PUBLICATIONS DE L'INSTITUT MATHÉMATIQUE
Nouvelle série, tome 35 (49), 1984, pp. 167–171

SOME HEURISTICS IN AUTOMATIC THEOREM PROVING

Dragoš Cvetković and Irena Pevac

Abstract. We propose two heuristics in automatic theorem proving: an analogy heuristic and a heurietic for detecting the logical equivalence of formulas. The first heuristic tries to establish an analogy between some subtheories of two non-analogous theories and to prove in this way a formula in one theory using a theorem in the second one. The second heuristic is related to the case when one should establish the logical equivalence of two formulas by instantiation of definitions. The structure of the definition set is represented by a digraph and a digraph coloring language is used.

When implementing the interactive programming system "GRAPH" for the classification and extension of knowledge in graph theory [**4**], [**5**] we came across some heuristics in automatic theorem proving which might be of more general interest.

To avoid technical discussions we consider here first order calculi without constants and functions. Beside the proper (basic) predicates b_i of the theory the re are other predicates d_i introduced by definitions of the form $d_i \Leftrightarrow F$, where F is a formula containing previously defined predicate letters. Arguments of predicatcs are not considered here. Therefore both, a predicate and the corresponding predicate letter will be denoted by the same symbol in further text. The form of the definition excludes recursive definitions but this is not essential limitation. Further, we shall denote by $L(F)$ the set of predicate letters occurring in a formula F. The predicates from the set $L(F)$ are called *direct predecessors* of d_i. A predicate d is a *predecessor* of d_i if $d = d_i$, or there is a predccessor k of d_i and d is a direct predecessor of k. The set of all predecessors of elements of $L(F)$ is denoted by $P(F)$. For an arbitrary formula F, let us introduce the *support* $S(F)$ of F as the set of those predecessors of elements of $L(F)$ which are basic predicates, and the *definition set* $D(F)$ of F as the set of predecessors of elements of $L(F)$ that are not basic predicates. In addition, the *level* of b_i is equal to zero, and the level of d_i is greater by one than the maximal level of elements of $L(F_i)$, F_i being the definiens of the definition of d_i.

AMS Subject Classification (1980): Primary 68G15

The definition structure can be suitably represented by a digraph G whose vertices are all $b_{i'}s$ and all $d_{i'}s$. If $d_i \Leftrightarrow F$ is a definition, an arc is going from each vertex denoting an element of $L(F)$ to d_i, and there are no other arcs in G.

Since we assume that axioms contain only basic predicates we can represent the axiom set as hypergraph on vertices $b_1, b_2, \ldots$ with hyperedges $L(A_1), L(A_2), \ldots$ where $A_1, A_2, \ldots$ are axioms.

Graph theory is an example of the theory with a large number of definitions and we had in particular it in mind when developing all these concepts. On the contrary some algebraic theories (e.g. group theory) can be developed using a few predicates and our theory would not be of greater advantage there.

1. An analogy heuristics. Two non-formal theories are called analogous if their formal theories are the same. This includes, for example: 1° the cases when two phenomena are described by the same differential equation (e.g. analogies of some mechanical and electrical quantities) and 2° mapping a theorem from an algebraic structure to a structure isomorphic to it (e.g. dual Boolean algebras and the corresponding duality principle). Examples and the role of such analogies are well known. We shall concentrate on the case when some subtheories of non-analogous theories are analogous. Capability of detecting such analogies would shift the power of any fully automatic or interactive theorem proved toward the area of nontrivial theorems. The analogy principle is little used in automatic theorem proving (cf. [**2**], [**6**]). Here we offer an analogy heuristic which might be incorporated into natural deduction theorem provers.

Let τ and τ' be first order calculi with the same deduction rules. A formula F from a theory τ is proved by analogy with theory τ' (we allow $\tau' = \tau$) if the following holds:

a) There exists a theorem F' of τ which is analogous to F in the sense that there is a bijection $f : L(F') \to L(F)$ between the sets of their predicates, which maps F' onto F, modulo renaming variables.

b) The mapping f can be extended to a mapping $f : F(L') \cup S(F') \to P(F)$ such that for any $d'_i \in L(F')$ if $d'_i \Leftrightarrow F_i(b'_{i_1}, b'_{i_2}, \ldots, b'_{i_k})$ in the theory τ' then $f(d'_i) \Leftrightarrow F_i(f(b'_{i_1}), f(b'_{i_2}), \ldots, f(b'_{i_k}))$ in the theory τ. While b'_{i_j} $(1 \leq j \leq k)$ are some basic predicates in τ', their images $f(b'_{i_j})$ need not be basic predicates of τ.

c) Formulas obtained from axioms of τ' by the mapping f are theorems of τ.

The proof of F is obtained by concatenating proofs of the mapped axioms of τ' in τ and the mapped proof of F' from τ' into τ. We say that the last part of the proof of F is done by analogy with the proof of F'.

In fact, a part of theory τ is isomorphically embedded into the theory τ by mapping f. Such kind of embedding of a theory into another has been used in the literature from other points of view (see, for example, [**7**], [**3**]). This isomorphic embedding enables one to construct a proof of F in τ by analogy with the proof of F' in τ'.

Instead of axioms of τ one can consider a set of theorems in τ' from which F' can be derived.

In case when the predicates from the definition set of F' in τ' have analogous definitions in τ an extension of f is easily determined. Otherwise, a way to enhance the chances for the success is to give the program the capability of introducing new definitions in τ during the process of detecting an analogy.

The corresponding program representing itself a heuristic should incorporate further heuristics for several subtasks: when to start the whole procedure, how long to search for the mapping f, and finally how much time to spend in attempts to prove the transformed axioms of the theory τ'.

To improve the described heuristic, we can allow the formula F in τ and the theorem F' in τ to be transformed by definitions (instantation or vice versa) or by logically valid formulas before detecting an analogy, or when the searching was not successful.

The actual usage of this heuristic in automatic theorem proving assumes that definition sets of the two theories are stored in the computer. In the system "GRAPH" a well organized set of definitions from graph theory is stored and, perhaps, the analogy heuristic acting between different parts of graph theory will be implemented.

2. Detecting equivalence of formulas. There is often a need in automatic theorem proving to recognize that two formulas are logically equivalent. We propose here a heuristic for that after developing necessary tools. We shall use a special coloring and recoloring of vertices of the digraph G. Vertices are either uncolored or colored by one of colors: blue, yellow, green, white. Let us introduce the following commutative composition of colors:

	green	blue	yellow	white	uncoloured
green	green	green	green	green	green
blue	green	blue	green	blue	blue
yellow	green	green	yellow	yellow	yellow

We shall say that a vertex colored blue or yellow or green is *transformed* if it is recolored by white and a direct predecessor y of x is recolored by the composition of the color of y and the previous color of x.

Suppose we want to establish whether formulas F_1 and F_2 are equivalent. If $L(F_1) \neq L(F_2)$ we try to instantiate some definiLions so that for new formulas F_1', F_2' ($F_1' \Leftrightarrow F_1$, $F_2' \Leftrightarrow F_2$) we have $L(F_1') = L(F_2')$. To avoid the explosion of the number of formulas when expanding all possible definitions we use some coloring of G.

Color $L(F_1)$ in blue and $L(F_2)$ in yellow. Then, of course, $L(F_1) \cap L(F_2)$ is colored in green while $L(F_1) \setminus L(F_2)$ remains blue and $L(F_2) \setminus L(F_1)$ remains yellow. We shall try to eliminate blue and yellow vertices by transforming them, or some green vertices. Obviously transforming a vertex means to instantiate the corresponding definition.

A blue or yellow vertex x is said to have property α if the following holds: there is a non-white vertex y, colored differently from x, such that x and y have a common predecessor. We follow two simple rules:
1. Transform always a vertex with property α and with a highest level (among vertices with property α).
2. If a blue or a yellow vertex is a predecessor of a green one (i.e. there is a path from a blue or yellow vertex to a green one) consider the smallest level green vertex on this path, and transform all the vertices on that path starting with the considered green one.

The process stops when none of these operations can be applied: If there are no blue or yellow vertices we are done. One should expand definitions determined by white vertices in both F_1 and F_2 in the order given by the vertex transformations of G proposed by the algorithm. Otherwise the attempt failed.

This algorithm can be improved since sometimes it is better to apply 2° before 1°. One can mark the coloring position when 2° is applicable and go back to it if further application of 1° has failed.

Once we got formulas F_1' and F_2' with $L(F_1') = L(F_2')$ transform them both to the prenex normal form, (we can consider F_1 and F_2 as closed) and match quantifiers (i.e. the corresponding variables) in all allowed ways. Here we have in mind the possibility of changing the order of the quantifiers of the same type. In kernels K_1 and K_2 of the prenex forms replace any occurrence of a predicate by a logical variable using the same variable for the same predicate (hawing in mind its arguments, too). So we obtain propositional formulas P_1 and P_2. If $P_1 \Leftrightarrow P_2$ is a tautology then $F_1 \Leftrightarrow F_2$ is true. If we incorporate the detecting of equality of terms on the level of arguments of the predicate, and further some properties on the level of predicates such as the symmetry, etc. the application of the proposed heuristic will be more successful.

Example. In arithmetical graph theory [**4**] we have a basic predicate $S1$ (X, Y, U) saying that "vertices X and Y are joined by line U", and definitions:

$R1(X,Y) \Leftrightarrow (\exists U) S1(X,Y,U)$ "X and X are adjacent",
$Q1(X) \Leftrightarrow (\forall Y) \neg R1(X,Y)$ "X is isolated",
$R2(X,U) \Leftrightarrow (\exists Y) S1(X,Y,U)$ "vertex X and line U are incident",
$P1(X) \Leftrightarrow (\forall X) R1(X,Y)$ "graph is complete".

To prove that to say "If graph is complete then there are no isolated vertices" is the same as to say "If there is a vertex not incident with a line then graph is not complete", we have to examine formulas: $P1 \Rightarrow \neg(\exists X) Q1(X)$ and $(\exists X)(\forall U) \neg R2(X,U) \Rightarrow P1$. The corresponding digraph G is given in Fig. 1 with its original and final colouring.

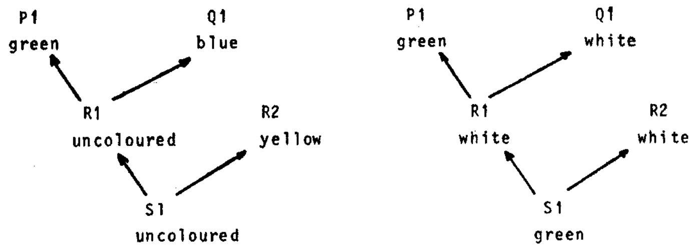

Fig. 1

We have to transform one after another $Q1$, $R1$ in the first formula, and $R2$ in the other one. The rest is technical.

Very often in automated theorem proving we have to compare a subgoal S with known theorems T_i stored in the computer. In order to avoid the whole procedure with each T_i, we should select those T_i which promise a success. We could classify the theorems T_i by some parameters related to coloring of G induced by T_i and S. These parameters can include number of yellow (blue) vertices, the existence of paths between yellow (blue) and green vertices, and the levels of yellow (blue) vertices.

The problem of definition instatiation in automatic theorem proving is discussed, for example, in [**1**], [**2**]. Experience shows that the definition instatiation should be carefully controlled and applied only if all other strategies fail or if it will, "do some good". It is recommended to instantiate definition of "strange" terms. In a formula of the form $A \Rightarrow B$ one should instantiate the definition of the main predicate in the conclusion B, or instantiate a definition into the hypothesis A if we find in a conjunctive position of A a possible match for B.

All these recommendations are covered by our recoloring heuristics. "Strange" terms can be explained as predicates with a high level. Instantiation of a definition will "do some good" if our recoloring heuristics are applicable and in particular it is applied to the case when it will lead to the matching of a conjunct of A with B in $A \Rightarrow B$. In our opinion, the instantatiation of the main predicate in B is not always recommendable; one should look at the coloring structure of the definition digraph and apply recoloring heuristics.

This heuristic is partially implemented in the system "GRAPH".

REFERENCES

[1] W.W. Bledsoe, P. Bruell, *A man-machine theorem-proving system*, Artificial Intelligence **5** (1974), 51–72.

[2] W.W. Bledsoe, *Non-resolution theorem proving*, Artificial Intelligence **9** (1977), 1–35.

[3] B.R. Boričić, *Equational reformulation of intaitionistic proposilional calculus and classical first-order predicate calculus*, Publ. Inst. Math. (Beograd), **29**(**43**) (1981), 23–28.

[4] D. Cvetkovć, "A project for using computers in further development of graph theory", *The Theory and Applications of Graphs*, (Proc. 4th Internat. Conf. Theory and Application of Graphs, Kalamazooo 1980) ed G. Chartrand, et all, John Wiley & Sons, New York-Chichester-Brisbane-Toronto-Singapore 1981, 285–296.

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

[6] R.E. Kling, *A paradigm for reasoning by analogy*, Artificial Intelligence **2** (1971), 147–178.

[7] S.B. Prešić, *Equational reformulation of formal theories*, Publ. Inst. Math. (Beograd) **19**(**33**) (1975), 131–138.

(Received 20 06 1983)