Mechanized Analysis Of a Formalization of Anselm's Ontological Argument by Eder and Ramharter[*]

John Rushby
Computer Science Laboratory
SRI International, Menlo Park CA USA

February 11, 2016

Abstract

Eder and Ramharter [7] propose requirements to be satisfied by formal reconstructions of informal arguments and illustrate these with their own reconstructions of Anselm's Ontological Argument: one in classical (higher-order) logic, and one in modal logic. I reproduce and mechanically check their classical reconstruction in the PVS verification system and present this as an illustration of the ease and benefit of mechanized analysis in this domain. I hope this may stimulate philosophers and theologians to investigate these tools from computer science, and to explore their application to topics of philosophical interest.

1 Introduction

The Ontological Argument is an 11h-century proof of the existence of God presented by St. Anselm of Canterbury in his *Proslogion* of 1078. It has been studied and debated by philosophers and theologians ever since. A modern rendition of the Argument is shown in Figure 1.

I have previously presented a mechanized analysis [17] using the PVS Verification System [14, 16] of a formalization of the Ontological Argument in modern classical logic due to Oppenheimer and Zalta [12, 13]. Separately, Benzmüller and Woltzenlogel Paleo presented a mechanized analysis [2, 3] using the Coq [5] and Isabelle-HOL [9] verification systems of a formalization in modal logic due to Gödel and Scott. And Ebnenasir and Tahat are developing a mechanized analysis [6] in PVS of Avicenna's proof of the "Necessary Existent." This proof is 150 years older than Anselm's argument and possibly one of its inspirations.

[*]This research was supported by SRI International.

1. We can conceive of something than which there is no greater
2. If that thing exists only in the mind and not in reality, then we can conceive of something greater—namely, a similar thing that does exist in reality
3. Therefore either the greatest thing exists in reality or it is not the greatest thing
4. Therefore the greatest thing necessarily exists in reality
5. That's God!

Figure 1: Modern Rendition of Anselm's Ontological Argument

Recently, Eder and Ramharter have published an interesting discussion on the requirements that should be satisfied by formal reconstructions of Anselm's argument, and they present reconstructions of their own using both classical and modal logic [7]. The immediate purpose of this paper is to present a mechanized analysis in PVS of Eder and Ramharter's classical reconstructions. My larger purpose (and, I think, that of the authors of the other mechanized verifications mentioned above) is to demonstrate the practicality and, I believe, the utility of applying mechanized verification systems to certain topics in philosophy, in the hope that philosophers and theologians will become interested in exploring the application of these tools to their problems.

Verification Systems are computer-based tools that combine a *specification language* (essentially a logic) with automated and user-directed theorem proving. They differ from the first-order theorem provers that philosophers may have experimented with, or been exposed to in classes on logic and AI, by the richness of their logics and the range and power of their automation. Most of these systems use higher-order logic, often extended with dependent typing and (e.g., in PVS) with predicate subtyping, and with means for defining recursive datatypes such as lists and trees (and *inter alia* thereby, the ordinals up to ε_0). They provide constructions for modularization and parameterization that are essential for specifications that may be thousands or tens of thousand of lines long, and for testing (e.g., by animation) the "fidelity" of such large specifications to the intent of their authors [11], and for ensuring their soundness (e.g., by guarantees of conservative extension for definitional constructions and by theory interpretation for axiomatic constructions). The theorem provers of verification systems generally combine high-performance decision procedures for important theories such as equality with uninterpreted functions, linear integer and real arithmetic, bitvectors, and arrays or functions of these, together with propositional and quantifier reasoning. Although the automation is often able to discharge the "endgame" of proofs containing hundreds of thousands of terms and variables in seconds, it generally operates as a "proof assistant" under human direction. These are mature and widely used systems: PVS, for example, was first released more than 20 years ago and has over 3,000 citations.

The value of verification systems is that they bring the power of mechanized calculation to the realm of logic, allowing rapid and reliable examination of large or complex formalized descriptions. These systems were primarily developed for the analysis of requirements, specifications, designs, and algorithms for computational systems and their implementations in hardware and software. However, they also are applied to some branches of mathematics and logic, especially those that provide "domain knowledge" for critical systems (e.g., 3-D geometry and dynamics for aircraft collision avoidance systems [10]). I hope they will find application also to suitable areas of philosophy, where I believe the benefit will derive from the precision and uniformity provided by their specification languages and the opportunity for experimentation and comparison afforded by the efficiency and security of their mechanized analysis.

This paper is intended for a general audience including philosophers and theologians; however, since I am neither of the latter, it may miss its target and I welcome feedback and suggestions. My previous paper on verification of an ontological argument [17] was written for computer scientists and assumes background knowledge and a perspective or world view that may not be shared by those from other fields. However, although the present paper is intended to be self-contained, it may be useful to have some familiarity with, or access to, the previous paper and also that of Eder and Ramharter [7]. The paper is organized as follows: Section 2 examines Eder and Ramharter's first-order reconstructions, Section 3 examines their higher-order reconstruction, and Section 4 concludes.

2 Eder and Ramharter's Discussion and First-Order Reconstruction

The first line of the presentation of the Ontological Argument in Figure 1 speaks of "*something* than which there is no greater." An alternative reading speaks of "*that* than which there is no greater" and this is the form espoused by Oppenheimer and Zalta [12] and mechanically analyzed in my earlier paper [17]. These prior works interpret *that* as a definite description, meaning assertion of the existence of a single, *unique* thing "than which there is no greater." In all forms of the Ontological Argument, it is necessary to assume the existence of *some* such thing (that is the import of "we can conceive of...") but formalizations that employ a definite description must make additional assumptions to establish its uniqueness: for example, that the relation *greater* (written $>$) is trichotomous.[1]

Eder and Ramharter are concerned with formal *reconstructions* (as opposed to improvements or emendations) of informal arguments and present seven requirements that reconstructions should satisfy. Among these are *local* and *global conformity* with what the original author actually said (locally and elsewhere). They argue that the definite description is not locally conformant with Anselm's text. For evidence, they cite Anselm's original (Latin) text from Chapter II of the *Proslogion* and provide a translation close to that of Line 1 in Figure 1, where the key word is *something* ("than which there is no greater"). Elsewhere in Chapter II, Anselm does say *that* ("than which there is no greater"), but Eder and Ramharter interpret it as a reference back to the *something.* Further evidence is that Anselm first mentions God's uniqueness in his Chapter III, so it is unlikely that he intended to introduce it silently in Chapter II; thus, global conformity, too, argues for *something* rather than *that.* We therefore follow Eder and Ramharter and use *something* ("than which there is no greater") in the mechanized analysis presented here.

We are now ready to specify these concepts in the language of the PVS verification system. We begin with fragments that are not, on their own, valid PVS. We will later combine them into a valid specification. Eder and Ramharter begin their reconstruction in first-order logic (they later shift to higher-order logic) with the following definition of a predicate G for "being a God":

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

That is, Gx (x is a God) exactly when there is nothing greater.

In PVS we can write this as follows; the `=` sign indicates that this is a definition and the range type `bool` (an abbreviation for Boolean) establishes that the function `God?` is a predicate. It is a convention that predicate names end with a question mark.

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

PVS can render its specifications in mathematical notation, but we will use the plain ascii form here.

PVS is a higher-order logic (meaning that quantification can extend over functions as well as individuals, and functions can take functions as arguments and return them as values). Higher-order logics require a typing discipline to ensure consistency, so we need to introduce suitable types for the variables `x` and `y` and the relation (i.e., predicate over two arguments) `>`. Now, the Ontological Argument is concerned with objects that exist "*in the understanding.*" Anselm's text speaks of this understanding being in the mind of a hypothesized individual (referred to

[1]That is, for all x, y, either $x > y$, or $y > x$, or $x = y$ (not necessarily exclusively).

as "The Fool"), but Eder and Ramharter argue that a more neutral interpretation, namely that of *being understandable*, is conformant. They observe that they could introduce a predicate U to capture this notion, but instead choose the "*understandable objects*" as the (implicit) domain of (first-order) quantification. In PVS, we will explicitly introduce an uninterpreted type of `U_beings` (for "*understandable beings*") and declare `x` and `y` to be of this type.[2] The relation `>` is simply an uninterpreted relation on (i.e., predicate on pairs of) `U_beings`.

PVS fragment
```
U_beings: TYPE

x, y: VAR U_beings

>(x, y): bool
```

Although the prefix form must be used when introducing the signature for `>`, it (and other familiar mathematical operators) may be used in infix form in expressions.

Next, we need a predicate that indicates whether an understandable object "*exists in reality.*" Eder and Ramharter use the symbol $E!$ for this, but we will use the identifier `re?` (for "really exists") and specify it as an uninterpreted predicate as follows.

PVS fragment
```
re?(x): bool
```

As noted, all treatments of the Ontological Argument take existence of some understandable object "than which there is no greater" as a premise. Eder and Ramharter write this as

ExUnd: $\exists x Gx$

and in PVS we render it as follows.

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

Our goal is to prove the claim that some such maximal object exists in reality, which Eder and Ramharter write as

God!: $\exists x(Gx \wedge E!x)$

and we write in PVS as follows.

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

[2]Unless explicitly indicated otherwise, quantification in PVS is over the type of the quantified variable(s). Types may be empty, and PVS takes care of associated soundness issues.

We cannot prove this claim without some additional premise to provide a connection between *greater* and *exists in reality.* Eder and Ramharter first propose the following premise (in words: if something does not exist in reality, then there is a greater thing).

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

In PVS, we write this as follows.

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

We now have enough to assemble the complete PVS theory shown in Figure 2.

```
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
```

Figure 2: First-Order Version of the Argument in PVS

We can prove the theorem with the following proof commands.

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

The first two instruct PVS to use the named lemmas or axioms as premises, and the third instructs it to use its general-purpose proof strategy, taking care to observe the polarity (i.e., positive vs. negative occurrences) of terms when searching for quantifier instantiations. PVS reports that the theorem is proved and its proofchain analysis (which checks that all proofs are complete, and also those of any lemmas they cite, plus any incidental proof obligations) 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?
```

Although we have now proved the desired conclusion, Eder and Ramharter consider this formalization of the Argument to be unsatisfactory because it violates two of their requirements for a good reconstruction. In particular, it "packs too much" of the argument into the premise `Greater1` and so the reconstructed proof does not follow the structure of Anselm's original. Furthermore, that premise cannot be considered *analytic* (another of their seven desiderata), by which they mean "we should attribute to the author only premises that he could have held to be true for conceptual (non-empirical) reasons. The premises should be direct consequences of conceptions presupposed by the author." `Greater1` cannot be considered analytic because it says nothing about what it means to be *greater* other than the contrived connection to *exists in reality.*

Accordingly, Eder and Ramharter next propose the following premises.

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

E!: $\exists x E!x$.

That is, an object that *exists in reality* is *greater* than one that does not, and there is some object that does *exist in reality.*

In PVS, these are written as follows and together replace `Greater1` in Figure 2.

PVS fragment

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

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

The same proof strategy as before succeeds in verifying the theorem with these new premises.

PVS proof

```
(lemma "ExUnd") (lemma "Greater2") (lemma "Ex_re") (grind :polarity? t)
```

Eder and Ramharter consider this formalization, too, to be an unfaithful reconstruction of Anselm's Argument. First, Anselm does not assume that something *exists in reality*, so `EX_re` violates the requirement for local conformity. Second, `Greater2` is no more analytic or less contrived than `Greater1`. Finally, these premises and proof strategy entirely miss a key point of Anselm's argument, which is that if a thing does not exist in reality, then a greater object would be *itself*, conceived as existing in reality. Eder and Ramharter employ higher-order logic to reconstruct the argument in a way that is faithful to these ideas, and we will verify their approach in the following section. Before doing so, however, I wish to make some observations on the first-order reconstructions presented above.

The second reconstruction, employing `Greater2` and `Ex_re`, is identical to that described in the earlier paper [17], where those axioms are named `reality_trumps` and `someone`, respectively. The equivalence may not be obvious, however, because the previous and current PVS specifications use apparently different types and expressions. In the notation of the current paper the previous paper uses the following definition and axiom.

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

P1: AXIOM nonempty?(greatest)
```

Sets and predicates are the same in higher-order logic, and the set comprehension notation in PVS is equivalent to lambda-abstraction. Hence, `greatest` is just a syntactic variation on our predicate `God?`; similarly, `P1` is equivalent to `ExUnd`. We can demonstrate this by postulating the following conjectures (`THEOREM`, `LEMMA`, `CLAIM` and `CONJECTURE` are synonyms in PVS).

```
                                                          PVS fragment
gr_God: CONJECTURE greatest = God?

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

These are proved by

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

and

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

respectively. The PVS proof strategy `apply-extensionality` establishes the equivalence of two functions (or predicates) by showing them to be equal for all values of their argument(s).

Next, we note that `greater1` is equivalent to the premise stated as axiom `P2` in the earlier paper, where it was shown to be directly circular: that is to say, the premise and conclusion were proved to be equivalent. That is not so here (i.e., `Greater1` cannot be proved from `God_re`); the reason is that the earlier treatment carries an additional premise (namely, trichotomy of `>`) needed to ensure that the definite description employed there is well-defined.

The simplicity of these demonstrations exemplifies what seems to me a potential benefit of mechanized analysis in this domain: it facilitates comparison of apparently different formulations of the same problem.

3 Eder and Ramharter's Higher-Order Reconstruction

In search of a more faithful reconstruction of Anselm's Argument, Eder and Ramharter observe that Anselm attributes properties to objects and that some of these (notably *exists in reality*) contribute to evaluation of the *greater* relation. They formalize this by hypothesizing some class $\mathcal{P}$ of "*greater*-making" properties on objects and then defining one object to be greater than another exactly when it has all the properties of the second, and more besides. They write this as follows

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 $\exists_{\mathcal{P}} F$.

This kind of restricted quantification is common in computer science (for example, it is useful in specifying the properties of partial functions) and *predicate subtypes* provide a convenient way to mechanize it [18]. Using predicate subtypes, we can specify Eder and Ramharter's **Greater 3** construction in PVS as follows.

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))
```

First, 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]`. In fact, 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`. Observe that `P` is a set, which is equivalent to a predicate in higher-order logic; in PVS, a predicate in parentheses denotes the corresponding predicate subtype, so `F` is defined here as a variable ranging over the members of `P`. Finally we define the relation `>` in the same way as Eder and Ramharter.

The formalized version of the Argument starts with *something* ("than which there is no greater"); if the selected something does not *exist in reality*, the proof strategy is to consider that *something* augmented with the property *exists in reality*. We need an additional premise to ensure that the augmented thing is an *understandable object*. Eder and Ramharter formulate this as follows.

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)
```

We use `FF(F) = F(x)` rather than the `FF(F) IFF F(x)` of Eder and Ramharter because `=` facilitates use of this premise as a rewrite rule, although this is not exploited here (`=` on the Booleans has the same truth table as `IFF`).

The statement of the theorem to be proved must be adjusted to ensure that `re?` is a member of the set `P` of *greater*-making properties.

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

The complete higher-order PVS specification is shown in Figure 3 and the PVS proof for `God_re` is shown below.

PVS proof
```
(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)))))
```

```
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
```

Figure 3: Higher-Order Version of the Argument in PVS

PVS proofs are instructions to a machine and are not really intended for human consumption: the best way to understand what is going on is to observe their operation one step at a time in the PVS prover, which displays its current state as a tree of sequents. However, a rough account of this proof is that the first block of 4 commands does some propositional simplification, expands the definition of the (built-in) set membership predicate, installs `ExUnd` as a premise, and then Skolemizes it; `x!1` is the Skolem constant representing the *something* in `ExUnd`. The next block of 2 commands performs a case analysis on whether this *something exists in reality*. If it does, we are done. Otherwise, in the next block of 3 lines we use the `Realization` axiom as a premise. We instantiate its top quantifier with the set of properties consisting of those possessed by `x!1`, plus `re?`. This is exactly the same tactic as used by Eder and Ramharter, who postulate a predicate or set $\mathcal{F}_E!$ defined as $Fg \vee F = E!$, where g is their Skolem constant from `ExUnd`. The final block of 12 lines perform routine manipulation (that would be entirely automated if PVS did better quantifier reasoning) to finish the proof.

Eder and Ramharter note that there is nothing in the proof that is specific to `re?`; thus, we can prove the generalization that for *any* property `a` in `P` (i.e., for any *greater*-making property), there is a God that has that property. In PVS, this is stated as follows, and the proof is the same as that shown above, with obvious small adjustments to replace `re?` by `a!1`, the Skolem constant for `a`.

PVS fragment
```
God_gen: THEOREM FORALL (a: (P)): EXISTS x: God?(x) AND a(x)
```

Eder and Ramharter state that Anselm's proof (in Chapter III of the Proslogion) of the *necessary* existence of God (or the inconceivability of the nonexistence of God) follows the same pattern with $\Box E!$ substituted for `a`. This seems vacuous to me, because $\Box E!$ is an uninterpreted predicate with nothing to bind it to the meaning associated with "necessity," but this observation also applies to $E!$ (or `re?` in PVS) as there is nothing that binds it to the interpretation "exists in reality." These deficiencies are those of Anselm's Argument and the formal reconstructions merely make them explicit. My earlier paper [17] exhibited a model for the axiomatization used there in which `U_beings` were interpreted by the natural numbers, `>` by `<` on these, and *exists in reality* by "is less than 4." Thus, although the theological interpretation is consistent with that formalization, it is hardly compelled by it, and the same is true here.

Eder and Ramharter consider the heart of Anselm's Argument, and hence something that must be preserved in a reconstruction, is that if something "than which there is no greater" does not exist in reality, then Anselm compares it with the *itself*, conceived as existing. To formalize 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 state that the Skolem constants a (from **Realization**) and g (from **ExUnd**) appearing in their formalization of the argument can be proved to be quasi-identical, $a \equiv_{\{E!\}} g$.[3]

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)
```

[3]Actually, they write $a \equiv_{E!} g$, which is type-incorrect; PVS detects such errors.

Then, in the PVS proof of `God_re`, at the penultimate step where the sequent appears as follows[4]

```
                                                          PVS sequent
God_re.2.2 :

[-1]  FORALL F: (F(x!1) OR F = re?) = F(x!2)
[-2]  God?(x!1)
[-3]  P(re?)
  |-------
{1}   re?(x!2)
[2]   re?(x!1)
```

we establish a subsidiary proof by issuing the following PVS proof command.

```
                                                          PVS proof step
 (case "quasi_id(x!1, x!2, singleton(re?))")
```

This creates a branch in the proof. On one branch, the formula stated in the `case` is installed as a premise, and in the other it is installed as a proof goal. On the first branch, we delete the new premise and continue the proof as before. On the second branch. we delete the formulas corresponding to -2, 1, and 2 (since we do not want the subsidiary proof to use these—that would be tantamount to proving the main theorem), flush the decision procedure state (for the same reason), then discharge the goal with `(lazy-grind)` (which is like the general-purpose strategy `grind` except it postpones quantifier instantiation until simplification has been performed).

Eder and Ramharter observe that a weaker form of the `Realization` premise, which guarantees the existence of an understandable object exemplifying all properties in a given set of properties but not necessarily *only* such properties, is enough to establish the conclusion to the Argument, but not quasi-identity of the compared objects. In PVS, the weakened premise is written as follows

```
                                                          PVS fragment
Realization_W: AXIOM
  FORALL (FF: setof[(P)]): EXISTS x: FORALL F: FF(F) => F(x)
```

(i.e., `=>` rather than `=`) and it is easy to confirm that this suffices to prove `God_re` but not the subsidiary goal described above.

Another of Eder and Ramharter's requirements for a satisfactory formal reconstruction of an informal argument is the following.

> "If the argument and, therefore, its reconstruction are deductively valid, the premisses should contain the conclusion in a non-obvious way. The

[4]The interpretation of a PVS sequent is that the conjunction of expressions above the turnstile should imply the disjunction of those below; top-level negations are eliminated by moving their expression to the other side of the turnstile.

conclusion has to be contained in the premisses; otherwise, the reasoning would not be deductive. But an argument can convince someone only if it is possible to accept the premisses without already recognizing that the conclusion follows from them. Thus, the desired conclusion has to be hidden in the premisses."

In my view, if their other requirements, such as local and global conformity, and analytic credibility of the premises, are satisfied, then the extent to which the conclusion is "hidden" in the premises reflects that attribute of the informal argument and need not be established as a separate requirement on the reconstruction.

However, mechanized explorations can help reveal how well hidden, or not, is the connection between the premises and conclusion of Anselm's Argument. For example, we already noted that the premise **Greater 1** is directly equivalent to the conclusion when the *greater* relation is assumed to be trichotomous.

In Eder and Ramharter's higher-order formulation, it is easy to show that anything "than which there is no greater" must have *all* the *greater*-making properties. In PVS this is stated as follows

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

and can be proved by the following proof commands.

PVS proof
```
(skolem-typepred)
(ground)
(expand "God?")
(lemma "Realization")
(inst - "{ G: (P) | G(x!1) OR G=a!1 }")
(skosimp)
(inst 1 "x!2")
(lazy-grind)
```

It is trivial to prove `God_gen` (and hence `God_re`) from `God_all`. For example, the following PVS proof command will do it.

PVS proof
```
(grind-with-lemmas :defs nil :lemmas ("ExUnd" "God_all"))
```

Here, `:defs nil` tells the prover not to expand definitions (specifically, that of `>`). This route might be the most concise way to derive `God_re`, but does not satisfy Eder and Ramharter's requirements for a reconstruction (as opposed to an emendation). On the other hand, `God_all` illustrates how much is "packed into" the combination of `Realization` and the definition of `>`. This is further illustrated by the observation that since all Gods have all *greater*-making properties, they must all be *quasi-identical*. In PVS, this can be written as follows

PVS fragment
```
all_God: THEOREM God?(x) & God?(y) => quasi_id(x, y, emptyset)
```

and can be proved by the following proof commands.

PVS proof
```
 (skosimp)
 (expand "quasi_id")
 (skolem-typepred)
 (lemma "God_all")
 (grind :if-match all)
```

It seems to me that results such as `God_all` and `all_God` highlight the specious character of Anselm's Argument. While the conclusion hides rather well within the premises of the informal argument, formal reconstruction and mechanized exploration expose it to scrutiny.

We have now reproduced in PVS all the constructions and proofs in classical logic presented by Eder and Ramharter, and explored some further consequences of these. We do not extend our reproduction to the modal logic constructions because I rather suspect that Benzmüller and Woltzenlogel Paleo will wish to do that.

4 Discussion

Readers will, I hope, be struck by the simplicity of this exercise. The resources of a verification system make it very easy to reproduce formalized constructions and proofs of the kind employed by Eder and Ramharter in something very close to mathematical vernacular, yet with the security of mechanized analysis built on a sound foundation: once you have learned to use a verification system, you can more or less type these formalizations straight in.

Given that it is feasible, even simple, to subject formalized arguments of this kind to mechanically assisted analysis, we might next ask if it is worthwhile. Some insight may be obtained by looking at existing formalizations of the Ontological Argument; there are several such, including [1,12,15]. All of these (of which Eder and Ramharter mention only [12]) require capabilities beyond first-order logic. Oppenheimer and Zalta [12] and Pottinger [15] employ definite descriptions and therefore require some logical machinery to ensure these are well-defined; both papers use free logic for this purpose and devote several pages to explaining and justifying the particular logic that they employ. Adams [1] and Pottinger [15] employ modal operators and likewise devote some pages to their introduction.

These idiosyncratic logics pose two difficulties. First, although the general principles of free logics and quantified modal logics are well understood, the reader must spend some time mastering (and possibly resolving doubts about) the particular logic employed by each paper. Thus, the technicalities of the formalization may obscure rather than clarify the philosophical problem at hand. Second, because each formalization uses a different logic, it is difficult to compare them: one must decide whether apparent differences in formalization are fundamental or are merely artifacts of the different notations and logics employed.

The reason that idiosyncratic logics are employed is that the standard languages of logic were created for study rather than use and are stripped down to basic concepts:

> "...our interest being less often in their actual and practical use as languages than in the general theory of such use and its possibilities in principle" [4, page 47].

In contrast, the specification languages of verification systems take the concepts and semantics of logic and engineer them into notations that provide the linguistic resources to formalize real systems. Thus, formalizations of the Ontological Argument require little, if any, augmentation to the language of a verification system; this allows attention to focus on the formalizations themselves and facilitates comparison of different formalizations. We saw this in the PVS renditions of Eder and Ramharter's formalizations and the earlier rendition [17] of Oppenheimer and Zalta's: PVS provides the logical resources (principally, an extensive type system) for sound treatment of the higher-order constructions employed by Eder and Ramharter and the definite description employed by Oppenheimer and Zalta. In cases where a formalization does need to augment the language of a verification system, the system provides mechanisms to do this in a sound and reusable manner, as Benzmüller and Woltzenlogel Paleo do for higher-order modal logic in Coq [3].

So, I think the first benefit that philosophers might obtain by employing verification systems is access to rich, well-engineered logical notations that would enable them to focus on the formalization of their problem, rather than the preliminaries of logic. And, of course, using a single notation will facilitate comparison of different formalizations.

The second benefit is that formal verification systems are truly formal: that is, they reduce the manipulation of logical formulas to mechanized calculation and thereby guarantee correctness. In contrast, some "formalizations" employ the symbols of logic but manipulate them in an informal manner. For example, Silvestre [19] provides a critical examination of Adam's formalization[5] [1] and reports that

> "...Adams uses a somehow incomplete calculus: the first-order part is ok, but he does not say from which calculus he draws his modal inferences. Consequently, we do not know which semantics might be associated with it (since there are many modal systems), neither if the resulting system with a specific semantics would be sound and complete."

[5] I am grateful to Bruno Woltzenlogel Paleo for bringing this paper to my attention.

The whole point of formalization is to allow attention to focus on the argument itself by eliminating concern about validity of the logical inferences employed. This is vitiated if the formalization has merely the appearance but not the substance of true formality.

Of course, it is very difficult, and tedious, to perform formal calculations by hand: mechanization is required to make formalization practical. The benefit of mechanization is most strongly realized when exploring and comparing alternative formalizations of the same or related problems. With discipline, it may be feasible to undertake one or two formalizations of the kind presented here with pencil and paper, but it is very hard to maintain the necessary degree of discipline and skepticism when exploring several subtly different renditions of the same problem. A mechanized verification system, on the other hand, brings the same discipline and implacable skepticism to the hundredth version as to the first.

It is important to note that mechanization must support every aspect of formal analysis: it is not enough to mechanize just the core logic, as simple first-order provers do. For example, Oppenheimer and Zalta used a free logic to deal with definedness issues related to use of the definite description in their treatment of the Ontological Argument [12] but, when they mechanized the analysis in a first-order prover, they had to deal with those issues outside the prover and introduced errors that caused them, mistakenly, to believe the prover had discovered a simplification in the Argument [8, 13].

The guarantee of validity provided by comprehensive mechanization would be the same whether the mechanization is fast or slow, but the raw speed and power of the deductive machinery provided by modern verification systems creates new opportunities: it facilitates experiments, and these are a third potential benefit.

An example of experimental investigations that could be undertaken is examination of the extent to which a given formalization uses circular reasoning, or "begs the question": that is, employs premises equivalent, or nearly so, to the conclusion to be proved. If we have premises p_1 and p_2 (which may be conjunctions of simpler premises) that are sufficient to prove the conclusion c, that is, p_1, $p_2 \vdash c$, then we can say that p_2 begs the question if it is equivalent to c under the remaining premises p_1: that is, $p_1 \vdash (p_2 = c)$.[6] It may be that p_2 does not beg the question under p_1 but does so under some additional premise x: that is, p_1, $x \vdash (p_2 = c)$. By the Deduction Theorem, this is equivalent to

$$p_1 \vdash x \supset (p_2 = c). \tag{1}$$

We could then say that the "size" of x represents how close p_2 is to begging the question. For example, the premise `Greater1` of Figure 2 begs the question under the

[6]Another way of looking at it is that in addition to using p_2 to prove c, we can also do the reverse: $p_1, c \vdash p_2$.

additional premise that > is trichotomous, and PVS can easily verify the following instance of (1).[7]

```
                                                          Future PVS
  begging: CLAIM trichotomous?(>) => (Greater1 = God_re)
```

We could argue that any reasonable ordering relation is trichotomous, so this is a "small" augmentation, and therefore Greater1 effectively begs the question in this formalization of the Argument. Observe that we can solve (1) for x, but its instances are likely to be ugly formulas; the experimental procedure enabled by efficient mechanization is to seek (through trial and error, guided by intuition and insight) a succinct and plausible x that is sufficient to discharge (1), thereby demonstrating that the formalization under examination begs the question.

The experiments suggested above evaluate individual formalizations against properties such as "begging the question." Another potentially useful application of efficient mechanization is comparison between different formalizations of the same argument. I earlier described how their uniform specification in the notation of a verification system could facilitate comparison of different formalizations, but those comparisons would be "by eyeball." With the efficient mechanization of a verification system, we can explore the differences in a more experimental manner: for example, we can ask if a construction (e.g., function or type) in one formalization is equivalent to that in another, or whether a premise of one implies that of another. We saw an example of this at the end of Section 2, where the conjectures gr_God and ne_Ex establish the equivalence of key constructions in Eder and Ramharter's second first-order formalization with those employed in one of Oppenheimer and Zalta's.

It seems to me that enabling such comparisons may be the most potent and useful application of verification systems to philosophical topics. The Ontological Argument, in particular, has been formalized and examined in many different ways by many different authors; these seem ripe for analysis and comparison in a uniform notation supported by mechanized proof. I suspect that centuries of debate and dispute could rapidly be clarified through such application of modern technology.

References

[1] Robert Merrihew Adams. The logical structure of Anselm's arguments. *The Philosophical Review*, 80(1):28–54, 1971. 15, 16

[2] Christoph Benzmüller and Bruno Woltzenlogel Paleo. Gödel's God in Isabelle/HOL. *Archive of Formal Proofs*, 2013. 1

[7]This is labeled "Future PVS" because the current version of PVS does not support use of formula identifiers within expressions (you must provide the actual formula), but a future version will do so.

[3] Christoph Benzmüller and Bruno Woltzenlogel Paleo. Interacting with modal logics in the Coq proof assistant. In *Computer Science—Theory and Applications: 10th International Computer Science Symposium in Russia, CSR 2015*, Volume 9139 of Springer-Verlag *Lecture Notes in Computer Science*, pages 398–411, Listvyanka, Russia, July 2015. 1, 16

[4] Alonzo Church. *Introduction to Mathematical Logic*, volume 1 of *Princeton Mathematical Series.* Princeton University Press, Princeton, NJ, 1956. 16

[5] *Coq home page.* `http://coq.inria.fr`. 1

[6] Ali Ebnenasir and Amer Tahat. An implementation of Avicenna's Necessary Existent Argument in PVS. Submitted, March 2015. 1

[7] Günther Eder and Esther Ramharter. Formal reconstructions of St. Anselm's Ontological Argument. *Synthese*, 192(9):2795–2825, October 2015. 1, 2, 3

[8] Paweł Garbacz. PROVER9's simplifications explained away. *Australasian Journal of Philosophy*, 90(3):585–592, 2012. 17

[9] *Isabelle home page.* `http://isabelle.in.tum.de/`. 1

[10] Anthony Narkawicz and César Muñoz. Formal verification of conflict detection algorithms for arbitrary trajectories. *Reliable Computing*, 17:209–237, December 2012. 3

[11] Patrick Oladimeji, Paolo Masci, Paul Curzon, and Harold Thimbleby. PVSio-web: A tool for rapid prototyping device user interfaces in PVS. In *Fifth International Workshop on Formal Methods for Interactive Systems: FMIS 2013*, Volume 69 of *Electronic Communications of the EASST*, London, England, June 2013. 2

[12] Paul E. Oppenheimer and Edward N. Zalta. On the logic of the Ontological Argument. *Philosophical Perspectives*, 5:509–529, 1991. Reprinted in *The Philosopher's Annual: 1991*, Volume XIV (1993): 255–275. 1, 3, 15, 17

[13] Paul E. Oppenheimer and Edward N. Zalta. A computationally-discovered simplification of the Ontological Argument. *Australasian Journal of Philosophy*, 89(2):333–349, 2011. 1, 17

[14] S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, Volume 607 of Springer-Verlag *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. 1

[15] Garrel Pottinger. A formal analysis of the Ontological Argument. *American Philosophical Quarterly*, 20(1):37–46, 1983. 15

[16] *PVS home page*. http://pvs.csl.sri.com/. 1

[17] John Rushby. The Ontological Argument in PVS. In Nikolay Shilov, editor, *Fun With Formal Methods*, St Petersburg, Russia, July 2013. Workshop in association with CAV'13, revised paper accepted for publication in *Informatica Didactica*. 1, 3, 8, 12, 16

[18] John Rushby, Sam Owre, and N. Shankar. Subtypes for specifications: Predicate subtyping in PVS. *IEEE Transactions on Software Engineering*, 24(9):709–720, September 1998. 9

[19] Ricardo Sousa Silvestre. On the logical formalization of Anselm's ontological argument. *Revista Brasileira de Filosofia da Religião*, 2(2):142–161, 2015. 16