AD-A136 282 OBJ-1 A STUDY IN EXECUTABLE ALGEBRAIC FORMAL SPECIFICATION(U) SRI INTERNATIONAL MENLO PARK CA 30 SEP 83 N00014-83-C-0333 1/1

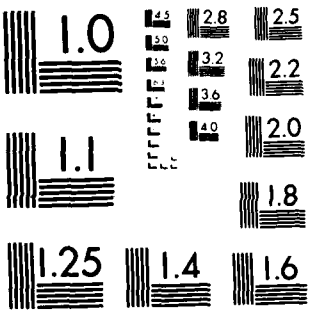
UNCLASSIFIED F/G 9/2 NL

END

DATE FILMED

4-84

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

12

October 4, 1983

Scientific Officer
Associate Director for Mathematical
and Physical Sciences, Research Programs
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217
Attention: Dr. Robert B. Grafton

Reference: Contract No. N00014-83-C-0333

Subject: Progress Report for Fiscal Year 1983

Dear Dr. Grafton:

Enclosed is one copy of our technical progress report on our work accomplished during fiscal year 1983.

Sincerely,

J. A. Goguen/mvo

Joseph Goguen
Project Leader
Computer Science Laboratory

JAG:mvo

cc: M. Denicoff, Leader, Information Sciences Division (2)
Administrative Contracting Officer, Code S0507A (1)
Director, Naval Research Laboratory, Code 2627 (6)
Defense Technical Information Center (12)
Office of Naval Research, Pasadena (1)

A136282

DTIC FILE COPY

DTIC
ELECTE
DEC 23 1983
S D
D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

SRI International
333 Ravenswood Ave • Menlo Park, CA 94025 • 415 326-6200 • TWX 910-373-1246 • TELEX 334463 • Facsimile 415 326-5512

83 10 12 266

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By Per Ltr. on file
Distribution/
Availability Codes
Dist | Avail and/or Special
A/1

To: Division Leader, Information Sciences Division, ONR

From: Joseph A. Goguen, SRI International

Re: End of FY Letter for contract N00014-82-C-0333,
OBJ-1, a Study in Executable Algebraic Formal Specification

Date: 30 September 1983

1 Progress

Our most significant technical results are discussed under three headings: Implementation; Applications and Examples; and Concepts and Foundations. We believe that we have demonstrated, through an ample collection of published examples, the feasability and applicability of an ultra high level programming language in which problems are described by writing equations, and are then solved by regarding those equations as rules for reducing expressions to answers. In addition, we have explored several issues that are significant to programming in general, including: (1) rapid prototyping; (2) reusability in programming; (3) types and modularity; (4) error handling and recovery; (5) database views and representations; (6) pattern matching; and (7) backtracking. A key to (2), (3) and (4) is the graceful integration of code and specification using a new notion called views. Moreover, we have significantly improved our experimental implementation of OBJ.

1.1 Implementation

Our UCI-Rutgers LISP implementation of OBJ has been extended by Prof. David Plaisted to OBJ-1, running on DEC-20s under TOPS-20 and on DEC-10s under TENEX. This implementation has a much better user interface and many useful new features, including the following:

1. **Interactive-incremental program developement.** When an error is detected by OBJ-1 while reading in a program file, comments are inserted into the file, and the user may subsequently edit that file, starting from the object where the errors were detected onward. When the editor (which is EMACS) is exited, OBJ-1 tries again to execute these objects. The objects that were previously accepted are left unchanged; however, there are commands to undo and edit any desired objects, or even whole files. Together with a very informative parser, this greatly increases programmer productivity by reducing the time required to find syntax errors, and by eliminating the need to re-process objects upon which an object containing a corrected error depended.

2. **Associative pattern matching.** Binary operators can be declared ASSOCIATIVE. The parser does not then require full parenthesisation, and we get all the effects of an associativity equation. Equations involving an associative operator permit a very general and simple form of pattern matching and can be used as "demons" for AI applications. An associative operation can, in addition, be declared COMMUTATIVE. This is equivalent to adding a commutative equation. An associative and commutative operator can also be declared IDEMPOTENT; this provides an efficient way of computing set-theoretic expressions and opens up interesting theorem-proving applications as explained below.

3. **Backtracking** is incorporated in a simple way that makes use of the error specification capabilities of the language, specifically the NONPRIMITIVE attribute declared for an operator. Again, this makes OBJ-1 a good candidate for AI applications (more on this below).

4. **Hash addressing for selected terms** is now possible, allowing big gains in efficiency by avoiding unnecessary recomputations. This is accomplished by the SAVERUNS attribute, declared for a given operator.

83 10 12

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

5. **Built-in objects** now include BOOLeans, NATurals, INTegers, and IDentifiers. In addition, parameterized definition of tuples of any desired length is built-in.

6. **Help and photo** facilities are now provided. Photo allows a record of the interactive session to be written on a separate file.

1.2 Applications and Examples

Applications and examples from rapid prototyping, parameterized programming, databases, and artificial intelligence are summarized according to the main area intended, since these areas overlap a good deal. The numbers in square brackets refer to papers listed in Section 3.

1. **Rapid prototyping**. Because OBJ-1 is executable, very high level, and has powerful parameterization and modularity, it is ideal for quickly bringing up a working prototype of a system, from which a final implementation can them be developed by stepwise refinement; see [5].

2. **Parameterized programming**. The parameterization mechanisms in OBJ make possible an entirely new style of programming which promises dramatic gains in reusability of code and programmer productivity. One can develop "universally applicable" algorithms of a given kind (e.g., sorting) that can be easily instantiated for a given application; see [6] and [9]. Explicitly providing **views** showing how a given module satisfies a given specification greatly increases the power and reliability of this style of programming; see [4].

3. **Databases**. Change of representation of data among different databases is a major practical problem. So is providing different views of the same database to different of users. OBJ can be used to give clear and elegant solutions to these two important problems; see [2].

4. **Artificial intelligence**. Conditional rewrite rules are essentially the same thing as the rules used in expert (rule based) systems. OBJ provides powerful ways of structuring such systems of rules, as well as a precise mathematical theory of what they are supposed to do. Another way to describe OBJ is that it is a language for constructing pattern directed inference systems. OBJ's pattern matching is further enhanced by built-in options for declaring operations associative, commutative, and/or idempotent; combining this with conditional rewrite rules yields quite powerful pattern directed "demons." OBJ can also be used as a decision procedure for ground terms of an equational theory, provided the equations satisfy some mild conditions when viewed as rewrite rules. Built-in associativity, commutativity, idempotence permits a simple implementation of a decision procedure for predicate calculus. Backtracking is also available via error conditions, thus allowing concise formulation of many typical AI search problems.

1.3 Concepts and Foundations

Work on both mathematical foundations of the existing OBJ-1, and developement of new concepts and ideas for future versions has been very active.

1. **Foundations of equational and error deduction** are given in [10], [11], and [13].

2. **Foundations for software module implementation** from algebraic specifications are given in [1].

3. **Foundations for specification languages** and their translations are given in [8].

4. **New ideas on parameterization and subsorts** with applications to databases and polymorphism are presented in [2], [4] and [14].

2 Presentations

Presentations were made at the following conferences: Rapid Prototyping (Columbia MD) [5]; Software Factory Experiences (Capri, Italy) [6]; 1982 ICALP (Aarhus, Denmark) [1]; Application of Algebra to Language Definition and Compilation (Fontainebleau, France) [10]; Semantics of Programming Languages (Bad Honnef, Germany) [8], [10]; Database Semantics and Interfaces (Philadelphia PA) [2]; Logics of Programming (Pittsburgh PA) [8]; Type Theory (Pittsburgh PA) [14]; and Reusability in Programming [4] (Newport RI).

In addition, lectures were given at the following places: Stanford University; Syracuse University; University of British Columbia; Simon Fraser University; Edinburgh University; Manchester University; Imperial College; Cambridge University; University of Barcelona; University of Milan; CNR, Cybernetics Research Inst, Naples Italy; University of Pennsylvania; SRI; and Xerox Palo Alto Research Center.

Finally, a university course was taught in San Sebastian, Spain, by Meseguer.

3 Publications

1. J. Goguen and J. Meseguer, "Universal Realization, Persistent Interconnection and Implementation of Abstract Modules," in *Proceedings, 9th International Colloquium on Automata, Languages and Programming* (Aarhus, Denmark) Springer-Verlag, Lecture Notes in Computer Science, 1982.

2. J. Goguen, "Merged Views, Closed Worlds and Ordered Sorts: Some Novel Database Features in OBJ," *Proceedings, of Workshop on Database Semantics and Interfaces*, University of Pennsylvania, 1982; to appear in *SIGMOD Notices*, ACM, 1983.

3. J. Meseguer, "Order Completion Monads," *Algebra Universalis*, vol. 16, pp 63-82, 1983.

4. J. Goguen, "Parameterized Programming," in *Proceedings, Workshop on Reusability in Programming*, (Newport RI), ITT, 1983.

5. J. Goguen and J. Meseguer, "Rapid Prototyping in the OBJ Executable Specification Language," in *Proceedings, Rapid Prototyping Workshop* (Columbia, Maryland) 1982. Also in *Software Engineering Notes*, ACM Special Interest Group on Software engineering, volume 7, number 5, 1983, pages 75-84.

6. J. Goguen, J. Meseguer and D. Plaisted, "Programming with Parameterized Abstract Objects in OBJ," in *Theory and Practice of Software Technology*, edited by D. Ferrari, M. Bolognani and J. Goguen, North-Holland, pages 163-193, 1983.

7. J. Goguen, "Future Directions for Software Engineering," in *Theory and Practice of Software Technology*, edited by D. Ferrari, M Bolognani and J. Goguen, North-Holland, pages 243-244, 1983.

8. J. Goguen and R. Burstall, "Introducing Institutions," *Proceedings, Logics of Programming Workshop*, Carnegie Mellon University, June 1983.

9. F. Meseguer, "Programacion Parametrizada en el Lenguaje OBJ-1," trabajo fin de carrera, Facultad de Informatica, Madrid, 1983.

10. J. Meseguer and J. Goguen, "Initiality, Induction and Computability," to appear in *Application of*

Algebra to Language Definition and Compilation, edited by M. Nivat and J. Reynolds, Prentice-Hall, 1984.

11. J. Goguen and J. Meseguer, "Completeness of Many-sorted Equational Logic," to appear in *Houston Journal of Mathematics*.

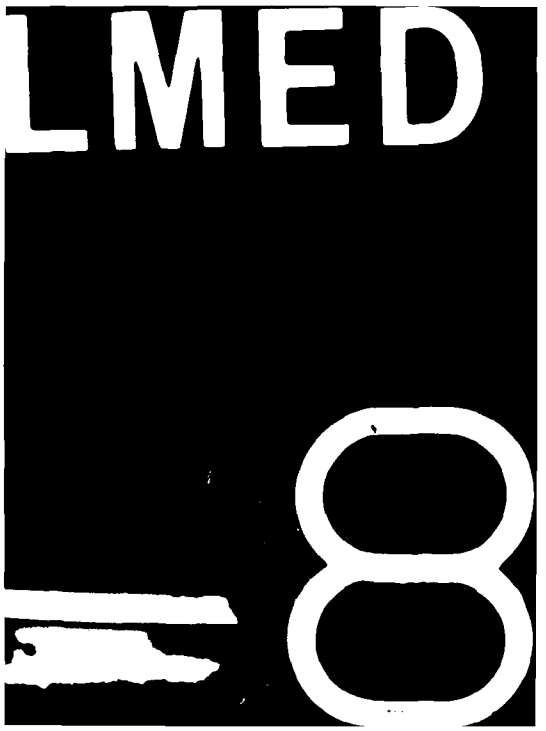
12. J. Goguen and J. Meseguer, "Correctness of Recursive Parallel Non-deterministic Flow Programs," to appear in *Journal of Computer and System Sciences* (special issue dedicated to Cal Elgot).

13. D. Plaisted, "An Initial Algebra Semantics for Error Presentations," submitted to *SIAM Journal of Computing*.

14. J. Goguen and J. Meseguer, "Algebraic Polymorphism," in preparation, SRI International, 1983.

4 Participants

The following personnel participated in the project at SRI International: J. A. Goguen, J. Meseguer, F. Meseguer, D. Plaisted, and D. Hare. In addition, we have had significant interactions with personnel at other insitutions who are implementing their own versions of OBJ. These include: D. Coleman and R. Gallimore of the University of Manchester; J. Weiner of the University of New Hampshire; and G. Mauri of the University of Milan.


LMED
8