Vol. 34 No. 2 Transactions of Information Processing Society of Japan Feb. 1993

Short Note

EμPS Compiler for Procedural Description of Flow Control Entirely Segregated from the Rules

AKIO SASAKI,† SHOICHI MAKI,†† SHIRYO YASUI,†† HIROSHI ICHISE,†† TOMOYUKI MINAMIYAMA,†† ICHIROH ANDOH ††

This paper proposes a new programming paradigm EμPS for implementing rule-based production systems as a set of named rule clusters and a separate procedural description of flow at the level of the clusters. The procedural description is compiled into a main rule cluster which governs the flow via goal-settings generated by the compiler.

1. Introduction

Since the advent of rule-based Production System (PS),[2),5)] the problem of procedural flow control has been of serious concern. Proposed and in use so far are basically a variety of means of implementing the flow in its most primitive form of a network of inter-cluster GOTO's,[1),3),4),7)] and purely procedural frameworks in which the user can interact with the PS.[6),8),9),11)]

This paper proposes a general programming paradigm EμPS that allows description of flow at the rule-cluster (cluster) level in procedural constructs in a form entirely segregated from the rules.

The procedural description of flow is compiled into a main cluster, as a set of rules that calls via goal-settings generated by the compiler the clusters each implemented and named for a task. A compiler has been experimentally implemented and the output of the compiler is a source program in an OPS family language μPS.

Presented in the following are the programming style, sample outputs and key design concept of the compiler, and experiences on two practical examples.

2. EμPS Environment

In the EμPS environment proposed, the entire PS is written as in the following for compilation.

```
DEFINE MAIN
  Procedural description of flow
  at the level of the clusters
END
DEFINE CLUSTER Cluster_name_1
  Rules for task 1
END
------
------
------
DEFINE CLUSTER Cluster_name_n
  Rules for task n
END
```

The compiler compiles the procedural descriptions into rules working as the "MAIN" cluster. And it processes the rules in the task clusters to insert condition elements into the LHS and actions into the RHS to implement goal-settings for the flow control intended.

Figure 1 shows sample outputs of the compiler showing a rule produced for a construct "FIRE Cluster_X" that calls a "Cluster_X" cluster in the MAIN program. And **Fig. 2** shows how the compiler manipulates a rule in the "Cluster_X" cluster with "DONE" in its RHS for returning to the main program.

It is possible to compile any procedural constructs in the MAIN program, and among the constructs compilable on the experimentally implemented compiler are Conditional/Unconditional call by name of a named cluster, Subroutine Call, Repeat-Until, etc.

† Sasaki & Associates Inc.
†† Toyo Communication Equipment Co. Ltd.

Aimed at in the EμPS environment is to provide the user with a powerful tool for automatic generation of goal-settings for flow control of any depth and complexity. It is powerful in the same sense as a conventional language compiler is powerful in saving mannual works mandatory in an assembly language environment.

The key design concept of the EμPS environment is to divide the MAIN cluster into subclusters each representing a line of the MAIN program. The flow control among the subclusters is made by goal-settings generated by the compiler using a control WME of class line__ storing implicit logical line numbers in its attribute ^number.

For calling a task-cluster in each subcluster for a line, another control WME of class fire__ is used for goal-setting (again generated by the compiler). To call a task-cluster, the attribute ^now of this WME is set to the name of the cluster to which control is transferred. In the cluster called, all the rules have a condition element to match the cluster name. And upon completion of the task, a rule in it with "DONE" in the RHS sets the cluster name back to MAIN.

```
(t1) FIRE Cluster_X
(t2) .......

-->            DEFINE RULE Line_t1
               IF
                &fire__     ^now=|MAIN|
                &line__     ^number=t1
               THEN
                MODIFY fire__   ^now t2
                MODIFY line__   ^number t3
               END
```

Fig. 1 The rule produced for a cluster call "FIRE Cluster_X". The t1 is numerals for the logical line number of the current line, t2 is the name "Cluster_X", and t3 is numerals for the logical line number of the next line.

```
DEFINE RULE Rule_name
 IF
  &..............
 THEN
  ...............
  DONE
 END

-->            DEFINE RULE Rule_Name
                IF
                 &..............
                 &fire__^now=|Cluster_X|
                THEN
                 ...............
                 MODIFY fire__  ^now |MAIN|
               END
```

Fig. 2 Insertions made by the compiler to a rule with a "DONE" in its RHS in the "Cluster_X" cluster.

All the rules in the MAIN cluster have a condition element to match the name MAIN, and upon return to it only these rules in the subcluster for the next line are instantiated according to the value of the attribute ^number of the control WME line__.

For implementing a Subroutine Call in the MAIN program, the logical line number of the line at which the call is made is pushed down into a Call Stack implemented using still another control WME. Nested REPEAT-UNTIL constructs require stacking operations as to the logical line numbers and values of the control variables, and means of checking the values of the control variables must be provided.

In the EμPS environment declarations and Make/Update actions upon those WME's for the stacks and other control mechanisms are automatically generated by the compiler.

3. Preliminary Experiments

Two sample PS's, one for Room Assignment (of 43 rules+52 line MAIN Program compiled into 123 rules in all), and another for simple Job Shop Scheduling(of 59 rules+30 line MAIN Program compiled into 93 rules in all), have been implemented, and the Job Shop Scheduler has been in trial use.

In both of the sample PS's, it took only a day to write and another day to debug the main flow at the level of task-clusters. And the implementor used all the rest of the time working on the task-clusters.

4. Conclusion

The proposed programming paradigm EμPS provides a means of describing procedural flow at the level of cluster in a form entirely segregated from the rules.

Explicit and implicit inter-cluster GOTO's implementable by proviously proposed means are not for readability. And direct interactions with the internal mechanisms of the PS in a procedural environment tend to be detrimental to the beauty of the basic concept of PS.

EμPS provides an essential and practical solution for the problem, and it now allows the user to implement intricate PS's with great ease implementing independent task clusters and describing the flow at the level of clusters in an well structured procedural form.

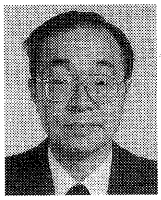
Recent advent of Rule Chip[10] forecasts a clear possibility of practical implementation of Rete Chip. A combination of EμPS and a Rete Chip would be an ideal environment for large, intricate, and speed conscious PS's.

Acknowledgements Dr. Shigeharu Yamada, and Genichiro Hattori, both exdirector of Research and Development Division of Toyo Communication Equipment Co. Ltd. gave the authors the opportunity to pursue the subject.

References

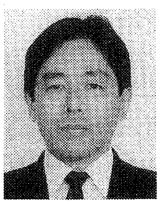
1) McDermott, J. C. and Forgy, C. L.: Production System Conflict Resolution Strategies, *Pattern Directed Inference Systems*, Waterman, D. A. and Hayes-Roth, F. eds., pp. 177-199, Academic Press (1978).
2) Forgy, C. L.: On the Efficient Implementation of Production Systems, Ph. D. Thesis, Carnegie-Mellon University (1979).
3) Davis, R.: Meta-Rules: Reasoning about Control, *Artif. Intell.*, Vol. 15, pp. 179-222 (1980).
4) Georgeff, M. P.: Procedural Control in Production Systems, *Artif. Intell.*, Vol. 18, pp. 175-201 (1982).
5) Forgy, C. L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp. 17-37 (1982).
6) Cruise, A. et al.: YES/L1: Integrating Rule-Based, Procedural, and Real-time Programming for Industrial Applications, *3rd IEEE CAIA*, pp. 134-139 (1987).
7) Matthews, B. et al.: Process Control with the G2 Real Time Expert System, *Proc. ACM IEA/AIE-88*, pp. 492-497 (June 1988).
8) Forgy, C. L.: The OPS83 User's Manual System version 3.0, Production Systems Technologies, Inc. (Apr. 1989).
9) Forgy, C. L.: The OPS83 Report System Version 3.0, Production Systems Technologies, Inc. (Apr. 1989).
10) Ueda, H. and Sugimoto, T.: A Coprocessor Dedicated for Inference Processing, *Turing Machine*, Vol. 3, No. 1, pp. 24-31 (1990) (in Japanese).
11) Ishida, T. et al.: Use of Procedural Programming Language for Controlling Production Systems, *7th IEEE CAIA*, pp. 71-75 (1991).

(Received July 20, 1992)
(Accepted December 3, 1992)



Akio Sasaki (Member)

Akio Sasaki was born in Hokkaido Japan in 1933, and received B. E., M. E., and Dr. Eng. degrees from E. E. Department of Hokkaido University Japan in 1955, 1957, and 1973. He also received Professional degree from E. E. Department of Columbia University N. Y. U. S. A. in 1965. He was with Toyo Cmmunication Equipment Co. Ltd. Tokyo from 1957 to 1963, with Electrotechnical Laboratory Tokyo from 1965 to 1970, and at MIT Mass. U. S. A. as a visiting research staff to the Project MAC from 1967 to 1968. He is now the president of Sasaki & Associates Inc., a software house in Shinjuku Tokyo. His research interests have been computer simulation, arithmetic schemes, artificial intelligence, and currently he is specially interested in automatic programming.

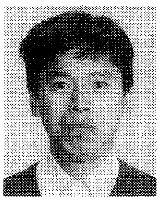


Shoichi Maki (Member)

Shoichi Maki was born in Hokkaido Japan in 1960, and received B. E. degree from Electrical Engineering Department of Hokkaido Institute of Technology in 1983. He joined Toyo Communication Equipment Co. Ltd. Tokyo in 1983. Since 1988, he has been with the R & D Division and has engaged in research and development in the area of systems software, artificial intelligence, automatic programming, and wireless communication. Among his research interests are operating system, parallel processing, automatic programming, data base system, and man-machine interface. Currently he is a supervising engineer at the Mobile Radio Communications Research Laboratory of the Division.

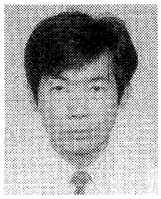
Shiryo Yasui (Member)

Shiryo Yasui was born in Hokkaido Japan in 1962, and received B. E. degree from Electrical Engineering Department of Tokyo University of Agriculture and Technology in 1985. He joined the R & D Division of Toyo Communication Equipment Co. Ltd. Tokyo in 1985, and since then he has engaged in research and development in the area of production system, CASE tool, work station for expert, and CAD system. Among his research interests are software tools, and CAD. Currently he is a supervising engineer at the Research Planning Department of the Division.



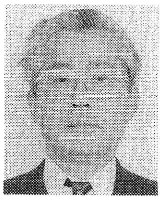
Hiroshi Ichise (Member)

Hiroshi Ichise was born in Nagano prefecture Japan in 1962. He received B. E., and M. E. degrees from Toyohashi University of Technology in 1985, and 1987. He joined the R & D Division of Toyo Communication Equipment Co. Ltd. Tokyo in 1987, and has engaged in research and development in the area of operating system, and language processor. Among his research interests are operating system, language processor, automatic programming, and object oriented programming environment. He is a member of IEICE and IEEE, etc.



Tomoyuki Minamiyama (Member)

Tomoyuki Minamiyama was born in Tochigi prefecture Japan in 1954, and received B. E., and M. E. degrees from the Electrical Engineering Department of Chuo University Tokyo in 1980, and 1982. He joined the R & D Division of Toyo Communication Equipment Co. Ltd. Tokyo in 1982 having engaged in research and development in those fields of signal processor, high level language processor, and production system. He was at MIT Mass. U. S. A. as a research staff from 1990 to 1992 pursuing learning model construction in the area of artificial intelligence. Of his special research interest is artificial intelligence, and he is interested in automatic programming as well. He is a member of AAAI and ACM.



Ichiroh Andoh (Member)

Ichiroh Andoh was born in Chiba prefecture Japan in 1934. He received B. E. degree from Electro-communications Department of University of Electro-communications Tokyo in 1957. He joined Toyo Communication Equipment Co. Ltd. Tokyo in 1957, and since then he has engaged in research and development in the area of hardware and operating systems for small computer. Currently he is an executive Director in charge of the R & D Division, and among his research interests are man-machine interface, artificial intelligence, agent system, and automatic programming. He is a member of IEICE, ACM, JSAI, and IEEE Computer Society.