An Implementation of FORPS on a NOVIX Beta Board

Christopher J. Matheus[1]
Instrumentations and Controls Division
Oak Ridge National Laboratory[2]

INTRODUCTION

FORPS is a Forth-based Production System (in the public domain) developed for fast, real time control problems [MATHUES86a]. It employs a very small and efficient inference engine (source <3k) to cycle through sets of IF-THEN production rules. The original version of FORPS was written in polyFORTH for a 68000 microcomputer. It has been applied to the real time control problem of robot obstacle avoidance [MATHEUS86a], and to the classic AI problem, The Towers of Hanoi [MATHEUS86b].

Recently, FORPS has been ported to the NOVIX Beta Board running novixFORTH. The translation from polyFORTH to novixFORTH was straightforward but was complicated by some problems with the NOVIX Beta Board. For timing comparisons, the Towers of Hanoi solution was run on the 6 MHz NOVIX with a realized speedup over the 10 MHz 68000 of nearly 30:1. With further optimization it is hoped that FORPS running on the NOVIX will approach speeds of 10,000 rules per second.

A BRIEF DESCRIPTION OF FORPS

For some of our work at ORNL we desired a production system which would allow the easy application of high level expert-type knowledge to real time control programs written in FORTH. This meant that, except for the IF...THEN production syntax, the contents of the rules had to be user definable FORTH words. The system also had to be as small and efficient as possible so that it could be used in real time applications without excessive consumption of computer resources. What resulted is FORPS, a small production system with a fast, near minimal inference engine, and a method for defining rules using FORTH words. Because it is so small and efficient FORPS does not contain many of the features of more extensive production systems. However, since FORPS is fully extensible, desired features can be added as required.

RULE DEFINITIONS

Rules are defined in FORPS as simple *RULE*...*IF*...*THEN*...*END* structures (see Fig. 1). Each rule is given a name and an optional PRIORITY value to be used during "conflict resolution" (i.e. when more than one rule is active the rule with highest priority is fired). The left hand side (LHS) or conditional portion of a rule contains standard FORTH words defined by the programmer. These words will be executed by the inference engine to determine if the rule is active -- the results they leave on the stack are ANDed to indicate if the rule is active. The words appearing in the right hand side (RHS) or action portion of the rule are executed by the inference engine whenever the rule is both active and is of highest priority from amongst all active rules.

---

[1] On leave from the University of Illinois at Champaign-Urbana.

[2] Operated by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy, under Contract No. DE-AC05-840R21400.

Consider the sample rule in Fig. 1 taken from an imaginary water control system. The word PRESSURE could be either a variable holding the pressure reading of a pipe or could reference an actual I/O port from which the pressure is read. If the value is higher than 100, TRUE is left on the stack (at run time). If the result of WATER-FLOW @ NOMINAL = is also TRUE then the rule will become active. If it also happens to be the active rule of highest priority it will fire -- i.e. "Water pressure is too high!" will be printed and OPEN-WATER-VALVE will be executed.

DICTIONARY ENTRY

When a rule is compiled a new entry is made in the FORTH dictionary under the rule's name. This entry actually becomes two words. The first word contains the code executed when the rule's condition is tested, and the second is the code executed if the rule is fired (see Fig. 2). The C-PFA points to the PFA of the conditional code, and the A-PFA points to the PFA of the action code. In addition to the compiled words of the LHS and RHS, the PFA also contains several words to mark the beginnings and ends of the condition and action code, and code to determine when the rule is active (see [MATHEUS86b] for an explanation of how this is accomplished).

THE RULE TABLE

To make the inference engine as fast as possible a table is created at the time of rule compilation to store all relevant rule information (see Fig. 3). This table contains four columns: C-PFA, A-PFA, active cell, and priority cell. The C-PFA and A-PFA columns contain pointers to the rules' C-PFA's and A-PFA's. The active cell is used for storage of the truth value that is calculated when the C-PFA of a rule is executed -- it is stored automatically by code in the C-PFA. The priority cell contains the rule's relative priority for use in conflict resolution.

THE INFERENCE ENGINE

The inference engine cycle is very simple:

1) test all conditionals, i.e. execute the C-PFA's
2) select the active rule of highest priority
3) fire the selected rule, i.e. execute its A-PFA
4) repeat until no rules are active

The rule table allows this process to be executed very efficiently by simply looping through the table to execute the C-PFA's and then looping through again to find the highest priority active rule. The high level code for the inference engine is shown in Fig. 4.

TRANSLATION TO NOVIX

novixFORTH was based on the design of polyFORTH. As a result, the vast majority of the FORPS ported without modification. There where however, a few problems encountered. Since polyFORTH for the 68000 uses 8 bit byte addressing, address offsets had to be changed in FORPS to account for NOVIX's 16 bit cell addressing. The NOVIX chip does not incorporate CFA's, but this difference actually simplified the code since it was no longer necessary to explicitly compile a colon at the beginnings of the C-PFA's and A-PFA's. The novixFORTH compiler failed to compile "COMPILE EXIT" as desired and so we were forced to compile the EXIT op-code explicitly as "32800 ,". It also became necessary to turn off the optimizing compiler when attempting to compile "COMPILE *if*" at the end of a definition. The familiar problem with NOVIX page boundary jumps (see [NOVIX86]) was never encountered -- for programs requiring a larger set of production rules, however, this could become a serious issue and is being looked into.

TIMING COMPARISONS

The FORPS solution to the Towers of Hanoi was translated to the NOVIX FORPS version by simply changing address offsets from bytes to cells. Timing tests where then conducted for towers of between five and ten disks. The results are shown in contrast to identical tests on a 10 MHz 68000 in Table 1. The relative speedup obtained was greater than 28:1. For the NOVIX this resulted in approximately 6200 rules being fired per second (with 2048 rules firing for 10 disks).

ANTICIPATED OPTIMIZATIONS

Our initial translation of FORPS from novixFORTH to polyFORTH did not attempt to take advantage of any features offered by the NOVIX chip. Thus, there is considerable room for optimization (see [NOVIX86]). The most obvious improvement would be to convert all DO LOOP's into the more natural and efficient #DO #LOOPS. This improvement alone should result in considerable time savings since DO LOOP's lie at the heart of the inference engine. It is also possible to study the FORPS code and optimize portions where two or more instructions might be combined into a single NOVIX operation [BRODIE85]. Finally, there are things in FORPS that could be changed to increase its speed (at the cost of readability and/or applicability). With there enhancements in mind, we anticipate having FORPS running on the NOVIX Beta Board at around 10,000 rules per second.

REFERENCES

[BRODIE85] Brodie, L. Programmer's Introduction to the NOVIX NC4000P Microprocessor, NOVIX, Inc. 1985.

[MATHEUS86a] Matheus, C.J. and Martin, H.L. FORPS: A Forth-based Production System and its Application to a Real Time Robot Control Problem, ASME International Computers in Engineering Conference, Chicago, July 1986.

[MATHEUS86b] Matheus, C.J. The Internals of FORPS - A Forth-based Production System, Journal of Forth Application and Research, 4, 1, 1986.

[NOVIX86] NC4000P User's Manual, NOVIX, Inc. 1986.

```
RULE: HIGH-PRESSURE   PRIORITY: 10
*IF*  PRESSURE @  100 >
      WATER-FLOW @ NOMINAL =
*THEN*  ." Water pressure is too high"
      OPEN-WATER-VALVE
*END*
```

Figure 1. Sample Rule

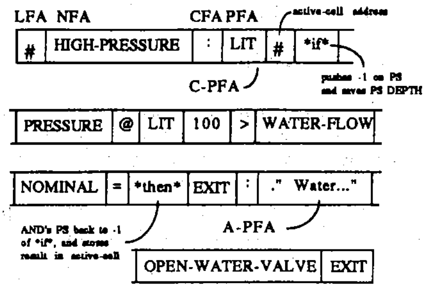

Figure 2. Dictionary Entry

Rule #	C-PFA	P-PFA	Active	Prior.
1	4020	4134	0	10
2	4148	4164	-1	1
:	:	:	:	:
n	4902	5030	-1	0

Figure 3. Rule Table

```
: FORPS  >RULE-TABLE @ 4-  >LAST-RULE !
         0 CYCLE !
  BEGIN
         1 CYCLE +!
         CLEAR-FIRES
         TEST-RULE-CONDS
         SELECT-BEST-RULE
         FIRE-RULE
         NO-ACTIVITY @
  UNTIL  ;
```

Figure 4. Inference Engine

Number of Disks	polyFORTH OMNIBYTE 68000 clock: 10 MHz	novixFORTH NOVIX Beta Board clock: 6 MHz
5	.29 sec.	.01 sec.
6	.58	.02
7	1.2	.04
8	2.3	.08
9	4.6	.16
10	9.3	.33

Table 1. Timing Comparisons