CAV Workshop "Fun With Formal Methods," St Petersburg, Russia, 13 July 2013
based on Crazy Ideas talk, 9 Nov 2012

The Ontological Argument In PVS

What Does This Really Prove?

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

PVS Proves The Existence Of God!

- The Ontological Argument is an 11th Century proof of the existence of God
- Almost everyone finds this topic interesting
- Believers and unbelievers alike
  - Many of those who studied and criticized the Argument were devout believers
  - Can something as ineffable as the existence of God can be subject to a mere logical demonstration?
- The proof raises quite deep issues in logic
  - Is the proof logically correct?
- And in the interpretation of logical proofs
  - What does this really prove?
- Just like formal methods in support of Safety Cases
- So I think it is a Fun way to introduce these topics

Classical Arguments for Existence of God

Teleological: argument from design

This is an empirical or *a posteriori* argument: it builds on empirical observations about the world

Hence is vulnerable to better understanding of empiricism, better observations, better explanations

- Hume, Darwin etc.

Cosmological: there must be a first (uncaused) cause

Or why is there something rather than nothing?

Also empirical, but less reliant on specifics

But depends on notion of cause

- Leibniz, Hume, Kant; current popularization: Holt

Ontological: next slide

This is a rational or *a priori* argument: it doesn't depend on observation

The Ontological Argument (St. Anselm, 11th C)

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 even 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.

The Ontological Argument: Modern Reading

- We can conceive of something than which there is no greater
- If that thing does not exist in reality, then we can conceive of a greater thing—namely, something that does exist in reality
- Therefore either the greatest thing exists in reality or it is not the greatest thing
- Therefore the greatest thing necessarily exists in reality
- That's God
  - Why it's the Christian God is another matter
  - Seems more like the Neo-Platonist "One"
  - Or Spinoza's "God or Nature"

Status of The Ontological Argument

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

Logic of the Ontological Argument

- Anselm himself gave two variants of the Argument
- The second asserts not the mere possibility that a maximally great something exists, but that it necessarily exists
- So several modern treatments use modal logics
  - Gödel, Plantinga
- Oppenheimer and Zalta make a good case that the basic argument can/should be interpreted in classical logic, but we need to be careful about existence

Existence

Two issues:

Existence in reality: this is not the same as $\exists$, which although it is pronounced "there exists" refers to an implicit domain of quantification and does not assert existence in reality (think "not $\forall$ not")

Quantifiers ranging over possibly nonexistent objects: can lead to unsoundness in first order logic

Oppenheimer and Zalta use Free Logic, which has an explicit existence predicate ($E!$) and adjusts the quantifier rules

Logic of the Ontological Argument (ctd.)

- The argument uses a definite description
  - The x such that some property ϕ: $\iota x \phi$
  - Here, "that (i.e., the x) than which there is no greater"
- These are tricky
  - "The present King of France is bald"
    - ⋆ Note, for those who learn about the world from CNN or the WSJ: France is a republic, it has no present king
  - Is this true, false, inadmissible?
  - If the former, its negation should be false
  - What is its negation?
- Related to the existence problem
  - Must not substitute definite descriptions into quantified expressions without being sure they are well defined

Oppenheimer and Zalta's Treatment

- Careful treatment in unmechanized Free Logic, 1991
- The treatment was later mechanized in Prover9, 2011
- Claimed that Prover9 discovered a much simpler proof
  - Prover9 uses classical First Order Logic
  - Not a Free Logic, lacks definite descriptions
  - So there's manual reformulation
  - Garbacz argues that is unsound
- I'll do it in PVS
  - A higher order logic
    - ⋆ With dependent typing and predicate subtypes
  - Provides sound and mechanically enforced treatment of existence and quantification, definite descriptions, and much else

Overview

- I'll first introduce PVS's treatment of definite descriptions
- Then do the Ontological Argument
- Then discuss the axioms, assumptions required
  - Is it a sound argument?
- Then some comparison with Oppenheimer and Zalta
- Finally, a crazy idea

Russell's Treatment of Definite Descriptions

- The present King of France is bald is 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.
- The sentence is false, because the first conjunct is false
- "The present King of France is not bald" also is false
- Rather contextual reading, we'd like an interpretation for The present king of France standing alone: e.g., $\iota x : \phi(x)$
- Can then say $bald(\iota x : \phi(x))$
- i.e., want to write $\iota x : \phi(x)$, where $\phi(x)$ is some predicate, subject to first two conditions (must exist, must be unique)
- How to enforce this requirement?

Definite Descriptions in PVS

- PVS is a higher-order logic
  - Functions can take functions as arguments, return them as values
  - Can quantify over functions
- Higher-order logics require types for consistency
- PVS extends simple type theory with predicate subtypes (and dependent types and structural subtypes)
- Typechecking in PVS is undecidable (i.e., requires theorem proving)
- But the circumstances that require theorem proving are very constrained, most typechecking is algorithmic
- When necessary, typechecker attaches proof obligations called Typecheck Correctness Conditions (TCCs) to specifications
- Analysis is not complete until all TCCs have been proved

Empty Types, and Sets in PVS

- PVS keeps track whether types are known to be empty or not
- If a type that may be empty is used in a context that requires a nonempty type, a TCC will be generated to force its nonemptiness to be proved
- Sets and predicates are the same in higher-order logic, and both are simply functions with range type Boolean (written `bool` in PVS)
- Easy to specify higher-order predicates `empty?`, `nonempty?`, and `singleton?` that indicate whether their set argument is empty or not, or is a singleton
- By convention, predicates often have names in ending in `?`
- A predicate name enclosed in parentheses denotes the corresponding subtype of the parent type
  - e.g., `x:  VAR (nonempty?[nat])`

Sets in PVS

```
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)

END Russell
```

Definite Descriptions in PVS

- We define a function `the`, that takes a singleton set as its argument and returns a value of that subtype

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

- Note, this is not a definition (there is no `=`)
- It just asserts the existence of a function with the given type
- So PVS generates a TCC to ensure this type in not empty

TCC

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

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

- Seems easy to prove: we know the argument is a singleton, just return its member, or any member
- Difficulty is constructing a name for that member

Choice Functions in PVS

- 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)
```

- Same as `the`, except domain merely needs to be `nonempty?`
- Given this, can discharge `the_TCC1` as follows

Proof Script

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

- The first of these instantiates the variable `x` in the TCC
- The second invokes one of PVS's more powerful general-purpose proof strategies

Choice Functions in PVS (ctd. 1)

- However, the invocation of `choose` introduces a TCC of its own to prove that `A` is nonempty

TCC

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

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

- Same difficulty as the TCC for `the`: finding a name for the function that provides an existential witness that this function type is nonempty
- Solve it by regress to a more primitive kind of choice function
- Hilbert defined a function ε (`epsilon` in PVS) that is a choice function for general (i.e., possibly empty) sets
- If its set argument is nonempty, returns a member of that set
- Otherwise, returns arbitrary value of the base type for the set

Choice Functions in PVS (ctd. 2)

- So the base type must be nonempty
- Ensure this by defining `epsilon` within a theory whose parameter is required to be nonempty

```
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
```

- Whenever `epsilons` theory is used, a TCC will be generated if necessary to establish nonemptiness of the instantiation for its type parameter

Choice Functions in PVS (ctd. 3)

- 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))`
    - Due to the range type specified for `choose`

- The second line instructs the prover to rewrite with `epsilon_ax[T]`, followed by another `grind` to clean up

Whew!

- Have succeeded in specifying definite descriptions in PVS as the function `the`
  - And have discharged all its attendant TCCs
  - Along the way, also defined the independently useful choice functions `choose` and `epsilon`
- Might seem a lot of work before we even get to the Argument
- In fact, all this is part of the PVS "Prelude"
  - Standard library built into the system
  - The `epsilons` theory supplied in the Prelude
  - The definitions we presented in theory `Russell` actually just part of a Prelude theory called `sets`
- Large tracts of logic are defined in the Prelude
- Many other branches of mathematics are formalized in other PVS libraries available from `http://pvs.csl.sri.com`

Now On To The Ontological Argument

- We can conceive of something than which there is no greater
- So we seem to need a type of things, or `beings`
- And some ordering `>` on them
- And then want `the` being that is maximal under this ordering
- We'll define `greatest` as the set of all beings that are maximal
- Then find conditions to ensure it is a singleton, and hence `the(greatest)` will be well-defined
- Surprisingly, Oppenheimer and Zalta discovered `>` doesn't need to be a true ordering, just needs what they called connectedness

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

- This is normally called trichotomy and is defined in the PVS prelude

Greatest

```
ontological: THEORY
BEGIN

  beings: TYPE

  x, y: VAR beings

  >: (trichotomous?[beings])

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

END ontological
```

TCCs

- Get TCC to ensure type asserted for constant > is nonempty

TCC

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

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

- Easily discharged by exhibiting the relation that relates everything to everything

Proof Script

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

- Next, want to specify we "can conceive of" "the greatest"
- Oppenheimer and Zalta introduce a predicate C to represent "can conceive of" but this seems unnecessary: so I omit it
- "The greatest" is `the(greatest)` in PVS
- PVS will generate TCC to prove `greatest` is a singleton
- Need additional constraint to make this so

Premise 1

- Oppenheimer and Zalta use a premise that asserts existence of maximal elements

Alternative

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

- Seems more direct to simply require `greatest` is a singleton
- But because of trichotomy, all we need is nonemptiness

continuation

```
P1: AXIOM nonempty?(greatest)

P1a: LEMMA singleton?(greatest)

the_greatest: beings = the(greatest)
```

- `P1a` is easily proved, and discharges the TCC from `the(greatest)`

Premise 2

- Next part of the argument states that if `the(greatest)` does not exist in reality, then there is a greater thing
  - Intuitively, something that does exist in reality
- O&Z use the $E!$ of Free Logic for "exists in reality"
- We'll use uninterpreted predicate `really_exists`
- Oppenheimer and Zalta formalize this step 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 later, I prefer to use a stronger premise, which I break into two parts
  - One axiom asserts there is some `being` that `really_exists`
  - Another asserts that `beings` that `really_exist` are `>` than those that do not

The Conclusion

- Can then prove the conclusion of the Argument
- Namely, that the(greatest) really_exists

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

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

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

- Proof is just ten routine steps in PVS: cite the axioms, expand definitions, and use predicate subtypes

Done!

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?
```

Not Quite!

- We have used three axioms and these could have introduced inconsistency
- PVS guarantees conservative extension for purely constructive specifications
- So one way to establish consistency of axioms is to exhibit a constructively defined model
- Can do this using PVS capabilities for theory interpretations
  - Interpret `beings` by the natural numbers `nat`
  - And > by < (so `the(greatest)` is 0)
  - And `really_exists` by "less than 4"
- PVS generates TCCs to prove that the axioms of the source theory are theorems under the interpretation

The Model

```
                                                      model
interpretation: THEORY
BEGIN

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

END interpretation
```

Proof Obligations for Consistency

TCCs

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

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

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

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

...continued
```

Proof Obligations for Consistency (ctd.)

TCCs

```
...continuation

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

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

- These are all easily proved
- So, our formalization of the Ontological Argument is sound
- And the conclusion is valid
- But what does it really mean?

Assurance Cases and Formal Verification

- An assurance case provides an argument to substantiate some claims (often concerning safety) based on evidence (about a system)
- This is like logic: formal verification provides mechanically checked proofs to verify conclusions based on premises
- So what's the difference?
- An assurance case can use formal verification
    - But pays attention to credibility of the premises and the interpretation of the conclusion
- The Ontological Argument is a paradigm example
    - The verification shows that it is valid
    - But does the theorem mean what we think it means?
    - And are the premises credible?
- I'll start with the premises I used cf. those of O&Z

Comparison with Oppenheimer and Zalta

- I drop "can conceive of": I don't think this matters
- My `P1` is equivalent to their `Premise_1`
    - Can prove each from the other
- My `someone` and `reality_trumps` are stronger than their `Premise_2`: former can prove the latter but not vice-versa
- Their `Premise_2` renders the proof circular!
    - Can prove `Premise_2` from `God_exists` and vice-versa
- Seems to have first been noted by Garbacz
- Arguably, `Premise_2` is closer to Anselm's original!

Oppenheimer and Zalta's Simplification

- O&Z formalized the Argument using the Prover9 first-order theorem prover
  - No first-order theorem prover automates Free Logic
  - Nor provides definite descriptions

  So these delicate issues are dealt with informally outside the system, and beyond the reach of automated checking

- Deductions performed by Prover9 actually used very little of their formalization

- This led them a much reduced formalization that Prover9 still found adequate

Oppenheimer and Zalta's Simplification (ctd.)

- Believed they had discovered a simplification to the Argument that
  - "not only brings out the beauty of the logic inherent in the argument, but also clearly shows how it constitutes an early example of a 'diagonal argument' used to establish a positive conclusion rather than a paradox"
- Garbacz disputes this
  - The simplifications flow from introduction of a constant (God) that is defined by a definite description
  - In the absence of definedness checks, this asserts existence of the definite description and bypasses the premises otherwise needed to establish that fact
- Lesson: first-order logic was designed for study, not for use

Premise 1 and Gaunilo's Objection

- Gaunilo was a contemporary of Anselm who used the strategy of Anselm's argument to deduce the (absurd) existence of "the most perfect island"
- `P1` is surely false for his interpretation
  - We can always add one more palm tree
- So `P1` blocks this objection, but is `P1` acceptable?
- Can think of the members of `greatest` as "gods"
  - Could be zero, none, many
- `P1` says there is at least one:
  - Equivalent to asserting "there is a god"
- Trichotomy of `>` then says ensures there is exactly one
- So these constraints are very close to asserting what we want to prove

Other Issues With >

- Some great-making properties are incompatible
  - e.g., being "perfectly just" and "perfectly merciful"
  - Exactly the "right amount" of punishment, vs. less than deserved
- Which is > the other?
- Not a problem: > is merely trichotomous
  - It is not an ordering relation in the usual sense
  - Can have both `just > merciful` and `merciful > just`
- A truly great being must surely be both just and merciful, and these are incompatible
  - A problem for theologians
  - But entirely independent of the Ontological Argument and therefore not a strong challenge to it

Intended Interpretation

- The constructive model provides a different interpretation than that intended by Anselm
- So, although `the(greatest)` and `really_exists` seem compatible with the intended interpretation
- They do not compel it
- In an assurance case we would not care
  - Provided the premises are true of our system
  - And the conclusion says something useful

  It does not matter if there are other interpretations
- But here, the goal is to compel the intended interpretation
- OTOH, surely do have an intended interpretation for safety

Conclusions

- We have formalized the Ontological Argument
- And verified its conclusion
- So the Argument is sound!
- But it is very close to circular
  - And slight variants are circular
- And it does not compel the intended interpretation
- I think it is a Fun example to introduce students to
  - Subtle issues in logic and mechanization
  - The interpretation and utility of formally verified claims

A Crazy Idea: Computational Philosophy

- Fitelson and Zalta propose "computational metaphysics"
  - Code stuff up in a mechanized logic
  - Let the automation rip
  - Examine the result for insights
- I think this is reasonable, but too modest
- A lot of philosophy is implicitly based on an anthropomorphic interpretation of knowledge, learning, deduction, language, communication, etc.
- As computer scientists we have a unique grasp of computational interpretations of these
  - From AI, robotics, machine learning, etc.
  - Cf. Searle's Chinese Room: he just doesn't get it
- I think this creates a potential for new insights on traditional philosophical questions

Some Suggested Reading

- Oppenheimer and Zalta's papers: just Google for them
- 36 Arguments for the Existence of God: A Work of Fiction by Rebecca Goldstein
- Types, Tableaus, and Gödel's God (Trends in Logic) by Mel Fitting
- Why Does The World Exist? An Existential Detective Story By Jim Holt
  - See also Freeman Dyson's review in NY Review of Books

And Homework

- Reconstruct Gödel's or Plantinga's proofs in PVS
  - Will need to embed a modal logic (which one?) in PVS
  - Embedding of LTL (S4) could serve as a model

  Hot news! Benzüller and Woltzenlogel-Paleo have done this (in Isabelle and Coq)

- Try to formalize and verify Avicenna's proof of the "Necessary Existent"
  - Older than the Ontological Argument
  - And arguably less of a logical "trick" and closer (for some) to the true source of belief