

2

Kestrel Institute

5 October 1990

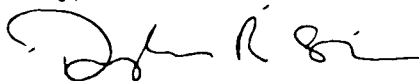
AD-A229 115

Dr. Andre van Tilborg, Code 1133  
Computer Science Division  
Office of Naval Research  
800 N. Quincy St.  
Arlington, VA 22217

Dear Andre,

Find enclosed two copies of the FY90 end-of-year letter for contract N00014-90-J-1733 and some recent papers.

Sincerely,



Douglas R. Smith

enclosures

OTIC  
LECTE  
OCT 15 1990  
D  
Co

P.I. Name: Douglas R. Smith  
Insitution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

### 1. Productivity Measures

Refereed papers submitted but not published: 4

Refereed papers published: 5

Unrefereed reports and articles: 1

Books or parts thereof not published: 4

Books or parts thereof published: 0

Patents filed but not granted: 0

Patents granted: 0

Invited presentations: 17

Contributed presentations: 3

Honors received: 2

Prizes or awards received: 0

Promotions obtained: 0

Graduate students supported: 0

Post-docs supported: 0

Minorities supported: 0



Accession For	
NPAS CRASH	<input checked="" type="checkbox"/>
DDIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Distribution/	

A-1

STATEMENT "A" Per Dr. A. van 1111borg  
ONR/Code 1133  
TELECON 10/11/90 VG

P.I. Name: Douglas R. Smith  
Institution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

## 2. Summary of Technical Results

Algorithms and data structures are among the primary constituents of computer software and thus are among basic objects of study in Computer Science. This project is concerned with the structure and automated design of algorithms and data structures. Our scientific hypothesis is that there exist general algorithm, data structure, and design concepts that underlie and explain most of the detailed structure of conventional software systems. By abstracting and formalizing these concepts and showing how to mechanize their application, we can prepare the way for the coming generation of automated software design environments.

Our approach involves identifying classes of algorithms that solve a broad range of useful problems. In particular we have emphasized formalizing abstract algorithms that make minimal assumptions about the structure of a problem. Once a class of algorithms has been identified we represent its essence as a theory, called an *algorithm theory*, [4]. There are a number of advantages to this axiomatic approach. First, it abstracts away concerns about programming language and style (e.g., functional vs. logical vs. imperative, recursive vs. iterative) and control strategy (e.g., top-down vs. bottom-up, sequential vs. parallel execution). These can be factored in as later decisions in the design process. Second, once and for all we can derive abstract programs (schemes) as theorems in the abstract theory and then apply them to concrete problems. Third, we can develop formal, automatable, and highly-reusable *design tactics* on the basis of the abstract theory. They provide generic methods for designing concrete algorithms from specifications for particular problems. Fourth, there is a strong analogy to abstract data types. For example, the concept of global search underlies a number of well-known data structures such as binary search trees, quad-trees, and B-trees, suggesting that algorithm design tactics could be extended to data structure design tactics. Finally we hypothesize that algorithmic theories can be combined to allow the inference of hybrid algorithms.

Project results during the last year are listed below.

1. *Problem Reduction Generators* - Problem reduction is a pervasive technique for solving problems by reducing a problem instance to a structure of subproblem instances. Solutions to the subproblem instances are composed to form a solution to the initial instance. Problem reduction generators are used to enumerate all solutions to a given problem. Problems are given by means of a formal specification (we use a specification language that extends first-order predicate calculus notation with set-theoretic datatypes).

We produced an algorithm theory and design tactic for the class of problem reduction generators [3]. This algorithm theory, called *complete problem reduction theory*, provides the logical basis for dynamic programming, branch-and-bound, game-tree search, and other well-known algorithm paradigms.

The design tactic extends the structure of a specified problem with the structure of complete problem reduction theory. If we decide to compute the characteristic recurrence equation of problem reduction theory from the bottom up, then we get the usual dynamic programming algorithms. Top-down control leads to branch-and-bound algorithms such as AO\* and game-tree algorithms such as alpha-beta and SSS\*.

The design tactic constructs a complete problem reduction theory for the specified problem through a combination of selecting of standard information from a library, deductive propagation of the consequences of choices, and verification of axioms. A major result of our project has been the development of deductive techniques for using axioms to help construct complex objects rather than simply to use them as verification conditions. For example, in complete problem reduction theory we use an axiom ("strong soundness") to deduce a specification for the reduction step of the algorithm given a choice of a standard composition step (which is usually a standard constructor algebra for the output domain).

The design tactic has been applied to about a dozen problems in order to explore its generality and the feasibility of its steps. The paper [3] presents a detailed treatment of the problem of enumerating optimal binary search trees. We plan to implement the tactic in order to more fully explore its feasibility and range of application.

This algorithm theory has been presented in a more general style than previously: the theory is indexed by a signature, as in algebraic theories. In previously published algorithm theories and design tactics we have had to fix the signature thereby limiting the applicability of the theory unnecessarily.

2. *KIDS and Challenge Problems* - KIDS (Kestrel Interactive Development System) is the testbed for our research on semiautomated program design [1, 2]. The system has components for performing algorithm design, deductive inference, program simplification, partial evaluation, finite differencing optimizations, data type refinement, compilation, and other development operations. To use KIDS for a new problem, the user first builds up a (first-order) domain theory that formalizes the concepts of the problem and provides laws for reasoning about the concepts. The domain theory provides the vocabulary for expressing a formal specification of the problem. Next the user applies the design, optimization, and refinement tools as needed in order to obtain an efficient and correct source-level program that implements the specification.

This past year we have used KIDS to respond to several challenge problems presented to us by non-Kestrel personnel. By accepting these challenges our aims were to deepen our understanding of automated algorithm design through applying KIDS to concrete problems and to demonstrate the effectiveness of KIDS by working problems that were not selected by personnel with an intimate knowledge of the system.

We briefly discuss two challenge problems: enumerating cyclic projective planes of order  $n$  and enumerating Costas arrays. The first problem was quite difficult, but resulted in

an algorithm that was significantly better than most programmers (including ourselves) could have designed given our simple understanding of the problem. Unfortunately we found out later that there is a considerable and deep literature on the problem under the rubric of combinatorial design theory. Exploiting this knowledge would have resulted in much better algorithm.

The Costas array derivation was more successful [2]. This is a problem arising in the design of sonar and radar signals with optimal ambiguity functions. The domain theory was built up over a period of about a week and enabled the derivation of an efficient backtrack algorithm. After we hand-refined the abstract data structures of the resulting program and manually translated it into C, we were able to enumerate all order 17 Costas arrays. This is as high as has been published in the literature by the various groups of mathematicians exploring this problem. This result is evidence that the machine-generated algorithm is comparable to the programs created by insightful and experienced human programmers.

There were several important lessons from these derivations and others like them. First, much of the hard work in performing a new derivation lies in building up the domain theory. As discussed below we have built up a library of some 25 theories, including basic theories such as finite set theory and finite map theory, as well as more specialized theories for scheduling and Costas arrays. The more basic theories are highly reusable so over time we can expect that theory building will involve more selection and adaptation. Our experience points to theory-building as an irreducible and time-consuming activity in future automated design environments.

Another lesson concerns the nature of the domain theories. What constitutes a well-formulated theory? Our experience has been that a well-formulated domain theory has simple laws for reasoning about the basic concepts. A related observation is that most of the laws needed to support design, optimization, and refinement in KIDS are distributive laws. Together these observations suggest that when building a new domain theory, we should seek conceptualizations that admit simple distributive laws. Our initial formulation of Costas array theory had quite complex laws and we eventually discarded it. A second attempt was successful and the basic concepts had very elegant distributive laws and the ensuing design process was straightforward.

3. *Theory Development in KIDS* - In light of the fundamental importance of domain theories in KIDS (and for formal methods of software design in general), we began to explore tools for supporting the development of domain theories. We prototyped a simple grammar for theories comprising imports, type definitions, concept definitions (functions), laws, and inference rules. We then built an interactive graphical interface and a collection of operations that could be used to help users build theories. Loading a theory recursively loads all imports and installs the concept definitions, laws, and inference rules in the KIDS active knowledge base. There is also a tool for automatically deriving distributive laws from a concept definition. The resulting laws can be transformed into nonlogical inference rules. Another operation allows the abstraction of new concepts from the body of any expression. For example, if a derived distributive law contains a complex subexpression  $S$ , then  $S$  can be abstracted out, named, and added to the theory. In this way the user may enter some definitions via a text editor, then use the system to derive new laws, rules, and concepts.

Of the roughly 700 rules in the KIDS system, we have encapsulated about 30% in about 25 domain theories. The remaining rules form an unstructured collection. A more systematic and complete attempt at formalizing the domain knowledge of KIDS awaits the results of the next item.

4. *Theories, Theory Interpretations, and Operations on Theories* - Theories and operations on theories have emerged at Kestrel as a pervasive foundation in our software design automation studies. Our algorithm design work is expressed in terms of abstract algorithm theories and interpretations between abstract and concrete algorithm theories. Related concepts of problem theories and program theories come into play. The use of (algebraic) theories to formalize abstract data types is well-known. We have been formalizing a grammar and compilation mechanism for theory interpretations as a way to formally express and use datatype refinements. For example, the implementation of sets as bit vectors is essentially an interpretation between set theory and bit vector theory. Generally, interpretations between theories provide the basis for refinement of specifications. The use of theories to express domain theories and various operations to help build them was discussed above. Thus theories and their operations play a fundamental role in expressing program design knowledge. Theories can also be used as a module mechanism in a specification/programming language, providing modules with a formally defined interface (not only the imported and exported types and operations, but the axioms that constrain their meaning too). For these reasons and others we have been exploring theories and their operations, extending the results of the OBJ3, LARCH, and Extended ML projects. Topics under active investigation include, parameterizing theories and theory interpretations by theories (a generalized polymorphism), partial constructors and theory invariants, expressing and compiling theory interpretations, object theories that include state, and temporal theories for reasoning about time.
5. *Problem Reformulation* - Our work on problem reformulation has been extended along several dimensions. Previous work on abstraction of domain and problem specifications has been extended to domains with hidden states, thereby facilitating scaling up to module and system abstraction. The foundation for abstraction through symmetry has progressed; we anticipate application to continuous domains during the coming year. A variant of Karmarkar's algorithm has been previously developed by hand using abstraction through symmetry, we anticipate that further work along these lines will provide the formal foundation for automating this development.
6. *Local Search* - Using permutation group theory, we have further developed our research on local search algorithms by formalizing a class of combinatorial local search algorithms based on permuting components of data structures representing feasible solutions. Previously, a design tactic based on these ideas had been developed and demonstrated on the synthesis of the Simplex algorithm using KIDS. We are currently extending our research on local search through a design tactic for primal/dual algorithms.

## References

- [1] SMITH, D. R. KIDS - a semi-automatic program development system. *IEEE Transactions on Software Engineering Special Issue on Formal Methods in Software Engineering* 16, 9 (September 1990), 1024-1043.
- [2] SMITH, D. R. KIDS: a knowledge-based software development system. In *Automating Software Design*, M. Lowry and R. McCartney, Eds., Live Oak Press, Menlo Park, 1991. to appear.
- [3] SMITH, D. R. *Structure and Design of Problem Reduction Generators*. Tech. Rep., Kestrel Institute, 1990. submitted to IFIP TC2 Working Conference on Constructing Programs from Specifications.
- [4] SMITH, D. R., AND LOWRY, M. R. Algorithm theories and design tactics. In *Proceedings of the International Conference on Mathematics of Program Construction, LNCS 375*, L. van de Snepscheut, Ed., Springer-Verlag, Berlin, 1989, pp. 379-398. (extended version to appear in *Science of Computer Programming*).

P.I. Name: Douglas R. Smith  
Institution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

### **3. Lists of Publications, Presentations, and Reports**

#### **3.1. Publications**

Lowry, M.R., Category Theory and Homomorphic Abstraction, Proceedings of Workshop on Change of Representation and Problem Reformulation, Menlo Park, CA, March 1990 (also appeared in Proceedings of Workshop on Algebraic Approaches to Problem Solving and Perception Tarrytown, New York, June 1990).

Lowry, M.R., Abstracting Domains With Hidden State, Proceedings of AAAI-90 Workshop on Automatic Generation of Approximations and Abstraction, Boston, MA, August 1990.

Lowry, M.R., Structure and Design of Local Search Algorithms, to appear in *Automating Software Design*, Eds. M. Lowry and R. McCartney AAAI Press, Menlo Park, CA, 1991.

Lowry, M.R., Software Engineering in the 21st Century, to appear in *Automating Software Design*, Eds. M. Lowry and R. McCartney AAAI Press, Menlo Park, CA, 1991.

Lowry, M. and McCartney, R., Editors, *Automating Software Design*, AAAI Press, Menlo Park, CA, 1991.

Smith, D.R., Automating the Development of Software, Proceedings of the NATO Symposium on Military Information Systems Engineering, Malvern, England, May 1990.

Smith, D.R., KIDS: A Semi-Automated Program Development System, IEEE Transactions on Software Engineering 16(9), Special Issue on Formal Methods, September 1990, 1024-1043.

Smith, D.R., Automating the Development of Software, Proceedings of the Fifth Annual Knowledge-Based Software Assistant Conference, Liverpool, New York, September, 1990, 13-27.

Smith, D.R. and Lowry, M.R., Algorithm Theories and Design Tactics, invited paper to appear in a special issue on the Mathematics of Program Construction, *Science of Computer Programming*, 1990.

Smith, D.R., KIDS: A Semi-Automated Program Development System, abstract to appear in Proceedings of SIGSOFT 90, Irvine, California, December, 1990.



Smith, D.R., KIDS: A Knowledge-Based Software Development System, to appear in *Automating Software Design*, Eds. M. Lowry and R. McCartney, Live Oak Press, Menlo Park, 1991.

Smith, D.R., Structure and Design of Problem Reduction Generators, submitted to IFIP WG2.1 TC2 Conference on Specification and Transformation of Programs, September 1990.

Smith, D.R., Structure and Design of Global Search Algorithms, accepted for publication in *Acta Informatica*.

### 3.2. Presentations

*Michael R. Lowry:*

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", Price Waterhouse Technology Center, 16 November 1989.

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", University of Ottawa Computer Science Colloquium, 21 November 1989.

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", University of Toronto Computer Science Colloquium, 23 November 1989.

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", AT&T Bell Laboratories, 29 November 1989.

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", Rutgers University Computer Science Colloquium, 30 November 1989.

Presented a talk entitled "A Comparison of Approaches to Problem Reformulation", Rutgers University Machine Learning Seminar, 1 December 1990.

Presented an informal talk entitled "Kestrel's Approach to Software Refinement", British Computer Society Third Refinement Workshop, 11 January 1990.

Presented a talk entitled "Category Theory and Isomorphic Reformulation", Workshop on Change of Representation and Problem Reformulation, Menlo Park, 24 March 1990.

Presented a talk entitled "Problem Reformulation through Abstraction, then Implementation", Workshop on Change of Representation and Problem Reformulation Menlo Park, 24 March 1990.

Presented a talk entitled "Category Theory and Homomorphic Reformulation", Workshop on Algebraic Approaches to Problem Solving and Perception, Tarrytown, New York, 27 June 1990.

*Douglas R. Smith*

Presented a talk entitled "Automating the Design of Algorithms", Computer Science Seminar, UCLA, Los Angeles, 14 November 1989.

Presented a talk entitled "Automating the Design of Algorithms", Information Technology Division, Naval Research Laboratories, Washington, DC, 20 December 1989.

Presented a talk entitled "REFINE" and a Refine demo, Presented a talk entitled "Automated Algorithm Design" and KIDS demo, Naval Postgraduate School, Monterey, California, February 1990.

Presented a talk entitled "KIDS: Knowledge-Based Software Development", San Francisco State University, San Francisco, California, 19 April 1990.

Presented a talk entitled "Operations on Theories in KIDS", IFIP WG2.1 meeting, Burton Manor, Burton, England, May 1990.

Presented a talk entitled "Automating the Development of Software", Symposium on Military Information Systems Engineering, Royal Signals and Radar Establishment, Malvern, England, 8-10 May 1990.

Presented a talk and demo entitled "KIDS: Knowledge-Based Program Development", Stanford University, Palo Alto, CA, 24 May 1990.

Presented a talk entitled "Automating the Design of Algorithms", ONR Contractor's Workshop, Moscow, Idaho, 20-21 June 1990.

Presented a paper entitled "Automating the Development of Software", gave demonstrations of KIDS, and participated in a panel on KBSA/CASE tools, Fifth Annual Knowledge-Based Software Assistant Conference, Liverpool, New York, 25 September 1990.

### **3.3. Technical Reports**

Smith, D.R., KIDS: A Semi-Automatic Program Development System, Technical Report KES.U.90.1, Kestrel Institute, Palo Alto, CA, April 1990, 48 pages.

P.I. Name: Douglas R. Smith  
Institution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

#### **4. Description of Research Transitions and DoD Interactions**

Perhaps the main "transition" of the ONR-sponsored work has been through our experimental development system, KIDS (described on the next page). We have received many requests for the system from researchers in software automation. Copies of KIDS are now installed at Rutgers Univ. (Mostow), Stanford Robotics Lab (Lowry/Binford), Information Sciences Institute, USC (Balzer, Feather) and the Catholic University of Louvain, Belgium (Sintzoff). Pending requests for copies of KIDS have come from over 40 sites in North America and Europe.

Several briefings to DOD personnel were given during the year by Dr. Smith:

Presented a talk entitled "Automating the Design of Algorithms", Information Technology Division, Naval Research Laboratories, Washington, DC, 20 December 1989.

Presented a talk entitled "REFINE" and a Refine demo; also presented a talk entitled "Automated Algorithm Design" and KIDS demo, Naval Postgraduate School, Monterey, California, February 1990.

Presented a talk entitled "Automating the Development of Software", Symposium on Military Information Systems Engineering, Royal Signals and Radar Establishment, Malvern, England, 8-10 May 1990.

Presented a paper entitled "Automating the Development of Software", gave demonstrations of KIDS, and participated in a panel on KBSA/CASE tools, Fifth Annual Knowledge-Based Software Assistant Conference, Rome Air Development Center, New York, 25 September 1990.

P.I. Name: Douglas R. Smith  
Institution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

## **5. Description of Software and Hardware Prototypes**

The Kestrel Interactive Development System (KIDS) provides an open architecture for experimenting with the semi-automated development of formal specifications into correct and efficient programs. The system has components for performing algorithm design, deductive inference, program simplification, partial evaluation, finite differencing optimizations, data type refinement and other development operations. Although their application is interactive, all of the KIDS operations are automatic except the algorithm design tactics which require some interaction at present. Over fifty programs have been derived using the system and we believe that KIDS could be developed to the point that it becomes economical to use for routine programming. We are not currently working on commercializing this system - it is regarded purely as an experimental testbed.

P.I. Name: Douglas R. Smith  
Institution: Kestrel Institute  
Telephone: (415) 493-6871  
E-mail: smith@kestrel.edu  
Contract Title: Theory of Algorithm Structure and Design  
Contract Number: N00014-90-J-1733  
Reporting Period: 1 Oct 89 - 30 Sep 90

## 6. Miscellaneous

Dr. Lowry was a program committee member for workshop "Change of Representation and Problem Reformulation", also served at same workshop as session chair for: "Applications of Category Theory to Representation Change and Problem Reformulation" Menlo Park, March 1990.

Dr. Lowry served as co-editor for *Automating Software Design* to be published by AAAI press in early 1991.

Dr. Lowry reviewed an NSF proposal.

Dr. Smith was appointed Lecturer in Computer Science, Autumn 1989, Computer Science Department, Stanford University. He taught CS-409, Knowledge-Based Software Environments, during the autumn quarter of 1989.

Dr. Smith is Co-Chairman and program committee member, IFIP TC2 Working Conference on Constructing Programs from Specifications, IFIP Working Group 2.1, Asilomar Conference Center, Pacific Grove, California, May 1991.

Dr. Smith is on the program committee for the Workshop on Logical Theory for Program Construction, to be held, 25 February - 1 March 1991, IBFI, Schloss Dagstuhl, Saarbrücken, Germany.

Dr. Smith reviewed 11 papers for various journals and books.