Formalizing Basic First Order Model Theory

John Harrison

Intel Corporation

- What and why?
- Plan of the talk
- Kreisel and Krivine's proof
- First order logic with equality
- Highlights of the HOL formalization
- An example: the hyperreals

What and why? (1)

There are a number of ‘classical’ results often proved in a first or second course on logic, all for classical first order logic:

- The Compactness theorem: if every finite subset of a set of formulas Δ has a model, then so does all of Δ.
- The (downward) Löwenheim-Skolem theorem: if a set of formulas (in a countable language) has a model then it has a model whose domain is a subset of $\mathbb{N}$.
- The Uniformity theorem (often called Herbrand's theorem): if the sentence $\exists x_1, \ldots, x_n.\ P[x_1, \ldots, x_n]$ holds in all models, then so does some disjunction of instances:

$$P[t_1^1, \ldots, t_n^1] \lor \ldots \lor P[t_1^k, \ldots, t_n^k]$$

What and why? (2)

All these results are purely semantic, and make no mention of provability or formal systems. However, it's commonest to prove them using the completeness theorem and syntactic arguments.

As part of our hobby of writing a textbook on logic, we wanted to present simple and elegant proofs that only use semantic notions.

Just such proofs are given by Kreisel and Krivine in their textbook on model theory (published by North-Holland, 1967).

We took the K&K proofs as our model, but wanted to check that we got all the details right, and see how the proofs could be improved. This is why we decided to formalize them in HOL.

Plan of the talk

Some work in formalizing logical notions inside theorem provers is motivated by ideas of using it for meta-level theorem proving or reflection. However, some of it is just for fun or curiosity about how difficult it is, e.g. Shankar's NQTHM proof of Gödel's First Incompleteness Theorem.

Our work belongs to the second category. However, we will show that it is certainly possible to apply it to other areas, by giving a simple construction of the hyperreals and proof of the transfer principle used in Nonstandard Analysis.

Since the details of the proofs are a bit much to get through in half an hour, we'll just present some highlights, draw some conclusions about how well the formalization went, then run through the hyperreal example. See the paper for more details of the proofs.

Kreisel and Krivine's proof

The idea behind K&K's proof is to proceed in three stages:

- Propositional logic (no quantifiers)
- First order logic (no special interpretation of equality relation)
- First order logic with equality (normal models only)

Each step builds on the previous one, without adding too much work. The most subtle part is the first jump, where we make the link between quantifier-free formulas of first order logic and formulas of propositional logic, with the help of Skolemization.

Formalization of syntax

We start by defining types of terms:

```
term = V num
     | Fn num (term list)
```

and formulas:

```
form = False
     | Atom num (term list)
     | --> form form
     | !! num form
```

Note that functions with the same tag and different arities are different. Connectives other than the primitive ones are defined in a fairly standard way, e.g.

```
|- Not p = p --> False
|- p || q = (p --> q) --> q
|- ?? x p = Not(!!x (Not p))
```

Syntactic notions

We define various syntactic notions such as the free variables in a term and in a formula:

```
|- (FVT (V x) = x INSERT EMPTY) /\
   (FVT (Fn f l) = LIST_UNION (MAP FVT l))
```

```
|- (FV False = EMPTY) /\
   (FV (Atom a l) =
       LIST_UNION (MAP FVT l)) /\
   (FV (p --> q) = FV p UNION FV q) /\
   (FV (!! x p) = FV p DELETE x)
```

where:

```
|- (LIST_UNION [] = EMPTY) /\
   (LIST_UNION (CONS h t) =
       h UNION LIST_UNION t)
```

Substitution

Substitution in terms is quite simple:

```
|- (termsubst v (V x) = v x) /\
   (termsubst v (Fn f l) =
       Fn f (MAP (termsubst v) l))
```

As usual, bound variables make the definition of substitution at the formula level a bit more complicated:

```
|- (formsubst v False = False) /\
   (formsubst v (Atom p l) =
       Atom p (MAP (termsubst v) l)) /\
   (formsubst v (q --> r) =
       formsubst v q --> formsubst v r) /\
   (formsubst v (!! x q) =
    let v' = valmod (x,V x) v in
    let z =
       if ?y. y IN FV (!!x q) /\ x IN FVT (v' y)
       then VARIANT (FV(formsubst v' q))
       else x in
    !!z (formsubst (valmod (x,V z) v) q))
```

where

```
|- valmod (x,a) v = (\y. if y = x then a else v y)
```

Semantics

This is the key part of the work. We define the value given to a term by a particular interpretation and valuation:

```
|- (termval M v (V x) = v x) /\
   (termval M v (Fn f l) =
        Fun M f (MAP (termval M v) l))

|- (holds M v False = F) /\
   (holds M v (Atom a l) =
        Pred M a (MAP (termval M v) l)) /\
   (holds M v (p --> q) =
        holds M v p ==> holds M v q) /\
   (holds M v (!! x p) =
        !a. a IN Dom M
            ==> holds M (valmod(x,a) v) p))
```

Useful theorems

The following expected results help to reassure us that the definitions are correct. First, only the effect of the valuation on the free variables matters:

```
|- (!x. x IN FV p ==> (v' x = v x))
     ==> (holds M v' p = holds M v p)
```

and the free variables of a substituted formula are:

```
|- FV (formsubst i p) =
     {x | ?y. y IN FV p /\ x IN FVT (i y)}
```

and the following covers whether a substituted formula holds:

```
|- holds M v (formsubst i p) =
   holds M (termval M v o i) p
```

Propositional logic

The first stage in the proof is to prove compactness for propositional logic. Here we have a different notion of a formula holding, under a 'valuation' that is now a truth assignment to atomic formulas:

```
|- (pholds d False = F) /\
   (pholds d (Atom p l) = d (Atom p l)) /\
   (pholds d (q --> r) =
       pholds d q ==> pholds d r) /\
   (pholds d (!!x q) = d (!!x q))
```

The key theorem, established by a fairly easy application of Zorn's Lemma, is compactness:

```
|- !s. psatisfiable s = ?v. !p. p IN s ==> pholds v p

|- (!p. p IN A ==> qfree p) /\
   (!B. FINITE B /\ B SUBSET A ==> psatisfiable B)
   ==> psatisfiable A
```

where `qfree` is a recursively defined predicate defining the set of quantifier-free formulas.

Skolemization

We prove that every set of formulas has a Skolem normal form.

```
|- qfree(SKOLEM p)

|- (?M. ~(Dom M = EMPTY) /\
        interpretation (language s) M /\
        M satisfies s) =
   (?M. ~(Dom M = EMPTY) /\
        interpretation
          (language {SKOLEM p | p IN s}) M /\
        M satisfies {SKOLEM p | p IN s})
```

We need to have an actual Skolemizing function rather than a pure existence assertions because of troubles with type variables.

The Skolemization is actually the hardest part of the proof by far because we need to show we can pick new Skolem functions independently for different formulas in the set. So although using Skolemization is intuitive for people, the details are still there under the surface.

Propositional vs. first order logic (1)

The relation between propositional and first order satisfiability of quantifier-free formulas needs a bit of care. We can switch between first order interpretation-valuation pairs and propositional valuations as follows:

```
|- prop_of_model M v (Atom p l) =
     holds M v (Atom p l)

|- canon_of_prop L d =
     terms (FST L),
     Fn,
     (\p l. d (Atom p l))
```

The model created by `canon_of_prop` is *canonical*, meaning that it is a model over the terms of the language with functions given their 'natural' interpretation as syntax constructors.

Propositional vs. first order logic (2)

The key relation between the first order and propositional worlds is:

```
|- qfree(p)
   ==> (pholds (prop_of_model M v) p = holds M v p)

|- qfree p ==> (holds (canon_of_prop L d) V p =
                         pholds d p)
```

This implies that a quantifier-free formula is a tautology iff it is valid in all (or just all canonical) models:

```
|- qfree(p) /\ (!d. pholds d p)
   ==> !M v. holds M v p

|- qfree(p) /\
   (!C v. canonical(language {p}) C ==> holds C v p)
   ==> !d. pholds d p
```

Propositional vs. first order logic (3)

This is not true for satisfiability. For example $P(x) \wedge \neg P(y)$ is propositionally satisfiable but not first order satisfiable. However we do get the same thing if we insist that *all substitution instances* are propositionally satisfiable:

```
 |- (!p. p IN s ==> qfree p) /\
     M satisfies s /\
     valuation(M) v
     ==> (prop_of_model M v) psatisfies s

 |- (!p. p IN s ==> qfree p) /\
    d psatisfies
      {formsubst v p | (!x. v x IN terms(FST L)) /\
                       p IN s}
    ==> (canon_of_prop L d) satisfies s
```

This immediately yields the Löwenheim-Skolem theorem, since a canonical model is countable.

Main results

With only slightly more effort we get the Compactness theorem at the same time:

```
 |- (!t. FINITE t /\ t SUBSET s
         ==> ?M. interpretation(language s) M /\
                 ~(Dom(M):A->bool = EMPTY) /\
                 M satisfies t)
   ==> ?C. interpretation (language s) C /\
           ~(Dom(C):term->bool = EMPTY) /\
           C satisfies s
```

The Uniformity theorem also follows easily. Using the above result on propositional validity of substitution instances seems to give more elegant and unified proofs of these results than those given by K&K.

First order logic with equality

It's straightforward to transform all these theorems into the world of first order logic with equality, where we restrict ourselves to *normal* models:

```
|- normal fns M =
        !s t v. valuation M v /\
                s IN terms fns /\ t IN terms fns
                ==> (holds M v (s == t) =
                     termval M v s = termval M v t)
```

The key lemma is:

```
|- (?M. interpretation (language s) M /\
        ~(Dom M = EMPTY) /\
        normal (functions s) M /\
        M satisfies s) =
   (?M. interpretation (language s) M /\
        ~(Dom M = EMPTY) /\
        M satisfies (s UNION Eqaxioms (language s)))
```

where `Eqaxioms` defines the set of 'equality axioms' (equivalence and congruence properties).

The reals as a model

For example, we can set up the HOL theory of reals as a model for a suitable formal language:

```
 |- RM = (UNIV,
          (\f l. if       f = 0 then &0
                 else if f = 1 then &1
                 else if f = 2 then pi
                 else if f = 3 then --(EL 0 l)
                 else if f = 4 then inv(EL 0 l)
                 else if f = 5 then abs(EL 0 l)
                 else if f = 6 then exp(EL 0 l)
                 ...
                 else if f = 13 then acs(EL 0 l)
                 else if f = 14 then atn(EL 0 l)
                 else if f = 15 then EL 0 l + EL 1 l
                 else if f = 16 then EL 0 l * EL 1 l
                 else if f = 17 then EL 0 l / EL 1 l
                 else @x. T),
          (\p l. if       p = 0 then (EL 0 l = EL 1 l)
                 else if p = 1 then EL 0 l <= EL 1 l
                 else if p = 2 then EL 0 l < EL 1 l
                 else if p = 3 then EL 0 l >= EL 1 l
                 else if p = 4 then EL 0 l > EL 1 l
                 else @x. T)
```

Axioms for the hypperreals

Now we define two sets of 'axioms', first, the set of all assertions in a language not using constant number 18 that are true for the reals:

```
 |- Real_axioms = { p | ~((18,0) IN functions {p}) /\
                        !v. holds RM v p }
```

and a set of assertions that the constant number 18 is bigger than all natural numbers:

```
  |- Infinity_axioms =
        { Atom 4 [Fn 18 []; Real_n n] | T }
```

where

```
 |- (Real_n 0 = Fn 0 []) /\
    (Real_n (SUC n) = Fn 15 [Fn 1 []; Real_n n])
```

The denotation of `Real_n n` in the standard model is just `&n`, the real constant n.

The hypperreals

It's now a straightforward application of compactness to show that we can find a model of all of these together, and we call this HYPM:

```
 |- interpretation
      (language(Real_axioms UNION Infinity_axioms))
      HYPM /\
    ~(Dom HYPM = EMPTY) /\
    normal
      (functions(Real_axioms UNION Infinity_axioms))
      HYPM /\
    HYPM satisfies (Real_axioms UNION Infinity_axioms)
```

Since a statement either holds in a model or it doesn't, we can show that the two models behave exactly the same for closed sentences not involving the constant 18, e.g.

```
 |- (FV(p) = EMPTY) /\
    ~((18,0) IN functions {p})
    ==> ((!v. holds RM v p) =
         (!v. valuation(HYPM) v ==> holds HYPM v p))
```

Conclusions

- Proofs generally work OK in HOL, but sometimes the types get in the way and keeping track of the language of terms is tedious.
- In some ways we have improved on the textbook originals, e.g. by isolating the lemma about all substitution instances holding propositionally.
- We have shown a simple construction of the hypperreals that yields the 'transfer' principle immediately. Using this, one could work in the hypperreals and transfer results back and forth.