Invited paper presented at the CAV Workshop *Fun With Formal Methods*, St. Petersburg, Russia, 13 July 2013.

The Ontological Argument in PVS[*]

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

Abstract

The Ontological Argument, an 11th Century proof of the existence of God, is a good candidate for *Fun With Formal Methods* as nearly everyone finds the topic interesting. We formalize the Argument in PVS and verify its correctness. The formalization raises delicate questions in formal logic and provides an opportunity to show how these are handled, soundly and efficiently, by the predicatively-subtyped higher-order logic of PVS and its mechanized support. The simplicity of the Argument, coupled to its bold conclusion, raise interesting issues on the interpretation and application of formal methods in the real world.

1 Introduction

The Ontological Argument is a proof of the existence of God. This is a topic that almost everyone, believer and unbeliever alike, finds deeply interesting. Formulation and verification of the Argument in a mechanized theorem prover is therefore an excellent candidate for *Fun With Formal Methods*. Furthermore, the formalization raises subtle issues in mathematical logic and thereby serves as a useful pedagogical vehicle to introduce students to these issues and how they are resolved in a mechanized system such as PVS.

Bertrand Russell observed "The Argument does not, to a modern mind, seem very convincing, but it is easier to feel that it must be fallacious than it is to find out precisely where the fallacy lies." He also reported earlier exclaiming "Great God in Boots! The Ontological Argument is sound!" Since PVS joins Russell in assuring us that the Argument is sound, but most will agree that its conclusion is troubling,[1] we are moved to examine its premises and the interpretation of its conclusion with special care and interest. Formal methods are often advocated in the assurance of

[*]This research was supported by SRI International.

[1]This is not about atheism: many of those who have analyzed and criticized the Ontological Argument were devout believers; rather, the question is whether something as ineffable as the existence of God can be subject to a merely *a priori* demonstration.

> Thus even the fool is convinced that something than which nothing greater can be conceived is in the understanding, since when he hears this, he understands it; and whatever is understood is in the understanding. And certainly that than which a greater cannot be conceived cannot be in the understanding alone. For if it is in the understanding alone, it can be conceived to exist in reality also, which is greater. Thus if that than which a greater cannot be conceived is in the understanding alone, then that than which a greater cannot be conceived is itself that than which a greater can be conceived. But surely this cannot be. Thus without doubt something than which a greater cannot be conceived exists, both in the understanding and in reality.

Figure 1: The Ontological Argument from Anselm's *Proslogion II* (translation by William Mann, quoted from [15])

critical systems, where the larger framework of argument in support of certification is generally referred to as a Safety or Assurance Case [4, 20]. A question that then often arises is "what is the difference between a formal verification and an assurance case?" to which part of the answer is that formal verification provides assurance that its conclusion follows logically from its premises, while the (rest of the) assurance case examines the credibility of the premises and the real-world interpretation of the formal conclusion. Thus, the Ontological Argument provides a paradigmatic illustration of this difference between formal verification and a complete assurance case.

The Ontological Argument was first formulated in the year 1078 by St. Anselm of Canterbury in his *Proslogion*, and it has been an object of study and dispute ever since. Many of the great philosophers and logicians have written on the topic, including Aquinas, Descartes, Leibniz, Hume, Kant (who gave it the name we use today), and Gödel [15].[2] An English translation of the original presentation appears in Figure 1.

More recently, Oppenheimer and Zalta made a careful study of the logic used in the Ontological Argument and gave a formalized proof (by hand) [12]. Later, they mechanically verified the Argument using the PROVER9 first-order theorem prover [14], from whose computations they extracted a simplified form of the argument, though the validity of their mechanization is disputed [6].

In this paper, I formally verify the Ontological Argument in the PVS verification system [17]. My formulation is based directly on Oppenheimer and Zalta's informal presentation, but its rendition in the higher-order, predicatively subtyped, logic of

[2] I am not a philosopher and am not familiar with the primary sources, so I cite just the secondary sources I found useful.

PVS raises interesting topics concerning definite descriptions, existence of constants, and quantification over possibly empty types.

The following section introduces some necessary preliminaries—in particular, the formalization of definite descriptions—and is followed in Section 3 by the formalization and verification of the Ontological Argument itself. Section 4 discusses the interpretation of the formal specification and relates this to similar questions in assessment of formal verifications in support of arguments for safety or security.

2 Preliminaries

A modern rendition of Anselm's argument is given in the following section as Figure 2 on page 8. Our goal is to formalize and verify this in PVS. Inspection of the Argument in Figure 2 shows that it revolves about "the greatest thing." In logic, an expression of the form "the such and such," or more formally "the x such that $P(x)$" is called a *definite description.* Early in the history of formal logic there was much debate about how to render definite descriptions formally and how to deal with expressions containing definite descriptions when there is no "such and such." The canonical example of such a non-referring definite description is "the present King of France" and the larger problem concerns the status of expressions such as "the present King of France is bald": is this *true*, *false*, or inadmissible? If either of the first two alternatives is chosen, then what about "the present King of France is *not* bald" (does it negate the previous valuation?).

Russell in 1905 gave a treatment for definite descriptions that is widely, though not universally, accepted today. He argued that "the present King of France is bald" should be interpreted as the conjunction of the following three claims.

1. There exists an x that is the present King of France,
2. Every x, y that is a present King of France satisfy $x = y$
   (i.e., the present King of France, if it exists, is unique),
3. Every x that is a present King of France, is bald.

In this reading, we see that "the present King of France is bald" is false, since the first of the three conjuncts is false; for the same reason, "the present King of France is *not* bald" also is false.

We would like to refine this rather contextual reading into one that is more compositional, so we seek a way to represent the definite description standing alone. Russell used the notation $\iota x\colon P(x)$ for the definite description "the x such that $P(x)$," where P is some predicate, and required it to satisfy the first of the three conjuncts above, that is: $P(\iota x\colon P(x))$ (i.e., $\iota x\colon P(x)$ is a witness to $\exists x\colon P(x)$), with the second conjunct as a side condition.

A compositional reading of "the present King of France is bald" could then have the form $\forall y \colon (y = \iota x \colon P(x)) \supset Q(y)$, where P is the predicate "is the present King of France," and Q is the predicate "is bald". A more succinct alternative (when $\iota x \colon P(x)$ is well-defined and unique) would be $Q(\iota x \colon P(x))$. We now have to decide what to do when, as here, $\iota x \colon P(x)$ fails to denote (i.e., there is no x such that $P(x)$), or the side condition fails (i.e., there is more than one such x). One resolution is to say that expressions such as $Q(\iota x \colon P(x))$ are inadmissible unless there is a unique x such that $P(x)$, and the issue then becomes one of enforcing such side conditions.

Now that we understand the general issue, let us see how this is handled in PVS. PVS is a higher-order logic (that is to say, functions can take functions as arguments and return them as values, and quantification can extend over functions). Higher-order logics require a typing discipline to ensure consistency, and PVS enriches the standard "simple" type theory of higher-order logic by allowing dependent types and predicate subtypes (it also supports structural subtypes). Typechecking in PVS is undecidable in general (i.e., it requires theorem proving), but the undecidable constructions are few and their impact is local, so the majority of its typechecking is algorithmic. When a circumstance is encountered that is algorithmically undecidable, the PVS typechecker attaches a proof obligation called a Typecheck Correctness Condition (TCC) to the theory concerned and no development involving that theory is considered complete (by the PVS tools) unless all its TCCs have been discharged. One circumstance that causes a TCC to be generated is when a term of one type is supplied where a subtype is expected (e.g., in x/y where x and y are numbers, y is expected to be in the subtype of numbers that are nonzero); another is when a type that may be empty is supplied where a nonempty type is required. Predicate subtypes allow much of a specification to be embedded in its types; this avoids cluttering the main development with side conditions and makes information available in a way that an automated theorem prover can easily locate and use. See [22] for several examples illustrating the utility of predicate subtypes.

In higher-logic, predicates are just functions with range type Boolean (written as `bool` in PVS), and sets are identified with predicates, with `setof[T]` and `pred[T]` simply being different names for the same type, namely `[T -> bool]`; similarly, `member(x, A)` and `A(x)` are synonyms in PVS. It is easy to specify the higher-order predicates `empty?`, `nonempty?`, and `singleton?` (by convention, predicates in PVS generally have names ending in `?`) that indicate whether a set is empty, nonempty, or has exactly one member. A predicate name enclosed in parentheses denotes the corresponding subtype, so that `(nonempty?)` and `(singleton?)` specify those subtypes (of sets) that are nonempty, or singletons. Definite descriptions are then specified as a function `the(P)`, where `P` is required to be of the singleton subtype, whose range type is `(P)`; any use of this function will generate a TCC that requires `singleton?(P)` to be proved for the argument `P`. In PVS, these concepts are specified as follows.

```
Russell [T: TYPE]: THEORY
 BEGIN

  x, y: VAR T
  A: VAR setof[T]

  empty?(A): bool = (FORALL x: NOT A(x))

  nonempty?(A): bool = NOT empty?(A)

  singleton?(A): bool = (EXISTS (x:(A)): (FORALL (y:(A)): x = y))

  the(P: (singleton?)): (P)

END Russell
```

This formal specification should be self-explanatory (one of the design goals of PVS is that it should closely correspond to mathematical vernacular). Observe the use of predicate subtyping in the definition of `singleton?`: the construction `EXISTS (x:(A)): ...` means, "there exists an `x` of type `T` satisfying the predicate `A(x)` such that ..." More interestingly, the declaration of `the` says that it takes a singleton set `P` as its argument and returns a member of that set as its value. Observe that this is not a definition for `the` (i.e., there is no `=` introducing a "body" for the function, in the way there is for `empty?` etc.); it merely specifies its type. Of course, the typing constraints are such that there is only one possible interpretation for the function and this provides an implicit definition.

When we typecheck the theory `Russell`, PVS reports that is has generated a TCC which, on inspection, is the following (`%` symbols introduce comments in PVS).

TCC

```
% Existence TCC generated (at line 14, column 2) for
    % the(P: (singleton?)): (P)

the_TCC1: OBLIGATION EXISTS (x: [P: (singleton?) -> (P)]): TRUE;
```

The explanation for this TCC is that our declaration for `the` is asserting existence of a constant (i.e., a function) of the higher type `[P: (singleton?) -> (P)]` and we need to be sure this type is nonempty, otherwise we have an inconsistency. (An example of an empty function type is one whose domain is empty but whose range is nonempty; observe that the function type whose domain and range are both empty is not itself empty: it has the empty function as a member.) Intuitively, it seems easy to discharge this TCC: the set `P` is a singleton and therefore nonempty, so any function that returns a member of this set will demonstrate nonemptiness of the type. The difficulty in constructing this proof is that we need a name for "a member of this set" and that is closely related to what we are trying to specify.

This reasoning is on the right track, however. Definite descriptions are closely related to “choice functions”; given a nonempty set, a choice function returns some member of the set. We can specify this as follows.

```
  choose(P: (nonempty?)): (P)
```

Observe that this declaration is the same as `the`, except its domain is merely required to be `nonempty?` rather than `singleton?`. Given this declaration, we can discharge `the_TCC1` by the following PVS proof commands.

Proof Script

```
(inst + "LAMBDA (A: (singleton?)): choose(A)")
(grind)
```

The first of these instantiates the variable `x` in the TCC by the function

```
    LAMBDA (A: (singleton?)):  choose(A),
```

and the second invokes one of PVS's more powerful general-purpose proof strategies.

This discharges the TCC from the definition of `the`, but the invocation of `choose` introduces one of its own.

TCC

```
% Existence TCC generated (at line 10, column 2) for
    % choose(p: (nonempty?)): (p)

choose_TCC1: OBLIGATION EXISTS (x: [p: (nonempty?) -> (p)]): TRUE;
```

This poses the same difficulty as the TCC for `the`, namely the problem of constructing a name for the function that provides an existential witness that this function type is nonempty. We overcome this difficulty in a similar way to that used before: we regress to a more primitive kind of choice function.

Hilbert defined a function he called ε and that we will call `epsilon` that is a choice function for general (i.e., possibly empty) sets: if its set argument is nonempty, it returns a member of that set; otherwise, it returns some arbitrary value of the base type for the set. Of course, the base type must be nonempty, and we can ensure this by defining `epsilon` within a theory whose parameter is required to be nonempty, as follows.

```
epsilons [T: NONEMPTY_TYPE]: THEORY
 BEGIN

  x: VAR T
  p: VAR setof[T]

  epsilon(p): T

  epsilon_ax: AXIOM (EXISTS x: p(x)) => p(epsilon(p))

 END epsilons
```

Because T is known to be nonempty, the definition of epsilon does not require a TCC to establish nonemptiness of its function type [setof[T] -> T]; however, whenever the epsilons theory is used, a TCC will be generated if necessary to establish nonemptiness of the instantiation for its type parameter.

We can now discharge choose_TCC1 by the following proof.

```
                                                          Proof Script
(then (inst + "LAMBDA (A: (nonempty?)): epsilon(A)") (grind))
(then (rewrite "epsilon_ax[T]") (grind))
```

The first line tells PVS to use the specified instantiation, then apply grind to any subgoals. The instantiation causes a TCC to be generated within the proof to ensure A(epsilon[T](A)) (this is due to the range type specified for choose); grind alone cannot prove this, so the next line instructs the prover to rewrite with epsilon_ax[T], followed by another grind to clean up.

We have now succeeded in specifying definite descriptions in PVS as the function the, and have discharged all its attendant TCCs. Along the way, we have also defined the independently useful choice functions choose and epsilon. This might all seem a moderately difficult endeavor before we even get to the Ontological Argument itself but, in fact, all this work has already been done and is incorporated in the PVS "Prelude," which is a standard library of PVS specifications and theorems that is built into the system. The epsilons theory is one of those supplied in the Prelude, and the definitions we presented in the theory Russell are actually just part of a Prelude theory called sets. Large tracts of logic are defined in the Prelude, and many other branches of mathematics are formalized in other PVS libraries that are available from the website http://pvs.csl.sri.com.

Before we leave the topic of definite descriptions, there is a related issue that needs to be explained. In classical first-order logic, all quantification ranges over some domain of discourse and all terms used in instantiations or generalizations are assumed to be in this domain. Of course, definite descriptions challenge this assumption because they may fail to denote. For example, starting from the identity $X = X$, where X is some constant, we can obtain the theorem $(\exists a\colon a = X)$ by existential generalization. If X abbreviates "the present King of France," then we have just proved that there is a King of France!

There have been attempts to construct variants of first-order logic that address this difficulty; they are called "Free Logics" (because they are logics "free of existence assumptions..." [11]). Free Logics generally introduce an explicit "existence" predicate $E!$ (not to be confused with $\exists!$, which is used for uniqueness) and adjust the quantifier rules appropriately. Oppenheimer and Zalta's examination of the logic of the Ontological Argument [12] uses a Free Logic to manage complications due to possibly nondenoting definite descriptions. However, no first-order theorem prover automates Free Logic, nor provides definite descriptions, so Oppenheimer and Zalta's computational exploration with PROVER9 uses classical first-order logic [14] and these delicate issues are dealt with informally outside the system, and beyond the reach of automated checking. Oppenheimer and Zalta's examination of the deductions performed by PROVER9 showed that very little of their formalization was actually used and they were then able to produce a much reduced formalization that PROVER9 still found adequate. This led them to believe they had discovered a simplification to the original 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 [6] disputes this claim and observes that 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.

In PVS, these issues of definedness, existence of constants, and quantification over possibly empty domains are addressed soundly by its logic and enforced by its automation. Quantification in PVS is over types, which may be empty: universal quantification over an empty type yields `TRUE`, and existential quantification yields `FALSE` (these are standard rules of higher-order logic). And as we saw in construction of the definite description and choice functions in PVS, TCCs are generated as necessary to ensure that terms denote and that constants are not asserted for empty types.

3 The Ontological Argument in PVS

A modern rendition of Anselm's Ontological Argument is shown in Figure 2. Our goal is to formalize and verify this in PVS.

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 a greater thing—namely, something 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 2: Modern Rendition of The Ontological Argument

The first step is to formalize the opening line "We can conceive of something than which there is no greater." This clearly requires some base type of "things" or, as we prefer, `beings` and an ordering relation `>` on this type. The claim then is that we can conceive of a maximal element under this ordering. There are two issues here: the properties required of `>` to ensure there is a maximal element, and the notion "can conceive of."

Perhaps surprisingly, Oppenheimer and Zalta discovered that `>` does not need to be a true ordering relation, it merely needs to have a property they call "connectedness," defined by

$$\forall x, y\colon\ x > y \vee y > x \vee x = y.$$

This property is also called *trichotomy*, and is defined under the related name `trichotomous?` in the PVS Prelude. Next, we define the set `greatest` comprising all `beings` that are maximal under this relation (i.e., the set of just those `beings` than which there is no greater); it could have many members, or be empty, or be a singleton. These considerations lead to the initial PVS specification shown below.

```
ontological: THEORY
BEGIN

  beings: TYPE

  x, y: VAR beings

  >: (trichotomous?[beings])

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

END ontological
```

This specification should be self-explanatory; the only novelty is use of set notation to define `greatest`. This is merely a syntactic variation on the notation we have seen before: the following specification would be entirely equivalent.

Alternative
```
  greatest(x): bool = NOT EXISTS y: y>x
```

Typechecking this specification generates a TCC requiring us to establish that the type asserted for the function `>` is nonempty.

TCC
```
% Existence TCC generated (at line 8, column 0) for
    % >: (trichotomous?[beings])

greaterp_TCC1: OBLIGATION EXISTS (x: (trichotomous?[beings])): TRUE;
```

This is easily discharged by exhibiting the relation that relates everything to everything.

Proof Script

```
(inst + "LAMBDA (x,y: beings): TRUE")
```

Next, we want to specify that we "can conceive of" "the greatest," where the latter is a definite description that can be written in PVS as `the(greatest)`. Oppenheimer and Zalta introduce a predicate C to represent "can conceive of" but this seems unnecessary: merely establishing that `the(greatest)` is well-defined seems sufficient.

The set `greatest` comprises just those `beings` than which there is no greater; as we noted before, this set could have many members, or be empty, or be a singleton. If we simply append mention of `the(greatest)` to the PVS specification shown above, PVS will force us to prove a TCC to establish that `greatest` is, in fact, a singleton. Nothing specified so far requires this to be true. Oppenheimer and Zalta introduce a *Premise 1* at this point which, purged of its mention of the predicate C, would be expressed in PVS as follows.

Alternative

```
Premise_1: AXIOM EXISTS x: NOT EXISTS y: y > x
```

This is equivalent to asserting nonemptiness of `greatest` and we consider it more perspicuous to state it in this way, as the axiom P1 in the following continuation of the PVS specification given above.

```
P1: AXIOM nonempty?(greatest)

P1a: LEMMA singleton?(greatest)

the_greatest: beings = the(greatest)
```

The lemma `P1a` is easily proved from `P1` by expanding definitions, and applying information recorded in predicate subtypes (including, crucially, trichotomy of `>`). Given `P1a`, it is trivial to discharge the TCC arising from the definite description defining `the_greatest`.

We now turn to the next line of the Ontological Argument, which makes reference to things (or `beings`) that exist "in reality." We cannot use the existential quantifier $\exists$ for this purpose because that has purely logical import (i.e., whether some value in the domain of quantification has a specified property). Oppenheimer and Zalta use the existence predicate $E!$ of Free Logic for this purpose in their informal examination [12], and an uninterpreted predicate `Ex1` in Prover9 [14]. We also introduce an uninterpreted predicate for this purpose, but name it `really_exists`. We use this to formalize the part of the argument that states that if `the(greatest)` does not `really_exist`, then there is a greater thing (intuitively, something that does `really_exist`). Oppenheimer and Zalta state this directly as their *Premise 2*, which would be rendered in PVS as follows.

```
                                                          Alternative
  Premise_2: AXIOM (NOT really_exists(x)) => EXISTS y: (y > x)
```

However, for reasons that are explained in Section 4, we prefer to use a stronger premise, which we break into two parts: one axiom that asserts there is some `being` that `really_exists`, and another that asserts that `beings` that `really_exist` are `>` than those that do not (this is essentially the formulation used by Barnes [2]). With these axioms, we can prove the conclusion of the Argument, namely, that `the(greatest) really_exists`

```
someone: AXIOM EXISTS x: really_exists(x)

reality_trumps: AXIOM (really_exists(x) AND NOT really_exists(y)) => x>y

God_exists: THEOREM really_exists(the(greatest))
```

The proof of this `THEOREM`, which incorporates steps 3 and 4 of the Argument in Figure 2, is just ten routine steps in PVS: cite the axioms, expand definitions, and use predicate subtypes.

Although PVS has now verified the theorem `God_exists`, we need to be sure all its TCCs and lemmas have been discharged. We can check this with the PVS Proof Chain Checker, which reports that the verification is, indeed, complete and generates the following list of dependencies.

```
                                                          Proof Chain
ontological.God_exists has been PROVED.

  The proof chain for God_exists is COMPLETE.

  God_exists depends on the following proved theorems:
    ontological.God_exists_TCC1
    ontological.P1a
    ontological.greaterp_TCC1

  God_exists depends on the following axioms:
    ontological.P1
    ontological.reality_trumps
    ontological.someone

  God_exists depends on the following definitions:
    ontological.greatest
    orders.trichotomous?
    sets.empty?
    sets.member
    sets.nonempty?
    sets.singleton?
```

Before we declare victory, there is one more item to attend to. We have used three axioms, and these could have introduced inconsistencies that render the verification nugatory. PVS guarantees the soundness of constructively defined specifications (i.e., those without axioms; technically, it guarantees "conservative extension"), so one way to verify consistency of specifications with axioms is to exhibit a constructively defined specification that provides a model for the axioms. This is mechanized in PVS using its facilities for theory interpretations as show below.

```
interpretation: THEORY
BEGIN

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

END interpretation
```

The `importing` clause supplies constructive interpretations to the uninterpreted types and constants of the `ontological` theory. Specifically, we interpret `beings` by the natural numbers (`nat` in PVS), the `>` ordering on `beings` by the `<` order on natural numbers (so `the(greatest)` is the number 0 in this interpretation), and `really_exists` by the predicate "less than 4." Typechecking a specification such as this causes PVS to generate a TCC for each axiom of the source theory that requires us to prove that its interpretation is a theorem of the interpreted theory.

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
```

```
                                                                    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 easy to prove, and their proof chains indicate no further dependencies.

We have now completed our formal verification of the Ontological Argument in PVS, apart from Step 5 of the informal presentation. We regard that step as a question of interpretation rather than verification and postpone it to the discussion in the following section.

4 Discussion

PVS has verified the theorem labeled `God_exists`, but does this purely formal demonstration really carry the import intuitively associated with that claim? Regulators deciding whether to certify an aircraft face similar challenges: does formal verification of the correctness of an item of software really substantiate or contribute to the claim for safety of the aircraft subsystem concerned? In both cases, there are three items that require further consideration before we make the leap from a formal demonstration to a conclusion about the real world: does the formally verified theorem really support the interpretation that corresponds to the real-world conclusion we wish to draw, are the premises and axioms used in the verification true in the real world, and does the whole formal construction truly represent the informal argument to which we are hoping to bring the benefit of mechanized analysis?

For the first of these, in the case of the Ontological Argument, we need to ask whether the PVS formula `really_exists(the(greatest))` truly represents the claim "God exists." There are two parts here: is the formal description `the(greatest)` equivalent to God, and does `really_exist` mean that this being really exists? Some treatments of the Ontological Argument begin by asserting that God *is* that than which there is no greater (a variant is that God *is* that being with all perfections) and the contribution of the Argument is to establish that this being really exists, but this just pushes the question whether `the(greatest)` is equivalent to God from interpretation of the conclusion to veracity of the premises.

Since my concern here is with logic and formal verification, rather than theology, I will merely aver that the formal term `the(greatest)` does seem to me to bear the interpretation "that than which there is none greater," and refer the reader to philosophical texts for discussion on whether that can further be interpreted as God (rather than, say, the Neo Platonic "One," or Spinoza's "God or Nature"). When certifying aircraft, we probably do not want to defer the question of interpretation to philosophers or theologians, but aircraft and their systems are indisputably real-world artifacts and the question then largely devolves to what we *know* about the relevant system and its environment (that is, its epistemology [21]). I will return to this topic later.

Next, we need to consider interpretation of the predicate `really_exists`. In the PVS specification this is an uninterpreted predicate and so it can carry any meaning, consistent with constraints that may be imposed by axioms. Here, the relevant axioms are `someone` and `reality_trumps`, which are fairly innocuous: they are certainly consistent with the intended interpretation, but surely do not characterize it strongly or uniquely.[3] Notice that if we were to impose additional axioms to constrain the interpretation of `really_exists`, our verification would be unaffected (unless they render the specification inconsistent), because nothing compels us to cite the additional axioms. Only if we *weaken* the axiomatization of `really_exists` so that `someone` and `reality_trumps` are no longer derivable will our verification (possibly) run into trouble—and we would not do this because these two properties are surely required to be true in the intended interpretation.

However, as we mentioned in Section 3, Oppenheimer and Zalta did actually use a weaker specification in their *Premise 2* (weaker in that it is implied by our axioms but not vice-versa). A rather shocking fact, which seems to have first been noticed by Garbacz [6], is that *Premise 2* renders the Argument circular! To see this, we can relabel the PVS rendition `Premise_2` as a `COROLLARY`, place it after the `THEOREM God_exists`, and then prove it using only `God_exists`, the definition of `greatest`, and trichotomy of `>`. We leave it as an exercise for the reader to prove that `God_exists` can also be proved from `Premise_2`.

Arguably, `Premise_2` more closely represents the original form of the Argument than our axioms `someone` and `reality_trumps`, so this circularity casts grave doubt on the merit of the Argument in its standard formulation. I do not believe my formulation is vulnerable to this source of doubt and, based on the considerations of the previous paragraphs, my opinion is that the PVS theorem `really_exists(the(greatest))` is consonant with its intuitive interpretation, but does not compel it.

Next, we need to consider whether the axioms employed in our PVS formalization are true in the intended interpretation. We have already considered `someone` and `reality_trumps` as they relate to `really_exists`, so it remains to consider

[3]On the other hand, they are not vacuous: perverse interpretations such as `really_exists` means "is colored blue" are surely blocked by `reality_trumps`.

reality_trumps as it relates to the relation >, the requirement that this relation is trichotomous, and the AXIOM P1.

It is, perhaps, surprising to note that the specification does not require > to be transitive, and so it is not an ordering relation in any standard sense (the weakest kind of ordering is a preorder, which is a relation that is both reflexive and transitive). If we temporarily ignore the trichotomy requirement, the set defined as greatest simply specifies the set of elements "than which there are none greater" with respect to a relation > that we are reading as "greater than" but that is uninterpreted and unconstrained. We could think of members of greatest as "gods" with respect to the > relation; this set of "gods" could be empty or nonempty. The axiom P1 specifies that it is nonempty, so we can think of it as asserting that there is at least one "god." The constraint that > is trichotomous then (implicitly) restricts the set to be a singleton, so that there is a unique "god" (which is therefore God). That is, P1 essentially asserts "there is a god" and trichotomy of > makes it unique. The axiom reality_trumps then (implicitly) asserts that this unique god really_exists. Thus, we see that the axioms and constraints of the PVS specification, although they seem innocuous, effectively encode the conclusion that we wish to draw (yet without being formally circular). In a sense, this must be true of any deductive demonstration, because deduction does not create new knowledge—it merely reveals what is implicit in the assumptions—but the directness (i.e., "near circularity") with which this applies to the Ontological Argument may come as a surprise.

Many of the classical objections to the Ontological Argument center on the properties and interpretation ascribed to the > relation. One of the earliest objections was raised by Anselm's contemporary Gaunilo, who used the strategy of Anselm's argument to deduce the (absurd) existence of "the most perfect island." That is, he interpreted beings as islands, and > as "more perfect than." Gaunilo's objection can be refuted by noting that P1 is surely false for his interpretation of >: there is no reason to think there are any maximally perfect islands, for we can always add one more palm tree.

Since Gaunilo's objection is blocked by P1, a plausible response would be to deny that P1 is an acceptable premise. I am sympathetic to this objection because, as we saw in the paragraphs above, P1 comes very close to assuming what we want to prove.

The next target for objection is trichotomy of >: this says that for any two (distinct) beings, one is > than the other. One objection is that some great-making attributes may be mutually incompatible [7]: examples are beings that are "perfectly just" and "perfectly merciful" (the first entails delivering exactly the "right amount" of punishment, while the latter may deliver less than is deserved). I believe there are really two objections here. The first is that it is surely difficult to rank justice and mercy: which is > than the other, a just being or a merciful one? This actually is not a problem; the terminology “greater than” may seduce us into thinking that > must be antisymmetric (which is true of any ordering stronger than a preorder), but this constraint is not present in the PVS formalization: it is perfectly acceptable to have both mercy > justice and justice > mercy (trichotomy requires at least one of these, but it does not prohibit both, and having both does not imply equality). The second objection is that a truly great being must surely be *both* just and merciful, and these are incompatible. This may be a problem for theologians, but is entirely independent of our formulation of the Ontological Argument and therefore not a strong challenge to it.

To summarize this discussion: the axioms and constraints used in the PVS formalization of the Ontological Argument seem consistent with the intended interpretation, as does its conclusion, but they do not compel that interpretation. In fact, the model that we used to demonstrate consistency of the axioms provides an interpretation quite different than that intended by Anselm. It seems to me that this weakens our confidence that the Argument has much value, but it is important to note that similar observations about a formal verification used in support of safety claims for an assurance case would not pose the same challenge to its value. The reason is that in formally verifying properties of systems, our axioms and constraints specify a model of the system and its environment, and our verified conclusion states some property of the model. If we can satisfy ourselves that the axioms and constraints are true for the real system, and that the conclusion does correspond to a property we care about in the real system, then we have accomplished something useful. It does not matter that there may be other interpretations for our axioms and constraints, and that our verified conclusion may carry a different import under those interpretations; all we need to care about is the fidelity of the modeling to the system under examination. The Ontological Argument is different, and in this sense is not a good paradigm for assurance or safety cases, because its very purpose is to compel belief in one interpretation.[4]

Thus, although we have formally verified the Ontological Argument, our examination of its formal premises and conclusion raise doubts about its value: the argument is close to circular (and, indeed, *is* circular in a closely related formalization that is arguably closer to the original), and it does not compel belief in the intended interpretation.

One response would be to challenge the entire basis for our formalization. For example, there is much discussion in the philosophical literature about the way

[4]It can be argued that in a safety case, we do wish to compel one interpretation for the term “safety.” Each theorem in a safety case will verify some property and we need to satisfy ourselves that the interpretation of each verified property is a contribution to safety, as intuitively understood. That seems fairly easy. The difficulty is to persuade ourselves that the conjunction of all the verified properties gives a “complete” account of safety for the system concerned; that is, that it compels the intended interpretation of safety.

"existence" (which we represent as `really_exists`) is used in the Ontological Argument. In particular, Kant denied that existence is a predicate. Those who use Free Logics would refute this, and there is much recent work on representing statements about fictional characters in formal logic, but there is opportunity to challenge our verification (and the Argument itself) on these grounds.

Anselm actually gave a second version of his Argument that used *necessary*, rather than simple, existence and this version has been studied, by Gödel among others, using modal logic to represent necessity. It is straightforward to embed modal logics within PVS ([19] does this for propositional linear temporal logic), so a modest exercise for motivated readers is to represent Gödel's form of the Ontological Argument in PVS.[5]

Another objection could be that the PVS formalization dispenses with the predicate `C` of Oppenheimer and Zalta's formulation; they use this to represent something being "in the understanding" as this expression is used the original form of the argument. The issue is that understanding the definite description may not be the same as conceiving that there is an object that exemplifies the property. In a later paper Oppenheimer and Zalta [13] use a much more elaborate treatment that distinguishes encoding and exemplifying a property. They propose a weakened form of their *Premise 1*, from which only a weakened form of the conclusion can be derived.

5 Conclusion

We have presented a formalization and verification of the Ontological Argument in PVS; along the way, we encountered definite descriptions and choice functions, and saw how PVS maintains soundness for these in the presence of possibly empty types. We established that our rendition of the argument is verified by PVS, and that the axioms we employed are consistent, and we considered their plausibility and the interpretation of the conclusion. We saw that the argument is very close to circular and that slight variants truly are circular, and we also saw that although the formal conclusion is consistent with the intended interpretation, it does not compel it.

These issues are all germane to any formal verification undertaken for the purpose of assurance in some larger context, but the Ontological Argument illustrates them in a particularly vivid manner that makes it a suitable and "fun" example for pedagogical purposes.

In the context of philosophy, we note that there are several university departments that espouse "computational philosophy" as a research area or degree subject (e.g., Pavia, Oxford), and there are conference series on the topic (e.g., the "AISB Symposium on Computing and Philosophy"), and there is also a rather more radical

[5]In August 2013, Benzmüller and Woltzenlogel-Paleo announced that they have formalized and verified Gödel's version in several automated verification systems [3]. They use a quantified modal logic, which requires a more complex embedding than the PVS embedding of propositional LTL.

proposal for a "computational metaphysics" [5]. Since it builds so directly on the prior work of Oppenheimer and Zalta, the present paper makes only a minor contribution to these endeavors (though it does illustrate why they should prefer to use a full verification system rather than a simple first order prover), but I hope that it provides a basis that others might extend to formal analyses of more philosophical interest. For example, there are numerous semi-formal presentations of the Ontological Argument that all differ slightly from each other (e.g., several are given in [15], and also see [18,23]) and it could be interesting to reformulate them in uniform mechanically checked renditions to see whether the differences are of any significance. In addition, there are formulations, such as those of Gödel and Plantinga, that use modal logic to derive the *necessary* existence of God (recall the footnote on page 17).

There are several classical objections to the Ontological Argument, of which I have mentioned only Gaunilo's. A recent paper by Millican [8] describes eight others and presents a reformulation that is claimed to refute all except Gaunilo's, and to reveal the one true and novel "fatal flaw" in the Argument. There are several ripostes, and counter-ripostes to this (e.g., [9,10,16]) and it could be interesting to formalize and verify these arguments and reformulations.

Finally, we could turn to the mechanically assisted investigation other metaphysical topics that have a strong logical component. Avicenna's proof of the "Necessary Existent" would be an interesting candidate for mechanized verification: this is older than Anselm's Ontological Argument and in some ways more interesting: it seems less of a logical "trick" and closer to what some would regard as a true source of belief [1].

Acknowledgments

I am grateful to Sam Owre (who, along with Shankar, is its main developer) for discussions on several finer points of the logic and implementation of PVS, and to Ed Zalta for showing that modern tools can be applied to old problems and for his stimulating and candid discussion of some of the issues in the formalization by him and Oppenheimer.

References

[1] Peter Adamson. *By All Means Necessary: Avicenna on God.* History of Philosophy without any gaps, Podcast, 2013. `http://www.historyofphilosophy.net/avicenna-god`.

[2] Jonathan Barnes. *The Ontological Argument.* Macmillan, London, UK, 1972.

[3] Christoph Benzmüller and Bruno Woltzenlogel-Paleo. *Formalization, Mechanization and Automation of Gödel's Proof of God's Existence.* arXiv.org, 2013. `http://arxiv.org/abs/1308.4526` and `https://github.com/FormalTheology/GoedelGod`.

[4] Peter Bishop, Robin Bloomfield, and Sofia Guerra. The future of goal-based assurance cases. In *DSN Workshop on Assurance Cases: Best Practices, Possible Obstacles, and Future Opportunities*, Florence, Italy, July 2004.

[5] Branden Fitelson and Edward N. Zalta. Steps toward a computational metaphysics. *Journal of Philosophical Logic*, 36(2):227–247, 2007.

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

[7] Kenneth Einar Himma. Ontological argument. In James Fieser and Bradley Dowden, editors, *Internet Encyclopedia of Philosophy.* April 2005.

[8] Peter Millican. The one fatal flaw in Anselm's argument. *Mind*, 2004. 437–476.

[9] Peter Millican. Ontological arguments and the superiority of existence: Reply to Nagasawa. *Mind*, 116(464):1041–1054, 2007.

[10] Yujin Nagasawa. Millican on the Ontological Argument. *Mind*, 116(464):1027–1040, 2007.

[11] John Nolt. Free logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Spring 2011 edition.

[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.

[13] Paul E. Oppenheimer and Edward N. Zalta. Reflections on the logic of the Ontological Argument. *Studia Neoartistotelica*, 4(1):28–35, 2007.

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

[15] Graham Oppy. Ontological arguments. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Winter 2012 edition.

[16] Graham Oppy. More than one flaw: Reply to Millican. *Sophia*, 46(3):295–304, 2007.

[17] Sam Owre, John Rushby, Natarajan Shankar, and Friedrich von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995. PVS home page: `http://pvs.csl.sri.com`.

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

[19] John Rushby. Formal verification of McMillan's compositional assume-guarantee rule. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, September 2001.

[20] John Rushby. New challenges in certification for aircraft software. In Sanjoy Baruah and Sebastian Fischmeister, editors, *Proceedings of the Ninth ACM International Conference On Embedded Software: EMSOFT*, pages 211–218, Association for Computing Machinery, Taipei, Taiwan, 2011.

[21] John Rushby. Logic and epistemology in safety cases. In SafeComp *2013: Proceedings of the 32nd International Conference on Computer Safety, Reliability, and Security*, number 8153 in Lecture Notes in Computer Science, pages 1–7, Springer-Verlag, Toulouse, France, September 2013.

[22] 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.

[23] Jordan Howard Sobel. *Logic and Theism: Arguments for and Against Beliefs in God.* Cambridge, 2003.

Appendix: The complete PVS Specification

The PVS specification and proof are available for download (as a PVS `dump` file)) at `http://www.csl.sri.com/users/rushby/abstracts/fwfm13`.

```
ontological: THEORY
BEGIN

beings: TYPE

x, y: VAR beings

>: (trichotomous?[beings])

greatest: SETOF[beings] = { x | NOT EXISTS y: y>x }

P1: AXIOM nonempty?(greatest)

P1a: LEMMA singleton?(greatest)

really_exists(x): bool

someone: AXIOM EXISTS x: really_exists(x)

reality_trumps: AXIOM (really_exists(x) AND NOT really_exists(y)) => x>y

God_exists: THEOREM really_exists(the(greatest))

P2: COROLLARY (NOT really_exists(x)) => EXISTS y: (y > x)

END ontological


interpretation: THEORY
BEGIN

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

END interpretation
```