Marktoberdorf NATO Summer School 2016, Lecture 3

Proofs and Assurance
The Case of The Ontological Argument

John Rushby

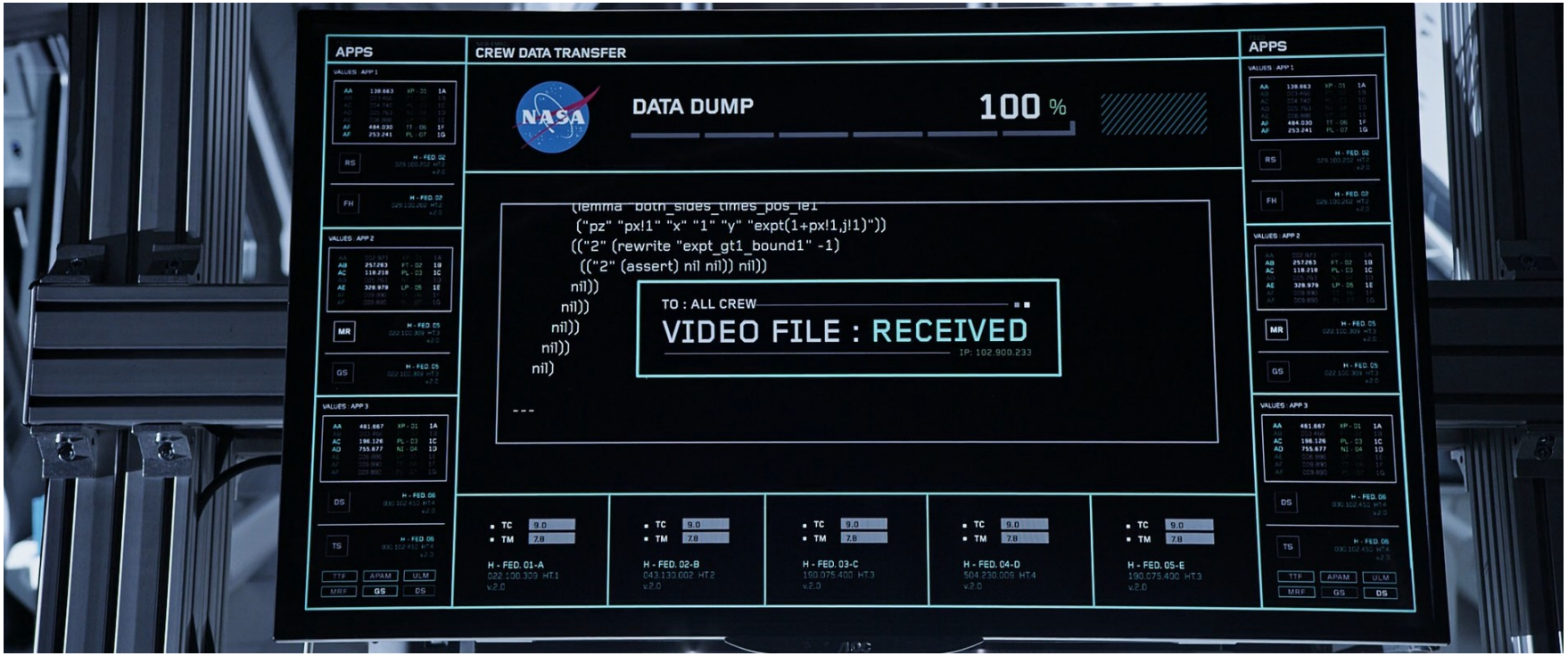
Computer Science Laboratory
SRI International
Menlo Park, California, USA

Overview

- We've seen model checking, synthesis, automated verification
  - Let's take a look at interactive theorem proving

- Formal methods establish that one string of symbols entails another string of symbols
  - We attach an interpretation to those strings and draw real-world conclusions
  - But how confident can we be in that connection?

Movie "The Martian" at 58 Minutes

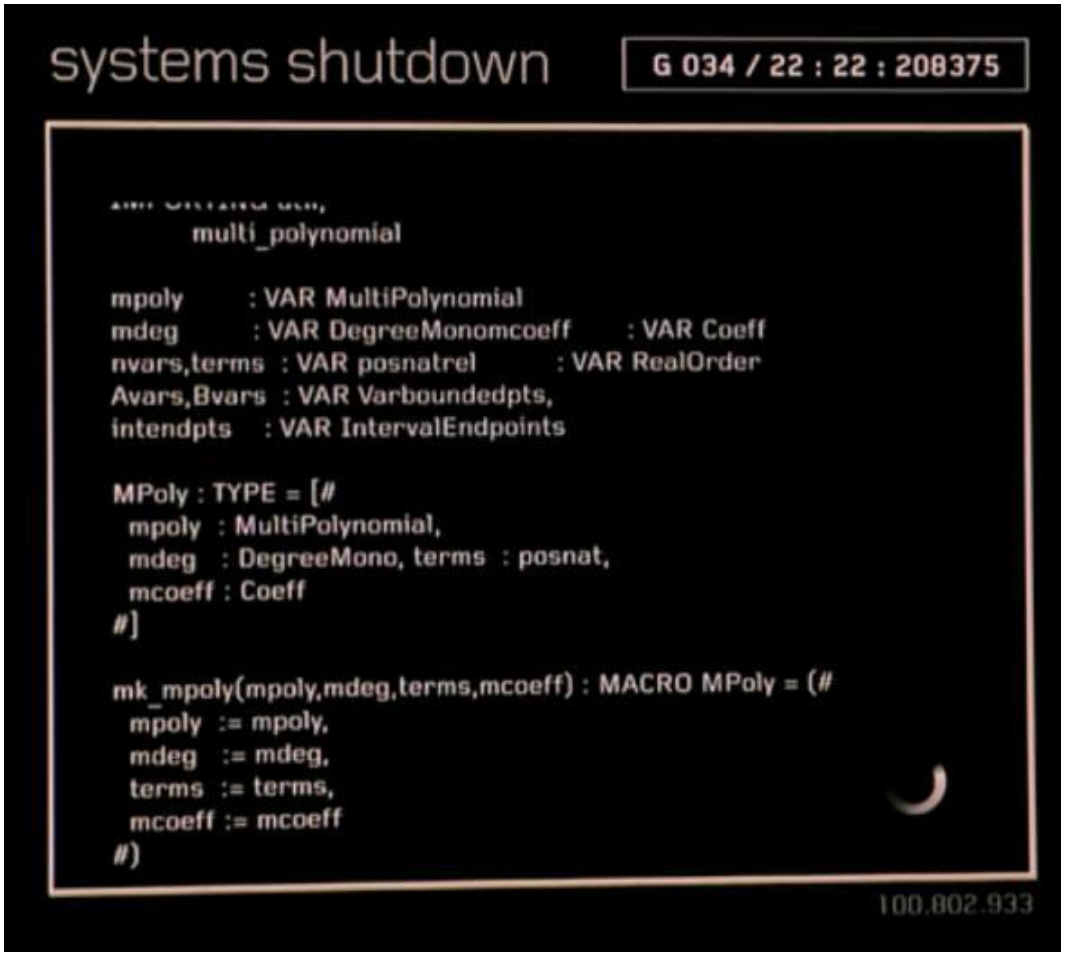


- Much speculation online what language is in the code snippet
- It's actually part of a PVS proof script for the theorem

$$\forall(n : \texttt{nat}, x : \texttt{posreal}) : 1 + n \times x \leq (1 + x)^n$$

by David Lester of Manchester U

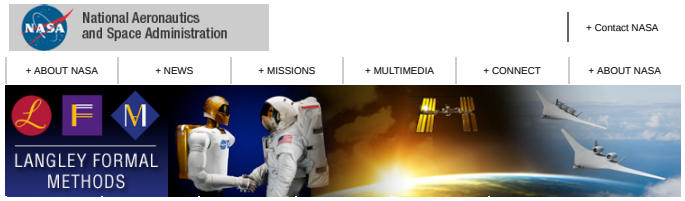
Movie "The Martian" at 1h:39m:03s and 1h48m53s



This is part of a PVS specification for a data structure representing multivariate polynomials, written by Anthony Narkawicz and César Muñoz of NASA

There's A Web Page About It

Written by César Muñoz of NASA



NASA PVS LIBRARY FEATURED IN THE MOVIE "THE MARTIAN"

Since the release of the science fiction film The Martian in 2015, movie fans have been speculating in internet forums about the source code that is displayed in computer screens in some scenes of the movie. Some fans have jokingly guessed "alien technology", others claimed to be "gibberish", and the most informed have noticed similarities with programming languages such as as Lisp, Prolog, and, even, Pascal.

Closest to the truth, the Internet Movie Database (IMDb) explains:

> Whenever Mark boots up a computer (ie. when finding the MAV) a sequence of source code is seen appearing on a screen. The code is written in PVS (Prototype Verification System), an experimental macro language which NASA actually uses and it's very plausible to appear on a future spacecraft. This particular chunk of code is from the already existing NASA PVS Library, and actually you can find that very piece of code as open source if you type a part of it into Google.

Indeed, the source code seen in the movie The Martian is written in PVS, a verification system developed by SRI International. It is also true that this particular code is part of the NASA PVS Library, a collection of formal theories developed and maintained by the Formal Methods Group of the Safety-Critical Avionics Systems Branch at NASA Langley Research Center. However, PVS is neither a programming language, nor a "macro language". PVS is a proof assistant. It consists of a specification language, i.e., a formal notation for defining mathematical objects and their properties, and an interactive theorem prover for verifying these properties using deductive rules. Both PVS specifications and proofs are displayed in the movie.

Code from the NASA PVS Library appears three times in the movie. In every instance, the code is unrelated to the movie's implied functionality. At 58 minutes, the following snippet of code from the file `power/exponentiation_aux.prf` is shown in a computer screen in the Hermes spacecraft.

```
("" (induct "n")
 (("1" (expand "expt") (("1" (propax) nil nil)) nil)
  ("2" (skosimp*)
   (("2" (expand "expt" 1)
     (("2" (inst - "px!1")
       (("2"
         (lemma "both_sides_times_pos_le1"
          ("pz" "px!1" "x" "1" "y" "expt(1+px!1,j!1)"))
         (("2" (rewrite "expt_gt1_bound1" -1)
           (("2" (assert) nil nil)) nil))
          nil))
        nil))
      nil))
    nil))
 nil)
```

It is implied in the movie that this text encodes video data. In reality, this text is the PVS internal representation of a proof of the the mathematical statement $\forall n, x\colon x > 0 \Rightarrow 1 + nx \leq (1 + x)^n$, where n is a natural number and x is a positive real number. Internally, PVS uses s-expressions to represent proofs. PVS is implemented in Lisp and s-expressions are often used in Lisp programs to represent data. These s-expressions are constructed by PVS from proof commands entered by the user such as `(induct "n")`, `(expand "expt")`, etc. This s-expressions reflects the fact that the proof, which was written by Prof. David Lester (U. of Manchester, UK), proceeds by induction on n.

The second and third appearances of the NASA PVS Library occur at times 1h:39m:03s and 1h48m53s, respectively, when the following code from

Bernstein/MPoly.pvs is displayed in a computer screen.

```
  mpoly        : VAR MultiPolynomial
  mdeg         : VAR DegreeMono
  mcoeff       : VAR Coeff
  nvars,terms  : VAR posnat
  rel          : VAR RealOrder
  Avars,Bvars  : VAR Vars
  boundedpts,
  intendpts    : VAR IntervalEndpoints

  MPoly : TYPE = [#
    mpoly  : MultiPolynomial,
    mdeg   : DegreeMono,
    terms  : posnat,
    mcoeff : Coeff
  #]

  mk_mpoly(mpoly,mdeg,terms,mcoeff) : MACRO MPoly = (#
    mpoly  := mpoly,
    mdeg   := mdeg,
    terms  := terms,
    mcoeff := mcoeff
  #)
```

The movie implies that this code is somehow related to the shutdown and startup scripts of the Mars Habitat (Hab) and the Mars Ascending Vehicle (MAV), respectively. Indeed, this code is a PVS specification of a data structure for representing multivariate polynomials. The code, which was written by Anthony Narkawicz (NASA) and César Muñoz (NASA), is part of a PVS formalization of a method for approximating the minimum and maximum values of a multivariate polynomial using Bernstein polynomial basis.

What about the claim by the Internet Movie Database (IMDb) that PVS code will appear in future spacecraft? Unlikely. As explained here, PVS is not a programming language but a proof assistant. Unless astronauts would like to kill the tedium in a long interstellar voyage by proving theorems, PVS won't be installed in future spacecraft computers. However, it is possible that computer programs, whose safety-critical algorithms have been formally verified in PVS, would appear in future aerospace systems. That is the case of separation assurance systems for air traffic management such as ACCoRD and detect and avoid systems for unmanned aircraft systems such as DAIDALUS.

The tag identifies links that are outside the NASA domain

NASA Official: César Muñoz
+ Contact NASA Langley
+ Contact NASA
Last Updated: January 14, 2013

So what is PVS?

Verification Systems/Proof Assistants

- There are several of these
- Unquantified First Order
  - ACL2 (USA)
- Higher Order
  - Coq (France)
  - HOL (UK)
  - Isabelle (Germany)
  - PVS (USA)
- Only PVS will get you home from Mars!

PVS

- I'm going to use PVS from SRI
- First released 1993
- Classical Higher-Order Logic with predicate subtypes
- One of the first provers with powerful decision procedures
  - Modern SMT solvers (ICS) evolved from these
  - But its quantifier reasoning is weak
- Winner of CAV Award 2015
- 3,000 citations

Next, PVS Proves The Existence Of God!

- The Ontological Argument is an 11th Century proof of the existence of God due to St. Anselm, Archbishop of Canterbury
- Can it really be true? Is it convincing?
- Almost everyone finds this topic interesting
- Believers and unbelievers alike
  - This is not about atheism: many of those who studied and criticized the Argument were devout believers
  - Can something as ineffable as the existence of God 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 actually mean? What does it really prove?
- Just like formal methods in support of Assurance Cases

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* (i.e., armchair) argument: it doesn't depend on observation

The Ontological Argument (Literal Translation)

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.

Fairly Neutral Modern Translation

- We can conceive of something/that than which there is no greater
- If that thing does not exist in reality, then we can conceive of a greater thing—namely, something (just like it) that does exist in reality
- Therefore either the greatest thing exists in reality or it is not the greatest thing
- Therefore the greatest thing (necessarily) exists in reality
- That's God

Anselm's Ontological Argument

- Formulated by St. Anselm (1033–1109)
  - Archbishop of Canterbury, though originally from Italy
  - Aimed to show Christian doctrine compatible with reason
  - Cf. Avicenna's earlier proof of The Necessary Existent
- Appears in his Proslogion
  - Written in Latin
  - With variations and developments
  - So scholars debate exact interpretation
- Disputed by his contemporary Gaunilo
  - Existence of a perfect island
- Widely studied and disputed thereafter
  - Descartes, Leibniz, Hume, Kant (who named it), Gödel

Analyses of Anselm's Ontological Argument

- Reconstruction
  - What did Anselm actually say?
  - Can we accurately formulate that in modern logic?
  - How do various formulations actually differ?
- Analysis
  - Is the argument sound?
  - Russell, on his way to the tobacconist: "Great God in Boots!—the ontological argument is sound!"
- Interpretation
  - Is it persuasive?
  - 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"

Günter Eder and Esther Ramharter's Reconstruction

- Appears in Synthese vol. 192, October 2015
- Three stages: first-order, higher-order, modal logic
- I will cover just the first two
- Later look at a variant due to Richard Campbell
- And versions due to Paul Oppenheimer and Ed Zalta
- My goal is to illustrate formal methods using theorem provers
  - Better than model checkers for some types of problem
- And the difference between proof and assurance
- Let's get started

First-Order: Understandable Objects, Gods

- Something is a God if there is nothing greater

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

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

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

PVS fragment

```
U_beings: TYPE

x, y: VAR U_beings

>(x, y): bool

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

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

First-Order: Conceive Of, Real Existence

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

PVS fragment

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

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

PVS fragment

```
re?(x): bool


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

First-Order: Additional Premises

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

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

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

PVS fragment

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

First-Order: Complete PVS Specification

```
ontological_arg: THEORY
BEGIN

  U_beings: TYPE

  x, y: VAR U_beings

  >(x, y): bool

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

  re?(x): bool

  ExUnd: AXIOM EXISTS x: God?(x)

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

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

END ontological_arg
```

First-Order: PVS Proof

- PVS can prove the theorem given the following commands

PVS proof

```
(use "ExUnd") (use "Greater1") (grind)
```

- First two instruct PVS to use named formulas as premises
- Third instructs it to use general-purpose proof strategy
- PVS reports that the theorem is proved

The Sequent (final step)

```
God_re :

{-1}  FORALL (x): (NOT re?(x) => (EXISTS y: y > x))
[-2]  EXISTS x: God?(x)
  |-------
[1]   EXISTS x: God?(x) AND re?(x)

Rule? (grind)

God? rewrites God?(x)
  to NOT (EXISTS y: y > x OR NOT x > y)
God? rewrites God?(x)
  to NOT (EXISTS y: y > x OR NOT x > y)
Trying repeated skolemization, instantiation, and if-lifting,
Q.E.D.
```

First-Order: Proofchain Analysis

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

PVS proofchain

```
ontological_arg.God_re has been PROVED.

   The proof chain for God_re is COMPLETE.

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

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

First-Order: Second Attempt

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

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

  E!: $\exists x E!x$

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

PVS fragment

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

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

First-Order: Second Attempt (ctd. 1)

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

Higher-Order

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

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

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

PVS fragment

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

re?: pred[U_beings]

F: VAR (P)

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

Continued. . .

Higher-Order (ctd.)

- In PVS we do this using predicate subtypes

PVS fragment

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

re?: pred[U_beings]

F: VAR (P)

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

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

Higher-Order: Realization

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

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

PVS fragment

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

Higher-Order Formulation in PVS

```
HO_ontological_arg: THEORY
BEGIN

   U_beings: TYPE

   x, y: VAR U_beings

   re?: pred[U_beings]

   P: set[ pred[U_beings] ]

   F: VAR (P)

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

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

   ExUnd: AXIOM EXISTS x: God?(x)

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

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

 END HO_ontological_arg
```

Higher-Order Proof in PVS (ugh)

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

Higher-Order: Quasi-id

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

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

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

PVS fragment

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

And reproduce the same proof

Interim Summary

- The rich logics of verification systems such as PVS can express abstract logical/mathematical models very naturally
- And can prove the theorems
- But all that these really establish is that one string of symbols entails another
- Need to ask whether these strings are consistent with
  - Ideally, compel

  The intended real-world interpretation
- And whether the premises are true in that interpretation
- Doubts are obvious here
  - That's why I chose this example

  But arise just the same in models of faults, time, causality

Models of Faults, Time, Causality

- A Note on Inconsistent Axioms in Rushby's "Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms," by Lee Pike. IEEE TSE 32(5), May 2006, pp. 347–348

- Note: we often should use axioms in formalizing assumptions
  - Our task is to describe the environment, not implement it

- Other examples of Ontological Argument raise more concerns

Oppenheimer and Zalta's Treatments

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

PVS fragment

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

P1: AXIOM nonempty?(greatest)
```

- E&R version

PVS fragment

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

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

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

Comparison of O&Z and E&R Reconstructions

- Equivalence

PVS fragment

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

These are proved, respectively, by

PVS proof

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

and

PVS proof

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

- So one potential value is in comparing different reconstructions; more generally, comparing different models for a system
- And verification is stronger than eyeballing

Circularity of Greater 1

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

Unintended Models

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

The Model

model

```
interpretation: THEORY
BEGIN

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

END interpretation
```

Proof Obligations for Consistency

TCCs

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

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

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

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

...continued
```

Proof Obligations for Consistency (ctd.)

TCCs

```
...continuation

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

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

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

Unintended Models (ctd.)

- In system verification does it matter if there are models other than the one intended?

- If the assumptions are true in the intended model and the conclusion is useful, surely that's enough—it doesn't matter if there are additional models

- Well, we do want only one interpretation of safety

Why Verification Systems and not Simple Provers?

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

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

Oppenheimer and Zalta's Simplification (ctd.)

- Believed they had discovered a simplification to the Argument that

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

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

Definite Descriptions

- O&Z version 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: France is a republic, it has no present king
  - Is this true, false, inadmissible?
- Must not substitute definite descriptions into quantified expressions without being sure they are well defined
- PVS deals with this by requiring x to be of a singleton type
- This is a predicate subtype and the TCC mechanism will force you to prove uniqueness of "that than which. . ."

Sophisticated Types: More on Quasi-Id

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

PVS fragment

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

- And all Gods are quasi-identical

PVS fragment

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

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

PVS fragment

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

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

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

Now strong_all_God does not typecheck

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

- Now strong_all_God does not typecheck

TCC

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

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

Other Variants

- Richard Campbell (ANU) wrote a book on The Argument (1976)
- Revising it for a new edition, he adopts Eder and Ramharter's Greater 3 and Quasi-id
- But criticizes Realization (says it is false, actually)
  - My take: what if some predicates are contradictory/incompatible?
- Full treatment is difficult (uses modal operators), but core is to replace Realization with the following

PVS fragment

```
DD: setof[(PosProps)] = singleton(Re?)


a9: AXIOM
   NOT Re?(x) => (EXISTS z: quasi_id(DD)(z, x) AND Re?(z))
```

- Leads to a very simple proof
- Do you consider a9 a credible axiom?

Modal Reconstructions

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

Summary

- We've now seen some ways of probing formal models
  - Do they beg the question?
  - Are they equivalent to other models?
  - Do they have unintended interpretations?
- These probings can help us formulate the assumptions and caveats that should form part of the assurance (sub)case associated with any use of formal methods
- The big questions are standard
  - Do we believe the premises?
    - ⋆ As interpreted for our system
  - Does the conclusion bear the interpretation we need?
  - Do we trust the theorem prover
    - ⋆ And the formalization of any subsidiary theories?

Coming Up

Next, we'll look at modeling human-machine interactions (a major source of critical failures) with infinite bounded model checking and synchronous observers.

References

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

[2] Paweł Garbacz. Prover9's simplifications explained away. *Australasian Journal of Philosophy*, 90(3):585–592, 2012

[3] Paul E. Oppenheimer and Edward N. Zalta. On the logic of the Ontological Argument. *Philosophical Perspectives*, 5:509–529, 1991

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

[5] 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

[6] John Rushby. Mechanized analysis of a formalization of Anselm's Ontological Argument by Eder and Ramharter. CSL technical note, SRI International, Menlo Park, CA, January 2016