Applying Formal Verification to Reflective Reasoning

R. Kumar[1] B. Fallenstein[2]

[1]Data61, CSIRO and UNSW
ramana@intelligence.org

[2]Machine Intelligence Research Institute
benya@intelligence.org

Beneficial Artificial Intelligence, Asilomar 2017

Formal Methods and Artificial Intelligence

What are formal methods?

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal methods for AI?

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal methods for AI?

- Proofs are premature: specifications for AI still unclear

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal methods for AI?

- Proofs are premature: specifications for AI still unclear
- For *highly reliable* systems, we would want a formal argument

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal methods for AI?

- Proofs are premature: specifications for AI still unclear
- For *highly reliable* systems, we would want a formal argument
- AI systems themselves might employ proofs for some tasks

Formal Methods and Artificial Intelligence

What are formal methods?

- *Mathematical models* of software/hardware systems
- Machine-checked *proofs* of theorems
- Wide field: what is proved, fidelity of model, effort required

Formal methods for AI?

- Proofs are premature: specifications for AI still unclear
- For *highly reliable* systems, we would want a formal argument
- AI systems themselves might employ proofs for some tasks

There is one area where formal methods could shed light now

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers
- Can reason only abstractly about a more powerful reasoner

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers
- Can reason only abstractly about a more powerful reasoner

Formal Logic as Model of Abstract Reasoning

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers
- Can reason only abstractly about a more powerful reasoner

Formal Logic as Model of Abstract Reasoning

- Concrete setting for study, and seems to generalise

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers
- Can reason only abstractly about a more powerful reasoner

Formal Logic as Model of Abstract Reasoning

- Concrete setting for study, and seems to generalise
- Gödel/Löb: "formal system that proves its own consistency must be inconsistent"

Formal Methods for Reflective Reasoning

Vingean Reflection

- AI systems may need to rely on other, *more powerful* agents:
  - Self-improving systems: their successors
  - Multi-agent environments: their peers
- Can reason only abstractly about a more powerful reasoner

Formal Logic as Model of Abstract Reasoning

- Concrete setting for study, and seems to generalise
- Gödel/Löb: "formal system that proves its own consistency must be inconsistent"
- Self-improving systems must avoid this *kind* of problem

Our FLI Grant Aims

Based on pen-and-paper work on reflective reasoning principles

Our FLI Grant Aims

Based on pen-and-paper work on reflective reasoning principles

Proposed Project

Implement a model of a reflective reasoning principle, to see:

- whether all the *details* work out, and
- how *hard* it is to do so.

Our FLI Grant Aims

Based on pen-and-paper work on reflective reasoning principles

Proposed Project

Implement a model of a reflective reasoning principle, to see:

- whether all the *details* work out, and
- how *hard* it is to do so.

Eventual Project

Assess how far theorem proving technology is from implementing reflective reasoning, and push it along.

Overview

- Reflective Reasoning: The Problem and Partial Solutions
- Our Progress on the Implementation
- Examples of Difficulties
- Outlook for the Future

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

- Cellular automaton with *embedded* robots
- Robots can construct/inspect/destroy/program other robots

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

- Cellular automaton with *embedded* robots
- Robots can construct/inspect/destroy/program other robots
- Task: Construct a Botworld agent that can self-modify into a *provably safe* agent of the same overall architecture
    - “safe” could mean, e.g., ensure some robot is not destroyed, and can ratchet up a minimum utility requirement

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

- Cellular automaton with *embedded* robots
- Robots can construct/inspect/destroy/program other robots
- Task: Construct a Botworld agent that can self-modify into a *provably safe* agent of the same overall architecture
  - "safe" could mean, e.g., ensure some robot is not destroyed, and can ratchet up a minimum utility requirement

Suggester-Verifier Architecture

Agent with two sub-programs:

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

- Cellular automaton with *embedded* robots
- Robots can construct/inspect/destroy/program other robots
- Task: Construct a Botworld agent that can self-modify into a *provably safe* agent of the same overall architecture
  - "safe" could mean, e.g., ensure some robot is not destroyed, and can ratchet up a minimum utility requirement

Suggester-Verifier Architecture

Agent with two sub-programs:

- Suggester: Sophisticated, untrusted code to compute agent's command plus a *proof* that it is no worse than a default

Reflective Reasoning Example Setup

Botworld: Environment for Studying Naturalistic Agents

- Cellular automaton with *embedded* robots
- Robots can construct/inspect/destroy/program other robots
- Task: Construct a Botworld agent that can self-modify into a *provably safe* agent of the same overall architecture
  - "safe" could mean, e.g., ensure some robot is not destroyed, and can ratchet up a minimum utility requirement

Suggester-Verifier Architecture

Agent with two sub-programs:

- Suggester: Sophisticated, untrusted code to compute agent's command plus a *proof* that it is no worse than a default
- Verifier: Simple, trustworthy code to *check* the suggester's proof, and output the suggested command or default

Problem and Approach

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, $T \vdash \Box_T \ulcorner \varphi \urcorner \implies \varphi$, violates Gödel/Löb

Problem and Approach

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, $T \vdash \Box_T \ulcorner \varphi \urcorner \implies \varphi$, violates Gödel/Löb

Partial Solutions

- Descending Trust: $T_{100} \vdash \Box_{T_{99}} \ulcorner \varphi \urcorner \implies \varphi$, $T_{99} \vdash \Box_{T_{98}} \ulcorner \varphi \urcorner \implies \varphi, \ldots$

Problem and Approach

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, $T \vdash \Box_T \ulcorner \varphi \urcorner \implies \varphi$, violates Gödel/Löb

Partial Solutions

- Descending Trust: $T_{100} \vdash \Box_{T_{99}} \ulcorner \varphi \urcorner \implies \varphi$, $T_{99} \vdash \Box_{T_{98}} \ulcorner \varphi \urcorner \implies \varphi, \ldots$
- Model Polymorphism: $T_{\kappa+1} \vdash \forall n. \Box_{T_\kappa} \ulcorner \varphi(\bar{n}) \urcorner \implies \varphi(n)$

Progress

Prerequisite Technology

- Programming Language (CakeML), formal specification, verified implementation
- Proof-producing translation from logic to CakeML
- Self-Verifying Theorem Prover (Candle) (work-in-progress)
- Proof-producing translation from (meta) logic to Candle

Specific to this Implementation

- Model-Polymorphism Library (work in progress)
- Botworld Formalisation
- Suggester-Verifier Design
- Partial Proof of Suggester-Verifier Correctness

Results

- Code on GitHub (machine-intelligence/Botworld.HOL)
- Upcoming presentation at AITP'17
- Draft report online

Difficulties 1

Reflective Programming

- suggester-verifier(sug,obs,def):
  1. run sug(obs,def), obtain (com,prf)
  2. if verify(obs,def,com,prf) then com
  3. else def
- Currently, step 1 is by splicing the suggester program into the suggester-verifier program

Difficulties 1

Reflective Programming

- suggester-verifier(sug,obs,def):
  1. run sug(obs,def), obtain (com,prf)
  2. if verify(obs,def,com,prf) then com
  3. else def
- Currently, step 1 is by splicing the suggester program into the suggester-verifier program
- Alternative: call an `eval` primitive
- Formal semantics, and verified implementation, for dynamic evaluation is *ongoing research*

Difficulties 2

Scaling Reflection Up

- Suggester's proof must include many definitions:
  - An internal copy of Botworld
  - Utility function on Botworld games
  - Machinery for model polymorphism

Difficulties 2

Scaling Reflection Up

- Suggester's proof must include many definitions:
  - An internal copy of Botworld
  - Utility function on Botworld games
  - Machinery for model polymorphism
- Reflection library (ITP'15): superlinear time in no. definitions

Difficulties 2

Scaling Reflection Up

- Suggester's proof must include many definitions:
  - An internal copy of Botworld
  - Utility function on Botworld games
  - Machinery for model polymorphism
- Reflection library (ITP'15): superlinear time in no. definitions
- All made in internal copy of logic used by Candle

Difficulties 2

Scaling Reflection Up

- Suggester's proof must include many definitions:
  - An internal copy of Botworld
  - Utility function on Botworld games
  - Machinery for model polymorphism
- Reflection library (ITP'15): superlinear time in no. definitions
- All made in internal copy of logic used by Candle

Partial Progress

- Alternative reflection library which axiomatises as many definitions as possible

Difficulties 2

Scaling Reflection Up

- Suggester's proof must include many definitions:
  - An internal copy of Botworld
  - Utility function on Botworld games
  - Machinery for model polymorphism
- Reflection library (ITP'15): superlinear time in no. definitions
- All made in internal copy of logic used by Candle

Partial Progress

- Alternative reflection library which axiomatises as many definitions as possible
- Automated machinery for quoting to bridge the various levels

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!

Formal Methods 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!

Formal Methods 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!

Formal Methods 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 (analogy: security of seL4 via architectural argument, becomes amenable to verification)