A formal proof of modal completeness for provability logic

Cosimo Perini Brogi [*]   Marco Maggesi [†]

[*]University of Genoa

[†]University of Florence

ITP 2021, Rome & The Internet
29 June – 01 July, 2021

Modal Logics: a large class

NON-NORMAL	NORMAL
Ability	Alethic
Conditional	Temporal
Belief revision	Dynamic
Probability	Epistemic
Deontic	Many Valued
Social Choice Theory	FDE
⋮	Provability
	⋮

Proof theory, foundations of mathematics, ordinal analysis,. . .

Modal Logics: a large class

Non-normal	Normal
Ability	Alethic
Conditional	Temporal
Belief revision	Dynamic
Probability	Epistemic
Deontic	Many Valued
Social Choice Theory	FDE
⋮	Provability
	⋮

Proof theory, foundations of mathematics, ordinal analysis,...

Gödel-Löb Logic

We consider a propositional modal language $\mathcal{L}_\Box$ whose formulas have one of the following forms:

$$p \mid \top \mid \bot \mid \neg A \mid A \wedge B \mid A \vee B \mid A \to B \mid A \leftrightarrow B \mid \Box A.$$

$\mathbb{GL}$ denotes the axiomatic calculus made of:

- Axioms of $\mathbb{CPC}$
- Axiom K : $\Box(A \to B) \to \Box A \to \Box B$
- Axiom GL : $\Box(\Box A \to A) \to \Box A$
- MP Rule $\dfrac{A \to B \qquad A}{B}$
- Nec Rule $\dfrac{A}{\Box A}$

Gödel-Löb Logic

We consider a propositional modal language $\mathcal{L}_\Box$ whose formulas have one of the following forms:

$$p \mid \top \mid \bot \mid \neg A \mid A \wedge B \mid A \vee B \mid A \to B \mid A \leftrightarrow B \mid \Box A.$$

$\mathbb{GL}$ denotes the axiomatic calculus made of:

- Axioms of $\mathbb{CPC}$
- Axiom K : $\Box(A \to B) \to \Box A \to \Box B$
- Axiom GL : $\Box(\Box A \to A) \to \Box A$
- MP Rule $\dfrac{A \to B \qquad A}{B}$
- Nec Rule $\dfrac{A}{\Box A}$

Arithmetical Realization

Let T be a theory of arithmetic such that $\mathsf{I}\Sigma_1 \subseteq \mathsf{T}$.
A function $* : \mathsf{Form}_{\mathcal{L}_\Box} \to \mathsf{Sent}_{\mathcal{L}_\mathsf{T}}$ is a realization if $(\Box A)^* := Prov_\mathsf{T}\ulcorner A^* \urcorner$ and it distributes over classical operators.

Soundness

For any $A \in \mathsf{Form}_{\mathcal{L}_\Box}$, if $\mathbb{GL} \vdash A$ then $\mathsf{T} \vdash A^*$ for any realization $*$.

Formalized Second Incompleteness Theorem

$\mathsf{T} \vdash \neg Prov_\mathsf{T}\ulcorner \bot \urcorner \to \neg Prov_\mathsf{T}\ulcorner \neg Prov_\mathsf{T}\ulcorner \bot \urcorner \urcorner$.

Henkin-Löb Theorem

For any T as before,

$$\mathsf{T} \vdash A \quad \text{iff} \quad \mathsf{T} \vdash Prov_\mathsf{T}\ulcorner A \urcorner \to A.$$

Solovay Theorem

Completeness (Solovay 1976)

For any $A \in \mathsf{Form}_{\mathcal{L}_\Box}$, if $\mathsf{I}\Sigma_1 \subseteq \mathsf{T}$ and $\mathbb{GL} \nvdash A$, then $\mathsf{T} \nvdash A^*$ for some realization $*$.

The only way to prove this result is by encoding in T a relational countermodel for A and "extract" from it a $*$ that does the job.

Modal Adequacy

$$\mathbb{GL} \vdash A \quad \text{iff} \quad TFT \vDash A$$

where TFT is the class of relational structures $\langle W, R, v \rangle$ where W is finite, $R \subseteq W \times W$ is transitive and $\langle W, R \rangle$ defines a *tree*.

Corollary (Decidability)

GL is an effectively decidable system.

Solovay Theorem

Completeness (Solovay 1976)

For any $A \in \text{Form}_{\mathcal{L}_\Box}$, if $\mathsf{I}\Sigma_1 \subseteq \mathsf{T}$ and $\mathbb{GL} \nvdash A$, then $\mathsf{T} \nvdash A^*$ for some realization $*$.

The only way to prove this result is by encoding in T a relational countermodel for A and "extract" from it a $*$ that does the job.

Modal Adequacy

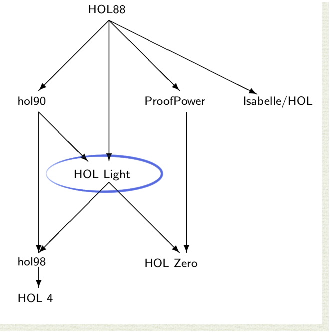
$$\mathbb{GL} \vdash A \quad \text{iff} \quad TFT \vDash A$$

where TFT is the class of relational structures $\langle W, R, v\rangle$ where W is finite, $R \subseteq W \times W$ is transitive and $\langle W, R\rangle$ defines a *tree*.

Corollary (Decidability)

GL is an effectively decidable system.

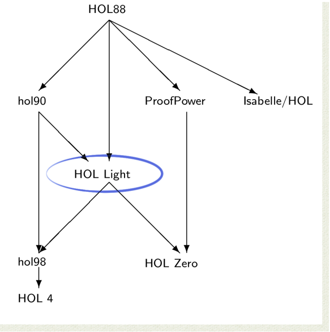
Brief glance at HOL Light



- Clean logical foundations ≈ *Principia Mathematica*
- LCF-style proof checker based on polymorphic simple type theory ≈ small class of *primitive inference rules* for creating theorems + *derived inference rules* to be programmed on top
  - ⇒ 10 primitive rules
  - ⇒ 2 conservative extension principles
  - ⇒ Axioms of choice, extensionality, and infinity
- Written as an OCaml program ≈ three datatypes for the logic: `hol_type`, `term`, and `thm`
- Goal-directed proof development ≈ tactic(al)s + automated methods (in the appropriate domains)

Despite its simple foundations, HOL Light includes a large library of mathematical results in topology, analysis, Euclidean geometry, QBF, arithmetic, FOL, ... and Incompleteness theorem!

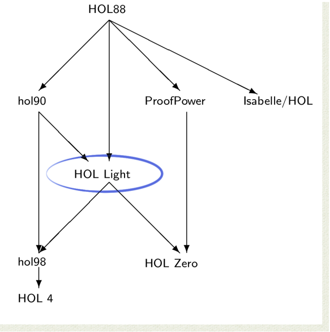
Brief glance at HOL Light



- Clean logical foundations ≈ *Principia Mathematica*
- LCF-style proof checker based on polymorphic simple type theory ≈ small class of *primitive inference rules* for creating theorems + *derived inference rules* to be programmed on top
  - ⇒ 10 primitive rules
  - ⇒ 2 conservative extension principles
  - ⇒ Axioms of choice, extensionality, and infinity
- Written as an OCaml program ≈ three datatypes for the logic: `hol_type`, `term`, and `thm`
- Goal-directed proof development ≈ tactic(al)s + automated methods (in the appropriate domains)

Despite its simple foundations, HOL Light includes a large library of mathematical results in topology, analysis, Euclidean geometry, QBF, arithmetic, FOL, ... and Incompleteness theorem!

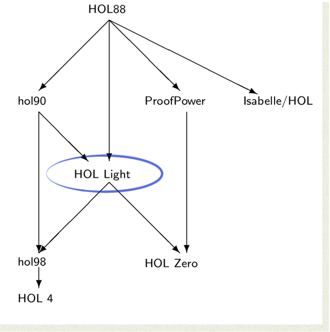
Brief glance at HOL Light



- Clean logical foundations ≈ *Principia Mathematica*
- LCF-style proof checker based on polymorphic simple type theory ≈ small class of *primitive inference rules* for creating theorems + *derived inference rules* to be programmed on top
  - ⇒ 10 primitive rules
  - ⇒ 2 conservative extension principles
  - ⇒ Axioms of choice, extensionality, and infinity
- Written as an OCaml program ≈ three datatypes for the logic: `hol_type`, `term`, and `thm`
- Goal-directed proof development ≈ tactic(al)s + automated methods (in the appropriate domains)

Despite its simple foundations, HOL Light includes a large library of mathematical results in topology, analysis, Euclidean geometry, QBF, arithmetic, FOL, ... and Incompleteness theorem!

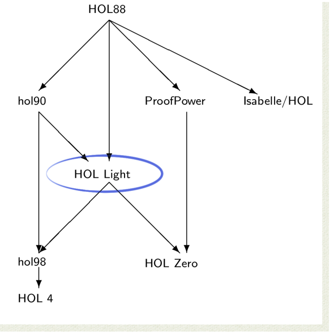
Brief glance at HOL Light



- Clean logical foundations ≈ *Principia Mathematica*
- LCF-style proof checker based on polymorphic simple type theory ≈ small class of *primitive inference rules* for creating theorems + *derived inference rules* to be programmed on top
  - ⇒ 10 primitive rules
  - ⇒ 2 conservative extension principles
  - ⇒ Axioms of choice, extensionality, and infinity
- Written as an OCaml program ≈ three datatypes for the logic: `hol_type`, `term`, and `thm`
- Goal-directed proof development ≈ tactic(al)s + automated methods (in the appropriate domains)

Despite its simple foundations, HOL Light includes a large library of mathematical results in topology, analysis, Euclidean geometry, QBF, arithmetic, FOL, ... and Incompleteness theorem!

Brief glance at HOL Light



- Clean logical foundations ≈ *Principia Mathematica*
- LCF-style proof checker based on polymorphic simple type theory ≈ small class of *primitive inference rules* for creating theorems + *derived inference rules* to be programmed on top
  - ⇒ 10 primitive rules
  - ⇒ 2 conservative extension principles
  - ⇒ Axioms of choice, extensionality, and infinity
- Written as an OCaml program ≈ three datatypes for the logic: `hol_type`, `term`, and `thm`
- Goal-directed proof development ≈ tactic(al)s + automated methods (in the appropriate domains)

Despite its simple foundations, HOL Light includes a large library of mathematical results in topology, analysis, Euclidean geometry, QBF, arithmetic, FOL, ... and Incompleteness theorem!

Syntax and relational semantics

A starting point

- We start with the inductive definition of our mono-modal propositional language, and with its interpretation w.r.t. standard relational structures;
- We identify the classes of structures we are interested in: transitive Noetherian frames, and more interestingly irreflexive transitive finite (`ITF`) frames.

N.B. The initial part of the formalization has been adapted from an embedding of the syntax and semantics of GL described in *The HOL Light Tutorial*. By now on, we develop the formalization towards different directions :

Give a formal study of the notions of *theoremhood* and *modal tautology* for GL, i.e. an adequacy theorem relating syntax and semantics for this specific logic.

Syntax and relational semantics

A starting point

- We start with the inductive definition of our mono-modal propositional language, and with its interpretation w.r.t. standard relational structures;
- We identify the classes of structures we are interested in: transitive Noetherian frames, and more interestingly irreflexive transitive finite (`ITF`) frames.

N.B. The initial part of the formalization has been adapted from an embedding of the syntax and semantics of GL described in *The HOL Light Tutorial*. By now on, we develop the formalization towards different directions :

Give a formal study of the notions of *theoremhood* and *modal tautology* for GL, i.e. an adequacy theorem relating syntax and semantics for this specific logic.

Our code

Our repository is publicly available from GitHub, and it is part of the official HOL Light distribution
`https://github.com/jrh13/hol-light/`, directory `GL`.

In the following, we briefly survey that code, partially relying on HOL Light syntax and vernacular.

Partial glossary

Informal notation	HOL notation	GL notation	Description
$\bot$	`F`	`False`	Falsity
$\top$	`T`	`True`	Truth
$\neg p$	`~ p`	`Not p`	Negation
$p \wedge q$	`/\`	`&&`	Conjunction
$p \vee q$	`\/`	`||`	Disjunction
$p \implies q$	`==>`	`-->`	Implication
$p \iff q$	`<=>`	`<->`	Biconditional
$\Box p$		`Box p`	Modal Operator
$p_1, \ldots p_N \vdash p$	`p1 ... pN |- p`		HOL theorem
$\vdash p$		`|-- p`	Derivability in $\mathbb{GL}$
$\mathcal{L} \vDash p$		`L |= p`	Validity in a class of frames $\mathcal{L}$
$\forall x.\, P(x)$	`!x. P(x)`		Universal quantification
$\exists x.\, P(x)$	`?x. P(x)`		Existential quantification
$\lambda x.\, M(x)$	`\x. M(x)`		Lambda abstraction
$x \in s$	`x IN s`		Set membership

Axiomatic Calculus

$\mathbb{GL}$

- Axioms of $\mathbb{CPC}$
- Axiom K : $\Box(A \to B) \to \Box A \to \Box B$
- Axiom GL : $\Box(\Box A \to A) \to \Box A$
- MP Rule $\dfrac{A \to B \qquad A}{B}$
- Nec Rule $\dfrac{A}{\Box A}$

Axiomatic Calculus, formalized

In HOL Light, we formalize that as

```
let GLaxiom_RULES,GLaxiom_INDUCT,GLaxiom_CASES = new_inductive_definition
  `(!p q. GLaxiom (p --> (q --> p))) /\
   (!p q r. GLaxiom ((p --> q --> r) --> (p --> q) --> (p --> r))) /\
   (!p. GLaxiom (((p --> False) --> False) --> p)) /\
   (!p q. GLaxiom ((p <-> q) --> p --> q)) /\
   (!p q. GLaxiom ((p <-> q) --> q --> p)) /\
   (!p q. GLaxiom ((p --> q) --> (q --> p) --> (p <-> q))) /\
   GLaxiom (True <-> False --> False) /\
   (!p. GLaxiom (Not p <-> p --> False)) /\
   (!p q. GLaxiom (p && q <-> (p --> q --> False) --> False)) /\
   (!p q. GLaxiom (p || q <-> Not(Not p && Not q))) /\
   (!p q. GLaxiom (Box (p --> q) --> Box p --> Box q)) /\
   (!p. GLaxiom (Box (Box p --> p) --> Box p))`;;
```

together with a derivability relation $|--$, operating on the previous axiom schemas by means of *modus ponens* and *necessitation*:

```
let GLproves_RULES,GLproves_INDUCT,GLproves_CASES =
   new_inductive_definition
   `(!p. GLaxiom p ==> |-- p) /\
     (!p q. |-- (p --> q) /\ |-- p ==> |-- q) /\
     (!p. |-- p ==> |-- (Box p))`;;
```

Modal Soundness

By applying induction on $|--$, it is not hard to prove that the calculus is sound w.r.t. our classes of frames:

Modal Soundness

```
let GL_TRANSNT_VALID = prove
 (`!p. (|-- p) ==> TRANSNT:(W->bool)#(W->W->bool)->bool |= p`,
  MATCH_MP_TAC GLproves_INDUCT THEN REWRITE_TAC[GLAXIOMS_TRANSNT_VALID]
  THEN MODAL_TAC);;


let GL_ITF_VALID = prove
 (`!p. |-- p ==> ITF:(W->bool)#(W->W->bool)->bool |= p`,
  GEN_TAC THEN STRIP_TAC THEN
  SUBGOAL_THEN `TRANSNT:(W->bool)#(W->W->bool)->bool |= p` MP_TAC THENL
  [ASM_SIMP_TAC[GL_TRANSNT_VALID];
   REWRITE_TAC[valid; FORALL_PAIR_THM] THEN MESON_TAC[ITF_NT]]);;
```

Corollary (Consistency)

```
let GL_consistent = prove
 (`~ |-- False`,
  REFUTE_THEN (MP_TAC o MATCH_MP (INST_TYPE [`:num`,`:W`] GL_ITF_VALID))
  THEN REWRITE_TAC[valid; holds; holds_in; FORALL_PAIR_THM;
                   ITF; NOT_FORALL_THM] THEN
  MAP_EVERY EXISTS_TAC [`{0}`; `\x:num y:num. F`] THEN
  REWRITE_TAC[NOT_INSERT_EMPTY; FINITE_SING; IN_SING] THEN MESON_TAC[]);;
```

Modal Soundness

By applying induction on $|--$, it is not hard to prove that the calculus is sound w.r.t. our classes of frames:

Modal Soundness

```
let GL_TRANSNT_VALID = prove
 (`!p. (|-- p) ==> TRANSNT:(W->bool)#(W->W->bool)->bool |= p`,
  MATCH_MP_TAC GLproves_INDUCT THEN REWRITE_TAC[GLAXIOMS_TRANSNT_VALID]
  THEN MODAL_TAC);;


let GL_ITF_VALID = prove
 (`!p. |-- p ==> ITF:(W->bool)#(W->W->bool)->bool |= p`,
  GEN_TAC THEN STRIP_TAC THEN
  SUBGOAL_THEN `TRANSNT:(W->bool)#(W->W->bool)->bool |= p` MP_TAC THENL
  [ASM_SIMP_TAC[GL_TRANSNT_VALID];
   REWRITE_TAC[valid; FORALL_PAIR_THM] THEN MESON_TAC[ITF_NT]]);;
```

Corollary (Consistency)

```
let GL_consistent = prove
 (`~ |-- False`,
  REFUTE_THEN (MP_TAC o MATCH_MP (INST_TYPE [`:num`,`:W`] GL_ITF_VALID))
  THEN REWRITE_TAC[valid; holds; holds_in; FORALL_PAIR_THM;
                   ITF; NOT_FORALL_THM] THEN
  MAP_EVERY EXISTS_TAC [`{0}`; `\x:num y:num. F`] THEN
  REWRITE_TAC[NOT_INSERT_EMPTY; FINITE_SING; IN_SING] THEN MESON_TAC[]);;
```

Modal Completeness

Our main goal has been proving in HOL Light the converse direction, namely

$$\text{If } \mathbb{GL} \nvdash A \text{ then } ITF \nvDash A$$

For many modal systems, it is common to use Henkin's "canonical model construction", where countermodels are made of maximal consistent sets of formulae and an appropriate accessibility relation.

These countermodels can then be filtrated, so that decidability of the system is obtained via the finite model property.

Notice, however, that for each modal system, a specific filtration is required.

Modal Completeness, formalized

The canonical model for $\mathbb{GL}$ *per se* does not belong to `ITF`, and some further constructions are necessary to derive a structure that does the job.

Boolos's textbook on provability logics shows that it is still possible to apply that style of reasoning to $\mathbb{GL}$, avoiding some problematic steps of Henkin's method.

By working with HOL Light, we show that, in fact, we can stick to the original idea by Henkin without invalidating the formalization, since all we need is

- Proving *several formal lemmas in* $\mathbb{GL}$ to reason about *finite* (maximal) consistent sets of formulae;
- Formalizing the key construction (`EXTEND_MAXIMAL_CONSISTENT`) by using *lists* (or finite sets) of formulae;
- Relying on the higher-order reasoning implemented in the system.

Modal Completeness, formalized

The canonical model for $\mathbb{GL}$ *per se* does not belong to `ITF`, and some further constructions are necessary to derive a structure that does the job.

Boolos's textbook on provability logics shows that it is still possible to apply that style of reasoning to $\mathbb{GL}$, avoiding some problematic steps of Henkin's method.

By working with HOL Light, we show that, in fact, we can stick to the original idea by Henkin without invalidating the formalization, since all we need is

- Proving *several formal lemmas in* $\mathbb{GL}$ to reason about *finite* (maximal) consistent sets of formulae;
- Formalizing the key construction (`EXTEND_MAXIMAL_CONSISTENT`) by using *lists* (or finite sets) of formulae;
- Relying on the higher-order reasoning implemented in the system.

Modal Completeness, formalized

The canonical model for $\mathbb{GL}$ *per se* does not belong to `ITF`, and some further constructions are necessary to derive a structure that does the job.

Boolos's textbook on provability logics shows that it is still possible to apply that style of reasoning to $\mathbb{GL}$, avoiding some problematic steps of Henkin's method.

By working with HOL Light, we show that, in fact, we can stick to the original idea by Henkin without invalidating the formalization, since all we need is

- Proving *several formal lemmas in* $\mathbb{GL}$ to reason about *finite* (maximal) consistent sets of formulae;
- Formalizing the key construction (`EXTEND_MAXIMAL_CONSISTENT`) by using *lists* (or finite sets) of formulae;
- Relying on the higher-order reasoning implemented in the system.

Modal Completeness, formalized

Completeness statement and explicit countermodel

```
COMPLETENESS_THEOREM
 |- !p. ITF:(form list->bool)#(form list->form list->bool)->bool |= p
         ==> |-- p

GL_COUNTERMODEL
 |- !M p.
      ~(|-- p) /\
      MAXIMAL_CONSISTENT p M /\ MEM (Not p) M /\
      (!q. MEM q M ==> q SUBSENTENCE p)
      ==>
      ~holds
         ({M | MAXIMAL_CONSISTENT p M /\ (!q. MEM q M ==> q SUBSENTENCE p)},
          GL_STANDARD_REL p)
         (\a w. Atom a SUBFORMULA p /\ MEM (Atom a) w)
         p M`
```

A better result, via bisimulation

Bisimulation

```
let BISIMIMULATION = new_definition
  `BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z <=>
   (!w1:A w2:B. Z w1 w2 ==> w1 IN W1 /\ w2 IN W2 /\
          (!a:string. V1 a w1 <=> V2 a w2) /\
          (!w1'. R1 w1 w1' ==> ?w2'. w2' IN W2 /\ Z w1' w2' /\ R2 w2 w2') /\
          (!w2'. R2 w2 w2' ==> ?w1'. w1' IN W1 /\ Z w1' w2' /\ R1 w1 w1'))`;;
```

Bisimulation Invariance Lemma

```
BISIMULATION_HOLDS
|- !W1 R1 V1 W2 R2 V2 Z p w1:A w2:B.
     BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z /\ Z w1 w2
     ==> (holds (W1,R1) V1 p w1 <=> holds (W2,R2) V2 p w2)
```

A better result, via bisimulation

Bisimulation

```
let BISIMIMULATION = new_definition
  `BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z <=>
   (!w1:A w2:B. Z w1 w2 ==> w1 IN W1 /\ w2 IN W2 /\
          (!a:string. V1 a w1 <=> V2 a w2) /\
          (!w1'. R1 w1 w1' ==> ?w2'. w2' IN W2 /\ Z w1' w2' /\ R2 w2 w2') /\
          (!w2'. R2 w2 w2' ==> ?w1'. w1' IN W1 /\ Z w1' w2' /\ R1 w1 w1'))`;;
```

Bisimulation Invariance Lemma

```
BISIMULATION_HOLDS
|- !W1 R1 V1 W2 R2 V2 Z p w1:A w2:B.
     BISIMIMULATION (W1,R1,V1) (W2,R2,V2) Z /\ Z w1 w2
     ==> (holds (W1,R1) V1 p w1 <=> holds (W2,R2) V2 p w2)
```

A better result, via bisimulation

We prove that the relation

```
\w1 w2. MAXIMAL_CONSISTENT p w1 /\ (!q. MEM q w1 ==> q SUBSENTENCE p) /\
        w2 IN GL_STDWORLDS p /\
        set_of_list w1 = w2
```

defines a bisimulation between the ITF-standard model based on maximal consistent *lists* of formulae and the model based on corresponding *sets* of formulae.

Finally, by the invariance principle stated by BISIMULATION_HOLDS, we have the desired version of modal completeness

Modal Completeness

```
COMPLETENESS_THEOREM_FINITE_SETS
 |- !p.
    ITF:((form->bool)->bool)#((form->bool)->(form->bool)->bool)->bool |= p
          ==> |-- p
```

A better result, via bisimulation

We prove that the relation

```
\w1 w2. MAXIMAL_CONSISTENT p w1 /\ (!q. MEM q w1 ==> q SUBSENTENCE p) /\
        w2 IN GL_STDWORLDS p /\
        set_of_list w1 = w2
```

defines a bisimulation between the ITF-standard model based on maximal consistent *lists* of formulae and the model based on corresponding *sets* of formulae.

Finally, by the invariance principle stated by BISIMULATION_HOLDS, we have the desired version of modal completeness

Modal Completeness

```
COMPLETENESS_THEOREM_FINITE_SETS
 |- !p.
     ITF:((form->bool)->bool)#((form->bool)->(form->bool)->bool)->bool |= p
           ==> |-- p
```

Application: Proving "formal" lemmas

- In natural deduction:

$$\frac{[A]}{A \to A} \to\!-intro$$

- In an axiomatic system:

1.	$(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$	Frege
2.	$A \to ((A \to A) \to A)$	a fortiori
3.	$(A \to (A \to A)) \to (A \to A)$	MP : $1, 2$
4.	$A \to (A \to A)$	a fortiori
5.	$A \to A$	MP : $3, 4$

- Does HOL Light automation help?

Application: Proving "formal" lemmas

- In natural deduction:

$$\frac{[A]}{A \to A} \to\!-intro$$

- In an axiomatic system:

1.	$(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$	Frege
2.	$A \to ((A \to A) \to A)$	a fortiori
3.	$(A \to (A \to A)) \to (A \to A)$	MP : $1, 2$
4.	$A \to (A \to A)$	a fortiori
5.	$A \to A$	MP : $3, 4$

- Does HOL Light automation help?

Application: Proving “formal” lemmas

- In natural deduction:

$$\frac{[A]}{A \to A} \to\!-intro$$

- In an axiomatic system:

1.	$(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$	Frege
2.	$A \to ((A \to A) \to A)$	a fortiori
3.	$(A \to (A \to A)) \to (A \to A)$	MP $: 1, 2$
4.	$A \to (A \to A)$	a fortiori
5.	$A \to A$	MP $: 3, 4$

- Does HOL Light automation help?

Proving 𝔾𝕃-lemmas, formally

Identity law

```
let GL_imp_refl_th = prove
 (`!p. |-- (p --> p)`,
  MESON_TAC[GL_modusponens; GL_axiom_distribimp; GL_axiom_addimp]);;
```

□ over ↔

```
let GL_box_iff_th = prove
 (`!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`,
  REPEAT GEN_TAC THEN MATCH_MP_TAC GL_imp_trans THEN
  EXISTS_TAC `(Box p --> Box q) && (Box q --> Box p)` THEN CONJ_TAC THENL
  [ALL_TAC; MATCH_MP_TAC GL_iff_imp2 THEN MATCH_ACCEPT_TAC GL_iff_def_th]
  THEN MATCH_MP_TAC GL_and_intro THEN CONJ_TAC THENL
  [MATCH_MP_TAC GL_imp_trans THEN EXISTS_TAC `Box (p --> q)` THEN
   REWRITE_TAC[GL_axiom_boximp] THEN MATCH_MP_TAC GL_imp_box THEN
   MATCH_ACCEPT_TAC GL_axiom_iffimp1;
   MATCH_MP_TAC GL_imp_trans THEN EXISTS_TAC `Box (q --> p)` THEN
   REWRITE_TAC[GL_axiom_boximp] THEN MATCH_MP_TAC GL_imp_box THEN
   MATCH_ACCEPT_TAC GL_axiom_iffimp2]);;
```

Leaving proof-search to HOL Light toolbox doesn't seem promising. . .

Proving $\mathbb{GL}$-lemmas, formally

Identity law

```
let GL_imp_refl_th = prove
 (`!p. |-- (p --> p)`,
  MESON_TAC[GL_modusponens; GL_axiom_distribimp; GL_axiom_addimp]);;
```

$\Box$ over $\leftrightarrow$

```
let GL_box_iff_th = prove
 (`!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`,
  REPEAT GEN_TAC THEN MATCH_MP_TAC GL_imp_trans THEN
  EXISTS_TAC `(Box p --> Box q) && (Box q --> Box p)` THEN CONJ_TAC THENL
  [ALL_TAC; MATCH_MP_TAC GL_iff_imp2 THEN MATCH_ACCEPT_TAC GL_iff_def_th]
  THEN MATCH_MP_TAC GL_and_intro THEN CONJ_TAC THENL
  [MATCH_MP_TAC GL_imp_trans THEN EXISTS_TAC `Box (p --> q)` THEN
   REWRITE_TAC[GL_axiom_boximp] THEN MATCH_MP_TAC GL_imp_box THEN
   MATCH_ACCEPT_TAC GL_axiom_iffimp1;
   MATCH_MP_TAC GL_imp_trans THEN EXISTS_TAC `Box (q --> p)` THEN
   REWRITE_TAC[GL_axiom_boximp] THEN MATCH_MP_TAC GL_imp_box THEN
   MATCH_ACCEPT_TAC GL_axiom_iffimp2]);;
```

Leaving proof-search to HOL Light toolbox doesn't seem promising. . .

Proving $\mathbb{GL}$-lemmas, cleverly

Having `COMPLETENESS_THEOREM_FINITE_SETS` we can approach the task by:

1. Applying the theorem;
2. Unfolding some definitions;
3. Trying to solve the resulting *semantic* problem by using automated first-order reasoning.

A tactic (and a rule) for $\mathbb{GL}$-derivability

```
let GL_TAC : tactic =
    MATCH_MP_TAC COMPLETENESS_THEOREM_NUM THEN
    REWRITE_TAC[valid; FORALL_PAIR_THM; holds_in; holds;
                ITF; GSYM MEMBER_NOT_EMPTY] THEN
    MESON_TAC[];;

let GL_RULE tm = prove(tm, REPEAT GEN_TAC THEN GL_TAC);;
```

Proving $\mathbb{GL}$-lemmas, cleverly

Having COMPLETENESS_THEOREM_FINITE_SETS we can approach the task by:

1. Applying the theorem;
2. Unfolding some definitions;
3. Trying to solve the resulting *semantic* problem by using automated first-order reasoning.

A tactic (and a rule) for $\mathbb{GL}$-derivability

```
let GL_TAC : tactic =
    MATCH_MP_TAC COMPLETENESS_THEOREM_NUM THEN
    REWRITE_TAC[valid; FORALL_PAIR_THM; holds_in; holds;
                ITF; GSYM MEMBER_NOT_EMPTY] THEN
    MESON_TAC[];;

let GL_RULE tm = prove(tm, REPEAT GEN_TAC THEN GL_TAC);;
```

Proving $\mathbb{GL}$-lemmas, cleverly

Almost there!

□ over ↔, again

```
# GL_RULE `!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`;;
    0..0..1..6..11..19..32..solved at 39
    0..0..1..6..11..19..32..solved at 39
  val it : thm = |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

In spite of this, the tactic needs to be improved:

```
GL_RULE `|-- (Box (Box False --> False) --> Box False)`;;
0..0..0..4..8..12..20..28..61..105..150..228..314..425..565..707..887..
1123..1397..1733..2128..2574..3101..3804..4572..5435..6457..7611..8898..
10310..11841..13585..15681..17896..20343..23033..25840..29215..33310..
37964..43266..49063..55099..61633..68918..76664..84798..93855..104554..
117586..132638..Exception: Failure "solve_goal: Too deep".
```

Proving $\mathbb{GL}$-lemmas, cleverly

Almost there!

□ over ↔, again

```
# GL_RULE `!p q. |-- (Box (p <-> q) --> (Box p <-> Box q))`;;
    0..0..1..6..11..19..32..solved at 39
    0..0..1..6..11..19..32..solved at 39
  val it : thm = |- !p q. |-- (Box (p <-> q) --> (Box p <-> Box q))
```

In spite of this, the tactic needs to be improved:

```
GL_RULE `|-- (Box (Box False --> False) --> Box False)`;;
0..0..0..4..8..12..20..28..61..105..150..228..314..425..565..707..887..
1123..1397..1733..2128..2574..3101..3804..4572..5435..6457..7611..8898..
10310..11841..13585..15681..17896..20343..23033..25840..29215..33310..
37964..43266..49063..55099..61633..68918..76664..84798..93855..104554..
117586..132638..Exception: Failure "solve_goal: Too deep".
```

Conclusions

Our repository shows that we can:

- Define the derivability relation w.r.t. the axiomatic system $\mathbb{GL}$;
- Prove in HOL Light several lemmas (approx. 120) *within* $\mathbb{GL}$;
- Formalize modal completeness in a very natural way, by relying on HOL Light toolbox.

As by-products, we have obtained:

- A "kernel" for further experiments on reasoning *within* and *about* modal axiomatic calculi by using HOL Light;
- An empirical analysis of "friendliness" and efficiency of the system w.r.t. proof automation for axiomatic calculi;
- An application of the bisimulation lemma with an eye on complexity issues.

Future directions:

- Integration with `Arithmetic` repository, and Solovay theorem?
- Improvement of `GL_TAC`/decision procedure, and mechanization of proof-search?

Conclusions

Our repository shows that we can:

- Define the derivability relation w.r.t. the axiomatic system $\mathbb{GL}$;
- Prove in HOL Light several lemmas (approx. 120) *within* $\mathbb{GL}$;
- Formalize modal completeness in a very natural way, by relying on HOL Light toolbox.

As by-products, we have obtained:

- A "kernel" for further experiments on reasoning *within* and *about* modal axiomatic calculi by using HOL Light;
- An empirical analysis of "friendliness" and efficiency of the system w.r.t. proof automation for axiomatic calculi;
- An application of the bisimulation lemma with an eye on complexity issues.

Future directions:

- Integration with `Arithmetic` repository, and Solovay theorem?
- Improvement of `GL_TAC`/decision procedure, and mechanization of proof-search?

Conclusions

Our repository shows that we can:

- Define the derivability relation w.r.t. the axiomatic system $\mathbb{GL}$;
- Prove in HOL Light several lemmas (approx. 120) *within* $\mathbb{GL}$;
- Formalize modal completeness in a very natural way, by relying on HOL Light toolbox.

As by-products, we have obtained:

- A "kernel" for further experiments on reasoning *within* and *about* modal axiomatic calculi by using HOL Light;
- An empirical analysis of "friendliness" and efficiency of the system w.r.t. proof automation for axiomatic calculi;
- An application of the bisimulation lemma with an eye on complexity issues.

Future directions:

- Integration with `Arithmetic` repository, and Solovay theorem?
- Improvement of `GL_TAC`/decision procedure, and mechanization of proof-search?

Many thanks
for your attention!