AD-A272 724

2

Applications of Sheaf Theory in Algorithm Design

Yellamraju V. Srinivas
Kestrel Institute, 3260 Hillview Avenue,
Palo Alto, CA 94304 (srinivas@kestrel.edu)

Final Report
ONR Contract N00014-92-C-0124
1 Oct 1992 to 30 Sep 93

DTIC
S ELECTE
NOV 17 1993
A

This document has been approved for public release and sale; its distribution is unlimited.

93-28149

93 11 16 098

~~93-26304~~

~~93 10 20036~~

Best
Available
Copy

PI Name:	Yellamraju V. Srinivas
PI Institution:	Kestrel Institute 3260 Hillview Avenue, Palo Alto, CA 94304
PI Phone Number:	(415) 493-6871
PI E-mail Address:	srinivas@kestrel.edu
Grant or Contract Title:	Applications of Sheaf Theory in Algorithm Design
Grant or Contract Number:	N00014-92-C-0124
Reporting Period:	1 Oct 1992 to 30 Sep 93

1 Productivity Measures

Refereed papers published: 3
Unrefereed reports and articles: 1
Contributed presentations: 1

DTIC QUALITY INSPECTED 5

Acce[illegible]

NTIS [illegible] √
DTIC [illegible]
U[illegible]
J[illegible]

By [illegible]
Distributio /

Availabil[illegible]

Dist	Avail a[illegible] Special
A-1	

PI Name:	Yellamraju V. Srinivas
PI Institution:	Kestrel Institute
	3260 Hillview Avenue, Palo Alto, CA 94304
PI Phone Number:	(415) 493-6871
PI E-mail Address:	srinivas@kestrel.edu
Grant or Contract Title:	Applications of Sheaf Theory in Algorithm Design
Grant or Contract Number:	N00014-92-C-0124
Reporting Period:	1 Oct 1992 to 30 Sep 93

2 Summary of Technical Results

The general goal of this project is the application of concepts from topology and sheaf theory to better explain and systematize some of the intricate aspects of algorithms. The basic technique is the adoption of a topological view of data structures, in contrast to the normal algebraic view. This approach was successfully applied to the derivation of pattern matching algorithms in [Srinivas 93, Srinivas 92]. In this project, this approach was extended to parsing algorithms for context-free grammars. This extension also yielded more insight into what a topological view of data structures entails, e.g., a connection to the object-oriented view of datatypes.

2.1 A topological view of data structures

We will briefly describe the characteristics of such a view. Consider the example of strings. Normally, they are described algebraically, as a monoid on some alphabet. That is, strings are constructed inductively using the operations of empty strings, singleton string, and concatenation. Functions on strings are defined using recursion on these constructors.

In the topological view, we concentrate on the internal structure of strings (e.g., a string is a total order) and maps between strings (e.g., adjacency-preserving map). The role of induction using constructors is played by the *covering* relationship: a string can be covered by a set of (possibly overlapping) substrings. Now, functions are defined using sheaves, which assign a value to a string by gluing together the values on elements of a cover (the analogue of recursive definitions).

This approach is usually more abstract than the algebraic approach, and allows us to concentrate on the essential aspects of an algorithm.

2.2 Application to parsing

The topology of data structures can be exploited in a similar manner in the derivation of parsing algorithms [Srinivas b]. Consider a context-free grammar for strings. Such a grammar specifies a set of strings (the language generated by the grammar) using both recursion and juxtaposition of strings. The two basic operations in a context-free grammar are

- rewriting via a production, i.e., replacing a non-terminal by a sentential form (e.g., $S \to aSa$ applied to bSb yields $baSab$), and
- alternation, which associates several sentential forms with a non-terminal (e.g., $S \to aSa \mid bSb \mid a \mid b$).

Corresponding to these two operations we can associate two topologies with a grammar. Replacing a non-terminal involves decomposing a sentential form and depends on the topology of the underlying data structure (strings, trees, etc.) of the language being generated. This topology is specific to the data structure and independent of the grammar.

Alternation in the right-hand side production translates to a union of languages, and generates a topology on languages [Walter 75]. Using this topology, the language generated by a grammar can be described as a fixed point [Manes and Arbib 86]. This topology is specific to the grammar and (almost) independent of the data structure.

By separating these two topologies and precisely characterizing their relationship, we can abstractly describe parsing algorithms and provide an intuitive explanation of such techniques as LR parsing. The collection of parses of a given terminal string has the structure of a sequence of sheaves, each sheaf representing the collection of (partial) derivation trees of depth up to n (for $n = 1, 2, 3, \ldots$). Different parsing algorithms (top-down, bottom-up, etc.) correspond to different design decisions and optimizations used in generating this sequence and extracting parses.

A sheaf-theoretic characterization of parsing nicely exposes the interplay between the algebraic and topological structure of a grammar. The sequence of parse sheaves above becomes a simplicial complex if composite productions are included (e.g., if $S \to AB$ and $A \to aA$, we get a composite production $S \to aAB$).

2.3 Connecting algebraic and topological descriptions of datatypes

There is a similar interplay in equationally described data types. Consider an abstract data type which is characterized by a set of operations satisfying some axioms (see, e.g., [Wirsing 90]). The collection of terms generated by the signature of the data type can be cast as a sequence of sheaves. Using this connection, the topological structure of an algebraically specified data type can be exposed. This, in turn, can provide some of the facilities of object-oriented programming style in the framework of algebraic specification [Srinivas a]. For example, we can systematically convert between the description of a tree using the operations empty and join (an algebraic description) and one using a set of nodes with a parent-child relationship satisfying some conditions (an object-oriented description).

2.4 Homomorphisms between datatypes

A topological description of dataypes, although conceptually simple, needs a lot of mathematical machinery. On the other hand, the algebraic style is simple and has been successfully used in the automated support of the program synthesis [Smith 90]. One result of this project is the insight that some of the benefits of the topological view can be obtained by adding a facility for defining homomorphisms to algebraic specifications. Homormorphisms relate the internal structure of two datatypes. For example, if a tree is described as a set of nodes together with a parent relationship, then it can be homomorphically mapped into a set of nested regions on a display; this technique is used in the specification of a graphical user interface in [Srinivas and Jüllig 93]. Note that the notion of homomorphism used here is somewhat different from that used in [Bird 87]; the latter operates at the level of constructors.

2.5 Sheaves and constraint satisfaction

The connection between sheaves and constraint satisfaction was noticed in [Srinivas 93] using Waltz's filtering algorithm for scene labeling. I explored this connection further in the context of type inference algorithms.

A constraint satisfaction problem consists of a hypergraph in which the nodes represent are variables and the (hyper) arcs represent constraints [Montanari and Rossi 91]. Each node/variable is labeled by a set of admissible values, and each arc by a relation which specifies a constraint. A solution to such a problem is an assignment of values to variables which satisfies all the constraints.

Consider the site of all subgraphs of a hypergraph with inclusions as arrows and surjective families as covers. Then the solutions to a constraint satisfaction problem form a sheaf: any consistent assignment must be assembled from consistent parts. Constraint satisfaction algorithms search for consistent assignments of values to variables, a process which can be succinctly described as *sheafification* starting from a functor which only labels each variable with a set of admissible values and each arc with a relation.

For the case of type inference, we treat an expression tree as a hypergraph: each node in the tree corresponds to a hypergraph node, and each function corresponds to a hyperarc connecting the arguments and the result. Each node is labeled by its possible types, and each hyperarc by the possible types of the function. Type inference consists of finding compatible families of type assignments. If polymorphism is included, the set of possible types for a node may be infinite: thus it has to be represented intensionally as an expression. In this case, finding a compatible family corresponds to finding a limit, which in turn requires *equalizers*. Equalizers in this context are nothing but most general unifiers [Rydeheard and Burstall 88]. This example illustrates the nice interplay between the topology of the expression tree and the algebra of the type expressions, a situation ideal for sheaves.

References

[Bird 87] Bird, R. S. An introduction to the theory of lists. In *Logic of Programming and Calculi of Discrete Design*, M. Broy, Ed., *NATO ASI Series*, Vol. F36. Springer-Verlag, Berlin, 1987, pp. 5–42.

[Manes and Arbib 86] Manes, E. G., and Arbib, M. A. *Algebraic Approaches to Program Semantics*. Springer-Verlag, New York, 1986.

[Montanari and Rossi 91] Montanari, U., and Rossi, F. Constraint relaxation may be perfect. *Artifical Intelligence 48* (1991), 143–170.

[Rydeheard and Burstall 88] Rydeheard, D., and Burstall, R. M. *Computational Category Theo.y*. Prentice-Hall, 1988.

[Smith 90] Smith, D. R. KIDS: A semiautomatic program development system. *IEEE Trans. Softw. Eng. 16*, 9 (Sept. 1990), 1024–1043.

[Srinivas a] Srinivas, Y. V. Augmenting algebraic specifications with structured sorts and structural subsorting. *In Preparation.*

[Srinivas b] Srinivas, Y. V. Deriving parsing algorithms using sheaves. *In Preparation.*

[Srinivas 92] Srinivas, Y. V. Derivation of a parallel matching algorithm. In *Second International Conference on Mathematics of Program Construction* (Oxford, UK, July 1992), *Lecture Notes in Computer Science*, Vol. 669, Springer-Verlag, pp. 323–343.

[Srinivas 93] Srinivas, Y. V. A sheaf-theoretic approach to pattern matching and related problems. *Theoretical Comput. Sci. 112* (1993), 53–97.

[Srinivas and Jüllig 93] SRINIVAS, Y. V., AND JÜLLIG, R. Formal specification of a graphical user interface. Tech. Rep. KES.U.93.4, Kestrel Institute, May 1993.

[Walter 75] WALTER, H. Topologies on formal languages. *Mathematical Systems Theory 9*, 2 (1975), 142–158.

[Wirsing 90] WIRSING, M. Algebraic specification. In *Formal Models and Semantics*, J. van Leeuwen, Ed., *Handbook of Theoretical Computer Science*, Vol. B. MIT Press/Elsevier, 1990, pp. 675–788.

PI Name:	Yellamraju V. Srinivas
PI Institution:	Kestrel Institute
	3260 Hillview Avenue, Palo Alto, CA 94304
PI Phone Number:	(415) 493-6871
PI E-mail Address:	srinivas@kestrel.edu
Grant or Contract Title:	Applications of Sheaf Theory in Algorithm Design
Grant or Contract Number:	N00014-92-C-0124
Reporting Period:	1 Oct 1992 to 30 Sep 93

3 Publications, Presentations, Reports

Publications:

1. SRINIVAS, Y. V. Derivation of a parallel matching algorithm. In *Second International Conference on Mathematics of Program Construction* (Oxford, UK, July 1992), *Lecture Notes in Computer Science*, Vol. 669, Springer-Verlag, 1993, pp. 323–343.

   Abstract: We present a derivation of a parallel version of the Knuth-Morris-Pratt algorithm for finding occurrences of a pattern string in a target string. We show that the failure function, the source of efficiency of the sequential algorithm, is a form of search in an ordered domain. This view enables the generalization of the algorithm both beyond sequential execution and the string data structure. Our derivation systematically uses a divide-and-conquer strategy. The computation tree so generated can be mapped onto time, yielding a naive sequential algorithm, onto a processor tree, yielding a parallel algorithm, or onto a data structure, yielding the failure function.

2. SRINIVAS, Y. V. A sheaf-theoretic approach to pattern matching and related problems. *Theoretical Comput. Sci. 112* (1993), 53–97.

   Abstract: We present a general theory of pattern matching by adopting an extensional, geometric view of patterns. Representing the geometry of the pattern via a Grothendieck topology, the extension of the matching relation for a constant target and varying pattern forms a sheaf. We derive a generalized version of the Knuth-Morris-Pratt string matching algorithm by gradually converting this extensional description into an intensional description, i.e., an algorithm. The generality of this approach is illustrated by briefly considering other applications: Earley's algorithm for parsing, Waltz filtering for scene analysis, matching modulo commutativity, and the n-queens problem.

3. Jüllig, R., and Srinivas, Y. V. Diagrams for software synthesis. In *Proceedings of the 8th Knowledge-Based Software Engineering Conference*

(Chicago, IL, September 20–23, 1993), IEEE Computer Society Press, 1993, pp. 10–19.

Abstract: We describe the formal environment at Kestrel for synthesizing programs. We show that straightforward formalization, persistently applied at all levels of system description and system derivation, produces a scalable architecture for a synthesis environment. The primitive building blocks of our framework are specifications, which encapsulate types and operations, and specification arrows, which are relations between specifications. The design of a system is represented as a diagram of specifications and arrows. Synthesis steps manipulate such diagrams, for example, by adding design detail to some specification, or by building new diagrams. A design history is a diagram of diagrams. Thus, we have a formal, knowledge-based, and machine-supported counterpart to such software engineering methodologies as CASE and OOP.

Presentations:

1. Homomorphisms Between Algebraic Specifications. Presentation given at the *IFIP WG2.1 Meeting*, Winnipeg, Canada, May 1993.

   Abstract: We introduce the specification-level counterparts to homomorphisms between algebras. These take the form of homomorphism specifications (homspecs), which specify a class of homomorphisms. Homspecs augment the algebraic specification framework by providing a way of encapsulating families of related functions. We briefly indicate several applicatins where homomorphisms are useful: (1) inductive function definitions, e.g., size of a sequence; (2) specification of divide-and-conquer algorithms, e.g., mergesort; (3) relating structured objects, e.g., subtrees; (4) abstraction/representation functions which implement one specification in terms of another, e.g., set-as-list; (5) specifying constraints in a graphical display.

Reports:

1. SRINIVAS, Y. V., AND JÜLLIG, R. Formal Specification of a Graphical User Interface, Technical Report KES.U.93.4, Kestrel Institute, Palo Alto, May 1993.

   Abstract: Formal development technology has advanced over the past two decades from the derivation of small, functional programs to the stage where it can be applied to realistic problems. We specify an interactive user interface for displaying internally stored data structures. We show how the code for propagating the changes to the internal representation or the display can be formally derived.

PI Name:	Yellamraju V. Srinivas
PI Institution:	Kestrel Institute
	3260 Hillview Avenue, Palo Alto, CA 94304
PI Phone Number:	(415) 493-6871
PI E-mail Address:	srinivas@kestrel.edu
Grant or Contract Title:	Applications of Sheaf Theory in Algorithm Design
Grant or Contract Number	N00014-92-C-0124
Reporting Period:	1 Oct 1992 to 30 Sep 93

4 Transitions and DoD interactions

NONE.

PI Name:	Yellamraju V. Srinivas
PI Institution:	Kestrel Institute
	3260 Hillview Avenue, Palo Alto, CA 94304
PI Phone Number:	(415) 493-6871
PI E-mail Address:	srinivas@kestrel.edu
Grant or Contract Title:	Applications of Sheaf Theory in Algorithm Design
Grant or Contract Number:	N00014-92-C-0124
Reporting Period:	1 Oct 1992 to 30 Sep 93

5 Software and Hardware Prototypes

NONE.