Meta-Logical Reasoning in Higher-Order Logic

Villadsen, Jørgen; Schlichtkrull, Anders; Hess, Andreas Viktor

Publication date:
2015

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Meta-Logical Reasoning in Higher-Order Logic

Jørgen Villadsen, Anders Schlichtkrull & Andreas Viktor Hess
Technical University of Denmark - DTU Compute

Abstract

The terms and formulas of the FOL language are defined as the datatypes tm and fm, respectively. Variables are indexed using de Bruijn indices. The semantics of the language is defined using the function \( \text{sem} \) where \( e \) is the environment, i.e. a mapping of variables to elements of the universe \('u\). \( f \) maps constants to elements of \('u\), and \( g \) gives the semantics of the predicates. Most of the cases of \( \text{sem} \) should be self-explanatory, but the \( \text{Uni} \) case is complicated. The details are not important here, but it uses the universal quantifier (\( ! \)) to consider all values of the universe \('u\). It also uses the lambda operator (\( \% \)) to keep track of the indices of the variables.

Examples of Meta-Logical Reasoning

By using HOL as the meta-language for FOL it is possible to make use of proof assistants such as Isabelle to reason about certain properties of FOL. For example, let \( \text{syn} \) be a proof system for FOL implemented as a predicate in Isabelle of type \( \text{fm} \rightarrow \text{bool} \) (an inductive definition). If it is sound then we can prove this in Isabelle as the theorem \( \text{syn} \text{ fml } \rightarrow \text{ f g . sem e f g fml} \). Likewise completeness \( \text{ ( e f g , sem e f g fml ) } \rightarrow \text{ syn fml } \) can be proved if the proof system is indeed complete. For example, in Berghofer (2007) a natural deduction proof system for FOL is proven sound and complete. A tool for teaching logic based on natural deduction has recently been developed and proved sound by Villadsen (2015).

Meta-logical reasoning using the formalized semantics also enables formal proofs of sentences using only the semantics of the object language. For example, we wish to show that the sentence \( \forall x . \forall y . P(y, x) \rightarrow P(y, x) \) is valid. We can prove this by first fixing two arbitrary elements \( u, w \) of the universe and show \( P(y, x) \rightarrow P(y, x) \) for an arbitrary environment updated to map variable \( x \) to \( u \) and \( y \) to \( w \), and then use the fact that any denotation \( g \) of predicates maps \( (P, u, w) \) to either true or false. In both cases \( P(y, x) \rightarrow P(y, x) \) holds. This proof sketch can then be extended to a readable proof in Isabelle. The sentence can even be proved automatically in Isabelle using the formalized semantics:

\[
\text{theorem } \, ! e f g . \, \text{sem } e f g (\text{Uni } (\text{Uni } \text{(Pre } 'P' \text{ (Var 0) (Var 1)})) \text{ (Pre } 'P' \text{ (Var 0) (Var 1)}))) \text{ by auto}
\]

On the other hand the sentence \( \forall x . \forall y . P(x, y) \rightarrow P(y, x) \) is not valid. Therefore we can ask Isabelle to search for a counterexample, using the \text{nitpick} command, and it is able to find one.

Finally, the approach we have presented can even be used with HOL as the object language, that is, the semantics of HOL and a proof system for HOL can also be formalized in HOL, most extensively by Kumar (2014), although of course self-verification is not possible due to the second incompleteness theorem. Paulson (2014) has recently formalized the incompleteness theorems in Isabelle.

Formalization in Isabelle

We consider a formalization in Isabelle of FOL with only binary predicates:

```
theory Semantics imports Main begin

type_synonym id = string

datatype tm = Var nat | Con id

datatype fm = Falsity | Pre id tm tm | Imp fm fm | Uni fm

primrec val :: "\( (nat \rightarrow 'u) \rightarrow (id \rightarrow 'u) \rightarrow tm \rightarrow 'u \)" where
"val e f (Var v)=ev "|
"val e f (Con c)=fc "

primrec sem :: "\( (nat \rightarrow 'u) \rightarrow (id \rightarrow 'u) \rightarrow (id \rightarrow 'u * 'u \rightarrow bool) \rightarrow fm \rightarrow bool \)" where
"sem e f g Falsity = False" |
"sem e f g (Pre a b) = g a (val e f a, val e f b) " |
"sem e f g (Imp p q) = (if sem e f g p then sem e f g else True) " |
"sem e f g (Uni p) = \( \lambda x . \text{sen } (\text{Xn. if n=0 then x else e } (n \cdot 1) ) f g p)\)"

end
```

The terms and formulas of the FOL language are defined as the datatypes tm and fm, respectively. Variables are indexed using de Bruijn indices. The semantics of the language is defined using the function \( \text{sem} \) where \( e \) is the environment, i.e. a mapping of variables to elements of the universe \('u\). \( f \) maps constants to elements of \('u\), and \( g \) gives the semantics of the predicates. Most of the cases of \( \text{sem} \) should be self-explanatory, but the \( \text{Uni} \) case is complicated. The details are not important here, but it uses the universal quantifier (\( ! \)) to consider all values of the universe \('u\). It also uses the lambda operator (\( \% \)) to keep track of the indices of the variables.

References


Villadsen, J., A. B. Jensen & A. Schlichtkrull (2015). NaDeA: A Natural Deduction Assistant with a Formal-

End