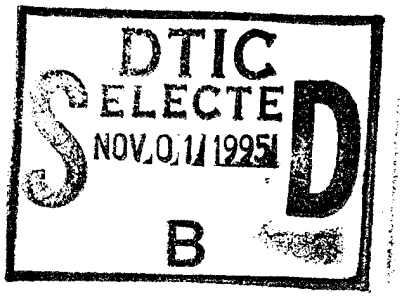
Final Report 0002/A001 • March 20, 1995

OBJECT-ORIENTED DESIGN AND SPECIFICATION

For the period 1 January 1990 through 31 December 1994

José Meseguer, Principal Scientist
Computer Science Laboratory

SRI Project ECU 8844



Prepared for:

Chief of Naval Research
Code 3330/Annual Report
Ballston Tower One
800 North Quincy Street
Arlington, VA 22217-5660

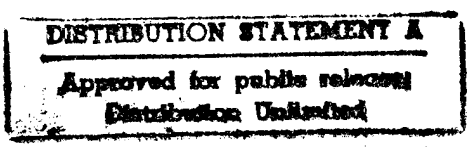
cc: Dr. Ralph Wachter, Scientific Officer

Contract No. N00014-90-C-0086

Approved:

Mark Moriconi, Director
Computer Science Laboratory

Donald Nielson, Vice President
Computing and Engineering Sciences Division



Object-Oriented Design and Specification

José Meseguer
SRI International, Menlo Park CA 94025

1 Introduction

This final report summarizes the research performed for the Office of Naval Research under Contract N00014-90-C-0086 on the topic "Object-Oriented Design and Specification." The project began on 1 January 1990 and ended on 31 December 1994. Dr. José Meseguer was the project leader. Drs. Patrick Lincoln, and Narciso Martí-Oliet and Mr. Timothy Winkler also worked on the project.

Early in the project, an important breakthrough took place with the discovery of rewriting logic, which was then further developed [25, 23, 26]. Rewriting logic has proved to be a very flexible multiparadigm logic [27] of great simplicity allowing the unification of equational programming, Horn logic programming, object-oriented programming, and concurrent programming. In particular, a very simple semantics can be given in rewriting logic to concurrent object-oriented programming [24, 29, 28] and to object-oriented databases [33]. This is particularly encouraging given that concurrent object-oriented programming and object-oriented databases are disciplines generally considered to lack a precise semantics.

Encouraged by these results, a preliminary language design for Maude, a wide-spectrum multiparadigm language based on rewriting logic and containing a subset called Simple Maude that can be efficiently compiled onto a wide variety of parallel machines has been developed [34, 27, 30, 28, 17]; and very encouraging experience about its suitability for specifying concurrent systems, AI problems, programming languages, and logics has been gathered [30, 20, 21, 32, 22]. In addition, some initial progress has been made on transformation and compilation techniques for the Simple Maude parallel programming sublanguage [17, 18].

Maude contains the equational language OBJ as its functional sublanguage. The OBJ3 system and its underlying theory have been developed with the support of the Office of Naval Research. OBJ3 is used throughout the world in more than 170 universities and research laboratories. Progress has been made in further developing OBJ3's underlying abstract data type theory [35, 31], and in improving the OBJ3 user manual [8].

Our work on rewriting logic and Maude during this project has stimulated their use by a growing community of international researchers in a variety of ways. For example, C. Kirchner, H. Kirchner, and Vittek are using it as a foundation for their ELAN language and applying it to the specification of computational systems [9, 39, 10]; Futatsugi has adopted it as the semantic basis of his Cafe language [6]; Denker and Gogolla have used Maude to give semantics to the TROLL *light* object-oriented language [5]; Viry has developed useful program transformation techniques for rewrite theories using completion methods [38]; Laneve and Montanari have shown how the heavy notation of the residual calculus can be replaced by a simpler semantic account using rewriting logic [11, 12]; Corradini, Gadducci, and Montanari have studied its relationships with other categorical models and with event structures [4], as well as the topic of infinite rewritings [3]; Levy and Agustí are applying it to their work on automated deduction [16, 14, 15]; Reichel has found it useful in his final coalgebra semantics for objects [36]; Lechner, Lengauer, and Wirsing have carried out an ambitious case study investigating the expressiveness of rewriting logic and Maude for object-oriented specification and have explored refinement concepts [13]; Talcott is using rewriting logic to give a concurrent semantics to actor systems [37]; and applications to Petri net algebraic specification have been developed by Battiston, Crespi, De Cindio, and Mauri [1], and by Bettaz and Maouche [2].

2 Accomplishments

Under contract N00014-90-0086, we have accomplished the following:

Rewriting Logic. The logical basis on which the main ideas developed in the project are based is provided by rewriting logic [25, 23, 26], a logic for reasoning correctly about *concurrent systems* having *states*, and evolving by means of *transitions*. The signature of a rewrite theory describes a particular structure for the states of a system—e.g., multiset, binary tree, etc.—so that its states can be distributed according to such a structure. The rewrite rules in the theory describe which *elementary local transitions* are possible in the distributed state by concurrent local transformations. The rules of rewriting logic allow us to reason correctly about which *general* concurrent transitions are possible in a system satisfying such a description. Thus, computationally, each rewriting step is a parallel local transition in a concurrent

system.

Alternatively, however, we can adopt a logical viewpoint instead, and regard the rules of rewriting logic as *metarules* for correct deduction in a *logical system*. Logically, each rewriting step is a logical *entailment* in a formal system. This second viewpoint is particularly fruitful when using rewriting logic as a logical framework [20, 21].

The computational and the logical viewpoints under which rewriting logic can be interpreted can be summarized in the following diagram of correspondences:

State	$\leftrightarrow$	*Term*	$\leftrightarrow$	*Proposition*
Transition	$\leftrightarrow$	*Rewriting*	$\leftrightarrow$	*Deduction*
Distributed Structure	$\leftrightarrow$	*Algebraic Structure*	$\leftrightarrow$	*Propositional Structure*

The last row of equivalences is actually quite important. Roughly speaking, it expresses the fact that a state can be transformed in a concurrent way only if it is nonatomic, that is, if it is *composed* out of smaller state components that can be changed independently. In rewriting logic this composition of a concurrent state is formalized by the *operations* of the signature Σ of the rewrite theory $\mathcal{R}$ that axiomatizes the system. From the logical point of view, such operations can naturally be regarded as user-definable *propositional connectives* stating the particular structure that a given state has. The papers [26, 22] give further discussion and examples illustrating the above correspondences between computational and logical concepts.

Semantics of Concurrent Objects and of Object-Oriented Databases. The naturalness with which concurrent object-oriented programming can be expressed in rewriting logic and can be unified with equational programming is particularly encouraging [24, 29, 28]. In a similar way, object-oriented databases can be given a logical semantics in rewriting logic [33]. In this way, the serious need for semantic foundations in these two areas has been satisfactorily met.

In addition, serious difficulties long recognized by many authors in integrating concurrency and inheritiance in object-oriented languages—the so-called "inheritance anomaly" problem—have been completely resolved using rewriting logic [29]; and equational functional programming is cleanly integrated with concurrent object-oriented program-

ming thanks to an embedding of equational logic within rewriting logic [27].

Preliminary Design of Maude. Rewriting logic is a very simple multiparadigm logic [27] on which to base a declarative wide-spectrum language unifying equational programming, Horn logic programming, object-oriented programming, and concurrent programming. Maude is our preliminary design for such a language [34, 27, 30, 28, 17]. In addition to *system modules* that are theories in rewriting logic, Maude provides explicit language support for *functional modules*—essentially identical to OBJ3 modules—and for *object-oriented modules*. Maude can be realized as an interpreter that executes such modules. We can view the modules executable in Maude as a commonly used subset of rewriting logic; however, nonexecutable specifications in arbitrary, finitely presented rewriting logic theories are possible within the language using *theories*, that can specify modules and can give formal requirements to the parameters of parameterized modules. In this way, a high-reuse "parameterized programming" methodology similar to that of OBJ but enjoying additional properties is achieved.

Besides the nonexecutable specification, executable specification, and rapid prototyping uses supported by Maude, machine-independent declarative parallel programming is supported in the Simple Maude subset, which can be efficiently compiled onto a wide variety of parallel machines. Program transformation techniques can then allow moving from nonexecutable specifications to executable ones, and from these to efficient parallel programs [17].

Specification Uses of Maude. Regarding the specificational uses of rewriting logic, an obvious question to ask is how general and natural rewriting logic is as a *semantic framework* in which to express different languages and models of computation. Our experience in this regard is quite encouraging. In several papers [26, 20, 28] we have been able to show that a wide variety of models of computation, including concurrent ones, can be naturally and directly expressed as rewrite theories in rewriting logic without any encoding or artificiality. As a consequence, models hitherto quite distant from each other can be naturally unified and interrelated within a common framework. This is particularly useful in the field of concurrency, where alternative models proposed as "basic" by different authors differ greatly, and also in attempts at

designing multiparadigm languages such as those combining functional and concurrent object-oriented programming.

In particular, we have shown that models and languages such as

- CCS,
- Petri nets,
- Actors,
- the UNITY language,
- the lambda calculus,
- equational languages, and
- concurrent object-oriented programming

can all be naturally expressed in rewriting logic. In addition, we have shown in [22] that rewriting logic has very good properties as a logic of change that avoids the *frame problem*, and that subsumes other logics previously proposed for this purpose.

In addition to all the uses already discussed, rewriting logic has also very good properties for specifying other logics in it, that is, as a *logical framework* [20, 21, 32]. Indeed, rewriting logic seems to have great flexibility to represent in a natural way many other logics, widely different in nature, including equational, Horn, and linear logics, and any sequent calculus presentation of a logic under extremely general assumptions about such a sequent presentation; moreover, quantifiers can also be treated without problems [20]. More experience in representing other logics is certainly needed, but we are encouraged by the naturalness and directness—often preserving the original syntax and rules—with which the logics that we have studied can be represented.

In summary, our experience with rewriting logic as a logical framework suggests that it has very good properties for this purpose in terms of:

- **Scope.** We actually conjecture that any finitely presented logic (for an adequate formal definition of "finitely presented logic" as a logic of practical interest) has a conservative representation in rewriting logic.
- **Representational adequacy.** The capacity for axiomatizing the syntactic constructs and structural properties of a logic as an order-sorted algebraic data type, as well as the rules of the logic

as rewrite rules, seems to make the "distance" between the logic and its representation negligible or non-existent in many cases.

Up to now, we have obtained faithful representations for:

- equational logic,
- Horn logic with equality,
- linear logic,
- logics with quantifiers, such as first-order classical and linear logics,
- any logic describable with a sequent calculus, including first-order classical, modal, linear, and intuitionistic logics.

Parallel Programming in Maude. We can usefully distinguish three parallel computing paradigms that in combination are sufficient for expressing with naturalness most parallel computing applications. These paradigms are:

1. **Parallel Symbolic Computing.** Functional, logic programming, and theorem-proving applications are typical of this paradigm.
2. **Highly Regular Data-Parallel Computing.** Many scientific computing applications, as well as cellular automata and systolic algorithms, are typical of this paradigm.
3. **Concurrent Object-Oriented Computing.** Many discrete event simulations, and many distributed AI and database applications can be naturally expressed and parallelized in this way.

A carefully chosen subset of rewriting logic gives rise to the multiparadigm parallel programming language Simple Maude, that is efficiently implementable on a wide range of parallel machines—including MIMD, SIMD, and SIMD/MIMD machines—and that can directly support the three paradigms of symbolic, object-oriented, and highly regular parallel computing. Specifically, Simple Maude supports:

- *Parallel Symbolic Computing*
  by **Term Rewriting**
- *Highly Regular Data-Parallel Computing*
  by **Graph Rewriting**

- *Concurrent Object-Oriented Computing*
  by **Object-Oriented Rewriting**

  Much more research is needed, but we have already carried out a preliminary language design for Simple Maude and have developed program transformation techniques bringing rewriting logic specifications into Simple Maude and optimizing Simple Maude programs [17]. In addition, compilation techniques for SIMD and MIMD/SIMD implementations and a prototype Simple Maude compiler for the Rewrite Rule Machine (RRM) [19] have also been developed [18].

Abstract Data Types and OBJ3. Maude contains the equational language OBJ as its functional sublanguage. Progress has also been made on theoretical and practical aspects of OBJ3. Important semantic properties of order-sorted abstract data types that make them strictly more expressive than many-sorted abstract data types have been studied in [31]; and computability properties of abstract data types clarifying their specificational power have been investigated in [35]. Several revisions and extensions of the OBJ3 user's manual [7]—that is over 100 pages long and contains an overview of OBJ3's semantics, the OBJ3 systems, its parameterized programming methodology, and many examples—have also been made; and we have continued distributing the system to universities and research laboratories worldwide.

References

[1] E. Battiston, V. Crespi, F. De Cindio, and G. Mauri. Semantic frameworks for a class of modular algebraic nets. In M. Nivat, C. Rattray, T. Russ, and G. Scollo, editors, *Proc. of the 3rd International AMAST Conference*, Workshops in Computing. Springer-Verlag, 1994.

[2] M. Bettaz and M. Maouche. How to specify nondeterminism and true concurrency with algebraic term nets. In M. Bidoit and C. Choppy, editors, *Recent Trends in Data Type Specification*, pages 164–180. Springer LNCS 655, 1993.

[3] A. Corradini and F. Gadducci. CPO models for infinite term rewriting. To appear in *Proc. AMAST'95.*

[4] A. Corradini, F. Gadducci, and U. Montanari. Relating two categorical models of term rewriting. To appear in *Proc. Rewriting Techniques and Applications, Kaiserslautern, April*, 1995.

[5] G. Denker and M. Gogolla. Translating TROLL *light* concepts to Maude. In H. Ehrig and F. Orejas, editors, *Recent Trends in Data Type Specification*, volume 785 of *LNCS*, pages 173–187. Springer-Verlag, 1994.

[6] K. Futatsugi and T. Sawada. Cafe as an extensible specification environment. To appear in *Proc. of the Kunming International CASE Symposium, Kunming, China, November*, 1994.

[7] J.A. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-88-9, Computer Science Laboratory, SRI International, August 1988.

[8] Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. Technical Report SRI-CSL-92-03, SRI International, Computer Science Laboratory, 1994. To appear in J.A. Goguen, editor, *Applications of Algebraic Specification Using OBJ*, Cambridge University Press.

[9] C. Kirchner, H. Kirchner, and M. Vittek. Designing constraint logic programming languages using computational systems. In P. van Hentryck and V. Saraswat, editors, *Selected Papers from the 1st PPCP Workshop*, 1995. MIT Press, to appear.

[10] H. Kirchner and P.-E. Moreau. Prototyping completion with constraints using computational systems. To appear in *Proc. Rewriting Techniques and Applications, Kaiserslautern, April*, 1995.

[11] C. Laneve and U. Montanari. Axiomatizing permutation equivalence in the λ-calculus. In H. Kirchner and G. Levi, editors, *Proc. Third Int. Conf. on Algebraic and Logic Programming, Volterra, Italy, September 1992*, volume 632 of *LNCS*, pages 350–363. Springer-Verlag, 1992.

[12] C. Laneve and U. Montanari. Axiomatizing permutation equivalence. *Mathematical Structures in Computer Science*, 1994. To appear.

[13] U. Lechner, C. Lengauer, and M. Wirsing. An object-oriented airport. To appear in *Proc. Tenth ADT + COMPASS Workshop on Specifica-*

tion of Abstract Data Types, Santa Margherita Ligure, Italy, May/June 1994, Springer LNCS, 1995.

[14] J. Levy. A higher order unification algorithm for bi-rewriting systems. In J. Agustí and P. García, editors, *Segundo Congreso Programación Declarativa*, pages 291–305, Blanes, Spain, September 1993. CSIC.

[15] J. Levy. *The calculus of refinements: a formal specification model based on inclusions.* PhD thesis, Universitat Politècnica de Catalunya, 1994.

[16] J. Levy and J. Agustí. Bi-rewriting, a term rewriting technique for monotonic order relations. In C. Kirchner, editor, *Proc. Fifth Int. Conf. on Rewriting Techniques and Applications, Montreal, Canada, June 1993*, volume 690 of *LNCS*, pages 17–31. Springer-Verlag, 1993.

[17] Patrick Lincoln, Narciso Martí-Oliet, and José Meseguer. Specification, transformation, and programming of concurrent systems in rewriting logic. In G.E. Blelloch, K.M. Chandy, and S. Jagannathan, editors, *Specification of Parallel Algorithms*, pages 309–339. DIMACS Series, Vol. 18, American Mathematical Society, 1994.

[18] Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Livio Ricciulli. Compiling rewriting onto SIMD and MIMD/SIMD machines. In *Proceedings of PARLE'94, 6th International Conference on Parallel Architectures and Languages Europe*, pages 37–48. Springer LNCS 817, 1994.

[19] Patrick Lincoln, José Meseguer, and Livio Ricciulli. The Rewrite Rule Machine Node Architecture and its Performance. In *Proceedings of CONPAR'94, Linz, Austria, September 1994*, pages 509–520. Springer LNCS 854, 1994.

[20] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. Technical Report SRI-CSL-93-05, SRI International, Computer Science Laboratory, August 1993.

[21] Narciso Martí-Oliet and José Meseguer. General logics and logical frameworks. In D. Gabbay, editor, *What is a Logical System?*, pages 355–392. Oxford University Press, 1994.

[22] Narciso Martí-Oliet and José Meseguer. Action and change in rewriting logic. In R. Pareschi and B. Fronhoefer, editors, *Theoretical Approaches*

to Dynamic Worlds in Computer Science and Artificial Intelligence. 1995. To appear.

[23] José Meseguer. Conditional rewriting logic: deduction, models and concurrency. In S. Kaplan and M. Okada (eds.) *Proc. CTRS'90*, Montreal, Canada, 1990, Springer LNCS 516, pp. 64-91, 1991.

[24] José Meseguer. A logical theory of concurrent objects. In *ECOOP-OOPSLA'90 Conference on Object-Oriented Programming, Ottawa, Canada, October 1990*, pages 101–115. ACM, 1990.

[25] José Meseguer. Rewriting as a unified model of concurrency. In *Proceedings of the Concur'90 Conference, Amsterdam, August 1990*, pages 384–400. Springer LNCS 458, 1990.

[26] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[27] José Meseguer. Multiparadigm logic programming. In H. Kirchner and G. Levi, editors, *Proc. 3rd Intl. Conf. on Algebraic and Logic Programming*, pages 158–200. Springer LNCS 632, 1992.

[28] José Meseguer. A logical theory of concurrent objects and its realization in the Maude language. In Gul Agha, Peter Wegner, and Akinori Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*, pages 314–390. MIT Press, 1993.

[29] José Meseguer. Solving the inheritance anomaly in concurrent object-oriented programming. In Oscar M. Nierstrasz, editor, *Proc. ECOOP'93*, pages 220–246. Springer LNCS 707, 1993.

[30] José Meseguer, Kokichi Futatsugi, and Timothy Winkler. Using rewriting logic to specify, program, integrate, and reuse open concurrent systems of cooperating agents. In *Proceedings of the 1992 International Symposium on New Models for Software Architecture, Tokyo, Japan, November 1992*, pages 61–106. Research Institute of Software Engineering, 1992.

[31] José Meseguer and Joseph Goguen. Order-sorted algebra solves the constructor-selector, multiple representation and coercion problems. *Information and Computation*, 103(1):114–158, 1993.

[32] José Meseguer and Narciso Martí-Oliet. From abstract data types to logical frameworks. To appear in Proceedings of the 10th Workshop on Abstract Data Types, Santa Margherita, Italy, June 1994, Springer LNCS, 1994.

[33] José Meseguer and Xiaolei Qian. A logical semantics for object-oriented databases. In *Proc. International SIGMOD Conference on Management of Data*, pages 89–98. ACM, 1993.

[34] José Meseguer and Timothy Winkler. Parallel programming in Maude. In J.-P. Banâtre and D. Le Mètayer, editors, *Research Directions in High-level Parallel Programming Languages*, pages 253–293. Springer LNCS 574, 1992. Also Technical Report SRI-CSL-91-08, SRI International, Computer Science Laboratory, November 1991.

[35] L. Moss, J. Meseguer, and J.A. Goguen. Final algebras, cosemicomputable algebras, and degrees of unsolvability. *Theoretical Computer Science*, 100:267–302, 1992.

[36] H. Reichel. An approach to object semantics based on terminal coalgebras. To appear in *Mathematical Structures in Computer Science*, 1995. Presented at *Dagstuhl Seminar on Specification and Semantics*, Schloss Dagstuhl, Germany, May 1993.

[37] C. Talcott. Semantics of component based distributed, open, heterogeneous systems. Manuscript, Stanford University, February 1995.

[38] P. Viry. Rewriting: An effective model of concurrency. In C. Halatsis et al., editors, *PARLE'94, Proc. Sixth Int. Conf. on Parallel Architectures and Languages Europe, Athens, Greece, July 1994*, volume 817 of *LNCS*, pages 648–660. Springer-Verlag, 1994.

[39] M. Vittek. *ELAN: Un cadre logique pour le prototypage de langages de programmation avec contraintes.* PhD thesis, Université Henry Poincaré — Nancy I, 1994.

OFFICE OF THE UNDER SECRETARY OF DEFENSE (ACQUISITION)
DEFENSE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22304-6145

IN REPLY REFER TO DTIC-OCC

SUBJECT: Distribution Statements on Technical Documents

TO: OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

1995 1031 022

1. Reference: DoD Directive 5230.24, Distribution Statements on Technical Documents, 18 Mar 87.

2. The Defense Technical Information Center received the enclosed report (referenced below) which is not marked in accordance with the above reference.

FINAL REPORT
N00014-90-C-0086
TITLE: OBJECT ORIENTED DESIGN
AND SPECIFICATION

3. We request the appropriate distribution statement be assigned and the report returned to DTIC within 5 working days.

4. Approved distribution statements are listed on the reverse of this letter. If you have any questions regarding these statements, call DTIC's Cataloging Branch, (703) 274-6837.

FOR THE ADMINISTRATOR:

1 Encl

GOPALAKRISHNAN NAIR
Chief, Cataloging Branch

FL-171
Jul 93

DISTRIBUTION STATEMENT A:

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

DISTRIBUTION STATEMENT B:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT C:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTORS; (Indicate Reason and Date Below). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT D:

DISTRIBUTION AUTHORIZED TO DOD AND U.S. DOD CONTRACTORS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT E:

DISTRIBUTION AUTHORIZED TO DOD COMPONENTS ONLY; (Indicate Reason and Date Below). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office Below).

DISTRIBUTION STATEMENT F:

FURTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date Below) or HIGHER DOD AUTHORITY.

DISTRIBUTION STATEMENT X:

DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DOD DIRECTIVE 5230.25, WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (Indicate date of determination). CONTROLLING DOD OFFICE IS (Indicate Controlling DoD Office).

The cited documents has been reviewed by competent authority and the following distribution statement is hereby authorized.

A

(Statement)

OFFICE OF NAVAL RESEARCH
CORPORATE PROGRAMS DIVISION
ONR 353
800 NORTH QUINCY STREET
ARLINGTON, VA 22217-5660

(Controlling DoD Office Name)

(Reason)

(Controlling DoD Office Address, City, State, Zip)

DEBRA T. HUGHES
DEPUTY DIRECTOR
CORPORATE PROGRAMS OFFICE

19 SEP 1995

(Signature & Typed Name)

(Assigning Office)

(Date Statement Assigned)