AN EXAMPLE OF FOL USING METATHEORY
FORMALIZING REASONING SYSTEMS
AND INTRODUCING DERIVED INFERENCE RULES

Richard W. Weyhrauch
Stanford University
Stanford, CA 94305

1. Introduction

This paper shows how FOL, Weyhrauch [1977, 1980], can be used to formalize and implement reasoning systems. The reasoning system we have chosen as an example is the implicational part of the propositional logic $P_1$ described in Church [1956, p. 72. The formulas of $P_1$ are either sentential constants, $\textsc{sentconsts}$, or are built up from other formulas using the implication symbol, $\supset$. For any formulas $A$, $B$, and $C$ the following are axioms:

$$\text{AXIOM 1}(A, B) \quad (A \supset (B \supset A))$$

$$\text{AXIOM 2}(A, B, C) \quad ((A \supset (B \supset C)) \supset ((A \supset B) \supset (A \supset C))).$$

We say that each axiom is a theorem and that, if $A$ is a theorem and $A \supset B$ is a theorem, then $B$ is a theorem. The latter inference rule is called modus ponens.

Suppose $A$ is some particular sentential constant. The following is a derivation showing that the particular formula $A \supset A$ is a theorem.

1. $A \supset ((A \supset A) \supset A)$ \hspace{1cm} AXIOM 1($A, A \supset A$)
2. $(A \supset ((A \supset A) \supset A)) \supset ((A \supset (A \supset A)) \supset (A \supset A))$ \hspace{1cm} AXIOM 2($A, A \supset A, A$)
3. $(A \supset (A \supset A)) \supset (A \supset A)$ \hspace{1cm} Modus Ponens 1, 2
4. $A \supset (A \supset A)$ \hspace{1cm} AXIOM 1($A, A$)
5. $A \supset A$ \hspace{1cm} Modus Ponens 4, 3

It is clear from the above proof that we could have used any formula in place of $A$. This suggests the following derived rule.

**OBVIOUS:** If $A$ is a formula then $A \supset A$ is a theorem.

The meta-theorem OBVIOUS although simple is a typical example of a derived rule that we might want to add to our system.

In section 2 we give the construction of META, an FOL context which formalizes the syntax and inference rules of $P_1$. (In Prolegomena [1980], FOL contexts were called $L/S$ pairs.) This formalization follows Church quite closely. Unfortunately Church's description of modus ponens is not in a form that we can use. We discuss the reason for this in section 3 where we also use the theorem-generating features of FOL to derive a more useful statement of the rule. In section 4 we define the FOL context, THEORY, which implements the reasoning system entailed by the declarative description of $P_1$ in META. After demonstrating how to use the FOL reflection principles to generate proofs in THEORY, we return our attention to META, and in section 5 we use the axiomatization of $P_1$ to prove the theorem OBVIOUS so that it can be used as a new derived rule and thus extends the expressive power of the system.

I am presenting this simple example because it is completely self contained and I do not want the details of the example to obscure the details of how the FOL features work. The important point here is not the complexity of the example, but rather the details of how FOL provides a formal framework for constructing reasoning systems in which we can both prove the correctness of the new reasoning principles and add them to the reasoning machinery available to us. The ability to reason about the inference rules essentially means that we can add provably correct new rules to our system.

The techniques described here have been used in the current FOL system to build a context META that contains an almost complete axiomatization of the FOL formalism itself. This formalism will be described in detail in Weyhrauch [1982]. This axiomatization has been used to do extremely complex examples. Aiello and Weyhrauch [1980] have used it for algebraic simplification. Talcott and Weyhrauch [1982] use it for reasoning about actions in a new formulation of the McCarthy situation calculus [McCarthy and Hayes [1980]]. The example of this paper was first run on the FOL system in the summer of 1977.
2. Formalizing Implicational Propositional Logic

We start by describing the propositional language of $P_i$ to FOL. The text below contains the actual FOL commands necessary to carry out this example. The commands to FOL are preceded by 5 stars, *****. The FOL response appears as lines of text beginning with line numbers. Even though this is a little harder to read than an explanation in English, I want to give some flavor of the complexity of interacting with FOL and to present a complete example.

**Well-formed formulas**, WFFs, are built by starting with **sentential constants**, SENTCONSTs, and using the function **make implication**, mkimp, to build new formulas out of old ones. The first command directs FOL's attention to the context META.

***** CHANGE TO META;
***** DECLARE SORT SENTCONST;
***** DECLARE INDVAR sentconst $\in$ SENTCONST;
***** DECLARE SORT WFF;
***** DECLARE INDVAR wff1, wff2, wff3 $\in$ WFF;
***** DECLARE OPCONST mkimp 2;

FOL has no defaults so it must be told everything. In particular since each FOL context has its own language we must tell it what identifiers are used in what way. The first and third FOL commands make SENTCONST and WFF into sorts, i.e., predicate constants of one argument. We also declare that SENTCONST is a variable which ranges over sentential constants and wff1, wff2, and wff3 are variables over well-formed formulas. mkimp is declared to be a function symbol (or operation constant) of two arguments. The axioms

***** AXIOM WFF1: $\forall$sentconst.WFF(sentconst) ;
WFF2: $\forall$wff1 wff2.WFF(mkimp(wff1,wff2)) ;;

state that every thing of sort SENTCONST is of sort WFF and that mkimp maps WFFs onto WFFs. In FOL another way of specifying this is to say that the sort WFF is more general than the sort SENTCONST; and to use the function map, FMAP, command to specify the sort of the value of a function given the sorts of its arguments. The commands are:

***** MOREGENERAL WFF $\geq$ {SENTCONST};
***** FMAP mkimp(WFF,WFF)=WFF;

Now that we know what formulas are, we need some facts about **implications**, IMPs.

***** DECLARE SORT IMP;
***** DECLARE OPCONST hypof(IMP)=WFF, concl(IMP)=WFF;

***** AXIOM IMPL1: $\forall$wff1 wff2.IMP(mkimp(wff1,wff2)) ;
IMPL2: $\forall$wff1 wff2.hypof(mkimp(wff1,wff2))=wff1 ;
IMPL3: $\forall$wff1 wff2.concl(mkimp(wff1,wff2))=wff2 ;
IMPL4: $\forall$wff1.(IMP(wff1)~wff1=mkimp(hypof(wff1),concl(wff1))) ;;

This axiom states the syntactic properties of implications: 1) if you make an implication (mkimp) of any two formulas, then it is an implication (IMP); 2) the hypothesis (hyp0f) if an implication is its first component; 3) the conclusion (concl) of an implication is its second component; 4) If you have an implication then you can reconstruct it by making an implication out of its hypothesis and its conclusion. These are facts about formulas.

We introduce the sort THEOREM.

***** DECLARE SORT THEOREM;

The idea that the axioms are theorems is formalized by the axioms

***** AXIOM
HILBERT1: Wff1 wff2.THEOREM(
mkimp(wff1,(mkimp(wff2,wff1))));
HILBERT2: Wff1 wff2 wff3.THEOREM(
mkimp(
mkimp(wff1,mkimp(wff2,wff3)),
mkimp(mkimp(wff1,wff2),mkimp(wff1,wff3))));