God in Coq: Modal Logic in Coq and Scott's Version of Gödel's Proof of God's Existence

(***Rough Diamond Proof Pearl***)

Christoph Benzmüller[1] and Bruno Woltzenlogel Paleo[2]

[1] Dahlem Center for Intelligent Systems, Freie Universität Berlin, Germany
c.benzmueller@gmail.com
[2] Theory and Logic Group, Vienna University of Technology, Austria
bruno@logic.at

1 Introduction

Gödel's proof of God's existence is challenging to formalize and verify because it requires an expressive logical language with modal operators ($\Diamond$ and $\Box$) and higher-order quantifiers. Therefore, it can serve as a good benchmark to measure and compare progress in the development of automated and interactive reasonging tools for higher-order modal logics. To fulfill this goal, formalizations [17] in TPTP THF [20], Coq [6] and Isabelle [16] have been completed so far, in this order. This paper focuses on the formalization in Coq.

An embedding of modal logics into higher-order logic with Henkin semantics [4, 3] provides the necessary theoretical foundation that enables the use of interactive theorem provers for modal logics as well. In the particular case of Coq, the implementation of this embedding is discussed in detail in Section 2 and Scott's version of Gödel's proof is shown in Section 3.

2 Modal Logic in Coq

A crucial aspect of modal logics [?] is that the so-called *necessitation rule* allows $\Box A$ to be derived if A is a theorem, but $A \rightarrow \Box A$ is not necessarily a theorem. Naive attempts to define the modal operators $\Box$ and $\Diamond$ may easily be unsound in this respect. To avoid this issue, the *possible world semantics* of modal logics can be explicitly embedded into higher-order logics [4, 3]. To this aim, a type for worlds must be declared and modal propositions should be not of type `Prop` but of a lifted type that depends on possible worlds. Possible worlds are connected by an accessibility relation.

```
(* Type for worlds *)
Parameter i: Type.

(* Type for individuals *)
Parameter u: Type.

(* Type of modal propositions *)
Definition o := i -> Prop.

(* Acessibility relation for worlds *)
Parameter r: i -> i -> Prop.
```

All modal connectives are simply lifted versions of the usual logical connectives. Notations are used to allow the modal connectives to be used as similarly as possible as the usual connectives. The prefix "`m`" is used to distinguish the modal connectives: if $\odot$ is a connective on type Prop, `m`$\odot$ is a connective on the lifted type `o` of modal propositions.

```
Definition mnot (p: o)(w: i) := ~ (p w).
Notation "m~  p" := (mnot p) (at level 74, right associativity).

Definition mand (p q:o)(w: i) := (p w) /\ (q w).
Notation "p m/\ q" := (mand p q) (at level 79, right associativity).

Definition mor (p q:o)(w: i) := (p w) \/ (q w).
Notation "p m\/ q" := (mor p q) (at level 79, right associativity).

Definition mimplies (p q:o)(w:i) := (p w) -> (q w).
Notation "p m-> q" := (mimplies p q) (at level 99, right associativity).

Definition mequiv (p q:o)(w:i) := (p w) <-> (q w).
Notation "p m<-> q" := (mequiv p q) (at level 99, right associativity).
```

Likewise, modal quantifiers are lifted versions of the usual quantifiers.

```
Definition A {t: Type}(p: t -> o) := fun w => forall x, p x w.
Notation "'mforall'  x , p" := (A (fun x => p))
  (at level 200, x ident, right associativity) : type_scope.
Notation "'mforall' x : t , p" := (A (fun x:t => p))
  (at level 200, x ident, right associativity,
    format "'[' 'mforall' '/ '  x  :  t , '/ '  p ']'")
  : type_scope.

Definition E {t: Type}(p: t -> o) := fun w => exists x, p x w.
Notation "'mexists' x , p" := (E (fun x => p))
  (at level 200, x ident, right associativity) : type_scope.
Notation "'mexists' x : t , p" := (E (fun x:t => p))
  (at level 200, x ident, right associativity,
    format "'[' 'mexists' '/ '  x  :  t , '/ '  p ']'")
  : type_scope.
```

The modal operators $\Diamond$ (*possibly*) and $\Box$ (*necessarily*) are defined according to their meanings in the possible world semantics.

```
(* Modal operator for 'necessarily' *)
Definition box (p: o) := fun w => forall w1, (r w w1) -> (p w1).

(* Modal operator for 'possibly' *)
Definition dia (p: o) := fun w => exists w1, (r w w1) /\ (p w1).
```

Although the embedding via the possible world semantics is necessary, it should be as transparent as possible to the user. Fortunately, basic Coq tactics such `intro`, `apply` and `split` automatically unfold the modal definitions and thus can be used with modal connectives and quantifiers as they are used with the usual connectives and quantifiers. Nothing else needs to be done. For the modal operators, however, the following simple tactics have been implemented, in order to allow the user to work with the concepts of *necessity* and *possibility* without having to unfold the definitions of modal operators and think in terms of possible worlds. These tactics remind rules of labelled calculi for modal logics.

```
Ltac box_intro w H := intros w H.

Ltac box_elim H w1 R1 Hn :=
  let P := match type of H with
```

```
             (* forall w0:i, r ?w w0 -> ?p w0 => p *)
             ((box ?p) _) => p
           end
  in cut (P w1); [intros Hn | apply (H w1 R1)].

Ltac dia_elim H w R newH := destruct H as [w [R newH]].

Ltac dia_intro w := (exists w; split; [assumption | idtac]).
```

Validity of a modal proposition is validity of its grounding on any world.

```
(* Modal validity of lifted propositions *)
Definition V (p: o) := forall w, p w.
```

The following lemmas are convenient

```
Lemma modus_ponens_inside_dia: V (mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)).
Proof.
intro.
intros p q.
intro H1.
intro H2.
dia_elim H1 w1 R1 H1.
exists w1.
split.
  exact R1.

  apply H2.
    exact R1.

    exact H1.
Qed.

Lemma not_dia_box_not: V (mforall p, ((m~ (dia p)) m-> (box (m~ p)) )).
Proof.
intro.
intro p.
intro H.
intros w1 H1.
intro H2.
apply H.
exists w1; split. exact H1.
exact H2.
Qed.

(* Modal accessibility axioms *)

Axiom reflexivity: forall w, r w w.

Axiom transitivity: forall w1 w2 w3, (r w1 w2) -> (r w2 w3) -> (r w1 w3).

Axiom symmetry: forall w1 w2, (r w1 w2) -> (r w2 w1).

(* Modal axioms *)
(* Here we show how they can be derived from the accessibility axioms *)
```

```
Theorem K: (V (mforall p, mforall q, (box (p m-> q)) m-> ( (box p) m-> (box q) ) )).
Proof.
intros w p q.
intros H1 H2.
intros w1 R1.
apply H1.
  exact R1.

  apply H2.
  exact R1.
Qed.


Theorem T: (V (mforall p, (box p) m-> p)).
Proof.
intros w p.
intro H.
apply H.
apply reflexivity.
Qed.

(* In strong modal logics, such as S5, iterations of modal operators can be collapsed *)
Theorem dia_box_to_box: V (mforall p, (dia (box p)) m-> (box p)).
Proof.
intro.
intro p.
intro H1.
destruct H1 as [w1 [R1 H1]].
intro. intro R0.
apply H1.
apply transitivity with (w2 := w).
  apply symmetry.
  exact R1.

  exact R0.
Qed.
```

3 Gödel's Proof of God's Existence

Moreover, automated reasoning tools such as Nitpick [8], LEO-II [5], Satallax [9], Sledgehammer [7] and Metis [15] have been used and a few interesting new facts have been discovered about this proof [?]. The formalization in Isabelle [?].

Attempts to prove the existence (or non-existence) of God by means of abstract ontological arguments are an old tradition in philosophy and theology. Gödel's proof [12, 13] is a modern culmination of this tradition, following particularly the footsteps of Leibniz. Gödel defines God as a being who possesses all *positive* properties. He does not extensively discuss what positive properties are, but instead he states a few reasonable (but debatable) axioms that they should satisfy. Various slightly different versions of axioms and definitions have been considered by Gödel and by several philosophers who commented on his proof (cf. [19, 2, 11, 1, 10]).

Dana Scott's version of Gödel's proof [18]. This formalization aims at being as similar as possible to Dana Scott's version of the proof

The numbering of axioms, definitions and theorems is exactly the same as in Scott's notes

The formal proofs follow the same structure of Scott's proof sketches and fill their gaps

Whenever a 'cut' or 'assert' uses a lemma mentioned in Scott's sketches, this is emphasized with a comment

```
Require Import Coq.Logic.Classical.
Require Import Coq.Logic.Classical_Pred_Type.

Require Import Modal.

Ltac proof_by_contradiction H := apply NNPP; intro H.


(* Constant predicate that distinguishes positive properties *)
Parameter Positive: (u -> o) -> o.


(* Axiom A1: either a property or its negation is positive, but not both *)
Axiom axiom1a : V (mforall p, (Positive (fun x: u => m~(p x))) m-> (m~ (Positive p))).
Axiom axiom1b : V (mforall p, (m~ (Positive p)) m-> (Positive (fun x: u => m~ (p x))) ).


(* Axiom A2: a property necessarily implied by a positive property is positive *)
Axiom axiom2: V (mforall p, mforall q, Positive p m/\ (box (mforall x, (p x) m-> (q x) )) m-> Pos


(* Theorem T1: positive properties are possibly exemplified *)
Theorem theorem1: V (mforall p, (Positive p) m-> dia (mexists x, p x) ).
Proof.
intro.
intro p.
intro H1.
proof_by_contradiction H2.
apply not_dia_box_not in H2.
assert (H3: ((box (mforall x, m~ (p x))) w)). (* Lemma from Scott's notes *)
  box_intro w1 R1.
  intro x.
  assert (H4: ((m~ (mexists x : u, p x)) w1)).
    box_elim H2 w1 R1 G2.
    exact G2.

    clear H2 R1 H1 w.
    intro H5.
    apply H4.
    exists x.
    exact H5.

  assert (H6: ((box (mforall x, (p x) m-> m~ (x m= x))) w)). (* Lemma from Scott's notes *)
    box_intro w1 R1.
    intro x.
    intro H7.
    intro H8.
    box_elim H3 w1 R1 G3.
    apply G3 with (x := x).
    exact H7.
    assert (H9: ((Positive (fun x => m~ (x m= x))) w)). (* Lemma from Scott's notes *)
    apply (axiom2 w p (fun x => m~ (x m= x))).
    split.
      exact H1.

      exact H6.
    assert (H10: ((box (mforall x, (p x) m-> (x m= x))) w)). (* Lemma from Scott's notes *)
      box_intro w1 R1.
      intros x H11.
      reflexivity.

      assert (H11 : ((Positive (fun x => (x m= x))) w)). (* Lemma from Scott's notes *)
        apply (axiom2 w p (fun x => x m= x )).
        split.
          exact H1.

          exact H10.

        clear H1 H2 H3 H6 H10 p.
        apply axiom1a in H9.
        contradiction.
Qed.


(* Definition D1: God: a God-like being possesses all positive properties *)
Definition G(x: u) := mforall p, (Positive p) m-> (p x).


(* Axiom A3: the property of being God-like is positive *)
Axiom axiom3: V (Positive G).


(* Corollary C1: possibly, God exists *)
Theorem corollary1: V (dia (mexists x, G x)).
Proof.
intro.
apply theorem1.
apply axiom3.
Qed.


(* Axiom A4: positive properties are necessarily positive *)
Axiom axiom4: V (mforall p, (Positive p) m-> box (Positive p)).


(* Definition D2: essence: an essence of an individual is a property possessed by it and necessar
Definition Essence(p: u -> o)(x: u) := (p x) m/\ mforall q, ((q x) m-> box (mforall y, (p y) m->
Notation "p 'ess' x" := (Essence p x) (at level 69).


(* Theorem T2: being God-like is an essence of any God-like being *)
Theorem theorem2: V (mforall x, (G x) m-> (G ess x)).
Proof.
```

```
intro.
intro g.
intro H1.
unfold Essence.
split.
  exact H1.

  intro q.
  intro H2.
  assert (H3: ((Positive q) w)).
    proof_by_contradiction H4.
    unfold G in H1.
    apply axiom1b in H4.
    apply H1 in H4.
    contradiction.

    cut (box (Positive q) w). (* Lemma from Scott's notes *)
      apply K.
      box_intro w1 R1.
      intro H5.
      intro y.
      intro H6.
      unfold G in H6.
      apply (H6 q).
      exact H5.

      apply axiom4.
      exact H3.
Qed.

(* At this point in Scott's notes there are two notes that are not necessary for the proof, *)
(* but it would be interesting to formalize them anyway *)

(* Definition D3: necessary existence: necessary existence of an individual is the necessary exem
Definition NE(x: u) := mforall p, (p ess x) m-> box (mexists y, (p y)).


(* Axiom A5: necessary existence is a positive property *)
Axiom axiom5: V (Positive NE).


Lemma lemma1: V ((mexists z, (G z)) m-> box (mexists x, (G x))).
Proof.
intro.
intro H1.
destruct H1 as [g H2].
cut ((G ess g) w).      (* Lemma from Scott's notes *)
  assert (H3: (NE g w)).       (* Lemma from Scott's notes *)
    unfold G in H2.
    apply (H2 NE).
    apply axiom5.

    unfold NE in H3.
    apply H3.
  apply theorem2.
  exact H2.
Qed.


Lemma lemma2: V (dia (mexists z, (G z)) m-> box (mexists x, (G x))).
Proof.
intro.
intro H.
cut (dia (box (mexists x, G x)) w).  (* Lemma from Scott's notes *)
  apply dia_box_to_box.

  apply (modus_ponens_inside_dia w (mexists z, G z)).
    exact H.

    box_intro w1 R1.
    apply lemma1.
Qed.


(* Theorem T3: necessarily, a God exists *)
Theorem theorem3: V (box (mexists x, (G x))).
Proof.
intro.
apply lemma2.
apply corollary1.
Qed.


(* Corollary C2: There exists a god *)
Theorem corollary2: V (mexists x, (G x)).
Proof.
intro.
apply T.
apply theorem3.
Qed.
```

Note: possible worlds machinery is hidden from the user.

4 Conclusions

cite related work about modal logics in Coq.

no need to bother about the embedding.

This work attests the maturity of contemporary interactive and automated deduction tools for classical higher-order logic and demonstrates the elegance and practical relevance of the embeddings-based approach. Most importantly, our work opens new perspectives for a computer-assisted theoretical philosophy. The critical discussion of the underlying concepts, definitions and axioms remains a human responsibility, but the computer can assist in building and checking rigorously correct logical arguments. In case of logico-philosophical disputes, the computer can check the disputing arguments and partially fulfill Leibniz' dictum: Calculemus — Let us calculate! The philosophical and theological importance of the correctness of this proof is beyond the scope of this paper.

We welcome feedback on

Acknowledgements: ToDo: thank Coq-Club members.

References

1. R.M. Adams. Introductory note to *1970. In *Kurt Gödel: Collected Works Vol. 3: Unpublished Essays and Letters*. Oxford University Press, 1995.
2. A.C. Anderson and M. Gettings. Gödel ontological proof revisited. In *Gödel'96: Logical Foundations of Mathematics, Computer Science, and Physics: Lecture Notes in Logic 6*. Springer, 1996.
3. C. Benzmüller and L.C. Paulson. Exploring properties of normal multimodal logics in simple type theory with LEO-II. In *Festschrift in Honor of Peter B. Andrews on His 70th Birthday*, pages 386–406. College Publications.
4. C. Benzmüller and L.C. Paulson. Quantified multimodal logics in simple type theory. *Logica Universalis (Special Issue on Multimodal Logics)*, 7(1):7–20, 2013.
5. C. Benzmüller, F. Theiss, L. Paulson, and A. Fietzke. LEO-II - a cooperative automatic theorem prover for higher-order logic. In *Proc. of IJCAR 2008*, volume 5195 of *LNAI*, pages 162–170. Springer, 2008.
6. Y. Bertot and P. Casteran. *Interactive Theorem Proving and Program Development*. Springer, 2004.
7. J.C. Blanchette, S. Böhme, and L.C. Paulson. Extending Sledgehammer with SMT solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013.
8. J.C. Blanchette and T. Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Proc. of ITP 2010*, number 6172 in LNCS, pages 131–146. Springer, 2010.
9. C.E. Brown. Satallax: An automated higher-order prover. In *Proc. of IJCAR 2012*, number 7364 in LNAI, pages 111 – 117. Springer, 2012.
10. R. Corazzon. Contemporary bibligraphy on the ontological proof (`http://www.ontology.co/biblio/ontological-proof-contemporary-biblio.htm`).
11. M. Fitting. *Types, Tableaux and Gödel's God*. Kluver Academic Press, 2002.
12. K. Gödel. Ontological proof. In *Kurt Gödel: Collected Works Vol. 3: Unpublished Essays and Letters*. Oxford University Press, 1970.
13. K. Gödel. *Appendix A. Notes in Kurt Gödel's Hand*, pages 144–145. In [19], 2004.
14. A.P. Hazen. On gödel's ontological proof. *Australasian Journal of Philosophy*, 76:361–377, 1998.
15. J. Hurd. First-order proof tactics in higher-order logic theorem provers. In *Design and Application of Strategies/Tactics in Higher Order Logics, NASA Tech. Rep. NASA/CP-2003-212448*, 2003.
16. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.
17. B. Woltzenlogel Paleo and C. Benzmüller. Formal theology repository (`https://github.com/FormalTheology/GoedelGod`).
18. D. Scott. *Appendix B. Notes in Dana Scott's Hand*, pages 145–146. In [19], 2004.
19. J.H. Sobel. *Logic and Theism: Arguments for and Against Beliefs in God*. Cambridge U. Press, 2004.
20. G. Sutcliffe and C. Benzmüller. Automated reasoning in higher-order logic using the TPTP THF infrastructure. *Journal of Formalized Reasoning*, 3(1):1–27, 2010.