OBVIOUS LOGICAL INFERENCES

Martin Davis

Courant Institute of Mathematical Sciences, New York, U.S.A.

ABSTRACT

A precise definition is given of a class of inferences in predicate logic which it la proposed to Identify with the class of "obvious" Inferences. A mechanism for Implementing "obvious inference" as a rule of Inference in proof checking systems is discussed.

I. INTRODUCTION

Automatic proofchecking systems should be able to certify the correctness of any Inference which users can aee as obviously correct. It should only be necessary for a user to specify the premises and conclusion in calling for such certification. In proofcheckIng systems for predicate logic based on natural deduction, e.g. Stanford FOL [3], [6], [7], this facility Is ordinarily available only for such special cases as tautological inference. The following example (taken from [5], p. 185) is instructive:

$$(\forall x)[(Fx \mathbin{\&} \sim Gx) \rightarrow (\exists y)(Hxy \mathbin{\&} Jy)]$$

$$(\exists x)[(Kx \mathbin{\&} Fx) \mathbin{\&} (\forall y)(Hxy \rightarrow Ky)]$$

$$(\forall x)[Kx \rightarrow \sim Gx]$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$(\exists x)(Kx \mathbin{\&} Jx]$$

Although this inference seems a bit complicated at a first glance it is really obvious to anyone experienced with elementary logic. (The second and third premises provide an individual c for which Fc, ~ Gc and (Vy)(Hcy * Ky) all hold. The first premise then gives an individual b such that Hcb and Jb. So Kb and Jb both must be true.) However a typical FOL proof would run to 17 lines. An "optimised" FOL proof using two calls to the TAUT tautological Inference checker still runs 9 lines. A user is compelled to step-by-step eliminate quantifiers, use proposltlonal calculus, and finally relnatate quantifiers.

In this communication, we will give a precise definition of a class of inferences in predicate logic we propose to identify with the classes of deductive inferences which, intuitively, are "obvious". We will describe a simple mechanism (based on a nonresolutlon Herbrand theorem-proving procedure) by means of which "obvious inference" can be Implemented as a rule of inference in natural deduction systems, and we will deacrlbe mi experimental version of this mechanism as an extension of FOL. In particular the 9 to 17 steps required In the example cited is reduced to 1 when the new mechanism is available.

II. OBVIOUS HERBRAND PROOFS

What makes an inference obvious? Equivalentlv, what makes an inference complicated? We propose a* an answer to the second question: an inference Is complicated when it requires multiple substitutions from the Herbrand universe in the same clause. In such a case, a human being must exercise considerable Ingenuity in order to find exactly the right substitutions from the typically infinite Herbrand universe. These considerations suggest the:

THESIS: An inference is obvious precisely when a Herbrand proof of its correctness can be given Involving no more than one substitution instance of each clause.

Returning to our example, It is equivalent to the unsatisflablIty of the list of clauses:

(1)	$\bar{F}x \vee Gx \vee Hx,g(x)$
(2)	$\bar{F}u \vee Gu \vee Jg(u)$
(3)	Kc
(4)	Fc
(5)	$\bar{H}cy \vee Ky$
(6)	$\bar{K}z \vee \bar{G}z$
(7)	$\bar{K}w \vee \bar{J}w$

A Herbrand proof of the correctness of the inference is obtained from the substitution:

$$x = u = z = a \,, \quad y = w = g(a)$$

by checking that the resulting list of clauses is truth-functionally unsatlsflabia. Indeed: there Is only one substitution instance of each clause used, In accord with our thesis.

[1]Work supported by NSF Grant MCS 60-02438.

III. AUTOMATING OBVIOUS INFERENCE

How could this treatment of our example be automated? A first thought might be to use binary resolution with the restriction that no clause may be resolved upon more than once. But it is readily seen that an inauspicious order of resolutions can defeat this method. Thus, resolving (5) with (6) would give

(8) Hey v Gy .

Since the clauses (1), (2), (3), (4), (7), (8) are readily seen to be satisfiable, further use of resolution on these clauses alone cannot lead to a refutation. Of course, a resolution theorem-prover equipped with suitable heuristics could easily deal with this example. But It is not clear how to design an efficient resolution theorem-prover which will effectively verify <u>all</u> obvious inferences. Fortunately the method of <u>linked conjuncts</u> [1], (2) (which historically preceded resolution), can be used quite nicely for this purpose. The method is based on the simple remark that a set of clauses which is minimally truth-functionally unsatisfiable (In the sense that any proper subset is truth-functlonally satlsflable) must be linked: that is, the negation of each literal occurring in one of the clauses must occur In one of the remaining clauses. Thus in order to test for <u>obvious iferences</u> one only need use the unification algorithm to search for all possible matches between literals and their negations <u>using only</u> a_ <u>single copy of each clause</u>.

Our proposal is thus to adjoin to natural deduction proofcheckers a new rule of inference, OBVIOUS. OBVIOUS is invoked along with a list of premises and a proposed conclusion. (In the context of FOL, OBVIOUS has the same syntax as TAUT.) Invoking OBVIOUS calls an algorithm consisting of:

(A) a preprocessor which negates the conclusion, Skolemlzes and produces a list of clauses;

(B) a procedure which searches the space of possible matches between literals and their negations;

(C) a satisfiabilty tester (e.g. one based on the Davis-Putnam procedure [2], pp. 25-26).

If (A), (B), and (C) lead to success, i.e. to an unsatlsflable linked conjunct, the system adds the proposed conclusion as a new line In the proof being constructed. Otherwise, the system returns a message such as NOT OBVIOUS.

IV. EXPERIMENTS WITH FOL

During a brief stay at the Stanford AI Laboratory, the OBVIOUS rule was Implemented along the lines discussed. A theorem-proving program based on the linked conjunct method that had been written by D. Mcllroy and Peter Hlnman (see [1]) In 1962 was resurrected and modified. The nodiflcatlon consisted of: (1) translation into the local dialect of LISP; (2) setting to 1 the parameters determining the maximum number of Instances of each clause to be permitted; (3) placing a Skolemlting conjunctive normal form preprocessor at the front end of the program.

Various proofs that had previously been checked by FOL were redone using the new OBVIOUS facility. It was found that the lengths of proofs were shortened by a factor of approximately 10. Opportunities for even more dramatic reduction abound In Filman's dissertation [4], which contains over a thousand lines of FOL.

A version of OBVIOUS for general use with FOL will have to be modified to accommodate the rich FOL faclltles for declaring variables to be of various "sorts". This can easily be done, either by modifying the unification algorithm subroutine in OBVIOUS to permit only substitutions of the correct sort, or by introducing into the preprocessor a preliminary procedure for making explicit the relativized quantifiers which the FOL sort mechanism suppresses.

REFERENCES

[1] Chlnlund, T. J., Davis, M., Hlnman, P. G., and Mcllroy, M. D., <u>Theorem-Proving by Matching</u>. Bell Labs., 1964.

[2] Davis, Martin, <u>Eliminating the Irrelevant from Mechanical Proofs</u>, Proc. Symp. Appl. Math., vol. 15 (1963), 15-30.

[3] Fllman, Robert E. and Weyhrauch, R. W., An FOL <u>Primer</u>, Stanford AIM 288, 1976.

[4] Fllman, Robert E., <u>The Interaction of Observation and Inference</u>, doctoral dissertation, Stanford University, 1979.

[5] Qulne, Wlllard Van Orman, <u>Methods</u> of <u>Logic</u>, Revised Edition, Henry Holt, 1959.

[6] Weyhrauch, R.W., A <u>Users Manual for</u> FOL, Stanford AIM 235.1, 1977.

[7] Weyhrauch, R. W., <u>Prolegomena to</u> a <u>Theory of Mechanised Formal Reasoning</u>, Artificial Intelligence 13 (1980), 133-170.