Journal of Automated Reasoning 3 (1987) 383–393.

Obvious Inferences

PIOTR RUDNICKI
Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada T6J 2H1.

(Received: 27 July 1986)

Abstract. The notion of 'obvious' inference in predicate logic is discussed from the viewpoint of proof-checker applications in logic and mathematics education. A class of inferences in predicate logic is defined and it is proposed to identify it with the class of 'obvious' logical inferences. The definition is compared with other approaches. The algorithm for implementing the 'obviousness' decision procedure follows directly from the definition.

Key words: Proof checking, theorem proving, unification, logic teaching.

1. Introduction

The core of an automatic proof-checking system is a decision procedure for accepting/rejecting presented inferences. A rejection does not mean that an inference is logically invalid, it simply mirrors the fact that the proof-checker was unable to certify the inference's correctness. Certainly, an invalid inference has to be rejected. Using a proof-checker is similar to a discussion between humans. One admits that one does not see why a conclusion follows from premises (even if it does in fact), but one agrees quickly that the adversary is right after being given additional explanation. The criterion for acceptance/rejection of valid logical inferences in a proof-checking system is said to define a class of 'obvious' inferences in the system.

It was proposed that 'automatic proof-checking systems should be able to certify the correctness of any inference which users can see as obviously correct' [4]. The key problem in putting the above proposal into work is in the word *users* since the proof-checking systems are used in various applications. In this paper we will focus on applications of proof-checking systems in teaching logic and mathematics at the introductory university level, thus the term *users* is fairly well defined.

A number of attempts to carry out and use an automatic proof-checking system in teaching logic have been tried. From the point of interest of this paper they fall into two categories. The first group includes those in which justification of inference validity requires the user to specify the premises and the rule(s) of inference (from certain set of rules) [13, 5]. In this kind of approach, since the rules involved in an inference are specified, the class of obvious inferences is equal to the class of valid inferences. In practice however, a nontrivial decision has to be made to define the set of allowed inference rules. In the second group there are proof-checking systems in which it is necessary to specify only the premises and conclusion before calling for inference validity certification [14, 15, 18, 1, 10, 6]. We are concerned with problems pertaining to this second class of proof-checking systems.

Checking an inference's validity is a sort of theorem proving. Because of the application's nature, a powerful theorem prover can not be applied. Firstly, the goal is to force students to write proofs in detail, thus not all correct inferences have to be accepted. Secondly, the proof-checker should be very efficient in rejecting invalid inferences even at the cost that some valid inferences will not be certified as correct. Thus we need to define a class of inferences which is a proper subset of all valid inferences and which a proof-checker always accepts. There is one issue which arises immediately. In defining the class of obvious inferences there is a danger of introducing a discrepancy between the user's sense of an obvious inference and what the proof-checker is able to accept.

2. Davis's Thesis

A direct motivation for writing this paper is [4] in which Davis proposes to identify *obvious* inferences with those correct inferences for which a Herbrand proof can be given involving no more than one substitution instance of each clause. This class of inferences is called *obvious Herbrand inferences*. (Independently, a similar criterion for obvious inferences was adopted in the MIZAR proof-checking system by Trybulec [16].) The following (suggested by J. Los [7]) is an example of inference that does not have an obvious Herbrand proof:

Example 1:

$$\begin{array}{l} sr\colon (\forall x)(\forall y)(p[x,y] \vee q[x,y]) \\ sq\colon (\forall x)(\forall y)(q[x,y] \rightarrow q[y,x]) \\ tp\colon (\forall x)(\forall y)(\forall z)((p[x,y] \wedge p[y,z]) \rightarrow p[x,z]) \\ tq\colon (\forall x)(\forall y)(\forall z)((q[x,y] \wedge q[y,z]) \rightarrow q[x,z]) \\ \hline (\forall x)(\forall y)(p[x,y]) \vee (\forall x)(\forall y)(q[x,y]) \end{array}$$

(The example was originally stated as: if two equivalence relations add up to the full relation then at least one of the initial relations is also full.)

The simplest Herbrand proof (known to the present author) of the above inference uses 5 substitution instances of *sr*, 3 of *sq*, 2 of *tp* and 3 of *tq*. Thus in a proof-checking system where the decision procedure for confirmation/rejection of inferences is based on the Davis's thesis, the following:

$$(\forall x)(\forall y)(p[x,y]) \vee (\forall x)(\forall y)(q[x,y]) \quad \text{by} \quad sr, sq, tp, tq$$

would be rejected, because each premise has to be used more than once, while

$$(\forall x)(\forall y)(p[x,y]) \vee (\forall x)(\forall y)(q[x,y]) \quad \text{by} \quad sr, sq, tp, tq, sr, sq, tp, tq, sr, sq, tq, sr, sr$$

would be accepted as correct, since each use of a premise is separately indicated. This fact was quickly discovered by users of the system (students), who, convinced that the theorem they were asked to prove was true, tried to use the power of the proof-checker by multiple references to given premises. This can be done in many obscure and hard to detect ways. There is no simple remedy for this, e.g. disallowing multiple references to premises for one inference in the input text to the proof-checker will not work. One can always rewrite a line of a proof to another place and label it differently. On the other hand testing whether all premises are pairwise distinct (or even pairwise not equivalent) would involve a considerable amount of work slowing the proof-checker and still not guarantee that all repetitions are detected.

The above example suggests that even quite complicated reasonings are obvious according to Davis's thesis. This may be a drawback in instructional systems. There are also other concerns. Applying the Davis thesis in practice may involve lengthy computations, especially in case of inferences which are eventually rejected. A bit of work is needed to test that the following:

$$(\forall x)(\forall y)(p[x, y]) \vee (\forall x)(\forall y)(q[x, y]) \quad \text{by} \quad sr, sq, tp, tq, sr, tq, sr, sq, sr$$

is not an obvious Herbrand inference. There are also examples of very simple inferences which violate the Davis's thesis. The following is not an obvious inference (in Davis's sense), even though it certainly ought to count as 'obvious'.

Example 2:

$$\begin{array}{l} (\forall x)(P[x] \to P[f(x)]) \\ P[a] \\ \hline P[f(f(a))] \end{array}$$

3. Other Approaches and Extremal Cases

Even in an introductory course in logic where a computer is used to check students' proofs one can think of employing more than one proof-checker. Without significantly changing the input language to the proof-checker one can exchange the checking modules and give the students a chance to work both with a very demanding proof-checker and with a more liberal one. Thus one can foresee a need to have classes of 'obvious' inferences predefined such that the proper choice could be made. It is quite reasonable that a proof-checker requiring the specification of rules of inference (even for sentential inferences) may be used at the very initial stage of teaching. But what should go next?

Since the propositional calculus is decidable, the notion of being obvious in sentential inferences has totally different flavor than in predicate calculus case. A propositional inference may look complicated and long but its correctness can always be trivially checked. This suggests that any predicate calculus inference with explicitly shown substitutions (which reduce it to a propositional calculus inference) might be counted as obvious. In this case the proof-checker does the certification of an inference considering only the substitutions indicated by the user. Let us treat this approach as the lower extremum. If we do not require the user to indicate all substitutions which reduce an inference to propositional calculus then one can foresee a spectrum of possible 'obviousness' classes. What should one take as an extreme on the other end?

Taking into account what has been said earlier about obvious Herbrand inferences one can suggest that the Davis's thesis for obviousness might be taken as the upper extreme. Namely, let us agree that it is sufficient for an inference to be complicated if *no* obvious Herbrand proof of its correctness can be given. One might also suggest that some amendments to Davis's thesis are needed to make inferences like Example 2 obvious.

In the initial design of the MIZAR project [15] the power of the proof-checking module was restricted aiming at quick rejection of incorrect inferences. The general idea was that a correct predicate calculus inference was accepted if certification of its correctness did not require to process substitution instances of two universally quantified premises at the same time. Furthermore, the terms used for the substitution were, only those ground terms which originally appeared in the inference. This approach was named 'non-cooperating universal sentences'. The following inference was not obvious in this system.

Example 3:

$$\begin{array}{l}(\forall x)(P[x] \rightarrow Q[x]) \\ (\forall x)(Q[x] \rightarrow R[x]) \\ \hline P[a] \rightarrow R[a]\end{array}$$

This approach results in quite long proofs, *cf.* [12]. Trybulec [17] coded in MIZAR several basic theorems in calculus on convergence of sequences which reuired 180 inference steps when only one universal premise was allowed per inference, in contrast to 50 inference steps when the Davis proposal was applied. However the class of inferences obvious according to MIZAR-2 criterion is not a subset of obvious Herbrand inferences. The following inference does not have an obvious Herbrand proof but it involves only one universally quantified premise:

Example 4:

$$\begin{array}{l}(\forall z)(P[x]) \\ \hline P[a] \wedge P[b]\end{array}$$

Other approaches defining obvious inferences used nonintuitive parameters to guide the search in the resolution process. E.g. in the implementation of the VERIFY rule of QUIP [14] a parameter was introduced which restricted the (usually infinite) Herbrand universe only to certain level of constant set of an inference ([2], p. 52). It resulted in counterintuitive behavior of the checker [14, 9].

4. A New Proposal

Let us take a look at the following example (taken from [11] p. 168).

Example 5:

$$\begin{array}{l} (\forall x)((F[x] \wedge \neg G[x]) \rightarrow (\exists y)(H[x, y] \wedge J[y])) \\ (\exists x)(K[x] \wedge F[x] \wedge (\forall y)(H[x, y] \rightarrow K[y])) \\ (\forall x)(K[x] \rightarrow \neg G[x]) \\ \hline (\exists x)(K[x] \wedge J[x]) \end{array}$$

The above was used in [4] to give a sample inference which was obvious according to his thesis. The correctness of that inference is equivalent to the unsatisfiability of the following list of clauses:

(1) $\neg F[x] \vee G[x] \vee H[x, g(x)]$

(2) $\neg F(u] \vee G[u] \vee J[g(u)]$

(3) $K[c]$

(4) $F[c]$

(5) $\neg H[c, y] \vee K[y]$

(6) $\neg K[z] \vee \neg G[z]$

(7) $\neg K[w] \vee \neg J[w]$

What makes testing of the above formula obvious? According to Davis' thesis it is the fact that with the substitutions:

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

we obtain a truth-functionally unsatisfiable set of clauses using only one substitution instance of each clause.

In the present proposal we link the 'obviousness' of an inference with the process of computing the substitution for variables which when applied (possibly) makes the set of corresponding clauses truth-functionally unsatisfiable. What, according to the present proposal, makes the above inference obvious? or, equivalently, what makes testing the unsatisfiability of the above list of clauses straightforward? Undoubtedly it includes the fact that only one substitution instance of each clause needs to be used to give a Herbrand proof. But in our opinion this is not all. Whichever method is used for unsatisfiability testing in this case (either the linked conjuncts [3] or resolution) it is easy to see what has to be done first. Namely: the only matches between complementary literals containing the predicate F that are possible occur when we put c/x and c/u. Thus if we assume that the set of clauses is minimally truth-functionally unsatisfiable, the indicated substitutions can be done putting us closer to a success.

Similarly, there are unique matings for H: we need the substitution c/x and $g(x)/y$. Hence, we say that the process of unsatifiability testing in this case is *directed*. However, considering literals containing K in the input clauses does not result in a unique substitution leading to complementary pairs.

The above discussion illustrates the essence of the proposed method.

DEFINITION: Let S be a set of Clauses $C_1, C_2, \ldots, C_k$. A pair of literals $\langle L_1, L_2\rangle$ is called an *initiating pair of S* iff

1° L_1 and L_2 occur in different clauses C_i and C_j respectively,

2° L_1 and $\neg L_2$ are unifiable,

3° either there is no other literal which occurs in a clause other than C_i except for L_2 which can unify with $\neg L_1$ or else there is no other literal which occurs in a clause other than C_j except for L_1 which can unify with $\neg L_2$.

In the last example the initiating pairs are as follows:

$\langle \neg F[x], F[c]\rangle$, $\langle \neg F[u], F[c]\rangle$,

$\langle G[x], \neg G[z]\rangle$, $\langle G[u], \neg G[z]\rangle$,

$\langle H[x, g(x)], \neg H[c, y]\rangle$,

$\langle J[g(u), \neg J[w]\rangle$.

Note that there is no initiating pair with predicate K.

DEFINITION: Let $\langle L_1, L_2\rangle$ be an initiating pair of literals of a given set of clauses S. The *initiating substitution in S by* $\langle L_1, L_2\rangle$ is the most general unifier of $\{L_1, \neg L_2\}$.

In the example above $\{c/x, g(c)/y\}$ is the initiating substitution by $\langle H[x, g(x)], \neg H[c, y]\rangle$.

The proposed characterization of obvious inferences will be identified with the following algorithm implementing a decision procedure for acceptance of inferences.

UNIQUE CONTINUATION ALGORITHM. Given is a set of clauses $S = \{C_1, C_2, \ldots, C_k\}$ corresponding to an inference I.

step 0: If S is truth-functionally unsatisfiable (at propositional calculus level), then stop, I is accepted.

step 1: If there is only one clause in S containing variables, then it is decidable whether S is unsatisfiable (see Appendix I). Stop, I is accepted iff S is unsatisfiable.

step 3: For each initiating pair $\langle L_1, L_2\rangle$ find the initiating substitution σ. If there are two initiating substitutions in conflict, i.e. substituting non-unifiable terms for a variable, then stop, I is rejected.

step 4: For each initiating pair $\langle L_1, L_2 \rangle$ and its initiating substitution σ do: if L_1 occurs in a clause C_i in S then replace C_i in S by $C_i\sigma$ and if L_2 occurs in a clause C_j in S then replace C_j in S by $C_j\sigma$. Go to *setp 0*.

In the above algorithm, as in obvious Herbrand proofs, we use only one substitution instance of each clause, but an exception is allowed. This exception is either the only clause containing variables or is not the the only one but all other substitutions were 'forced' in a unique way. The 'forced' substitutions result from existence of initiating pairs of literals. If there are no initiating pairs with more than one clause containing variables left, the inference is not accepted. This is a factor in making inferences complicated. Maybe we need only one substitution instance of each clause (which satisfies Davis's thesis), but one has to try numerous possible substitutions before finally finding the one which works. The proposed method requires that finding the needed substitutions is straightforward.

It is proposed:

An inference is *obvious* only in case when it is accepted by the unique continuation algorithm.

The presented algorithm can be seen as a restricted version of the linked conjuncts method [3] and resembles the linked inference principle [19].

The algorithm when applied to our example works as follows. There are the following initiating substitutions:

$\{c/x\}$	by	$\langle \neg F[x], F[c] \rangle$
$\{c/u\}$	by	$\langle \neg F[u], F[c] \rangle$
$\{x \leftrightarrow z\}$	by	$\langle G[x], \neg G[z] \rangle$
$\{u \leftrightarrow z\}$	by	$\langle H[u], \neg G[z] \rangle$,
$\{c/x, g(x)/y\}$	by	$\langle H[x, g(x)], \neg H[c, y] \rangle$,
$\{g(u)/w\}$	by	$\langle J[g(u), \neg J[w] \rangle$.

The symbol $\leftrightarrow$ indicates equivalence of variables and that the substitution is to be done after choosing a representative of an equivalence class. The application of the above substitutions results in the following set of clauses for further testing:

(1) $\neg F[c] \vee G[c] \vee H[c, g(c)]$

(2) $\neg F[c] \vee G[c] \vee J[g(c)]$

(3) $K[c]$

(4) $F[c]$

(5) $\neg H[c, g(c)] \vee K[g(c)]$

(6) $\neg K[c] \vee \neg G[c]$

(7) $\neg K[g(c)] \vee \neg J[g(c)]$

which is truth-functionally unsatisfiable. We now present an example of an obvious Herbrand inference which is not accepted by the unique continuation algorithm.

Example 6:

$$
\begin{array}{l}
(\forall x)(\forall y)(R[x, y] \vee R[y, x]) \\
(\forall x)(\forall y)((S[x, y] \wedge S[y, x]) \rightarrow x = y) \\
(\forall x)(\forall y)(R[x, y] \rightarrow S[x, y]) \\
\hline
(\forall x)(\forall y)(S[x, y] \rightarrow R[x, y])
\end{array}
$$

The correctness of the above inference is equivalent to unsatisfiability of the following list of clauses:

(1) $R[x, y] \vee R[y, x]$

(2) $\neg S[u, v] \vee \neg S[v, u] \vee u = v$

(3) $\neg R[w, z] \vee S[w, z]$

(4) $S[a, b]$

(5) $\neg R[a, b]$

Though the inference seems to be very simple it is not accepted since there is no initiating pair of literals.

5. Implementation and Experiments

The implementation of the unique continuation algorithm is straightforward using unification. The implementation is now in progress and in the meantime many experiments are being carried out by hand. In appendix II there are 3 versions of a proof of a simple theorem each using a different definition of obviousness for an inference step. A few things observed in this experiments are worth mentioning. The proposed method is sensitive to redundant premises. A redundant premise often causes there to be no initiating pair of literals and the algorithm stops early. But in the application of proof-checkers in teaching logic we believe this should be treated as an advantage.

The defined class of obvious inferences is not a subset of the class defined by obvious Herbrand proofs. A substantial number of inferences which have obvious Herbrand proofs are not accepted by the proposed algorithm (Example 6). There are however cases of certain simple inferences not obvious according to Davis' thesis which are certified by our method as correct (Examples 2 and 4). It appeared in many examples that the proposed method is more efficient in rejecting invalid inferences than finding out that there is no obvious Herbrand proof for the inferences (see discussion of Example 1). This method also does not allow users' tricks in avoiding a detailed proof by making multiple references to a premise in a single inference step. These two issues are of practical importance in computer aided teaching of proving techniques.

Appendix I

LEMMA. *It is decidable whether a set of clauses $S = \{C_1, C_2, \ldots, C_k\}$ in which variables occur in only one clause is unsatisfiable.*

Proof. Let the clause in which variables occur be C_1. All literals occurring in $C_2, \ldots, C_k$ are ground literals. If $k = 1$ then S is satisfiable. So assume $k > 1$. If C_1 initially contains a pair of unifiable complementary literals, then by the lifting lemma ([2], p. 84) one can replace C_1 in S by its substitution instance $C_1\sigma$ (for certain substitution σ) removing the pair of unifiable complementary literals and thus obtaining a set S'. S is unsatisfiable if and only if S' is unsatisfiable. So, let us assume that C_1 does not contain a pair of complementary unifiable literals. If there is a deduction of the empty clause $\square$ from S then there are two complementary literals L_1 and L_2 which appear as unit clauses in the process of resolution and at least one of them is a ground literal. Hence, in the process of resolution it is enough to consider only those resolvents for which C_1 is a parent clause, which contain terms unifiable with ground terms initially present in S. But there is only a finite amount of such resolvents.

Appendix II

We present three versions of proving the first example from the text. The structure of each proof is exactly the same. All of them are recorded in MIZAR-MSE but each with different requirements for an inference acceptance by the proof-checker. MIZAR key words are in bold. The premises are recorded as follows:

sr: **for** x, y **holds** p[x, y] **or** q[x, y];

sq: **for** x, y **st** q[x, y] **holds** q[y, x];

tp: **for** x, y, z **st** p[x, y] & p[y, z] **holds** p[x, z];

tq: **for** x, y, z **st** q[x, y] & q[y, z] **holds** q[x, z];

Version 1: Only one universally quantified premise is allowed in an inference. (10 inference steps)

```
(for  x, y holds p[x,y]) or (for x, y holds q[x,y])
 proof  given a, b being individual such that
   1: not p[a,b]; then
      q[a,b] by sr; then
   2: q[b,a] by sq;
  let x, y be individual;
   auxstep: for z holds q[a,z]
```

```
        proof let z be  individual;
          11: now assume not p[z,b]; then
                      q[z,b] by sr; then
                      q[z,a] by tq,2;
                  hence q[a,z] by sq end; == of now
          p[a,z] or q[a,z] by sr;
         hence q[a,z] by tp,1,11 end; == of auxstep
      3: q[a,x] & q[a,y] by auxstep; then
         q[x,a] by sq;
    hence q[x,y] by tq,3 end == of proof
```

Version 2: With Davis' proposal for obviousness, but not allowing multiple references to premises in a single inference step. (5 inference steps)

```
    (for  x, y holds p[x,y]) or (for x, y holds q[x,y])
     proof  given a, b being individual such that
       1: not p[a,b]; then
       2: q[b,a] by sr,sq;
      let x, y be  individual;
       auxstep: for z holds q[a,z]
         proof let z be  individual;
            p[z,b] or q[a,z] by sr,tq,sq,2 ;
          hence q[a,z] by sr,tp,1 end; == of auxstep
      q[a,x] by auxstep;
     hence q[x,y] by auxstep,sq,tq end == of proof
```

Version 3: Inference checking according to the new proposal. (6 inference steps)

```
     (for  x, y holds p[x,y]) or (for x, y holds q[x,y])
     proof  given a, b being individual such that
       1: not p[a,b]; then
       2: q[b,a] by sr,sq;
      let x, y be  individual;
       auxstep: for z holds q[a,z]
         proof let z be  individual;
           now assume not p[z,b]; then
                q[z,a] by sr,tq,2;
               hence q[a,z] by sq end; == of now
          hence q[a,z] by sr,tp,1 end; == of auxstep
       q[x,a] by auxstep,sq; then
     hence q[x,y] by auxstep,tq end == of proof
```

Acknowledgements

My thanks are to Pawel Gburzynski and Jeff Pelletier for discussions while preparing this paper.

References

1. Blaine, Lee, 'Programs for Structured Proofs', in *University-Level Computer-Assisted Instruction at Stanford: 1968 1980*, Patrick Suppes (ed.), IMSSS Stanford University, 81–119 (1981).
2. Chang, Chin-Liang and Lee, Richard Char-Tung, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, NY (1973).
3. Davis, Martin, 'Eliminating the Irrelevant from Mechanical Proofs', in *Proc. Symp. Appl. Math.*, vol. 15, 15–30 (1963).
4. Davis, Martin, 'Obvious Logical Inferences', *Proceedings of the 7th IJCAI*, Vancouver, 530 531 (1981).
5. Garson, James and Mellema, Paul, 'Computer-Assisted Instruction on Logic: EMIL', *Teaching Philosophy*, 453–478 (1980).
6. Ketonen, Jussi, EKL – A Mathematically Oriented Proof Checker', in *7th International Conference on Automated Deduction*, R. E. Shostak (ed.), Napa, 65–79 (1984).
7. Los, Jerzy, Personal communication (1983).
9. Marinov, Vesko, Personal communication (1980).
10. Prazmowski, Krzysztof and Rudnicki, Piotr, *MIZAR MSE Primer*, ICS PAS Report No. 529, Warsaw (1983).
11. Quine, W. V. O., *Methods of Logic*, Holt, Rinehart and Winston, Inc. (1972).
12. Rudnicki, Piotr and Drabent, Wlodzimierz, 'Proving Properties of Pascal Programs in MIZAR 2, *Acta Informatica*, 311–331, 699 707 (1985).
13. Schagrin, Morton L., Rapaport, William J. and Dipert, Randall R., *Logic: a Computer Approach*, McGraw Hill Book Company (1985).
14. Smith, R. L., Graves, W. H., Blaine, L. H. and Marinov, V G., 'Computer-Assisted Axiomatic Mathematics: Informal Rigor', in *Computers in Education*, O. Lecarme and R. Lewis (ed.), North Holland/American Elsevier, 803–809 (1975).
15. Trybulec, Andrzej, 'The MIZAR-QC/6000 Logic Information Language', *Bulletin of the Association for Literary and Linguistic Computing*, 136–140 (1978).
16. Trybulec, Andrzej, Personal communication (1981).
17. Trybulec, Andrzej, Unpublished report, University of Connecticut (1984).
18. Weyhrauch, R. W., 'Prolegomena to a Theory of Mechanized Formal Reasoning', *Artificial Intelligence* 12, 133–170 (1980).
19. Wos, L., Veroff, R., Smith, B. and McCune, W., 'The Linked Inference Principle II: The User's Viewpoint', in *7th International Conference on Automated Deduction*, R. E. Shostak (ed.), Napa, 316–332 (1984)