Modal Logics in the Coq Proof Assistant

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

Abstract. This paper describes an embedding of higher-order modal logics in the Coq proof assistant. Coq's capabilities are thus extended in a minimalistic manner, which is nevertheless sufficient for the formalization of significant modal proofs. The elegance, flexibility and convenience of this approach, from a user perspective, are illustrated here with the successful formalization of Gödel's ontological argument.

1 Introduction

Modal logics extend the usual formal logic language with modal operators ($\Box$ and $\Diamond$) and are characterized by the *necessitation rule*, according to which $\Box A$ is a theorem if A is a theorem, although $A \rightarrow \Box A$ is not necessarily a theorem. Various informal concepts, such as *necessity and possibility*, *knowledge and belief*, and *temporal globality and eventuality*, have been formalized with the help of modal operators. Therefore, modal logics [9] are well-established for a wide variety of application domains, ranging from philosophy [**?**] to formal software verification [**?**].

However, general automated reasoning support for modal logics is still not as well-developed as for classical logics. Current deduction tools for modal logics are often limited to propositional, quantifier-free, fragments [**?**] or tailored to particular modal logics and their applications [**?**]. Recently, first-order automated deduction techniques based on tableaux, sequent calculi and connection calculi have been generalized to cope with modalities directly and a few new provers have been implemented [**?**].

Another recently explored possibility [4, 3, 25] is the embedding of first-order and higher-order modal logics into classical higher-order logics, for which existing higher-order automated theorem provers [**?**,13] exist. Embedding is flexible, because various modal logics can be easily supported by adding their characteristic axioms. The embedding approach can also be easily adapted to support multiple modalities, and it is possible to replace constant domain quantifiers by varying or cumulative domain quantifiers. Moreover, the approach is relatively simple to implement, because it does not require any modification in the

* This work has been supported by the German Research Foundation (DFG) under grant BE2501/9-1.

source code of the higher-order prover. The prover can be used as is, and only the input files provided to the prover must be especially encoded. Furthermore, the efficacy and efficiency of the embedding approach has been confirmed in philosophically interesting and relevant benchmarks [26]. These qualities make embedding a convenient approach for *fully automated* reasoning.

However, one may wonder whether the embedding approach is adequate also for *interactive* reasoning, when the user proves theorems by interacting with a proof assistant. The main goal of this paper is to strudy this question. Our answer is positive. The `Coq` proof assistant was chosen because of the authors' greater familiarity with the tactic language of this system.

One major initial concern was whether the embedding could be a disturbance to the user. Fortunately, by using `Coq`'s `Ltac` tactic language, we were able to define intuitive new tactics that hide the technical details of the embedding from the user. The resulting infra-structure for modal reasoning within `Coq` (as described in Section 2) provides a user experience where modalities can be handled transparently and straightforwardly. Therefore, a user with basic knowledge of modal logics and `Coq`'s tactics should be able to use (and extend) our implementation with no excessive overhead.

In order to illustrate the use of the implemented embedding, we show here the formalization of Scott's version [**?**] of Gödel's ontological argument for God's existence (in Section 3). This proof was chosen mainly for two reasons. Firstly, it requires not only modal operators, but also higher-order quantification. Therefore, it is beyond reach of specialized propositional and first-order (modal) theorem provers. Secondly, this argument addresses an ancient problem in Philosophy and Metaphysics, which has nevertheless received a lot of attention in the last 15 years, because of the discovery of the modal collapse [**?**]. This proof lies in the center of a vast and largely unexplored application domain for automated and interactive theorem provers.

The simpler ontological argument of Anselm has been automatically verified with `PVS` by Rushby [22] and with first-order theorem by Oppenheimer and Zalta [12]. Gödel's argument was automatically verified in our previous work on embedding-based fully automated modal theorem proving [**?**,7]. But this paper presents the first fully interactive and detailed formalization of this proof in a proof assistant.

2 Modal Logic in Coq

A crucial aspect of modal logics [9] 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].

The embedding is related to labeling techniques [16]. However, the expressiveness of higher-order logic is exploited here to encode the labels within the logic language itself. To this aim, a type for worlds must be declared and modal

propositions should be not of type `Prop` but of a lifted type `o` that depends on possible worlds:

```
Parameter i: Type. (* Type for worlds *)
Parameter u: Type. (* Type for individuals *)
Definition o := i -> Prop. (* Type of modal propositions *)
```

Possible worlds are connected by an accessibility relation, which can be represented in `Coq` by a parameter `r`, as follows:

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

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 mequal (x y: u) := fun w: i => x = y.
Notation "x m= y" := (mequal x y) (at level 99, right associativity).

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 accordingly to their meanings in the possible world semantics.

```
Definition box (p: o) := fun w => forall w1, (r w w1) -> (p w1).
Definition dia (p: o) := fun w => exists w1, (r w w1) /\ (p w1).
```

Fig. 1. propositional rules

$$\frac{\bot}{A}\ \bot_E \qquad \frac{\begin{array}{c}\overline{A}\ n\\ \vdots\\ B\end{array}}{A \to B}\ \to_I^n \qquad \frac{A \quad A \to B}{B}\ \to_E$$

$$\frac{A \quad B}{A \wedge B}\ \wedge_I \qquad \frac{A \wedge B}{A}\ \wedge_{E_1} \qquad \frac{A \wedge B}{B}\ \wedge_{E_2}$$

Fig. 2. double negation elimination

$$\frac{\neg\neg A}{A}\ \neg\neg_E$$

To maximally preserve user intuition in interactive modal logic theorem proving, the embedding via the possible world semantics should be as transparent as possible to the user. Fortunately, basic Coq tactics such as `intro`, `apply` and `split` automatically unfold the modal definitions. They can thus be used with modal connectives and quantifiers just as they are used with the usual connectives and quantifiers. 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.

```
Ltac box_i := let w := fresh "w" in let R := fresh "R"
              in (intro w at top; intro R at top).
```

Fig. 3. quantifier rules

$$\frac{A[\alpha]}{\forall x_\tau.A[x]}\ \forall_I \qquad \frac{\forall x_\tau.A[x]}{A[t]}\ \forall_E \qquad \frac{A[t]}{\exists x_\tau.A[x]}\ \exists_I \qquad \frac{\exists x_\tau.A[x]}{A[\beta]}\ \exists_E$$

eigen-variable conditions:
if ρ is a $\forall_I$ inference eliminating a variable α, then any occurrence of α in the proof should be an ancestor of the occurrence of α eliminated by ρ;
if ρ is a $\exists_E$ inference introducing a variable β, then any occurrence of β in the proof should be a descendant of the occurrence of β introduced by ρ.

Fig. 4. Rules for Modal Operators

$$\frac{\omega : \boxed{\begin{array}{c}\vdots\\ A\end{array}}}{\Box A}\ \Box_I \qquad \frac{\Box A}{w : \boxed{\begin{array}{c}A\\ \vdots\end{array}}}\ \Box_E$$

$$\frac{w : \boxed{\begin{array}{c}\vdots\\ A\end{array}}}{\Diamond A}\ \Diamond_I \qquad \frac{\Diamond A}{\omega : \boxed{\begin{array}{c}A\\ \vdots\end{array}}}\ \Diamond_E$$

eigen-box condition:
ω must be a fresh name for a box.

boxed assumption condition:
assumptions should be discharged within the box where they are made.

```
Ltac box_elim H w1 H1 := match type of H with
      ((box ?p) ?w) => cut (p w1);
                       [intros H1 | (apply (H w1); try assumption)] end.

Ltac box_e H H1:= match goal with | [ |- (_ ?w) ] => box_elim H w H1 end.

Ltac dia_e H := let w := fresh "w" in let R := fresh "R" in
                (destruct H as [w [R H]]; move w at top; move R at top).

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

Validity of a modal proposition is validity of its grounding on any world. A non-intrusive notation and a tactic that grounds the proposition to an arbitrary world are provided.

```
Definition V (p: o) := forall w, p w.
Notation "[ p ]" := (V p).
Ltac mv := match goal with [|- (V _)] => intro end.
```

The following lemmas are quite convenient. The first is a kind of modus ponens applicable to formulas under the scope of a modal operator. The second allows pushing negations inside modalities.

```
Lemma mp_dia:
  [mforall p, mforall q, (dia p) m-> (box (p m-> q)) m-> (dia q)].
Proof. mv.
intros p q H1 H2. dia_e H1. dia_i w0. box_e H2 H3. apply H3. exact H1.
Qed.

Lemma not_dia_box_not: [mforall p, (m~ (dia p)) m-> (box (m~ p))].
Proof. mv.
intro p. intro H. box_i. intro H2. apply H. dia_i w0. exact H2.
Qed.
```

The most well-known modal logics can be distinguished by their acceptance of the following reachability axioms. **S5** is the modal logic that accepts all three.

```
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 logic "axioms" can be derived from the reachability axioms.

```
Theorem K:
  [ mforall p, mforall q, (box (p m-> q)) m-> (box p) m-> (box q) ].
Proof. mv.
intros p q H1 H2. box_i. box_e H1 H3. apply H3. box_e H2 H4. exact H4.
Qed.

Theorem T: [ mforall p, (box p) m-> p ].
Proof. mv.
intro p. intro H. box_e H H1. exact H1. apply reflexivity.
Qed.
```

In strong modal logics, such as S5, iterations of modal operators can be collapsed. This is a controversial principle used in Gödel's and Scott's versions of the proof.

```
Theorem dia_box_to_box: [ mforall p, (dia (box p)) m-> (box p) ].
Proof. mv.
intros p H1. dia_e H1. box_i. box_e H1 H2. exact H2. eapply transitivity.
  apply symmetry. exact R.
  exact R0.
Qed.
```

3 Gödel's Proof of God's Existence

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 [17, 18] is a modern culmination of this tradition, following particularly the footsteps of Leibniz. Various slightly different versions of axioms and definitions have been considered by Gödel and by several philosophers who commented on his proof (cf. [24, 2, 15, 1, 14]). The formalization shown in this section aims at being as similar as possible to Dana Scott's version of the proof [23]. The formulation and numbering of axioms, definitions and theorems is the same as in Scott's notes. Even the Coq proof scripts follow all the steps in Scott's notes. Scott's assertions are emphasized with comments. In contrast to the formalization in Isabelle [7], where automation via Metis [20] and Sledgehammer [10] using tools such as Nitpick [11], LEO-II [5] and Satallax [13] has been successfully employed, the formalization in Coq used no automation. This was a deliberate choice, mainly because it allowed a qualitative evaluation of the convenience of the possible world semantic embedding approach for *interactive* theorem proving. Moreover, in order to formalize precisely Scott's version and not another automatically found version, automation would have to be heavily limited anyway. Furthermore, the deliberate preference for simple tactics (mostly *intro*, *apply* and the modal tactics described in the previous section) results in proof scripts that could be read as natural deduction proofs. This hopefully makes the formalization more accessible to those who are not experts in Coq's tactics but are nevertheless interested in Gödel's proof.

```
Require Import Coq.Logic.Classical Coq.Logic.Classical_Pred_Type 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 :
  [ mforall p, (Positive (fun x: u => m~(p x))) m-> (m~ (Positive p)) ].

Axiom axiom1b :
  [ 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: [ mforall p, mforall q,
  Positive p m/\ (box (mforall x, (p x) m-> (q x) )) m-> Positive q ].

(* Theorem T1: positive properties are possibly exemplified *)
Theorem theorem1: [ mforall p, (Positive p) m-> dia (mexists x, p x) ].
Proof. mv.
```

```
intro p. intro H1. proof_by_contradiction H2. apply not_dia_box_not in H2.
assert (H3: ((box (mforall x, m~ (p x))) w)). (* Scott *)
  box_i. intro x. assert (H4: ((m~ (mexists x : u, p x)) w0)).
    box_e H2 G2. exact G2.
    clear H2 R H1 w. intro H5. apply H4. exists x. exact H5.
  assert (H6: ((box (mforall x, (p x) m-> m~ (x m= x))) w)). (* Scott *)
    box_i. intro x. intros H7 H8. box_elim H3 w0 G3. eapply G3. exact H7.
    assert (H9: ((Positive (fun x => m~ (x m= x))) w)). (* Scott *)
      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)). (* Scott *)
        box_i. intros x H11. reflexivity.
        assert (H11 : ((Positive (fun x => (x m= x))) w)). (* Scott *)
          apply (axiom2 w p (fun x => x m= x )). split.
            exact H1.
            exact H10.
          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: [ Positive G ].

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

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

(* Definition D2:
   essence: an essence of an individual is a property possessed by it
   and necessarily implying any of its properties *)
Definition Essence(p: u -> o)(x: u) :=
    (p x) m/\ mforall q, ((q x) m-> box (mforall y, (p y) m-> (q y))).
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: [ mforall x, (G x) m-> (G ess x) ].
Proof. mv. 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). (* Scott *)
```

```
      apply K. box_i. intro H5. intro y. intro H6.
      unfold G in H6. apply (H6 q). exact H5.

      apply axiom4. exact H3.
Qed.

(* Definition D3:
   necessary existence: necessary existence of an individual
   is the necessary exemplification of all its essences *)
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: [ Positive NE ].

Lemma lemma1: [ (mexists z, (G z)) m-> box (mexists x, (G x)) ].
Proof. mv.
intro H1. destruct H1 as [g H2]. cut ((G ess g) w). (* Scott *)
  assert (H3: (NE g w)).       (* Scott *)
    unfold G in H2. apply (H2 NE). apply axiom5.
    unfold NE in H3. apply H3.
  apply theorem2. exact H2.
Qed.

Lemma lemma2: [ dia (mexists z, (G z)) m-> box (mexists x, (G x)) ].
Proof. mv.
intro H. cut (dia (box (mexists x, G x)) w).  (* Scott *)
  apply dia_box_to_box.
  apply (mp_dia w (mexists z, G z)).
    exact H.
    box_i. apply lemma1.
Qed.

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

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

4 Conclusions

The successful formalization of Scott's version of Gödel's proof of God's existence indicates that the *embedding* of modal logics into higher-order logics via the *possible world semantics* is a viable approach for interactive theorem proving within modal logics. The minimalistic implementation of the embedding (Section 2) takes special care to hide the underlying possible world machinery from the user. An inspection of the proof scripts in Section 3 shows that this goal has been

achieved. The user does not have to explicitly bother about worlds and their mutual reachability; the provided tactics for modalities do the job for him/her. Moreover, for subgoals that do not involve modalities, the user has all the usual interactive tactics at his/her disposal.

HOL Light, Isabelle

The theological implications of the verified correctness of this proof certainly depend on a critical discussion of the underlying concepts, definitions and axioms, which is beyond the scope of this paper. Clearly, the application of theorem proving technology to the domain of theoretical philosophy can — as already pictured by Leibniz — be very fruitful for both areas. In fact, new results of philosophical interest have been obtained by our work [6]. Moreover a first, basic infrastructure for interactive and automated reasoning in higher-order modal logics has been created and will be further improved in the future.

Acknowledgements: we thanks Cedric Auger and Laurent Théry, for their answers to our questions about Ltac in the Coq-Club mailing-list.

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. C. Benzmüller and B. Woltzenlogel Paleo, Formalization, Mechanization and Automation of Gödel's Proof of God's Existence. *arXiv*:1308.4526, 2013.
7. C. Benzmüller and B. Woltzenlogel Paleo, Gödel's God in Isabelle/HOL. *Archive of Formal Proofs*, 2013.
8. Y. Bertot and P. Casteran. *Interactive Theorem Proving and Program Development*. Springer, 2004.
9. P. Blackburn, J.v. Benthem, and F. Wolter (eds.), *Handbook of Modal Logic*. Elsevier, 2006.
10. J.C. Blanchette, S. Böhme, and L.C. Paulson. Extending Sledgehammer with SMT solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013.
11. 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*, no. 6172 in LNCS, pages 131–146. Springer, 2010.
12. P.E. Oppenheimera and E.N. Zalta. A Computationally-Discovered Simplification of the Ontological Argument. *Australasian Journal of Philosophy*, 89(2):333-349, 2011.

13. C.E. Brown. Satallax: An automated higher-order prover. In *Proc. of IJCAR 2012*, number 7364 in LNAI, pages 111 – 117. Springer, 2012.
14. R. Corazzon. Contemporary bibligraphy on the ontological proof (`http://www.ontology.co/biblio/ontological-proof-contemporary-biblio.htm`).
15. M. Fitting. *Types, Tableaux and Gödel's God*. Kluver Academic Press, 2002.
16. D.M. Gabbay. *Labelled Deductive Systems*. Clarendon Press, 1996.
17. K. Gödel. Ontological proof. In *Kurt Gödel: Collected Works Vol. 3: Unpublished Essays and Letters*. Oxford University Press, 1970.
18. K. Gödel. *Appendix A. Notes in Kurt Gödel's Hand*, pages 144–145. In [24], 2004.
19. A.P. Hazen. On gödel's ontological proof. *Australasian Journal of Philosophy*, 76:361–377, 1998.
20. 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.
21. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Number 2283 in LNCS. Springer, 2002.
22. J. Rushby. The Ontological Argument in PVS. In *Proc. of CAV Workshop "Fun With Formal Methods"*, St. Petersburg, Russia, 2013.
23. D. Scott. *Appendix B. Notes in Dana Scott's Hand*, pages 145–146. In [24], 2004.
24. J.H. Sobel. *Logic and Theism: Arguments for and Against Beliefs in God*. Cambridge U. Press, 2004.
25. 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.
26. B. Woltzenlogel Paleo and C. Benzmüller. Formal theology repository (`https://github.com/FormalTheology/GoedelGod`).