THEOREM PROVING IN TYPE THEORY

Peter B. Andrews and Eve Longini Cohen
Mathematics Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

As one aspect of the endeavor to create new intellectual tools for mankind, we wish to enable computers to prove, and to assist in the proofs of, theorems of mathematics and (eventually) other disciplines which have achieved the requisite logical precision. For this purpose, a particularly suitable formal language is Church's formulation [4] of type theory with A-conversion. in this language traditional mathematical notations can be expressed very directly, and the intuitive distinctions between different types of mathematical entities (such as numbers, functions, and sets of functions) are made syntactically explicit.

The program for proving theorems of type theory which we discuss is intended to provide experience relevant to such a project, and was developed with the aid of Charles E, Blair and John J. Grefenstette.

After reducing the negation of the sentence to be proved to a set of clauses (basically as in the resolution method), the program seeks an acceptable mating by the method outlined in [3]. The unifying substitutions are found by Huet's algorithm [6], augmented by heuristics to minimize branching of the search tree, and a procedure for deleting nodes which are essentially supersets of other nodes. The semi-decision procedures which search for a potentially acceptable mating and for the associated unifying substitutions operate in parallel, and interact so that information acquired by each procedure limits the other's search. After an acceptable mating is found, the computer constructs from it a more traditional refutation, using substitution, cut (ground resolution), and simplification of disjunctions as rules of inference.

The program can be run in automatic or interactive mode. The interactive system embodies a set of logical rules which is in a certain sense complete [1], but the user must provide some of the substitutions for predicate variables. (As shown in [2], even completeness in this weak sense is not trivial.) The program in automatic mode is not logically complete for type theory (though it is complete when applied to sentences of first order logic), since no practical method is known for automatically generating all required substitutions for predicate variables. This is the fundamental theoretical problem of automatic theorem-proving in type theory, and no practical general solution to it seems imminent, since substitutions for predicate variables often express the important concepts in a mathematical proof.

This research was supported by NSF Grants DCR71-01953-A04 and MCS76-06087.

As noted in [1], for certain purposes axioms of extensionality, descriptions, or choice must be taken as hypotheses, Actually, the introduction of Skolem functions to eliminate essentially existential quantifiers involves an implicit use of the Axiom of Choice (AC) in type theory, so the system can prove certain consequences of AC, such as

$$\forall Y\,\exists Z,\ [R_{o\alpha Y Z}] \Rightarrow \text{ap}_{M\,\forall Y_\S}\ \text{on}\ i\ f^F n^Y J$$

Among the theorems which can be proved completely automatically are Cantor's Theorems that a set has more subsets than members, and that there are more functions on a non-unit set than members of the set. (Thus there are uncountably many functions of natural numbers.) Following [5], the Cantor Theorem for Sets can be expressed by the sentence $\sim\exists H\ \forall S\ \exists j.\ [H\ j = S]$, which asserts that there is no function H from individuals to sets which has every set S in its range. The computer decides to substitute for S_{oi} the wff $[\lambda Xj.\sim H\ X_t X_\ell]$, which denotes the set $(x|\sim xeHx)$, and expresses the key idea in the classical diagonal argument.

REFERENCES

[1] Peter B. Andrews, "Resolution in Type Theory", Journal of Symbolic Logic 36 (1971), 414-432.

[2] Peter B. Andrews, "Resolution and the Consistency of Analysis", Notre Dame J. of Formal Logic XV (1974), 73-84.

[3] Peter B. Andrews, "Refutations by Matings", IEEE Transactions on Computers C-25 (1976), 801-807.

[4] Alonzo Church, "A Formulation of the Simple Theory of Types", Journal of Symbolic Logic 5 (1940), 56-68.

[5] Gerard P. Huet, "A Mechanization of Type Theory", Third International joint Conference on Artificial Intelligence, Stanford, 1973, 139-146.

[6] Gerard P. Huet, "A Unification Algorithm for Typed A-Calculus", Theoretical Computer Science 1 (1975), 27-57.

[7] D. C. Jensen and T. Pietrzykowski, "Mechanizing a)- Order Type Theory Through Unification", Theoretical Computer Science 3 (1976), 123-171.