An OPS5 Expert System Converted to FORPS

James L. Rash
Code 531.1
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

ABSTRACT

FORPS, a simple Forth-based production system, has been used to reimplement TRAPS, an expert system originally programmed in OPS5 to perform input data checking in Goddard's Communications Link Analysis and Simulation System (CLASS). The FORPS version of TRAPS is implemented both on the CLASS computer and on an IBM PC/AT compatible machine (Hewlett-Packard Vectra). This paper presents results of the TRAPS reimplementation; compares FORPS with OPS5, including efficiency on the HP Vectra; and offers suggestions for enhancements to FORPS.

INTRODUCTION

Rule-Based Programming. An expert system is a computer program which can perform as well as a human expert in some problem domain. Typically, expert systems are created using a nonprocedural programming methodology known as rule-based programming, in which knowledge (rules) concerning the problem domain is kept separate from the data (facts) that characterize the current state of the problem domain. Programs written in conventional computer languages rely on hard-coded execution paths locked into the object program at compilation time. Execution of a rule-based program, on the other hand, follows a more cyclic or "data driven" path which is unrelated to the order of the rules in the rule base. Execution (firing) of a rule is followed by a fresh comparison (matching) of the conditional parts of the rules in the rule base with patterns in the facts data base. In a process called "conflict resolution", one of the matched rules is then selected to fire. This match-select-fire cycle is controlled by the inference engine [Brownston 85].

OPS5. Numerous production system languages have been developed. One of the oldest and most popular is OPS5 (Official Production System, Version 5), developed in the 1970's by Charles L. Forgy at Carnegie-Mellon University. Early implementations were in BLISS, MACLISP, and FRANZ LISP [Forgy 81], but today OPS5 implementations exist in other languages including Forth [Dress 86] and C.

FORPS. FORPS (FORth-based Production System) was developed by Chris Matheus in 1986 at Oak Ridge National Laboratory. It was designed to be fast and efficient, maintaining the usual Forth environment and thereby providing a tool to develop knowledge based systems for real time applications [Matheus 86].

THE OPS5 VERSION OF TRAPS

The Trajectory Preprocessing System (TRAPS) was developed as an expert system to detect certain kinds of errors in spacecraft mission trajectory data supplied to the CLASS computer. Built as a prototype on an IBM PC, TRAPS was intended both to serve as a

model for other expert systems and to provide a vehicle for comparing various approaches to solving problems using procedural languages as well as expert system tools [Rash 86].

Implemented in the OPS5+ software package from Computer Thought Corp., TRAPS comprises 49 rules based on six "human rules" expressing requirements to be satisfied by spacecraft mission trajectory data. No calls to user-written routines in the host language (C) were required in the OPS5 version.

THE FORPS VERSION OF TRAPS

The availability of FORPS provided an opportunity to compare OPS5 with another production system tool. To make a fair comparison, the original TRAPS application was translated from OPS5 to FORPS, rule-for-rule in most cases, and then run on the same computer (an 8 MHz Hewlett-Packard Vectra, which is compatible with the IBM PC/AT). A version of FORPS obtained from its inventor, and running under Laxen & Perry's F83 on IBM PCs and compatibles, was used. The FORPS version of TRAPS has also been implemented in SS-FORTH on the CLASS computer.

COMPARISON OF FORPS AND OPS5

Efficiency. Table 1 presents results of test runs using a typical TRAPS input data file containing 163 records. The table compares the original OPS5 version to two new versions written in FORPS, one comprising 48 rules and the other, 35 rules. Instead of some dozen rules to generate output, as in the original OPS5 version, the 35-rule version uses only one rule (which had to be supported by several custom routines in Forth). The table shows that the 35 rule version runs 30 percent faster than the original, but the 48 rule version is only 18 percent faster.

However, for applications with much larger rule bases, FORPS would be at an ever increasing disadvantage relative to OPS5 because FORPS evaluates the left-hand side of every rule in the rule base during each match-select-fire cycle. The Rete match algorithm used in OPS5, on the other hand, takes advantage of

TABLE 1. Summary of Test Results

Tool Used	# Rules in Rule Base	Processing Efficiency (Records/Second)
OPS5+	49	6.0
FORPS	48	7.1
FORPS	35	7.9

"temporal redundancy" of data in the facts base to avoid the inefficiency of evaluating all rules during each cycle [Brownston 85, Chap. 6].

Host Language Support. Offsetting the efficiency advantage of the Rete algorithm in OPS5 is the fact that, in FORPS, host language routines are so easily incorporated. This is important in debugging applications and in developing applications which

must run fast or interface with other systems such as data bases and hardware devices. Efficiency can be greatly improved with appropriate use of procedural routines. OPS5 has a host language interface, but this provides far less than the tight coupling capability available with the FORPS/Forth combination, and in fact represents a relative weakness of OPS5.

Knowledge Representation. Knowledge is represented in OPS5 in the form of rules as well as data objects, which are defined and referenced by a special, built-in pattern description language. Custom Forth code had to be developed in the FORPS version of TRAPS to provide facilities for pattern matching and knowledge representation. Though the necessity of devising routines for such essential components of a production system language suggests basic inadequacy of FORPS, it is noted that FORPS was deliberately designed as a simple inference engine around which can be built any desired expert system facilities.

Conflict Resolution. Controlling rule selection is crucial in production systems because the execution path is data driven rather than instruction driven. The primary means of control in FORPS is rule priority assignment. This sufficed for TRAPS, but more difficult problems would call for other control strategies, which, with Forth, could be devised specifically for the problem at hand. OPS5 has no built-in rule priority scheme, but instead more elaborate strategies based on "recency" (how recently a fact was added or modified).

User Environment. User environment is implementation-dependent. OPS5+ provides a "develop mode" which includes a useful window and mouse-menu interface to aid in application development and debugging. FORPS has no such features, but the normal Forth environment somewhat offsets this lack.

Readability. Figure 1 presents one of the OPS5 rules, while Figure 2 shows the corresponding rule in the FORPS version of TRAPS. This is one of the simpler rules but it serves to

```
(p link-support-3-switch-counting
   (goal=process-a-record)
   { (current-support ^current-support-code <cs>
                      ^switch-count <k> ) <c> }
   (record ^link-support-3 { <s> <> <cs> } ^link-3 <> 0 )
  -->
   (modify <c> ^current-support-code <s>
               ^switch-count (compute <k> + 1) ))
```

FIGURE 1. Example of an OPS5 Rule.

```
RULE:  LINK-SUPPORT-3-SWITCH-COUNTING     PRIORITY:  10
 *IF*
    GOAL=PROCESS-RECORD
    $CURRENT-SUPPORT-CODE  LINK-SUPPORT-3  $=  NOT
    LINK-3  " 0"  $=  NOT
 *THEN*
    LINK-SUPPORT-3  $CURRENT-SUPPORT-CODE  $!
    SWITCH-COUNT  1+  TO  SWITCH-COUNT
 *END*
```

FIGURE 2. Example of a FORPS Rule.

illustrate a subjective comparison to be made: readability. The author previously judged OPS5 rules more readable than FORTRAN, but now regards FORPS rules as more readable than OPS5 rules.

Memory Requirements. Typical of Forth applications, memory requirements are low in FORPS -- less than 2 Kbytes for FORPS itself, while the entire TRAPS program added less than 20 Kbytes (including space for facts, compiled rules, and ancillary Forth code). Total memory required, including F83, was less than 42 Kbytes. In contrast, OPS5+ requires 256 Kbytes just to load, plus space for working memory and compiled rules.

Implementations. As a final comparison, it is noted that FORPS is in the public domain. It has been implemented in F83, C-Forth, polyFORTH (tm, FORTH, Inc.) and SS-FORTH, and should be easily adaptable to other Forth dialects. By contrast, while there are many implementations of OPS5, by far most are implemented in LISP on rather large machines; to the author's knowledge OPS5 is not available in a public domain version for microcomputers (although it should be possible to port one of the public domain LISP versions to a PC).

CONCLUSIONS

The FORPS implementation of TRAPS discussed in this paper appears to be at least as viable as the OPS5 implementation for the following reasons: (1) it processes input data faster than the OPS5+ version; (2) it seems, subjectively, to be more readable and therefore should be more maintainable; and (3) the ability to incorporate custom Forth code is expected to make it more adaptable to special future requirements.

The combination of Forth and FORPS should be viewed as a serious alternative to OPS5 for smaller expert system applications. However, FORPS would be inadequate for general use in developing larger expert systems because it lacks features and capabilities often taken for granted in expert system development tools. A list of enhancements to resolve this would well include knowledge representation schemes such as objects/frames with inheritance, a pattern matcher based on the Rete algorithm, flexible conflict resolution strategies, and an attractive user environment based on windows.

REFERENCES

[Brownston 85] L. Brownston, R. Farrell, E. Kant, and N. Martin, Programming Expert Systems in OPS5, An Introduction to Rule-Based Programming, Addison-Wesley, Reading, Massachusettes, 1985.

[Dress 86] W. B. Dress, "REAL-OPS, A Real-Time Engineering Applications Language For Writing Expert Systems", Proceedings, 1986 Rochester Forth Conference.

[Forgy 81] C. L. Forgy, OPS5 User's Manual, CMU-CS-81-135, Carnegie-Mellon, University, Department of Computer Science, 1981.

[Matheus 86] C. J. Matheus, "The Internals of FORPS: A FORth-based Production System", The Journal of Forth Application and Research, Vol. 4, No. 1.

[Rash 86] J. L. Rash, "A Prototype Expert System in OPS5 for Data Error Detection", Proceedings, 1986 Rochester Forth Conference.