PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série, tome 38 (52), 1985, pp. 207–213

HEURISTIC FOR AVOIDING SKOLEMIZATION IN THEOREM PROVING

Irena Pevac

Abstract. Some further possibilities of the theorem prover of the system GRAPH [**2**]-[**5**] are described. Instead of skolemization of a formula, we transform the subgoal into an equivalent form or into a stronger subgoal with less quantifiers, in a human like manner. A heuristic is proposed how to choose, among several possibilities, a stronger subgoal which would be proved more easily. Although an incomplete method, it is suitable for interactive work since human behavior is imitated. If machine's choice is not suitable, the user can appropriately modify the transformation.

1. Introduction. In [**3**] the theorem prover of the system GRAPH is described and in [**2**] the main features of knowledge organization are given. The prover is a system that proves theorems in classical first order logic, and some extensions of that by subgoaling, splitting, rewriting, simplification and some other procedures. We shall suppose here that the reader is familiar with these papers. In [**3**] a heuristic for selecting definition or lemma instantiations is proposed by which we get a subgoal which is easier to prove. Generally, we create a proof tree whose leaves are subgoals with the property: each predicate from the conclusions appears also on the left hand side of the main implication of the subgoal. When we delete universal (existential) quantifiers on the top of the right (left) side of the subgoal, it is, generally speaking, still too complicated to be proved easily. At this moment the elimination of other quantifiers becomes actual.

Most of the authors use skolemized formulas in their provers. ln some cases when the machine is not able to find the most general substitution some further improvements are offered such as multiple copies rule in [**1**].

The author of the present paper is of the opinion that the skolemization is not suitable in interactive provers, because it is too far from human way of creating the proof. In this paper it is described, how to delete some of quantifiers in a human like manner and, hence, to avoid skolemization.

AMS Subject Classification (1980): Primary 68G15

We start similarly as in the usual skolemization procedure. In the first step we delete universal quantifiers being the top operations of the subgoal. Further, we delete those quantifiers which could come as universal to the top. This will be done without transforming the subgoal i. e. without moving the quantifiers to the top. After that the rest of quantifiers will be pushed towards the predicates as close as possible.

We say that a formula ${}_1$I is stronger than F_2 (or F_2 is weaker then F_1) if $F_1 \Rightarrow F_2$ is true.

In the second step we propose a way of creating a stronger subgoal to be further proved instead of the current one, based on the use of valid formulas:

$$(\forall X)F(x) \Rightarrow F(t_1) \wedge F(t_2) \wedge \cdots \wedge F(t_n);$$
$$F(t_1) \vee F(t_2) \vee \cdots \vee F(t_n) \Rightarrow (\exists x)F(x).$$

In the fully automatic mode the machine itself finds where and when the transformation is permitted, and the terms $t_1, t_2, \ldots, t_n$ are chosen automatically. In a way, this part is analogous to the use of multiple copies rule (i.e. it is used in similar situations in the formula) but in [**1**] it is not possible to decide automatically how many copies should be taken in advance. In addition, our method solves partially the unification level problem at the same time.

In the interactive mode, the human selects a quantifier by telling the name of the variable it bounds. We accept the convention that variables bounded by different quantifiers are differently denoted, and no free variable is denoted by a symbol used for a bound variable. The system first checks whether its position in the subgoal allows such a transformation. In the case of a positive answer it offers the user the choice of terms. If the user does not approve the machine's choice he will be asked to tell his choice of terms. Using the human idea of the global strategy of the proof the unification is solved on higher level.

Let us now introduce some definitions and a theorem which will be used in further development of the above ideas. Let us recursively assign a sign + or - to each subformula of the given formula as described in [**1**].

Definition. The whole formula has positive sign.

If $A \wedge B$ or $A \vee B$ is positive(negative), then so are A and B.

If $\neg A$ is positive (negative), then A is negative (positive).

If $A \Rightarrow B$ is positive (negative), then A is negative (positive), and B is positive (negative).

If $(\forall x)A$ or $(\exists x)A$ is positive (negative), then so is A.

Let us introduce a presentation of the formula in the tree form in the following way. Each inner node of tree contains the logical operation and the sign of the subformula whose root is this operation. The nodes on the leaves contain predicates and the corresponding sign.

Example 1. The formula $(\exists X1)(\exists X2)\neg X1 = X2 \vee (\forall X)(\forall Y)(X = Y \vee (\exists Z)R1(X,Z)) \Rightarrow (\forall X3)(\exists Y1)R1(X3,Y1)$ is presented by a tree on Fig. 1.

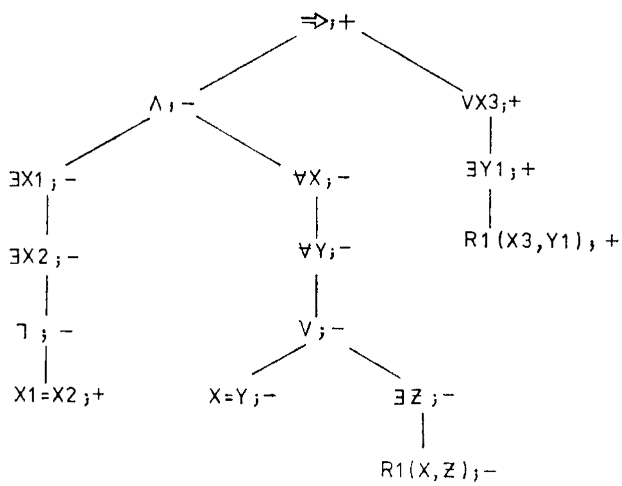

Fig. 1

The following theorem is useful in creating new subgoals.

Theorem. *If we replace a positive (negative) subformula of some formula G with a stronger (weaker) formula the resulting formula is stronger then G.*

An analogous theorem for creating a weaker resulting formula can be formulated, and the proof is obtained easily by induction on the number of logical operations in the formula.

2. Elimination of quantifiers based on equivalent logical transformations. Let us mention the following logically valid formulas:

$$A \vee (\forall x)B(x) \Leftrightarrow (\forall x)(A \vee B(x)), \quad A \vee (\exists x)B(x) \Leftrightarrow (\exists x)(A \vee B(x)),$$

$$A \wedge (\forall x)B(x) \Leftrightarrow (\forall x)(A \wedge B(x)), \quad A \wedge (\exists x)B(x) \Leftrightarrow (\exists x)(A \wedge B(x)),$$

$$A \Rightarrow (\forall x)B(x) \Leftrightarrow (\forall x)(A \Rightarrow B(x)), (\exists x)B(x) \Rightarrow A \Leftrightarrow (\exists x)(A \Rightarrow B(x)),$$

$$(*) \quad (\forall x)B(x) \Rightarrow A \Leftrightarrow (\exists x)(B(x) \Rightarrow A), (\exists x)B(x) \Rightarrow A \Leftrightarrow (\forall x)(B(x) \Rightarrow A)$$

$$(*) \quad \neg(\forall x)B(x) \Leftrightarrow (\exists x)\neg B(x), \quad (\exists x)B(x) \Leftrightarrow (\forall x)\neg B(x),$$

$$(\forall x)(\forall y)F(x,y)(\forall y)(\forall x)F(x,y),$$

$$(\exists x)(\exists y)F(x,y) \Leftrightarrow (\exists y)(\exists x)F(x,y),$$

where x is not free in A.

If we apply these formulas from left to right the quantifiers are moved up in the tree. When using the logically valid formulas labeled by an asterisk, we can see that moving through negation or moving through implication from the left results in changing the type of the quantifier (i.e. universal becomes existential and conversely) as well as changing the sign of the subformula at the time same. As we can delete a universal quantifier on the top of the tree, we are interested to see, in advance, which quantifier could be moved to the top as universal, without performing such transformations.

Since the top operation is positive by definition we can see that the necessary condition is that the quantifier is universal and signed positively or existential and signed negatively.

Finally, as we are not permitted to change the order of universal and existential quantifiers we have the following algorithm for deleting the quantifiers that could come as universal to the top of subgoal without performing actual transformation of the subgoal.

ALGORITHM. For each positive universal quantifier or negative existential quantifier let us examine the path from this node to the top of the tree. If there are no nodes labeled by a positive existential or negative universal quantifier on this path the quantifier can be moved to the top as a universal one. Hence, we shall delete all the quantifiers satisfying the above condition together with the bound variable. After that we try to push the rest of quantifiers to predicates as close as possible. Now we use the above logically valid formulas from the right to the left as long as possible. In this case we actually perform the transformations. The main features of these transformations are described in [**4**].

Example 2. In the formula from example 1. we have the following nodes $\exists X1; -$, $\exists X2; -$, $\exists Z; -$, and $\forall X3; +$ as candidates for elimination. The first, second and fourth could be deleted as explained above. The resulting formula is $\neg X1 - X2 \wedge (\forall X)(\forall Y)(X = Y \vee (\exists Z)R1(X, Z)) \Rightarrow (\exists Y1)R1(X3, Y1)$. After pushing the rest of quantifiers closer to predicates we have $\neg X1 = X2 \wedge (\forall X)((\forall Y)X = Y \vee (\exists Z)R1(X, Z)) \Rightarrow (\exists Y1)R1(X3, Y1)$.

3. Quantifier elimination by introducing a stronger subgoal. Using the result stated in theorem above we shall try to replace a positive subformula of the type $(\exists x)F(x)$ with the disjunction $F(t_1) \vee F(t_2) \vee \cdots \vee F(t_n)$. Analogously, a negative, subformula of the type $(\forall x)F(x)$ will be replaced with the conjunction $F(t_1) \wedge F(t_2) \wedge \ldots F(t_n)$. Of course, when performing such transformation, we have to find out how many disjuncts (conjuncts) we need, and which terms $t_1, t_2, \ldots, t_n$ should be taken. We propose the following heuristic.

Let G denote a current subgoal and let $(A1, \operatorname{sgn} A1)$, $(A2, \operatorname{sgn} A2), \ldots,$ $(An, \operatorname{sgn} An)$ be all the subformulas of G of the type $Ai \equiv (Qiy_i)Fi(y_i)$ with Qi denoting the universal (existential) quantifier if $\operatorname{sgn} Ai$ is negative (positive) respectively. If there is more then one such subformula in G we shall, select one with quantifier attacking less predicates and acting on a shorter subformula. Without loss of generality we shall suppose that selected subformula is negative of type $(\forall y)F(y)$. Let us extract predicate letters attacked by this quantifier and let us mark the places in these predicates where variable y occurs. For each symmetric predicate we mark also the places where the commutation is permitted. After that we have to find out all the occurrences of the extracted predicates in the rest of formula G. Finally, we pick up the free variables or ground terms from the marked positions, and among them we select those which are not free in formula F. The free variables in F will not be avoided only in the case when none of the predicates of F is symmetric. If the variables on marked positions are bound we apply this procedure to the next quantifier. Once we have terms $t_1, t_2, \ldots, t_n$ we have determined the logically valid formula which will be used to create a new stronger subgoal. This new subgoal is sent to the block of "trivial transformations" where the subgoal is splitted and reduced as described in [**3**]. Since some new subgoals could be created after applying this block, we try to prove each of them (see [**3**]). For the unproved subgoals we repeat the elimination of quantifier and the above mentioned, until all the subgoals are proved, or nothing that has been mentioned can be performed.

The proposed heuristic is especially efficient in cases when the predicates of the subformula happen to be equivalence relations or at least symmetric relations. It is based on the idea that there should be an interaction between the predicates of the considered subformula and those placed in the rest of formula. The symmetry of some predicate is specially noted in the theorem prover of the system GRAPH. It is incorporated as a part of system's knowledge and it is not again proved during the proof of the main goal.

Let us now illustrate the above by some examples. Although the examples are of necessity drawn from arithmetical graph theory introduced in [**2**] we hope that heuristic can be useful outside this area too.

Example 3. Let us continue with the formula from Example 2.

$$\neg X1 = X2 \wedge (\forall X)(\forall Y)X = Y \vee (\exists Z)R1(X, Z)) \Rightarrow (\exists Y1)R1(X3, Y1).$$

In the tree of the formula we search for the nodes having universal quantifier labeled $+$ or the nodes having existential quantifier labeled $-$. There are three such nodes $\forall X; -$, $\forall Y; -$, $\exists Y1; +$. The node $\forall Y; -$ will be considered first. The only predicate attacked by this quantifier is $=$. Since $=$ is a symmetric predicate we shall mark both places. Now we search for the $=$ in the rest of formula. As $X1$ and $X2$ happen to be free and they do not appear in the considered subformula, consequently we have determined $X1$ and $X2$ as terms. Hence, we shall use valid formula $(\forall Y)F(Y) \rightarrow F(X1) \wedge F(X2)$. The resulting formula is $\neg X1 = X2 \wedge (\forall X)(X = X1 \wedge X = X2 \vee (\exists Z)R1(X, Z)) \Rightarrow (\exists Y1)R1(X3, Y1)$. The subgoal remains unchanged after "trivial transformations" given in [**3**] and a direct testing of the validity as given in [**3**] is not successful. So, we start again with the quantifier elimination. Now the candidates are the following nodes: $\forall X; -$ and $\exists Y1; +$. We choose $\exists Y1$ as it attacks only one predicate. After marking the predicate and the relevant places, we can see that both marked positions of $R1$ in the rest of the formula contain bound variables. We take both positions because of symmetry of $R1$. (As explained in [**2**], $R1(X, Y)$ is symmetric predicate with the meaning "X and Y are adjacent"). Further, we try the next candidate i.e. the node $\forall X; -$. The predicate letters in the subformula are $=$ and $R1$. Both are symmetric and we mark their both places. The marked positions of relevant predicates in the rest of formula contain $X1, X2, X3, Y1$. The variable $Y1$ is bound and $X1$ and $X2$ are free in the considered subformula. The valid formula of the form $(\forall X)F(X) \to F(X3)$ is used for creation of a stronger subgoal. The result is $\neg X1 = X2 \wedge ((X3 = X1 \wedge X3 = X2) \vee (\exists Z)R1(X3, Z)) \Rightarrow (\exists Y1)R1(X3, Y1)$. Applying the block of "trivial transformations" to this subgoal it will be split into the following two subgoals:

$$\neg X1 = X2 \wedge X3 = X1 \wedge X3 = X2 \Rightarrow (\exists Y1)R1(X3, Y1)$$
$$\neg X1 = X2 \wedge (\exists Z)R1(X3, Z) \Rightarrow (\exists Y1)R1(X3, Y1).$$

Both of them can be proved in the block for testing the validity of a subgoal.

In some further examples the outline of the proof will be given. The meanings of predicates occurring in further examples are the same as given in [**2**] i.e. $R4(X, Y)$ means "X and Y are joined by a walk", and $S2(X, Y, K)$ means "X and Y are joined by a walk of length K".

Example 4.

$$R4(X, Y) \wedge \neg(\forall X1)(\forall Y1)R4(X1, Y1) \Rightarrow (\exists Z)(\neg R4(X, Z) \wedge \neg R4(Y, Z))$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow (\exists Z)(\neg R4(X, Z) \wedge \neg(R4(Y, Z))$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow (\neg R4(X, X1) \wedge \neg R4(Y, X1)) \vee (\neg R4(X, Y1) \wedge \neg R4(Y, Y1))$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow \neg R4(X, X1) \vee \neg R4(X, Y1)$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow \neg R4(X, X1) \vee \neg R4(Y, Y1)$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow \neg R4(Y, X1) \vee \neg R4(X, Y1)$$
$$R4(X, Y) \wedge \neg R4(X1, Y1) \Rightarrow \neg R4(Y, X1) \vee \neg R4(Y, Y1).$$

Example 5.

$$(\exists X1)(\exists Y1)\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow (\exists Z)(\neg X = Z \wedge \neg R1(X, Z))$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow (\exists Z)(\neg X = Z \wedge \neg R1(X, Z))$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow (\neg X = X1 \wedge \neg R1(X, X1)) \vee (\neg X = Y1 \wedge \neg R1(X, Y1))$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow \neg X = X1 \vee \neg X = Y1$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow \neg X = X1 \vee \neg R1(X, Y1)$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow \neg R1(X, X1) \vee \neg X = Y1$$
$$\neg X1 = Y1 \wedge (\forall Y)\neg R1(X, Y) \Rightarrow \neg R1(X, X1) \vee \neg R1(X, Y1).$$

Example 6.

$$(\forall X)(\neg S2(X, Y1, 0) \Rightarrow S2(X, Y1, 1)) \Rightarrow (\exists K)(S2(X1, Y1, K))$$
$$(\forall X)(\neg S2(X, Y1, 0) \Rightarrow S2(X, Y1, 1)) \Rightarrow S2(X1, Y1, 0) \vee S2(X1, Y1, 1).$$

4. Some final comments. Let us note that in the case of stating the subgoal from Example 6. in an equivalent form as

$$(\forall X)(X \neq Y1 \Rightarrow S2(X, Y1, 1)) \Rightarrow (\exists K)S2(X1, Y1, K)$$

(as $S2(x, y, 0) \Leftrightarrow x = y$ holds) the machine choice would be the valid formula of the form $F(1) \Rightarrow (\exists K)F(K)$ which is not good. The same would happen if the system does not know that $x \neq y \Leftrightarrow \neg x = y$ holds.

We can overcome this problem by consulting the digraph of definitions and lemmas described in [**3**]. The human does the same having the global knowledge of the theory considered.

Another problem appears in the case when the marked positions contain bound variables, and when we select the next quantifier, it could happen that marked positions contain bound variables from the first quantifier. One way of handling this situation in fully automatic work would be to take us many different new variables as many different bound variables we have on the marked places.

Of course, the problem of incompleteness of the heuristic remains. But most of the experts in theorem proving area have agreed that for real effectiveness of the prover we must not strangle the mechanism that does it by making sure that it handles every case. The idea here, was to make a procedure of dealing with and deleting of quantifiers that is near to the human way of creating proofs and, hence, to make the man-machine interaction easier.

Finally, let us finish with the remark from [**7**]: "It is quite probable that heuristicaly-driven provers would not be complete, but then people, who are the best theorem provers so far, probably are not complete either."

REFERENCES

[1] W. Bledsoe, M. Tyson, *The UT Interactive Prover*, Univ. Texas Math. Dept., Memo ATP 17A, june, 1978.

[2] D. Cvetković, *Discussing graph theory with a computer IV, Knowledge organization and examples of theorem proving*, Proc. Fourth Yugoslav Seminar on Graph Theory, Novi Sad, 1983.

[3] D. Cvetković, I. Pevac, *Man-machine theorem proving in graph theory*, to appear in Artificial Intelligence.

[4] D. Cvetković, I. Pevac, *Algorithms for transforming the first order sentences in their natural form*, Publ. Elektrotehn. Fak., Ser. Mat. Fiz. 735–762 (1982).

[5] D. Cvetković, I. Pevac, *Discussing graph theory with a computer III, Man-machine theorem proving*, Publ. Inst. Math., (Beograd) (N. S.) **34(48)** (1983), 37–47.

[6] D. Pastre, *Automatic theorem proving in set theory*, Artificial Intelligence **10** (1978), 1–27.

[7] W. Bledsoe, L. Henschen, *What is automated theorem proving?*, J. Automated Reasoning, **1** (1985), 23–28.

Matematički institut
Kneza Mihajlova 35/I
11000 Beograd
Yugoslavija

(Received 03 10 1984)
(Revised 20 05 1985)