AD-A219 848

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		FINAL Report, 1 Aug 88 to 30 Sep 89

4. TITLE AND SUBTITLE
DEDUCTIVE COMPUTER PROGRAMMING

5. FUNDING NUMBERS
AFOSR-88-0281
61102F 2304/A2

6. AUTHOR(S)
Zohar Manna, Professor

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Stanford University
Computer Science Department
Stanford, CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

AFOSR-TR- 90-0333

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFSOR/NM
Building 410
Bolling AFB, DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFOSR-88-0281

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In an effort to make the research accessible to a wider audience besides the scholarly journals, the PI has published two volumes of the book "The Logical Basis of Computer Programming". This book requires only an intuitive understanding of sets, relations, functions, and numbers. Despite the elementary approach, the text presents some novel research results, including: theories of strings, trees, lists and finite sets which are particulary suited to theoren-proving and program-synthesis applications; formalization of parsing; a noncluasal version of skolemization; a treatment of mathematical induction in the deductive-tableau framework. The implemented tableau system combines features lacking elsewhere such as producing proofs by mathematical inducing.

14. SUBJECT TERMS	15. NUMBER OF PAGES
	8
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

DEDUCTIVE COMPUTER PROGRAMMING

Final Technical Report (revised):
Department of the Air Force
Grant AFOSR 88-0281
Expiration Date: September 30, 1989

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

TECHNICAL SUMMARY

Our research concentrated on the following topics:

AUTOMATED DEDUCTION

- **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.

- **TABLEAU, An Interactive Graphic Deductive System** ([BMW])

We have collaborated with a team in developing an interactive theorem prover, TABLEAU, based on the deductive-tableau framework. Implemented on an Apple Macintosh computer, the system exploits the graphical interface of that machine in communicating with the user. Rather than relying on the keyboard, the user may select with a mouse which step in the proof to attempt next. Although directing the proof is the responsibility of the user, the system intervenes if the user attempts an illegal step. The system can construct proofs in propositional and predicate logic, in theories of the nonnegative integers, trees, and tuples, and in new theories introduced by the user. The system is now in final stages of development and documentation.

The system has a combination of features lacking elsewhere. In particular,

- It can handle theorems with both universal and existential quantifiers.
- It can produce proofs by mathematical induction, including well-founded induction.
- It has special provisions for reasoning about equality. Furthermore, the convenient interface of the system makes it far easier to use in constructing a detailed proof than it is to prove the same result by hand, a feature unfortunately not shared with many systems.

The system is being augumented now in several directions:

- Introduce the capability of adding new deduction rules. This would facilitate the application of the system to new theories.
- Introduce a facility for extracting information from proofs. This information could be a program, a plan, or a database transaction.
- Allow the gradual automation of the system. In particular, we would like to automate certain routine and repetitive aspects of the proof process.

- **A Resolution Approach to Temporal Proofs** ([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.

We have investigated the use of the system for verifying concurrent programs. We also provided analogous resolution systems for other useful modal logics, such as certain modal logics of knowledge and belief.

LOGIC PROGRAMMING

- **Logic Programming Semantics: Techniques and Applications** ([B1],[B2])

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.

- **Temporal Logic Programming** ([AM1])

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*.

REAL-TIME SYSTEMS

- **Real-Time Reasoning in Temporal Logic** ([AH1])

We introduce a real-time temporal logic for the specification of reactive systems. The novel feature of our logic, TPTL (Time Propositional Temporal Logic), is the adoption of temporal operators as quantifiers over time variables; every modality binds a variable to the time(s) it refers to.

TPTL is demonstrated to be both a natural specification language as well as a suitable formalism for verification and synthesis. We present a tableau-based decision procedure and model-checking algorithm for TPTL. Several generalizations of TPTL are shown to be highly undecidable.

- **Real-time Logics: Complexity and Expressiveness** ([AH2])

The theory of the natural numbers with linear order and monadic predicates underlies propositional linear temporal logic. To study temporal logics for real-time systems, we combine this classical theory of infinite state sequences with a theory of time, via a monotonic function that maps every state to its time. The resulting theory of timed state sequences is shown to be decidable, albeit nonelementary, and its expressive power is characterized by Omega-regular sets. Several more expressive variants are proved to be highly undecidable.

This framework allows us to classify a wide variety of real-time logics according to their complexity and expressiveness. In fact, it follows that most formalisms proposed in the literature cannot be decided. We are, however, able to identify two elementary real-time temporal logics as expressively complete fragments of the theory of timed state sequences, and give tableau-based decision procedures. Consequently, these two formalisms are well-suited for the specification and verification of real-time systems.

THEORY OF PROGRAMMING

- **A Hierarchy of Temporal Properties** ([MP2])

We proposed a classification of temporal properties into a hierarchy which refines the known *safety-liveness* classification of properties. The classification is based on the different ways a property of finite computations can be extended into a property of infinite computations.

This hierarchy was studied from three different perspectives, which were shown to agree. Respectively, we examined the cases in which the finitary properties, and the infinitary properties extending them, are unrestricted, specifable by temporal logic, and specifiable by predicate automata. The unrestricted view leads also to a topological characterization of the hierarchy as occupying the lowest two levels in the Borel hierarchy.

For properties that are expressible by temporal logic and predicate automata, we provide a syntactic characterization of the formulae and automata that specify properties of the different classes. The temporal logic characterization strongly relies on the use of the past temporal operators.

Corresponding to each class of properties, we presented a proof principle that is adequate for proving the validity of properties in that class.

- **Specification and Verification by Predicate Automata** ([MP3])

We examined the possibility of specifying and verifying temporal properties using an extension of finite-state automata, called *predicate automata*. These automata extend the conventional notion of automata in three respects. The first extension is that the conditions for transitions between states can be arbitrary predicates expressed in a first-order language. The second extension is that these automata inspect *infinite* input sequences, and hence a more complex acceptance criterion is needed. The third extension is that non-determinism is interpreted *universally*, rather than *existentially*, as is the case in conventional non-deterministic finite-state automata. This means that if the automata can generate several possible runs, in response to a given input, then it is required that *all* runs are accepting.

By introducing conventions for representing automata in a structured form, we demonstrated that specification of temporal properties by automata can become very legible and understandable, and presents a viable alternative to their formulation in temporal logic.

A single proof rule was presented for proving that a given program satisfies a property specifiable by a predicate automaton. The rule was shown to be sound and relatively complete.

REACTIVE SYSTEMS

- **Formal Approaches to the Construction of Correct Reactive Programs** ([MP1])

Reactive programs are programs whose role is to maintain an on-going interaction with their environment, rather than to produce a computational final result on termination. Programs belonging to this class are usually described by some of the attributes: concurrent, distributed, real-time, and typical examples of such programs are: embedded programs, communication networks, control

programs of industrial plants, operating systems, real-time programs, etc. Clearly, the correct and reliable construction of such programs is one of the most challenging programming tasks existing today.

Due to the special character of these programs, the formal approach to their specification and development must be based on the study of their *behavior*, rather than on the function or relation they compute, an approach which is adequate for computational and sequential programs.

We developed formal methods for the specification, verification and development of reactive programs, based on the formalism of *temporal logic* that has been specifically developed to reason about behaviors of reactive programs.

Among the topics we investigated are:

- Modeling reactive programs as fair transition systems.
- The language of temporal logic and its usage for the specification of program properties and system requirements.
- A classification of specifications into the classes of *safety* properties, *responsiveness* properties, etc.
- Methodologies for formal verification of the safety properties of a reactive program, and development approaches derived from them.
- Methodologies for formal verification of the responsiveness properties of a reactive program, and derived development approaches.
- The case of finite-state programs and automatic verification tools for this case.

- **Reactive Systems: Specification and Verification** ([MP4]–[MP6])

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.

PUBLICATIONS

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

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

TEXTBOOKS

*[MW] Z. Manna and R. Waldinger, "Logical Basis for Computer Programming", Volume 2: Deductive Systems, 642 pp., Addison-Wesley Pub., Reading, MA, 1989.

[MP1] Z. Manna and A. Pnueli, "The Temporal Logic of Reactive Programs", Volume 1: The Languages, Volume 2: Verification, Springer-Verlag (to appear, 1990).

CONFERENCE PAPERS (IN PROCEEDINGS)

*[MP2] Z. Manna and A. Pnueli, "The anchored version of the temporal framework" (invited paper), In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Lecture Notes in Computer Science 354, Springer-Verlag, 1989, pp. 201–284.

[MP3] Z. Manna and A. Pnueli, "Specification and verification of concurrent programs by ∀-automata", In *Temporal Logic in Specification*, Lecture Notes in Computer Science 398, Springer-Verlag, 1989, pp. 124–164.

*[MP4] Z. Manna and A. Pnueli, "Completing the temporal picture", 16th International Colloquium on Automata, Languages, and Programming (ICALP), Stresa, Italy (July 1989), Lecture Notes in Computer Science 372, Springer-Verlag, 1989, pp. 534–558.

CONTRIBUTION TO BOOKS

[MP5] Z. Manna and A. Pnueli, "An exercise in the verification of multi-process programs" (invited paper), in *Beauty is our business*, Springer-Verlag, 1990 (in press).

JOURNAL ARTICLES

*[AM1] M. Abadi and Z. Manna, "Temporal logic programming", Journal of Symbolic Computation, Vol. 8, No. 3 (Sept. 1989), pp. 277Π-295.

[AM2] M. Abadi and Z. Manna, "Nonclausal deduction in first-order temporal logic", Journal of the ACM (to appear April 1990).

[MP6] Z. Manna and A. Pnueli, "Completing the temporal picture", Theoretical Computer Science Journal (in press).

PAPERS BY PH.D. STUDENTS

Supervised by Zohar Manna

*[AH1] R. Alur and T.A. Henzinger, "A Really Temporal Logic", 13th IEEE Symposium on Foundations of Computer Science, 1989, 164Π-169.

[AH2] R. Alur and T.A. Henzinger, "Real-time Logics: Complexity and Expressiveness", Symposium on Logic in Computer Science, Philadelphia, PA, June 1990.

[B1] M. Baudinet, "Temporal Logic Programming is Complete and Expressive", Proceedings of the Sixteenth ACM Symposium on Principles of Programming Languages, Austin, Texas, January 1989, pp. 267–280.

[T] J. Traugott, "Deductive Synthesis of Sorting Programs", Journal of Symbolic Computation, Vol. 7, No. 6, June 1989, pp. 533–572.

PH.D. THESIS

Supervised by Zohar Manna

*[B2] M. Baudinet, "Logic Programming Semantics: Techniques and Applications", Ph.D. Dissertation, Computer Science Department, Stanford University, 1989.

USER MANUAL

[BMW] R. Burback, Z. Manna and R. Waldinger et. al., "Using the Deductive Tableau System", User Manual, Computer Science Department, Stanford University, 1990.