NASA/CR—2000-209947

Progress in the Semantic Analysis of Scientific Code

Mark Stewart
Dynacs Engineering Company, Inc., Brook Park, Ohio

Prepared for the
Computational Aerosciences Workshop
sponsored by the High Performance Computing and Communications Program
Moffett Field, California, February 15–17, 2000

Prepared under Contract NAS3–98008

National Aeronautics and
Space Administration

Glenn Research Center

November 2000

This report contains preliminary findings, subject to revision as analysis proceeds.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

Available electronically at http://gltrs.grc.nasa.gov/GLTRS

PROGRESS IN THE SEMANTIC ANALYSIS OF SCIENTIFIC CODE

Mark Stewart
Dynacs Engineering Company, Inc.
2001 Aerospace Parkway
Brook Park, Ohio 44142
Mark.E.Stewart@grc.nasa.gov
216–977–1163

Existing software analysis tools use the semantics of the programming language to check our codes: Are variables declared and initialized? Do variable types match? Where do memory leaks and memory errors occur? However, the meaning or semantics that a code developer builds into his/her code extends far beyond programming language semantics. Scientific code developers use variables to represent physical and mathematical quantities (mass, derivative), expressions of quantities to represent physical formulae (Navier-Stokes equation), loops to apply these formulae in a domain, and conditional expressions to control execution. These semantic details are crucial when developers and users try to understand and check their scientific and engineering codes; further, their analysis is manual, time-consuming, and error-prone.

This paper reports progress in an experiment to automatically recognize and check these physical and mathematical semantics. The experimental procedure combines semantic declarations with a pattern recognition capability; the code (1)

```
C?  MA == mass, ACC == acceleration                         (1)
    FF = MA * ACC
```

contains two semantic declarations for MA and ACC, and with Newton's law among the recognizable patterns, the procedure recognizes this code as force assigned to FF. These formula patterns are represented in and recognized by parsers[1]. The conclusions of this procedure are displayed for the user as shown in Figure 1. A more detailed explanation of this procedure and its extensions is given in Reference 2.

This experiment's objective is to understand the limits of this automatic recognition procedure: Does it apply to a wide range of scientific and engineering codes? Can it reduce the time, risk, and effort required to develop and modify scientific code?

Previous work[2] demonstrated that scientific concepts and formulae could be represented and recognized. In fact, for part of one reacting flow code (Figure 2), 50% of the operations can be recognized. However, this preliminary work posed several more questions: Can additional semantic details be represented and recognized? How well do the recognition rules work in blind test cases? What are the limitations of this procedure?

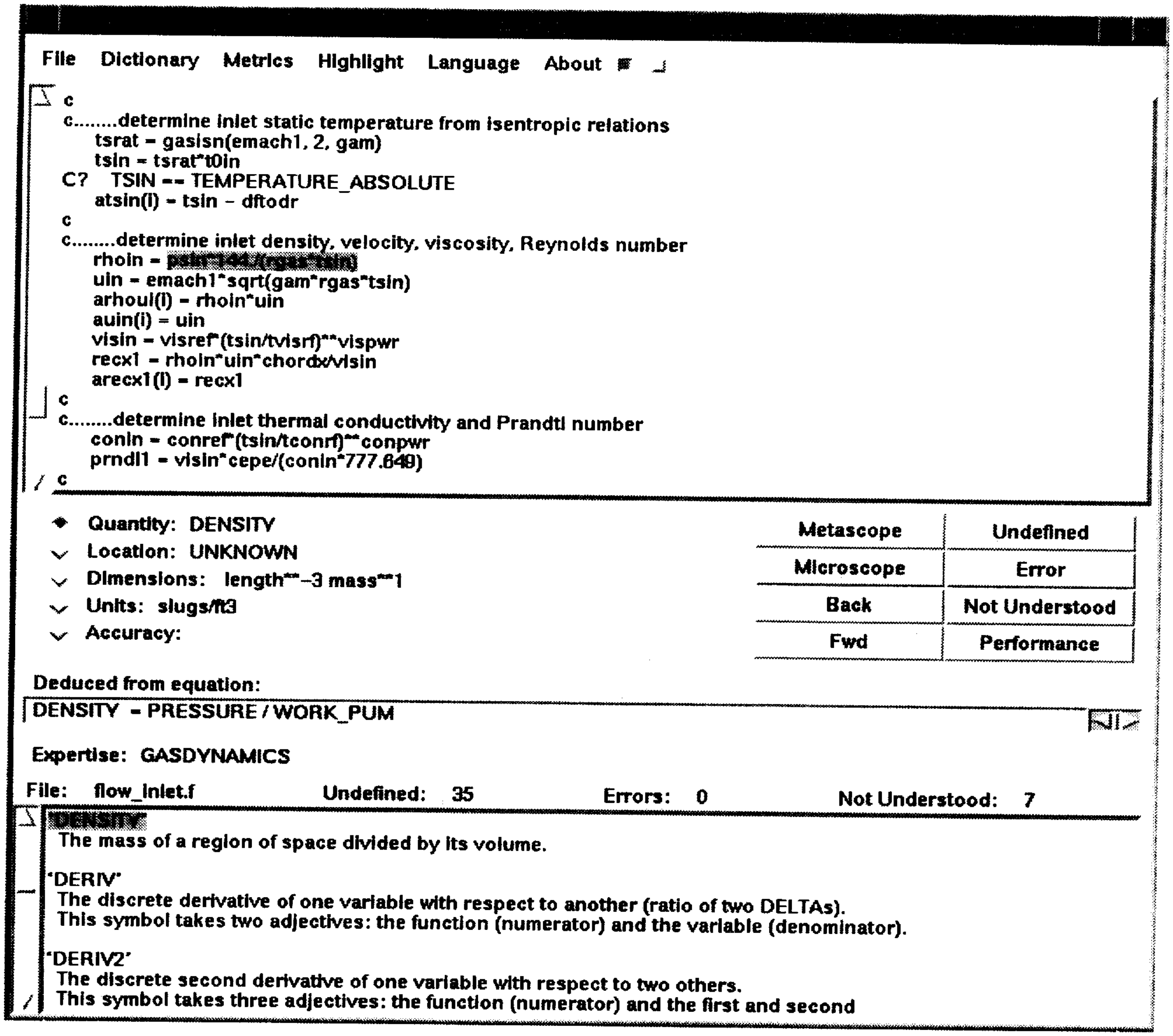

Figure 1: GUI display for the semantic analysis program. The top window displays a user's code; variables and expressions may be selected for explanation. The middle region explains this selected text. In this case, the physical quantity is density, it does not have a grid location, and it has the displayed dimensions, units, and derivation. The bottom region displays the semantic dictionary/lexicon.

To answer these questions, the procedure's representation and recognition of semantic details has been significantly extended, including expert parsers for vector analysis, object analysis (the object of the formula), array reference/assignment analysis. Also, existing expert parsers have been refined and extended. A measure of the expert parsers is given in Table 1. Table 2 samples the rules represented in these parsers.

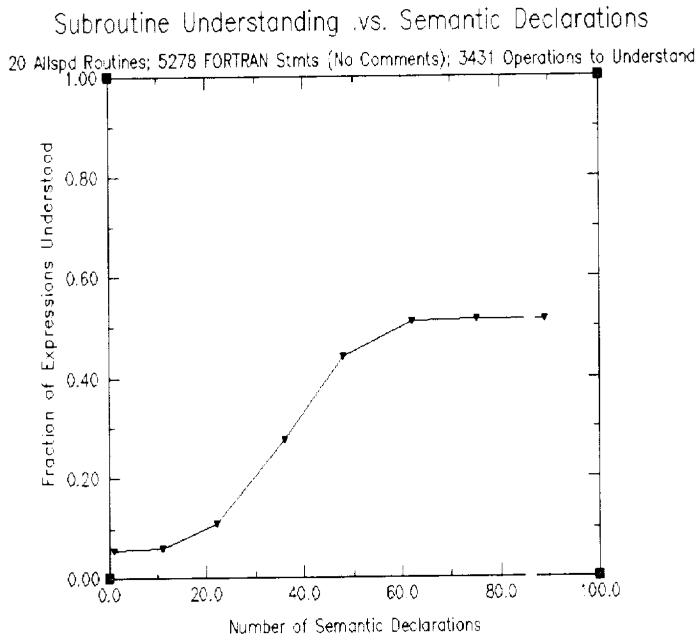

Figure 2: Graph showing the increase in expression understanding as semantic declarations are added to twenty subroutines from the ALLSPD code. The subroutines contain 5278 non-comment FORTRAN statements and 3431 operations to understand. Further work will increase the understanding fraction. The analysis results reflect the analysis code's quality and not the quality or ability of the ALLSPD code.

Aspect Analyzed	Parsers	Parser Rules	Fundamental Equations
Quantity-Math	5	772	72
Quantity-Physical	3	766	114
Value / Interval	2	223	27
Grid Location	4	1801	235
Geometrical Entity	1	447	20
Vector Entity	1	300	15
Non-Dimensional	1	72	5
Dimensions	1	59	10
Units	1	71	14
Object Analysis	1	128	10
Array Analysis	2	121	3

Table 1: Aspect analyses performed by the semantic analysis procedure including number of parsers for each aspect, number of Yacc[1] parser rules, and fundamental equations. Equation (1) corresponds to a fundamental equation; some equations require several parser rules.

Mathematical, Numerical Quantity	**Physical Quantity**	**Physical Quantity**
$q \Leftarrow q + 0$	$p \Leftarrow F / area$	$^{o}C \Leftarrow {}^{o}K - 273.15$
$q \Leftarrow q * 1$	$F \Leftarrow m * A$	$^{o}F \Leftarrow 1.8 * {}^{o}C + 32$
$0 \Leftarrow q_1 - q_2$	$W \Leftarrow F * length$	$\partial m/\partial t \Leftarrow \rho * U * A$
$\Delta q \Leftarrow q_1 - q_2$	$E_k \Leftarrow ½ * m * U^2$	$\nu \Leftarrow \mu / \rho$
Polynomials	$R_u \Leftarrow k * N_A$	$Pr \Leftarrow C_p \ \mu / k$
$\Sigma q \Leftarrow q + q + \ldots$	$R \Leftarrow R_u$ / Mol. wt.	Reynolds $\Leftarrow \rho * U * length/\mu$
$q^2 \Leftarrow q * q$	$R \Leftarrow C_p - C_v$	$u*\partial u/\partial x - (1/\rho)*\partial p/\partial x$
$2q \Leftarrow q_1 + q_2$	$C_p \Leftarrow \Sigma$ (Mass Fract.* C_p)	$U_\theta \Leftarrow r\,\Omega$
$\Delta^2 q \Leftarrow q - 2q + q$	$\gamma \Leftarrow C_p / C_v$	$(\partial m/\partial t)_{corr} \Leftarrow \partial m/\partial t \sqrt{\theta} / \delta$
$\partial q/\partial x \Leftarrow \Delta q / \Delta x$	$w \Leftarrow p / \rho$	Circum $\Leftarrow 2\,\pi\,r$
$\partial^2 q/\partial x^2 \Leftarrow \Delta^2 q / \Delta^2 x$	$c^2 \Leftarrow \gamma * p / \rho$	vol $\Leftarrow$ length * area
$\partial q/\partial y \Leftarrow \partial q/\partial x * \partial x/\partial y$	$p / \rho \Leftarrow R * T$	area $\Leftarrow$ length * length
$\nabla \cdot \mathbf{q} \Leftarrow$ expression	$e_i \Leftarrow 1/(\gamma-1) * p / \rho$	**Grid Location, Geometrical Entity**
$\nabla \times \mathbf{q} \Leftarrow$ expression	$e_k \Leftarrow ½ * U^2$	
$\nabla^2 q \Leftarrow$ expression	$h \Leftarrow e_i + w$	$l \Leftarrow l_1 \pm l_2$
$\mathbf{q}_1 \cdot \mathbf{q}_2 \Leftarrow$ expression	$h_o \Leftarrow h + e_k$	$l \Leftarrow l_1$ */ l_2
$\mathbf{q}_1 \times \mathbf{q}_2 \Leftarrow$ expression	$M \Leftarrow U / c$	$g \Leftarrow g_1 \pm g_2$
Jacobian $\Leftarrow$ expression	$P \Leftarrow const * T^{\gamma/\gamma-1}$	$g \Leftarrow g_1$ */ g_2
Number Value, Number Interval	**Vector Entity**	**Non-Dimensionalization, Dimensions, Units**
$n \Leftarrow n_1 \pm n_2$	$v \Leftarrow v_1 \pm v_2$	$D \Leftarrow D_1$ ±*/ D_2
$n \Leftarrow n_1$ */ n_2	$v \Leftarrow v_1$ */ scalar	$D \Leftarrow$ ftn(D_1)
$n \Leftarrow n_1$ ** n_2	surface $\Leftarrow v_1 * v_2$	$d \Leftarrow d_1$ ±*/ d_2
$n \Leftarrow$ ftn(n_1)	scalar $\Leftarrow$ scalar ± scalar	$d \Leftarrow$ ftn(d_1)
$r \Leftarrow r_1 \pm r_2$	scalar $\Leftarrow$ scalar */ scalar	$u \Leftarrow u_1$ ±*/ u_2
$r \Leftarrow r_1$ */ r_2	scalar $\Leftarrow$ Dot Product	$u \Leftarrow$ ftn(u_1)
q = Math/Numerical Quantity; l = Grid Location; g = Geometrical Entity; v = Vector Entity; n = Number Value; r = Number Interval; D = Non-Dimensionalization; d = Dimensions; u = Units		

Table 2: A sampling of expert parser rules used in the semantic analysis method. Many rules are condensed. Due to decomposition a single operation may involve multiple independent aspects (units, grid location and quantity for *x_coordinate – x_coordinate*), and several rules from this table can apply to it.

To understand the procedure's generality, that is, if the rules and recognition capability can apply to a range of codes, the procedure's performance was tested on large blind test cases. Semantic declarations for solution variables and coordinates were included in the ADPAC code (a 3D Navier-Stokes, curvilinear coordinate, turbomachinery code with 86k lines of code (loc)) and the ENG10 code (an axisymmetric, curvilinear coordinate, engine simulation code with 20k loc). The fraction of operations recognized is shown in Figure 3. These baseline results provide some initial evidence of generality, however, how these measurements improve as the procedure develops further is most important.

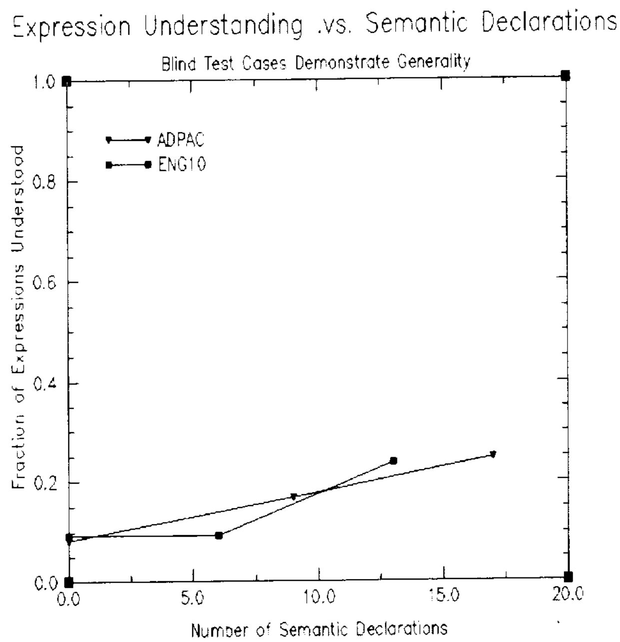

Figure 3: Graph showing the increase in expression understanding as semantic declarations are added to two blind test cases. The ADPAC codes contain 86k loc, and the ENG10 code contains 20k loc. Further work will increase the understanding fraction. The analysis results reflect the analysis code's quality and not the quality or abilities of the ADPAC or ENG10 codes.

Assessing the future of this procedure is problematic, however experience indicates that three issues will determine success. First, the large number of formulae used in scientific codes—even within a field—makes it difficult, but not a priori impossible, to capture the knowledge necessary for recognition. Second, although one rule application or inference is necessary to recognize equation (1), and the formula sqrt $(u_x^2 + u_y^2 + u_z^2)$ involves six inferences, $O(10^2)$ inferences are often required as expressions are evaluated and combined. Needing many inferences to find a result magnifies the risk of failure since an unknown inference, a limitation of this procedure, or a coding error will terminate the inference chain and leave the result unidentified. Hence, success of this method depends on good coverage of the domain knowledge, a robust semantic analysis procedure, and stable procedure coding. Third, representation of semantic details has not been a major problem, however continued success in representing knowledge is important.

Future work will pursue two questions. First, can formulae be added to the expert parsers so that the knowledge domain is sufficiently covered for good recognition of general codes? Second, can the procedure be perfected to a useful scientific software tool? The best way to answer these questions is to develop the procedure further while testing it on more codes.

[1]A.V. Aho, R. Sethi, and J.D. Ullman, *Compilers: Principles, Techniques, and Tools* (Reading: Addison-Wesley, 1986).
[2]M.E.M. Stewart, and S. Townsend, "An Experiment in Automated, Scientific-Code Semantic Analysis," AIAA-99-3276, (June 1999).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)

2. REPORT DATE
November 2000

3. REPORT TYPE AND DATES COVERED
Final Contractor Report

4. TITLE AND SUBTITLE
Progress in the Semantic Analysis of Scientific Code

5. FUNDING NUMBERS
WU–725–10–11–00
NAS3–98008

6. AUTHOR(S)
Mark Stewart

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Dynacs Engineering Company, Inc.
2001 Aerospace Parkway
Brook Park, Ohio 44142

8. PERFORMING ORGANIZATION REPORT NUMBER
E–12194

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR—2000-209947

11. SUPPLEMENTARY NOTES

Prepared for the Computational Aerosciences Workshop sponsored by the High Performance Computing and Communications Program, Moffett Field, California, February 15–17, 2000. Project Manager, Karl Owen, Computing and Interdisciplinary Systems Office, NASA Glenn Research Center, organization code 2900, 216–433–5895.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category: 61 Distribution: Nonstandard
Available electronically at http://gltrs.grc.nasa.gov/GLTRS
This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

This paper concerns a procedure that analyzes aspects of the meaning or semantics of scientific and engineering code. This procedure involves taking a user's existing code, adding semantic declarations for some primitive variables, and parsing this annotated code using multiple, independent expert parsers. These semantic parsers encode domain knowledge and recognize formulae in different disciplines including physics, numerical methods, mathematics, and geometry. The parsers will automatically recognize and document some static, semantic concepts and help locate some program semantic errors. These techniques may apply to a wider range of scientific codes. If so, the techniques could reduce the time, risk, and effort required to develop and modify scientific codes.

14. SUBJECT TERMS
Software engineering; Computational fluid mechanics

15. NUMBER OF PAGES
12

16. PRICE CODE
A03

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