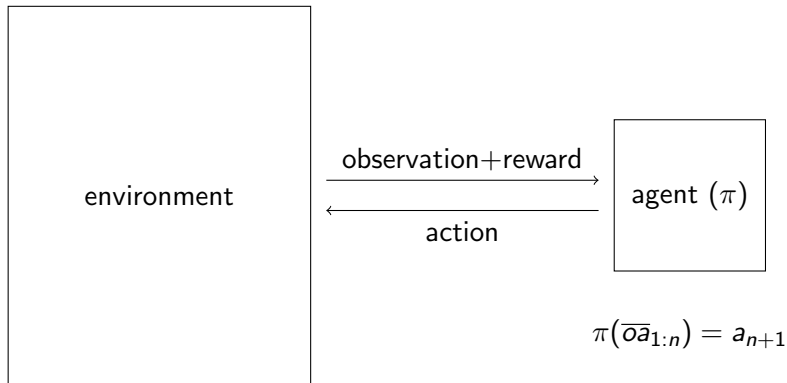
- ▶ Botworld
- ▶ Formalisation in HOL

Reflective Reasoning

The Agent Framework



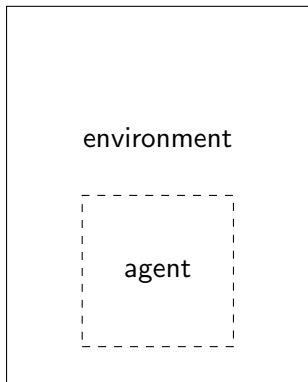
The Agent Framework



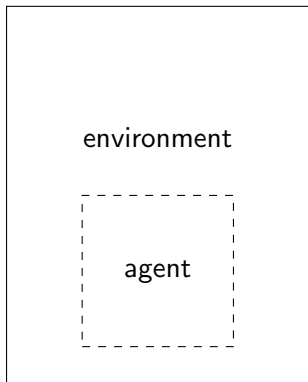
Cartesian boundary

- ▶ agent computed outside environment

Reality is not Cartesian



Reality is not Cartesian



$$\pi_n(o_n) = (a_{n+1}, \lceil \pi_{n+1} \rceil)$$

Vingean Principle

*One can reason only **abstractly** about a stronger reasoner*

Vingean Principle

*One can reason only **abstractly** about a stronger reasoner*

Relevance

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

Vingean Principle

*One can reason only **abstractly** about a stronger reasoner*

Relevance

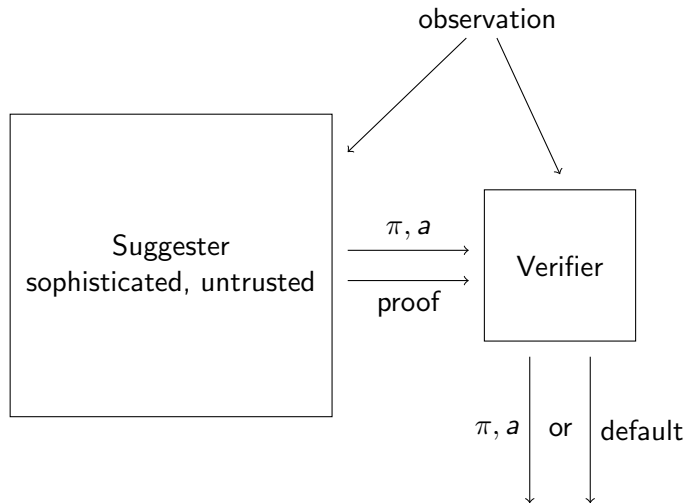
Self-improving system must reason about programs it cannot run:
its successors

Approach

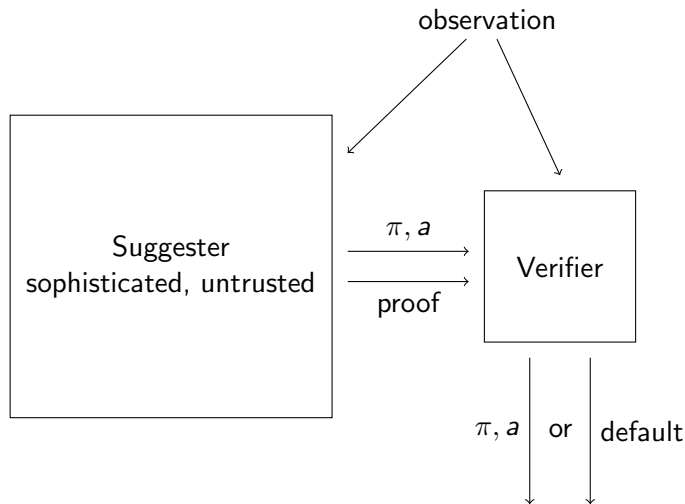
Formal logic as a model of abstract reasoning

Suggester-Verifier Architecture

Suggester-Verifier Architecture

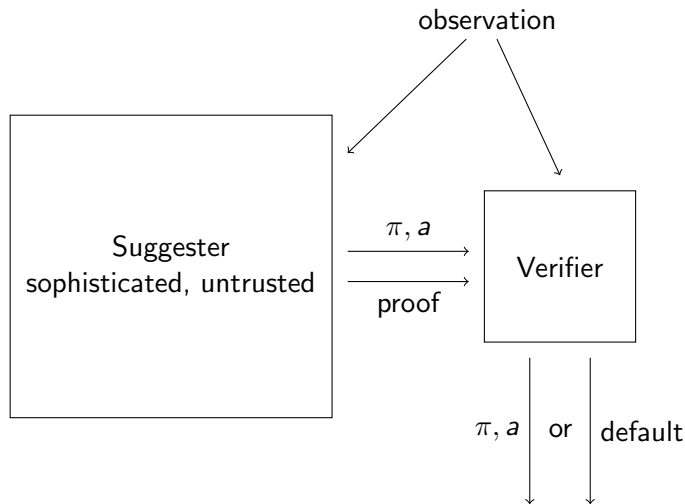


Suggester-Verifier Architecture



Verify: $\vdash u(h(\pi, a)) \geq u(h(\text{default}))$

Suggester-Verifier Architecture



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, $T \vdash \Box_T \ulcorner \varphi \urcorner \implies \varphi$, is inconsistent.
(Gödel/Löb)

Partial Solutions

Descending Trust

$$T_{100} \vdash \Box_{T_{99}} \lceil \varphi \rceil \implies \varphi, T_{99} \vdash \Box_{T_{98}} \lceil \varphi \rceil \implies \varphi,$$

...

Partial Solutions

Descending Trust

$$T_{100} \vdash \Box_{T_{99}} \lceil \varphi \rceil \implies \varphi, T_{99} \vdash \Box_{T_{98}} \lceil \varphi \rceil \implies \varphi,$$

...

Model Polymorphism

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

Model Polymorphism

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

Model Polymorphism

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

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

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

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

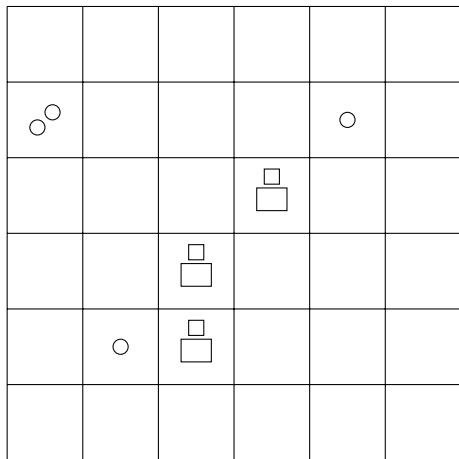
Take $\varphi(n) \equiv n \leq \kappa \implies \text{Safe}(a, n)$

$$\forall a. \Box \lceil \forall n \leq \overline{t+1} + \kappa. \text{Safe}(\bar{a}, n) \rceil \implies$$

$$\forall n \leq t + \kappa. \text{Safe}(a, n)$$

Implementation

Botworld: Concrete Framework for Embedded Agents



Robots can construct/inspect/destroy/program other robots

Botworld Formalisation

Semantics

- ▶ $\text{step} : \text{state} \rightarrow \text{state}$

Botworld Formalisation

Semantics

- ▶ $\text{step} : \text{state} \rightarrow \text{state}$
- ▶ Robots run policies in CakeML

Botworld Formalisation

Semantics

- ▶ $\text{step} : \text{state} \rightarrow \text{state}$
- ▶ Robots run policies in CakeML

Counterfactuals

- ▶ state-with-hole for proposed action

Botworld Formalisation

Semantics

- ▶ $\text{step} : \text{state} \rightarrow \text{state}$
- ▶ Robots run policies in CakeML

Counterfactuals

- ▶ state-with-hole for proposed action
- ▶ $\text{steph} : \text{s-w-h} \rightarrow \text{a} \rightarrow (\text{obs}, \text{state}) \text{ option}$

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 (π, a) or (π', 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 (π, a) or (π', a') accordingly

Reflection Library

Automation for: $\Box \lceil \text{LCA } \bar{k} \implies P \rceil \text{ implies } \text{LCA } (k + 1) \implies P$

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 > 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