

INTELLIGENT VISUALISATION OF INNER BEHAVIOUR FOR ENCODE

Hatala, M., Mach, M., Paralic, J. & Sabol, T.

Key words: configuration design, forward chaining systems, visualisation, debugging

1 INTRODUCTION

The ENCODE is the name of the joint research project (CIPA-CT94-0149) solved under the Copernicus program. At this symposium the project was introduced a year ago [1]. The aim of the project is to develop a methodology and a system operating on the PC platform facilitating creation of the wide range of knowledge based systems (applications) for solving configuration design problems. The ENCODE methodology will support three basic levels of system development [2]: conceptual, implementation, and execution. The ENCODE architecture follows the basic ideas of the VITAL system. The conceptual level is based on the OCML (Operational Conceptual Modelling Language) developed at The Open University. The implementation level is supported by various modules (e.g. forward chaining, backward chaining, constraint satisfaction, frames, etc.). The execution level will enable to represent the control structures and particular tasks in the target language (Common Lisp and CLIPS).

2 FORWARD CHAINING

Forward Chaining (FC) is considered to be one of the basic mechanisms for the implementation of the conceptual level tasks. The OCML includes language structures for the specification of FC resembling tasks which enable direct implementation in the Common Lisp target language. The second target language - CLIPS - is a rule based forward chaining system enhanced with functions and an object system.

The typical FC system uses rules consisting of two parts: antecedent and consequent. The system maintains a set of facts (working memory) which are true for the solved problem. The antecedent part of the rule is a condition testing the presence of facts in the working memory and relations between facts. If the antecedent part of the rule is satisfied the rule is 'activated' and it is placed into the agenda. In each cycle one of the activated rules in the agenda is selected and 'fired', i.e. its consequent part is executed. As a result the contents of the working memory is changed which causes deactivation of some rules and activation of some others. The process continues while either the goal is reached, system is halted, or there are no rules in the agenda.

3 VISUALISATION AND DEBUGGING OF FORWARD CHAINING SYSTEMS

The large amount of activated rules in agenda and the selection strategy of the rule to fire make the operation of such a system very difficult to trace. The debugging of the set of hundreds rules in hundreds of cycles is not possible by classic

techniques. A very useful technique for visualisation of execution of FC systems called Viz has been described for the

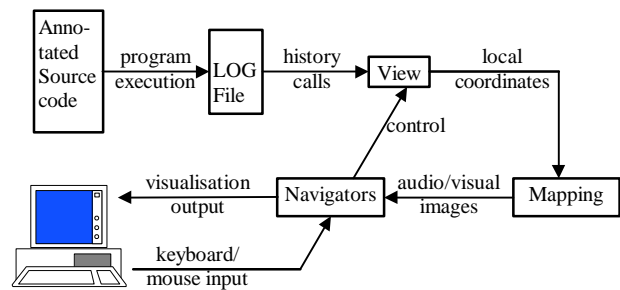


Figure 1

first time in [3]. Figure 1 shows the scheme of the Viz.

The visualisation module for FC systems based on the Viz methodology has been implemented within the ENCODE project. However, the independent visualisation component enables to visualise any FC system. The only requirements are that the FC system writes the log file of its execution and an external filter converting the log file into the component's internal structures is provided (in Common Lisp). The current implementation contains filters for the CLIPS and OCML FC component.

4 VISUALISATION MODULE CHARACTERISTICS

In the Viz the system writes a log file and this is replayed and visualised to the user. To visualise execution of the CLIPS we have modified its source code of the system and added to commands for switching the logging on and off. The modification is described in the [4].

The log file is read into the visualisation module through the filter. The execution is then visualised at two levels of detail. The TRI table shown in Figure 2 has been used for the visualisation of the execution at the coarse level of detail. The left pane of the table contains names of the rules, functions and groups. The rules could be organised into the groups which are displayed instead of the rules. This enables to reduce the amount of information displayed.

The right pane is an array of graphical symbols. Each column represents a cycle, symbols in the column represent states of rules in this cycle. Three different symbols appear in the array and represent states: passive (rule in the cycle is not activated), active, and fired. Symbols are coloured and are easily distinguishable by the user. Each symbol and rule name are

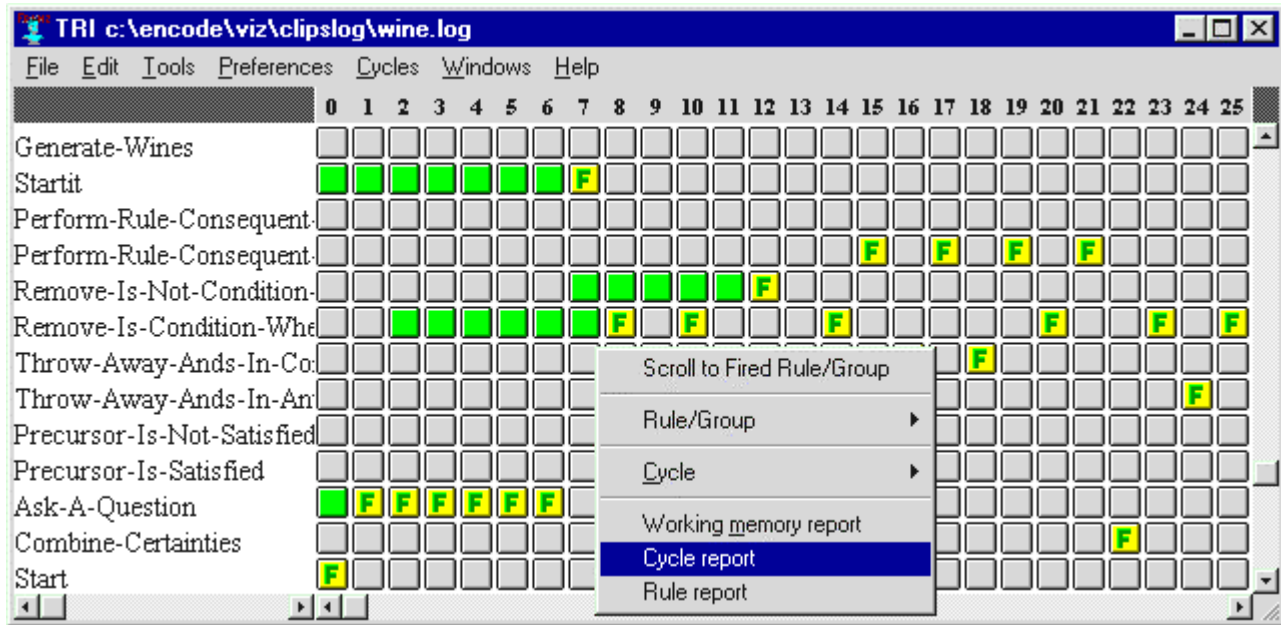


Figure 2

mouse sensitive and a popup menu is associated with them. Through the menus the user can set up the visualisation module parameters and display the information about the execution in the form of reports. Figure 1 shows the popup menu for the symbol from the right pane.

The report is a textual window displaying detailed information about a particular object or an event. Three types of report are implemented: rule report, cycle report, and working memory report. The contents of each report is highly customisable. The working memory report displays the contents of the working memory (facts) in the cycle and changes of the working memory (retracted and asserted facts). The cycle report (Figure 3) displays the information about the rules in the cycle: rules in the agenda, fired rule, deactivated and activated rules. It is possible to bind displaying the working memory report together with the cycle report. The rule report contains all information about the rule: a definition, statistics when the rule was activated, deactivated and fired, and antecedents with which the rule was fired.

5 CONCLUSIONS AND FUTURE WORK

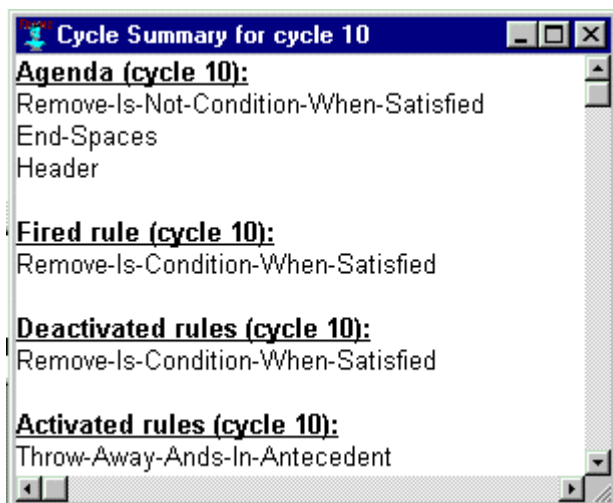


Figure 3

The visualisation module for execution of FC systems has been implemented as an independent module operating along with the ENCODE workbench and with the CLIPS system. Its use with other FC systems is possible and requires to make minor changes in the FC system to generate log file and writing the filter for visualisation module.

Some FC systems enable to use backward chaining (BC, goal directed) rules in the program. Because the ENCODE system will provide the BC mechanism we are working on the module within our visualisation system enabling to visualise a BC rules execution. The new module will be highly interactive execution graph.

6 REFERENCES

- [1] Sabol, T., Mach, M., Hatala M., Paralic J.: Knowledge based design in CLIPS. Proc. of 6th Int. DAAAM Symposium „Intelligent Manufacturing Systems”, p.293-294, Krakow, Poland, October 1995.
- [2] Newell, A.: The Knowledge Level, Artificial Intelligence, Vol. 18, p.87-127, 1982.
- [3] Domingue, J., Price, B., Eisenstadt, M.: Viz: a framework for describing and implementing software visualisation systems. In: D.Gilmore and R.Winder (Eds.) *User Centered Requirements for Software Engineering Environments*, Springer-Verlag, 1992.
- [4] Hatala, M.: CLIPS modification supporting software visualisation, TR-Encode-TUKE-1-95 (ENCODE Project Report), Technical University, Kosice, 1995.

AUTHORS

Marek HATALA (hatala@ccsun.tuke.sk)
 Dr. Marian MACH (machm@ccsun.tuke.sk)
 Jan PARALIC (paralic@ccsun.tuke.sk)
 Dr. Tomas SABOL (sabol@ccsun.tuke.sk)
 Department of Cybernetics and Artificial Intelligence, Technical University, Letna 9/B, 041 20 Kosice, SLOVAKIA,
<http://www.tuke.sk/tu/fei/kkui/research/aig/encode/encode.html>