Logik Cafe, Vienna 23 May 2016

Mechanized Analysis of Reconstructions of Anselm's Ontological Argument

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Overview

- Why am I here?
  - Computer Scientists confront philosophical problems
    - ⋆ Could use your help
  - We have tools and perspectives that may be useful to you
    - ⋆ Hope to gain your interest

  Focus here on the latter
- Verification Systems: powerful theorem provers
- Example application: Anselm's Ontological Argument
  - Specifically, the reconstruction of Eder and Ramharter
  - Also Oppenheimer and Zalta, Campbell (and Gödel)

  Can this add value?
- Back to the start: opportunities for collaboration?

Verification Systems

- General purpose systems developed over the last 30 years
  - For reasoning about correctness of computational systems
    - ⋆ Algorithms, protocols, software, hardware
    - ⋆ HMI, requirements, biological systems
  - Integrate a specification language
    - ⋆ Essentially a rich logic, invariably higher-order
  - And mechanized deduction
    - ⋆ Combine interactive and automated theorem proving
    - ⋆ Decision procedures, SAT and SMT solvers, model checkers
  - Plus stuff for managing large formal developments
    - ⋆ Often tens of thousands of lines
- Recent focus is more specialization, more automation

Popular Verification Systems

- Unquantified First Order
  - ACL2 (USA)

- Higher Order
  - Coq (France)
  - HOL (UK)
  - Isabelle (Germany)
  - PVS (USA)
    - ⋆ This is what I will use, first released 1993
    - ⋆ Classical Higher-Order Logic with predicate subtypes
    - ⋆ Winner of CAV Award 2015, 3,000 citations

Compared with Simple First Order Provers

- Verification systems tackle the whole problem
- Must be able to specify anything
  - Without going outside the system
- Want guarantees of soundness (conservative extension)
- And ways of demonstrating consistency of axiomatizations
  - Theory interpretations
- And ways to explore intuition (e.g., testing)
- Want modularity (theories and parameterization)
- And ways to manage and ensure consistency of large developments
- Want automation for common CS theories
  - Equality, arithmetic, bitvectors, arrays etc.
- Etc.

Anselm's Ontological Argument

- Formulated by St. Anselm (1033–1109)
  - Archbishop of Canterbury
  - Aimed to justify Christian doctrine through reason
  - Cf. Avicenna's earlier proof of The Necessary Existent
- Disputed by his contemporary Gaunilo
  - Existence of a perfect island
- Widely studied and disputed thereafter
  - Descartes, Leibniz, Hume, Kant (who named it), Gödel
- Russell, on his way to the tobacconist: "Great God in Boots!—the ontological argument is sound!"
- The later Russell: "The argument does not, to a modern mind, seem very convincing, but it is easier to feel convinced that it must be fallacious than it is to find out precisely where the fallacy lies"

Analyses of Anselm's Ontological Argument

- Reconstructions
  - What did Anselm actually say?
  - Can we accurately formulate that in modern logic?
- Analysis
  - Is the argument sound?
  - If not, where is the flaw, and can it be repaired?
  - Other questions and lines of inquiry
- For computer scientists (a reason for my interest)
  - An assurance case is an argument that aims to justify a claim (typically about safety) on the basis of evidence (premises) about the system
  - The Ontological Argument is a good illustration of how this differs from proof
  - I became aware of it through Susanne Riehemann, who worked in our lab and is married to Ed Zalta

Anselm's Argument

- Appears in his Proslogion
  - With variations and developments
  - Written in Latin
- So scholars debate exact interpretation
- Here's a fairly neutral modern translation
  - We can conceive of something/that than which there is no greater
  - If that thing does not exist in reality, then we can conceive of a greater thing—namely, something (just like it) that does exist in reality
  - Therefore either the greatest thing exists in reality or it is not the greatest thing
  - Therefore the greatest thing (necessarily) exists in reality
- That's God

Günter Eder and Esther Ramharter's Reconstruction

- Appears in Synthese vol. 192, October 2015
- Three stages: first-order, higher-order, modal logic
- I will cover just the first two
  - Leave the third to Benzmüller and Woltzenlogel-Paleo
- My goal is to show that it is quite easy to represent and mechanize this in a verification system
- I will not comment (much) on E&R's reconstruction
  - That's a task for philosophers
  - But I hope to show that mechanized support could aid the discussion

First-Order: Understandable Objects, Gods

- Something is a God if there is nothing greater

  Def C-God: $Gx :\leftrightarrow \neg\exists y(y > x)$

  Here, x and y range over the "understandable objects," which is the implicit range of first-order quantification

- PVS is higher-order, so we need to be explicit about types

PVS fragment

```
U_beings: TYPE

x, y: VAR U_beings

>(x, y): bool

God?(x): bool = NOT EXISTS y: y > x
```

The `VAR` declaration saves us having to specify each appearance; overloaded infix operators like `>` use prefix form in declarations; the `?` in `God?` is just a naming convention for predicates; the `=` indicates this is a definition

First-Order: Conceive Of, Real Existence

- The Argument says we can conceive of something than which there is no greater (i.e., a God); interpret this as a premise
- **ExUnd:** $\exists x Gx$
- In PVS we render it as follows.

PVS fragment

```
ExUnd: AXIOM EXISTS x: God?(x)
```

- Real existence is not the $\exists$ of logic, but a predicate
  - E&R write it as $E!$, I use `re?`
- Our goal is to prove that a God exists in reality
- **God!:** $\exists x(Gx \wedge E!x)$
- We write this in PVS as follows

PVS fragment

```
re?(x): bool


God_re: THEOREM EXISTS x: God?(x) AND re?(x)
```

First-Order: Additional Premises

- Cannot prove this without additional premise to connect >, $E!$
- Note, nothing so far says > is an ordering relation
- First attempt

  Greater 1: $\forall x(\neg E!x \rightarrow \exists y(y > x))$

  If `x` does not exists in reality, then there is a greater thing
- In PVS, we write this as follows.

PVS fragment

```
Greater1: AXIOM FORALL x: (NOT re?(x) => EXISTS y: y > x)
```

First-Order: Complete PVS Specification

```
ontological_arg: THEORY
BEGIN

  U_beings: TYPE

  x, y: VAR U_beings

  >(x, y): bool

  God?(x): bool = NOT EXISTS y: y > x

  re?(x): bool

  ExUnd: AXIOM EXISTS x: God?(x)

  Greater1: AXIOM FORALL x: (NOT re?(x) => EXISTS y: y > x)

  God_re: THEOREM EXISTS x: God?(x) AND re?(x)

END ontological_arg
```

First-Order: PVS Proof

- PVS can prove the theorem given the following commands

`(lemma "ExUnd")(lemma "Greater1")(grind :polarity? t)`	PVS proof

- First two instruct PVS to use named formulas as premises
- Third instructs it to use general-purpose proof strategy, observing the polarity (i.e., positive vs. negative occurrences) of terms when searching for quantifier instantiations
- PVS reports that the theorem is proved

First-Order: Proofchain Analysis

- Proof is a local concept
- Proofchain analysis checks that all proofs are complete, and also those of any lemmas they cite, plus any incidental proof obligations
- It provides the following report

PVS proofchain

```
ontological_arg.God_re has been PROVED.

   The proof chain for God_re is COMPLETE.

   God_re depends on the following axioms:
    ontological_arg.ExUnd
    ontological_arg.Greater1

   God_re depends on the following definitions:
    ontological_arg.God?
```

First-Order: Second Attempt

- E&R observe Greater 1 is not a faithful reconstruction
  - Not analytic: no a priori reason to believe it
  - Argument does not follow Anselm's structure
- Eder and Ramharter next propose the following premises

  Greater 2: $\forall x \forall y(E!x \wedge \neg E!y \rightarrow x > y)$, and

  E!: $\exists x E!x$

  An object that exists in reality is > than one that does not, and there is some object that does exist in reality.
- In PVS, these are written as follows and replace `Greater1`

PVS fragment

```
Greater2: AXIOM FORALL x, y: (re?(x) AND NOT re?(y)) => x > y

Ex_re: AXIOM EXISTS x: re?(x)
```

First-Order: Second Attempt (ctd. 1)

- Same PVS proof strategy as before proves the theorem
- E&R consider this version unfaithful also
- Hence the higher-order treatment
- Higher-order:
  - Functions can take functions as arguments
  - And return them as values
  - Can quantify over functions
  - Need types to keep things consistent
  - Predicates are just functions with range type Boolean

Higher-Order

- Anselm attributes properties to objects and some of these, notably $E!$, contribute to evaluation of >
- Hypothesize some class $\mathcal{P}$ of "greater-making" properties on objects; define one object to be greater than another exactly when it has all the properties of the second, and more besides

  Greater 3: $x > y :\leftrightarrow \forall_{\mathcal{P}} F(Fy \rightarrow Fx) \wedge \exists_{\mathcal{P}} F(Fx \wedge \neg Fy)$,

  where $\forall_{\mathcal{P}} F$ indicates that the quantified higher-order variable F ranges over the properties in $\mathcal{P}$, and likewise for $\exists_{\mathcal{P}} F$
- In PVS we do this using predicate subtypes

PVS fragment

```
P: setof[ pred[U_beings] ]

re?: pred[U_beings]

F: VAR (P)

>(x, y): bool =
  (FORALL F: F(y) => F(x)) AND (EXISTS F: F(x) AND NOT F(y))
```

Continued. . .

Higher-Order (ctd.)

- In PVS we do this using predicate subtypes

PVS fragment

```
P: setof[ pred[U_beings] ]

re?: pred[U_beings]

F: VAR (P)

>(x, y): bool =
  (FORALL F: F(y) => F(x)) AND (EXISTS F: F(x) AND NOT F(y))
```

- We let P be some set of predicates over U_beings
- Previously, we specified re? by re?(x): bool, but here we specify it to be a constant of type pred[U_beings]
- These are syntactic variants for the same type; we use the latter form here for symmetry with the specification of P, so that is clear that re? is potentially a member of P
- P is a set, equivalent to a predicate in HO logic; in PVS, predicate in parentheses denotes corr. predicate subtype
- So F is a variable ranging over the members of P

Higher-Order: Realization

- Anselm starts with something than which there is no greater
- If that something does not exist in reality, consider same thing augmented with the property of existence in reality
- Problem is, that may not be an understandable object
- E&R use additional premise realization to ensure that it is
- **Realization:** $\forall_{\mathcal{P}}\mathcal{F}\exists x\forall_{\mathcal{P}}F(\mathcal{F}(F)\leftrightarrow Fx)$

  This says that for any set $\mathcal{F}$ of properties in $\mathcal{P}$, there is some understandable object x that has exactly the properties in $\mathcal{F}$
- Eder and Ramharter use the locution $\forall_{\mathcal{P}}\mathcal{F}$ to indicate a third-order quantifier over all sets of properties in $\mathcal{P}$
- In PVS, we make the types explicit and the corresponding specification is as follows.

PVS fragment

```
Realization: AXIOM
  FORALL (FF: setof[(P)]): EXISTS x: FORALL F: FF(F) = F(x)
```

Higher-Order Formulation in PVS

```
HO_ontological_arg: THEORY
BEGIN

   U_beings: TYPE

   x, y: VAR U_beings

   re?: pred[U_beings]

   P: set[ pred[U_beings] ]

   F: VAR (P)

   >(x, y): bool =
      (FORALL F: F(y) => F(x)) & (EXISTS F: F(x) AND NOT F(y))

   God?(x): bool = NOT EXISTS y: y > x

   ExUnd: AXIOM EXISTS x: God?(x)

   Realization: AXIOM
    FORALL (FF:setof[(P)]): EXISTS x: FORALL F: FF(F) = F(x)

   God_re: THEOREM member(re?, P) => EXISTS x: God?(x) AND re?(x)

 END HO_ontological_arg
```

Higher-Order Proof in PVS (ugh)

```
(ground)
(expand "member")
(lemma "ExUnd")
(skosimp)
 (case "re?(x!1)")
(("1" (grind))
  ("2"
  (lemma "Realization")
  (inst - "{ G: (P) | G(x!1) OR G=re? }")
   (skosimp)
  (inst + "x!2")
  (ground)
  (("1"
    (expand "God?")
    (inst + "x!2")
    (expand ">")
    (ground)
    (("1" (lazy-grind)) ("2" (grind))))
   ("2" (grind)))))
```

Higher-Order: Quasi-id

- The heart of Anselm's Argument
  - If ExUnd does not exist in reality
  - Then compare it with the itself, conceived as existing

  A reconstruction must preserve this
- Eder and Ramharter define two objects to be quasi-identical, written $\equiv_{\mathcal{D}}$, if they have the same greater-making properties apart from those in some subset $\mathcal{D} \subseteq \mathcal{P}$:

  Quasi-id: $x \equiv_{\mathcal{D}} y :\leftrightarrow \forall_{\mathcal{P}} F(\neg\mathcal{D}(F) \rightarrow (Fx \leftrightarrow Fy))$
- Eder and Ramharter prove that the Skolem constants a (from **Realization**) and g (from **ExUnd**) appearing in their formalization of the argument are quasi-identical: $a \equiv_{\{E!\}} g$
- In PVS, we define quasi-identity as follows

PVS fragment

```
quasi_id(x, y: U_beings, D: setof[(P)]): bool =
   FORALL (F: (P)): NOT D(F) => F(x) = F(y)
```

And reproduce the same proof

Interim Conclusions

- I hope you agree: this was straightforward
- But does it add value?
- Eder and Ramharter made no errors!
- But I think there are opportunities beyond bug-finding
- Let's look at some related examples

Oppenheimer and Zalta's Treatments

- I previously mechanized a version of O&Z's reconstruction
- It is identical to the first-order version of E&R with Greater 2
- But that may not be obvious due to different types and representations
- O&Z version

PVS fragment

```
greatest: setof[U_beings] =
     { x | NOT EXISTS y: y > x }

P1: AXIOM nonempty?(greatest)
```

- E&R version

PVS fragment

```
God?(x): bool = NOT EXISTS y: y > x

ExUnd: AXIOM EXISTS x: God?(x)
```

- Sets and predicates are the same in higher-order logic, and the set comprehension notation in PVS is equivalent to lambda-abstraction, so we can conjecture equivalence. . .

Comparison of O&Z and E&R Reconstructions

- Equivalence

PVS fragment

```
gr_God: CONJECTURE greatest = God?
ne_Ex: CONJECTURE nonempty?(greatest) IFF EXISTS x: God?(x)
```

These are proved, respectively, by

PVS proof

```
(apply-extensionality) (grind :polarity? t)
```

and

PVS proof

```
(grind :polarity? t)
```

- So one potential value is in comparing different reconstructions
- And verification is stronger than eyeballing

Circularity of Greater 1

- The first attempt (with Greater 1) is also in O&Z
- PVS shows it to be directly circular: Greater 1 can be proved from the conclusion and vice-versa
- I.e., the formulation begs the question
- Not so here, because O&Z use a definite description and need an additional premise (trichotomy of >) to establish uniqueness of God?
- However, it is surely plausible to suppose that something than which there is no greater is also greater than everything else (i.e., it cannot be unrelated)
- And that is enough for circularity
- I think these kinds of exploration are another potential value in mechanization

Unintended Models

- The version with Greater 2 uses two axioms (three in O&Z's version) and these could introduce inconsistency
- PVS guarantees conservative extension for purely constructive specifications
- So one way to establish consistency of axioms is to exhibit a constructively defined model
- Can do this using PVS capabilities for theory interpretations
  - Interpret `beings` by the natural numbers `nat`
  - And `>` by `<` (so `the(greatest)` is 0)
  - And `really_exists` by "less than 4"
- PVS generates TCCs (proof obligations) to prove that the axioms of the source theory are theorems under the interpretation

The Model

model

```
interpretation: THEORY
BEGIN

IMPORTING ontological {{
  beings := nat,
  > := <,
  really_exists := LAMBDA (x: nat): x<4
}} AS model

END interpretation
```

Proof Obligations for Consistency

TCCs

```
% Mapped-axiom TCC generated (at line 56, column 10) for
    % ontological
    %      beings := nat,
    %        > := restrict[[real, real], [nat, nat], boolean](<),
    %        really_exists := LAMBDA (x: nat): x < 4

model_P1_TCC1: OBLIGATION nonempty?[nat](greatest);

% Mapped-axiom TCC generated (at line 56, column 10) for
    % ontological
    %      beings := nat,
    %        > := restrict[[real, real], [nat, nat], boolean](<),
    %        really_exists := LAMBDA (x: nat): x < 4

model_someone_TCC1: OBLIGATION EXISTS (x: nat): x < 4;

...continued
```

Proof Obligations for Consistency (ctd.)

TCCs

```
...continuation

% Mapped-axiom TCC generated (at line 56, column 10) for
    % ontological
    %      beings := nat,
    %        > := restrict[[real, real], [nat, nat], boolean](<),
    %        really_exists := LAMBDA (x: nat): x < 4

model_reality_trumps_TCC1: OBLIGATION
  FORALL (x, y: nat): (x < 4 AND NOT y < 4) => x < y;
```

- These are all easily proved
- So, our formalization of the Ontological Argument is sound
- And the conclusion is valid
- But it does not compel a theological interpretation

Why Verification Systems and not Simple Provers?

- O&Z formalized a version of the Argument that employs a definite description
  - Used a Free Logic to deal with definitional concerns
- Then mechanized it with Prover9 first-order theorem prover
  - No first-order theorem prover automates Free Logic
  - Nor provides definite descriptions

  So these delicate issues are dealt with informally outside the system, and beyond the reach of automated checking
- Deductions performed by Prover9 actually used very little of their formalization
- This led them a much reduced formalization that Prover9 still found adequate

Oppenheimer and Zalta's Simplification (ctd.)

- Believed they had discovered a simplification to the Argument that

  > "not only brings out the beauty of the logic inherent in the argument, but also clearly shows how it constitutes an early example of a 'diagonal argument' used to establish a positive conclusion rather than a paradox"

- Garbacz refutes this
  - The simplifications flow from introduction of a constant (God) that is defined by a definite description
  - In the absence of definedness checks, this asserts existence of the definite description and bypasses the premises otherwise needed to establish that fact
- Lesson: mechanization needs to deal with the whole problem
- See PVS treatment of O&Z's version for sound mechanization of definite descriptions

Sophisticated Types: More on Quasi-Id

- There is a lot "packed into" these definitions
- E.g., can prove all Gods have all greater-making properties

PVS fragment

```
God_all: THEOREM FORALL (a: (P)): God?(x) => a(x)
```

- And all Gods are quasi-identical

PVS fragment

```
all_God: THEOREM God?(x) AND God?(y)
   => quasi_id(x, y, emptyset)
```

- Günter Eder observes it is not intended to use emptyset here
- We can enforce that

PVS fragment

```
gr_props(D: setof[(P)]): bool = member(re?, D)

strong_quasi_id(x,y: U_beings, D: (gr_props)): bool =
  FORALL (F:(P)): NOT D(F) => F(x) = F(y)

strong_all_God: THEOREM God?(x) AND God?(y)
    => strong_quasi_id(x, y, emptyset)
```

Now strong_all_God does not typecheck

Sophisticated Types: More on Quasi-Id (ctd.)

- Now strong_all_God does not typecheck

TCC

```
 % Subtype TCC generated (at line 60, column 69) for  emptyset
    % expected type  (gr_props)
  % unfinished
strong_all_God_TCC1: OBLIGATION
  FORALL (x, y: U_beings): God?(y) AND God?(x)
      IMPLIES gr_props(emptyset[(P)]);
```

- Predicate subtypes allow much of the specification to be embedded in the types
- Keeps the formulas uncluttered
- Typechecking generates proof obligations
- Allows richly expressive specification

Possible Concrete Opportunity

Richard Campbell:

- Eder & Ramharter's paper has proved invaluable. Their two definitions Quasi-Id and Greater 3 have enabled me to develop a formalization, mostly in first-order logic, which I believe accurately represents Anselm's reasoning

  They have made it possible, for the first time, to achieve that without having to invoke implicit premises or background assumptions to deduce, from what Anselm actually asserts as premises, the conclusions he actually draws (apart from one obvious lacuna in his argumentation)

- But he criticizes realization (says it is false, actually)
- Has a treatment that uses modal operators for the Fool's introspection ("possibly thought that") that avoids this
- Some difficult issues, would be cool to check it mechanically

Modal Reconstructions

- Anselm goes on to establish the necessity of God's existence
- And his perfection, etc.
- Seems natural to employ a modal logic for necessity
- CS employs temporal logics (interpreted over sequences)
- But combinations of general modal and first- or higher-order logics are difficult
- Gödel left a modal version of the Argument in his nachlass
- Christoph Benzmüller and Bruno Woltzenlogel-Paleo have mechanized this (using Coq and Isabelle) and detected and fixed an inconsistency
  - "...their work has received a media repercussion on a global scale"
- They first embed higher-order modal logic in Isabelle
- Then represent Scott's version of Gödel's proof in that
- They explore consistency, modal collapse, make strong claims

Interim Conclusions (redux)

- Uniform, comprehensive notation facilitates comparisons
  - Eliminates need for idiosyncratic constructions
- Mechanization further facilitates this
- Recall equivalence of Greater 2 and O&Z's treatment
- When automated and fast, mechanization enables exploration of variants, conjectures, etc.
  - By hand, you can do one or two of these
  - But hard to maintain discipline for many of them
  - Mechanization brings same skepticism to 100th as to first
- Example: version with Greater 1 is circular under plausible additional premise
- Can venture reliably into difficult areas
  (e.g., quantified modal logics)
- It's fun! Students might enjoy it

Opportunities: Collaboration Between CS and Philosophy

- CS has logical tools, and the theories to support them
  - That may be useful in philosophy
  - E.g., I speculate that we could demonstrate equivalence, or sharpen differences, in various formulations of the Ontological Argument
  - Are the any other a priori arguments? Proclus? Avicenna?
- Computer Science has problems of a philosophical nature
  - E.g., an assurance case is an argument that aims to justify a claim (typically about safety) on the basis of evidence (premises) about a system
    - ⋆ Not a proof: there are uncertainties, unknowns
    - ⋆ So we encounter topics in epistemology: Bayesian epistemology, confirmation theory
    - ⋆ And dialectics: resolving contested arguments
    - ⋆ Elsewhere: emergence
  - Would benefit from dialog with philosophers

Collaboration Between CS and Philosophy (ctd.)

- Fitelson, Zalta et al propose computational metaphysics
  - Code problems up in logic, let the automation rip
  - Examine the detritus for insight
- I have a more radical proposal: computationally-informed philosophy: warning Crazy Idea ahead
  - Some philosophical topics might benefit from a computational (i.e., strong AI) perspective
    - ⋆ E.g., free will, consciousness, ethics
    - ⋆ Postulate a robot, not a human
    - ⋆ But this requires suitable interpretations for duality of computation and humanity
      - ◇ i.e., not the Chinese Room
  - Will need combination of CS and philosophical insight

Thank You

Papers:

- Eder and Ramharter's reconstructions:
  http://www.csl.sri.com/users/rushby/abstracts/er-ontarg16
  This was written for a general audience

- Oppenheimer and Zalta's reconstruction:
  http://www.csl.sri.com/users/rushby/abstracts/fwfm13
  This was written for a Computer Science audience