(4)

TABLOG:

THE DEDUCTIVE TABLEAU

PROGRAMMING LANGUAGE

Final Technical Report:
Department of the Navy
Contract N00039-84-C-0211 (task 7)
Expiration Date: November 20, 1987

by
Zohar Manna, Professor
Computer Science Department
Stanford University
Stanford, California 94305

Sponsored by

Defense Advanced Research Projects Agency (DoD)
1400 Wilson Boulevard
Arlington, Virginia 22209-2389

"TABLOG: THE DEDUCTIVE TABLEAU PROGRAMMING LANGUAGE"

Issued by Space and Naval Warfare Systems Command

Under Contract #N00039-84-C-0211, Task 7

"The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TABLOG: THE DEDUCTIVE TABLEAU PROGRAMMING LANGUAGE		5. TYPE OF REPORT & PERIOD COVERED final technical report 11/21/85-11/20/87 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Zohar Manna		8. CONTRACT OR GRANT NUMBER(s) N00039-84-C-0211, Task 7
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS SPAWAR 3241C2 Space and Naval Warfare Systems Command Washington, D.C. 20363-5100		12. REPORT DATE November 1988 13. NUMBER OF PAGES 5
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) ONR Representative - Mr. Robin Simpson 202 McCullough Stanford University Stanford, CA 94305		15. SECURITY CLASS. (of this report) Unclasified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See attached report.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TECHNICAL SUMMARY

Our research concentrated on the following topics:

The Deductive-Tableau System ([MW1][MW2])

Theorem provers have exhibited super-human abilities in limited, obscure subject domains but seem least competent in areas in which human intuition is best developed. One reason for this is that an axiomatic formalization requires us to state explicitly facts that a person dealing in a familiar subject would consider too obvious to mention; the proof must take each of these facts into account explicitly. A person who is easily able to construct an argument informally may be too swamped in detail to understand, let alone produce, the corresponding formal proof. A continuing effort in our research is to make formal theorem proving more closely resemble intuitive reasoning. One case in point is our treatment of special relations.

In most proofs of interest for program synthesis, certain mathematical relations, such as equality and orderings, present special difficulties. These relations occur frequently in specifications and in derivation of proofs. If their properties are represented axiomatically, proofs become lengthy, difficult to understand, and even more difficult to produce or discover automatically. Axioms such as transitivity have many consequences, most of which are irrelevant to the proof; including them produces an explosion in the search space.

For the equality relation, the approach that was adopted early on is to represent its properties with rules of inference rather than axioms. In resolution systems, two rules of inference, paramodulation (Wos and Robinson) and E-resolution (Morris), were introduced. Proofs using these rules are shorter and clearer, because one application of a rule can replace the application of several axioms. More importantly, we may drop the equality axioms from the clause set, thus eliminating their numerous consequences from the search space.

We have discovered two rules of inference that play a role for an arbitrary relation analogous to that played by paramodulation and E-resolution for the equality relation. These rules apply to sentences employing a full set of logical connectives; they need not be in the clause form required by traditional resolution theorem provers. We intend both these rules to be incorporated into theorem provers for program synthesis.

Employing the new special-relations rules yields the same benefits for an *arbitrary* relation as using paramodulation and E-resolution yields for equality: proofs become shorter and more comprehensible and the search space becomes sparser.

The TABLOG language and its implementation ([M][MMW1][MMW2][MMW3])

Logic programming uses formal proofs as the computation paradigm. That is, a logic program is a theory, expressed in a given logic, that captures some properties of the real world. The execution of such a program is the proof of some theorem in this theory.

TABLOG is a new logic-programming language ([M][MMW1][MMW2]) based on quantifier-free first-order logic with equality, using the proof rules of the deductive-tableau theorem-proving method as the execution mechanism.

The main features of TABLOG are consequences of the use of full first-order logic. In particular, TABLOG incorporates all the standard connectives, not only implication and conjunction, but also equality, negation and equivalence. Programs are nonclausal: they do not need to be in Horn-clause

form or any other normal form. Programs can compute relations (as in PROLOG) or functions (as in LISP), whichever is more appropriate; this improves the clarity and the efficiency of programs. Terms are lazy-evaluated to make the use of functions more convenient. No cut annotation is required as the system can detect such optimizations dynamically.

Three deduction rules are used for the execution of the programs: *nonclausal resolution* (case analysis), *equality replacement* (repalcement of equal terms), and *equivalence replacement* (repalcement of equivalent subsentences).

We have developed of a compiler for TABLOG; this compiler will produce code for a virtual TABLOG machine, similar to the Warren abstract machine. This compiler, written in TABLOG itself, will support a new syntax, which includes types and an elaborate notion of modules and generic modules. The virtual machine was implemented on a Sun workstation.

A Resolution Approach to Temporal Proofs ([A][AM1][AM2])

A novel proof system for temporal logic was developed. The system is based on the classical non-clausal resolution method, and involves a special treatment of quantifiers and temporal operators.

Soundness and completeness issues of resolution and other related systems were investigated. While no effective proof method for temporal logic can be complete, we established that a simple extension of the resolution system is as powerful as Peano Arithmetic.

The use of temporal logic as a programming language was explored. We suggested that a specialized temporal resolution system could effectively interpret programs written in a restricted version of temporal logic.

We also provided analogous resolution systems for other useful modal logics, such as certain modal logics of knowledge and belief.

Temporal Logic Programming ([A][AM1][AM3])

Temporal logic is a formalism for reasoning about a changing world. Because the concept of time is directly built into the formalism, temporal logic has been widely used as a specification language for programs where the notion of time is central. For the same reason, it is natural to write such programs directly in temporal logic. We developed a temporal logic programming language, TEMPLOG, which extends classical logic programming languages, such as PROLOG, to include programs with temporal constructs. A PROLOG program is a collection of classical *Horn clauses*. A TEMPLOG program is a collection of *temporal Horn clauses*, that is, Horn clauses with certain temporal operators. An efficient interpreter for PROLOG is based on *SLD-resolution*. We base an interpreter for TEMPLOG on a restricted form of our temporal resolution system, *temporal SLD-resolution*.

Logic Programming Semantics: Techniques and Applications ([B1]–[B3])

It is generally agreed that providing a precise formal semantics for a programming language is helpful in fully understanding the language. This is especially true in the case of logic-programming-like languages for which the underlying logic provides a well-defined but insufficient semantic basis. Indeed, in addition to the usual model-theoretic semantics of the logic, proof-theoretic deduction plays a crucial role in understanding logic programs. Moreover, for specific implementations of logic programming, e.g. PROLOG, the notion of deduction stategy is also important.

We provided semantics for two types of logic programming languages and develop applications of these semantics. First, we propose a semantics of PROLOG programs that we use as the basis of a proof method for termination properties of PROLOG programs. Second, we turn to the temporal logic programming language TEMPLOG of Abadi and Manna, develop its declarative semantics, and then use this semantics to prove a completeness result for a fragment of temporal logic and to study TEMPLOG's expressiveness.

In our PROLOG semantics, a program is viewed as a function mapping a goal to a finite or infinite sequence of answer substitutions. The meaning of a program is then given by the least solution of a system of functional equations associated with the program. These equations are taken as axioms in a first-order theory in which various program properties, especially termination or non-termination properties, can be proved. The method extends to PROLOG programs with extra-logical features such as *cut*.

For TEMPLOG, we provide two equivalent formulations of the declarative semantics: in terms of a minimal temporal Herbrand model and in terms of a least fixpoint. Using the least fixpoint semantics, we are able to prove that TEMPLOG is a fragment of temporal logic that admits a complete proof system. This semantics also enables us to study TEMPLOG's expressiveness. For this, we focus on the propositional fragment of TEMPLOG and prove that the expressiveness of propositional TEMPLOG queries essentially corresponds to that of finite automata.

REFERENCES

Research papers and Ph.D. theses supported by this contract.

* indicates papers that are attached as part of this report.

[A] M. Abadi (supervised by Z. Manna), *Temporal-logic theorem proving*, Ph.D. Thesis, Computer Science Dept., Stanford University (1986).

[AM1] A. Abadi and Z. Manna, "Nonclausal temporal deduction," *Logic of Programs Conference*, Brooklyn, NY, Lecture Notes in Computer Science 193 (R. Parikh, ed.), Springer-Verlag, June 1985, pp. 1–15.

[AM2] A. Abadi and Z. Manna, "A timely resolution," *Symposium on Logic of Computer Science*, Cambridge, MA, June 1986, pp. 176–186.

*[AM3] A. Abadi and Z. Manna, "Modal theorem proving," *8th International Conference on Automated Deduction*, Oxford, England, Lecture Notes in Computer Science 230 (R. Parikh, ed.), Springer-Verlag, July 1986, pp. 172–189.

*[AM4] A. Abadi and Z. Manna, "Temporal logic programming," *4th Symposium on Logic Programming*, San Francisco, CA, Sept. 1987, pp. 4–16. To appear also in the *Journal of Symbolic Computation* (1989).

*[B1] M. Baudinet, "Proving Termination Properties of PROLOG Programs: A Semantic Approach", *Proceedings of the Third Annual Symposium on Logic in Computer Science*,, pp. 336–347, Edinburgh, Scotland, July 1988.

*[B2] M. Baudinet, "Temporal Logic Programming is Complete and Expressive", *Proceedings of the Sixteenth ACM Symposium on Principles of Programming Languages*, Austin, Texas, January 1989.

[B3] M. Baudinet (supervised by Z. Manna), *Logic Programming Semantics: Techniques and Applications*, Ph.D. Thesis, Computer Science Dept., Stanford University (1989).

[M] Y. Malachi (supervised by Z. Manna), *Nonclausal Logic Programming*, Ph.D. Thesis, Computer Science Dept., Stanford University, 1986.

*[MMW1] Y. Malachi, Z. Manna, and R. Waldinger, "TABLOG - The deductive tableau programming language," *ACM Symposium on LISP and Functional Programming*, Austin, Texas (Aug. 1984), pp. 323–330. Also in *Logic Programming: Functions, Relations, and Equations* (D. DeGroot and G. Lindstrom, eds.), Prentice-Hall, Englewood Cliffs, NJ, 1986, pp. 365–394.

[MMW2] Y. Malachi, Z. Manna, and R. Waldinger, "TABLOG: Functional and relational programming in one framework," *IEEE Software*, Vol. 3, No. 1 (Jan. 1986), pp. 75–76.

*[MMW3] E. Muller, Z. Manna, and R. Waldinger, "The TABLOG Language," Draft, 1988.

*[MW1] Z. Manna and R. Waldinger, "Special relations in automated deduction," *Journal of the ACM*, Vol. 33, No. 1 (Jan. 1986), pp. 1–59.

*[MW2] Z. Manna and R. Waldinger, "The origin of the binary-search paradigm," *Science of Computer Programming* Journal, Vol. 9, No. 1 (August 1987), pp. 37–83.