UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
AD-A216 670	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
	5. MONITORING ORGANIZATION REPORT NUMBER(S): AFOSR-TR- 89-0570
6a. NAME OF PERFORMING ORGANIZATION: Stanford University; 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: AFOSR
6c. ADDRESS (City, State, and ZIP Code): 660 Arguello Way, Encina Hall, Stanford, CA 94305-6060	7b. ADDRESS (City, State, and ZIP Code): Building 410, Bolling, AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR; 8b. OFFICE SYMBOL (If applicable): NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-87-0149
8c. ADDRESS (City, State, and ZIP Code): Building 410, Bolling, AFB DC 20332-6448	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 61102F; PROJECT NO. 2304; TASK NO. A2; WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): Deductive Computer Programming

12. PERSONAL AUTHOR(S): Professor Manna

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)
Final	FROM 1 Mar 87 TO 28 Feb 88	

15. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

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

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.

90 01 09 179

~~90 5 11 036~~

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code): (202) 767- ; 22c. OFFICE SYMBOL: NM

DTIC ELECTE JAN 09 1990 S D

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

DEDUCTIVE COMPUTER PROGRAMMING

AFOSR-TR- 89-0570

Final Technical Report:
Department of the Air Force
Grant AFOSR 87-0149
Expiration Date: February 29, 1988

DTIC COPY INSPECTED

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

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

TECHNICAL SUMMARY

Our research concentrated on the following topics:

- **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.

- **The TABLOG language and its implementation** ([MMW])

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.

- **Logic: The Calculus of Computer Science** ([MW])

The research papers in which we have presented the deductive approach to program synthesis has been addressed to the usual academic readers of the scholarly journals. In an effort to make this work accessible to a wider audience, including computer science undergraduates and programmers, we have developed a more elementary treatment in the form of a two-volume book, *The Logical Basis for Computer Programming*, Addison-Wesley (Manna and Waldinger [85c]).

This book requires no computer programming and no mathematics other than an intuitive understanding of sets, relations, functions, and numbers; the level of exposition is elementary. Nevertheless, the text presents some novel research results, including

- theories of strings, trees, lists, finite sets and bags, which are particularly well suited to theorem-proving and program-synthesis applications;
- formalizations of parsing, infinite sequences, expressions, substitutions, and unification;
- a nonclausal version of skolemization;
- a treatment of mathematical induction in the deductive-tableau framework.

- **Verification of Concurrent Programs** ([MP])

We studied in detail the proof methodologies for verifying temporal properties of concurrent programs. Corresponding to the main classification of temporal properties into the classes of *safety* and *liveness* properties, appropriate proof principles were presented for each of the classes.

We developed proof principles for the establishment of *safety* properties. We showed that essentially there is only one such principle for safety proofs, the invariance principle, which is a generalization of the method of intermediate assertions. We also indicated special cases under which these assertions can be found algorithmically.

The proof principle that we developed for *liveness* properties is based on the notion of well-founded descent of ranking functions. However, because of the nondeterminancy inherent in concurrent computations, the well-founded principle must be modified in a way that is strongly dependent on the notion of *fairness* that is assumed in the computation. Consequently, three versions of the well-founded principle were presented, each corresponding to a different definition of fairness.

REFERENCES

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

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

*[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).

[MP] Z. Manna and A. Pnueli, *Temporal Specification and Verification of Concurrent Program*, Textbook, to appear 1989.

[MW] Z. Manna and R. Waldinger, *Logical Basis for Computer Programming*, Textbook, Addison-Wesley Pub., Volume 2: Deductive Systems (to appear, 1989).

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