Intellectual Performance using Dynamical Expert Knowledge in Seismic Environment

Vadim Stefanuk
Institute for Information Transmission Problems,
Russian Academy of Science,
Bolshoy Karetny per. 19
127051 Moscow, Russia
stefanuk@iitp.ru

Abstract—The paper introduces the notion of Dynamic Expert System. It shows that by restarting a common (static) expert system periodically it is possible to cope with dynamic environments. This quasi-static approach to dynamics is suitable if the environment is changing slowly enough in comparison with the inference engine operation and the user reaction time. Implementation of this scheme in "pure" Expert System shell meets no difficulties. However in practice some problems may occur due to the side effects in rules and attached procedures. These problems and their relation to classical AI issues are considered in details. The designed system has been applied to the task of seismology forecast, which contains the dynamical factors of both numerical and heuristic nature.

The resulting dynamic expert system never stops, occasionally interrogating the user when it suspects that some of the previously entered data are obsolete. In this sense the computer system behaves as an “alive creature.”

I. Introduction

In eighties there was a certain interest to the reactive planning research. This interest also was supported by the general feeling of dissatisfaction with classical planners when facing with dynamic, time dependent reality [1].

In several application domains for advanced diagnosis strategies the same property hold as for reactive planning, though it is not the same problem [2]. An example application in [2] is the intention recognition in cooperative help to recognize the user's goal and intentions.

Anytime property of a real-time planning system was introduced to mean that the system can make a quick, dumb solution if that requested, but still is able to reason intelligently if given more time. The ERE planner [3] has the anytime property.

There was also research in real-time expert systems [3]. Real-time expert system must often work in limited time and adapt to a changing environment, its input being usually a sensor data.

Several researchers noted that "real-time" traditionally is taken to imply "need to be fast". However in reactive planning or real-time expert systems it is not necessarily so. In a number of applications the main issue is that the time for planning or decision making is limited, but the limit might not necessarily be tight.

The problems of dynamics are being overcome in various ways. Thus, in reactive planning the execution system can be guided by a highly adaptable plan that either hand-coded or generated at compiler time [1].

The notion of reactive diagnosis to denote diagnosis problems where time limitation and adaptation to changes are important factors was introduced in [2]. In conclusion to the paper its author says in particular: "A more intricate problem in the notion of adaptiveness, how can we expect to reuse results from earlier proofs when the circumstances change? The author plans to investigate these questions for the particular application domain of intelligent help".

Present paper addresses similar questions in a straightforward manner through the notion of Dynamic Expert Systems. The goals of our paper are:

- to propose a new architecture for the reactive or Dynamic Expert System;
- to study the problems of combination of time-dependence and fuzziness in case of changing environment;
- to consider certain classical AI issues such as the frame problem, side-effects and fuzzy inference in the way they appear in our version of Dynamic Expert Systems;
- to show an example application of the proposed architecture in a real problem domain.

Common Expert System shells are usually oriented to problems for which the Knowledge Base (KB) and Data Base (DB) items, after they entered by the Knowledge Engineer, do not subject to changes during the whole session of interaction with the user. Though the user may occasionally modify DB items, the different sessions are essentially independent. Such a system may be called *static expert system* as its knowledge base is a fixed repository of information.

Yet, in a number of real applications DB and KB may considerably vary within one session of user interaction with ES. For example, during one session of the earthquake forecast in a certain area some new evidences may arrive and some old evidences may become obsolete, both processes influencing which rules in KB are to be relevant.

Below we will discuss the proposed architecture and the problem domain, which was used to demonstrate its advantages. As our quasi-static system is built upon a traditional static expert system, in the following paragraph we will described briefly the static expert system which was used as the basis for the dynamical architecture.

II. Static Expert System ZNATOK

Our static expert system shell ZNATOK resembles the shell described in [5]. It is made as a cardboard rule-based system, which uses attributes for storing data and a simple stack arrangement as the inference engine.

A typical "pure" rule is shown in Fig. 1.

```
IF (AND (A1:=V1) (A2:=V2) ... (An:=Vn)) THEN (Am:=Vm)
```

Fig. 1. The format of a “pure” rule

```
((|subgoal processed| |yes|
                (with-window *instruction-window*
                (send *instruction-window* :clear-screen)
                (send *query-window* :clear-screen)
                (send (histo-window1) :clear-screen)
                (send (histo-window2) :clear-screen)
                (FORMAT (histo-window1) "*** Middle-term forecast~% the
                  week of  earthquake:~%")
                (histo |the week of earthquake| (histo-window1))
                (FORMAT (histo-window2) "*** Short-term forecast~% the
                  hour of  earthquake:~%")
                (histo |the hour of earthquake| (histo-window2))
                (wait-key)
                (send *query-window* :clear-screen)
                (FORMAT *query-window* "*** Expected power: ~A")
                (wait-key)))
  (|subgoal chosen| |monthly probability and power forecast|)
  (|total time estimation| |yes|)
  (|power estimation| |yes|)
                                        )
```

Fig. 2. A rule with attached procedures

However the system ZNATOK acquires its efficiency and usefulness trough an extensive usage of various attached procedures (see example in Fig. 2).

Besides these traditional attached procedures like those shown in Figure 2 in the ZNATOK Expert System Shell, there is a number of so-called *control procedures*, influencing the performance the inference engine itself. This feature provides the necessary flexibility to the static shell ZNATOK that by now was used in a number of applications in medicine, tutoring, civil engineering, project evaluation [6], [7] and etc..

III. Treatment of Fuzzy Values in ZNATOK

Before going further note that ZNATOK was able to cope with fuzziness.

The fuzzy concepts, met in the assertions (rules) and in the data, are treated in the system ZNATOK in a traditional way.

The main idea is to accumulate the evidences in favor of a certain hypothetical fact in the process of functioning of Inference Engine of the ES in order to decide what hypothesis is to be preferred.

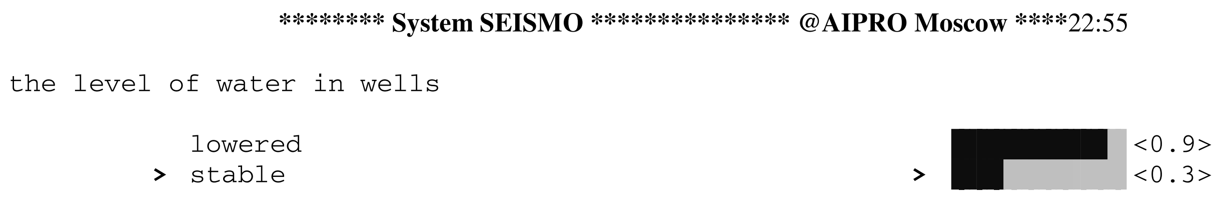
In the paper [8] an axiomatic approach was proposed to create a regular way to produce formulae for combining evidences under various formal constrains. This approach found a support in [9] where it was somewhat emphasized. We will not go into details here pointing only that the combination formulae used in ZNATOK reminds the one used in MYCIN.

Many examples might be found in the area of earthquake forecast for long, middle and short term predictions.

The next rule (see Fig. 3) shows how the attribute "the hour of earthquake " receives a fuzzy value. Besides that, a fuzzy value for an attribute may be directly entered by the user in the way demonstrated in Fig. 4.

```
((|the hour of earthquake|              (:F   (|in 18 hours| 0.0)
                                              (|in 20 hours| 0.7)
                                              (|in 22 hours| 0.8)
                                              (|in 24 hours| 0.9)
                                              (|in 26 hours| 0.8)
                                              (|in 28 hours| 0.7)
                                              (|in 30 hours| 0.0)
                                              )
                                                           )
(|the earthquake will be in a day| |yes|))
```

Fig. 3. A rule, which assigns fuzzy values



```
******** System SEISMO *************** @AIPRO Moscow ****22:55

the level of water in wells

           lowered                                          <0.9>
         > stable                                        >  <0.3>
```

Fig. 4. A fuzzy value might be directly entered by the user

IV. Quasi-static System Architecture

After some modification the described static architecture of ZNATOK type may be used in a quasi-static mode. This new mode results from a repeating restart of the static shell within one session of user interaction with the ES. During each restart cycle virtually all the data used (and all the rules involved) may be reconsidered, if of course some grounds for such reconsideration are found in the ES Knowledge Base.

It appears the proposed quasi-static schema resolves the classical Frame Problem: how to remove all the consequences of a datum if the datum is not valid any more. At least it is true for a "pure" ES shell (an analog of the "pure Lisp") with the “pure” rules of Figure 1, in which side-effects are absent (see however a figure below).

In case of a dynamic environment the attached procedures cause major problems due to possible side-effects related to them.

V. Side-effects Handling

Side-effects in a computer system may be subdivided into two classes: external and internal ones.

The notion of internal to the system side-effects is slightly more sophisticated and it was studied in [10] in connection with the recursion removal problem. The internal side-effects are related to the evaluation of attributes, rules, procedures and other forms that get their values in the normal run of the system [7].

One might find it useful to consider *relative side-effects*, relevant to the pairs of expressions (or forms) evaluated sequentially [10]. These are special cases of side-effects. We say that the form F1 produces a relative side-effect on the form if and only if the evaluation of the form F1 may change the result of evaluation of F2.

When a ES shell of ZNATOK is being applied to a concrete static domain the Knowledge Engineer has to be careful using the side-effects to achieve a desired behavior of the system. This task is not simple and requires from the KE a deep knowledge of the system architecture and the problem domain.

This task becomes even more complex when the system is used in the quasi-static mode. In the quasi-static mode the number of possible side-effects are doubled. For each relative side-effect that the form A1 produces on the form A2 in the static case, in the quasi-static case one has to consider in addition a possibility of relative side-effect which the form A2 produces on A1 at the next restart cycle .

From the other side, used properly side-effects present a very powerful way to achieve flexibility and efficiency. Some hints for this may be found in [10].

One of the important goal of Knowledge Engineer is to achieve an *operational consistency* by avoiding undesirable side-effects. A practical way to check the operational consistency is to run the dynamic shell in a number of static environments. In a static environment the quasi-static ES should behave exactly as the original static ES.

It is obvious from the description of quasi-static system architecture that the Inference Engine must be a deterministic one. Otherwise the operational consistency is unobtainable.

To achieve both operational consistency and efficiency it is necessary to introduce types for the different attributes used in the ES. For this purpose there three types DYNAMIC, STATIC and FUZZY are used in the system. The DYNAMIC type serves to limit the number of attributes reconsidered from cycle to cycle. FUZZY type saves efforts in evaluation of fuzzy variables by distinguishing fuzzy and ordinary variables.

VI. Seismology Data

In the present paper we are not considering the very tough problem of direct knowledge acquisition, i.e. the direct transfer of the knowledge of experts on seismology to the ES Knowledge Base. Thus, we avoid consideration of the problem of special "logic of time and space" [11] used by many experts. Both the time and the space are treated as regular physical phenomena in our system.

However, it turned out to be impossible to avoid the use of fuzzy concepts as they constitute the basis for almost all the assertions of Experts concerning the earthquake forecast. In particular, the fuzziness permits a seismologist to express a certain imprecision in measuring time intervals.

The earthquake forecast involves both observations based on the laws of physics and the observations of a heuristic nature. The law of linear time accumulation of the tension in rocks may be taken as an example of purely physical phenomenon. It may be easily taken into account by a corresponding attached procedure for computation of the tension, which gradually accumulates with time.

Here is a typical heuristic observation that may be found in the relevant literature: "As a rule, on the eve of a strong earthquake a predecessor occurs consisting in a local displacement. Yet on the eve of an average earthquake it is the whole area that is displaced."

From this and many similar examples found in the seismic literature it follows that the fuzziness and time dependence are intrinsic properties of the problem area.

VII. Implementation and Results

The quasi-static ES shell ZNATOK 2.0 was implemented in Common Lisp. It is very compact and runs practically on any PC.

The study of relevant literature on seismology and consultations with practitioners in the domain let us outline the most essential requirements for a working prototype of ES for earthquake forecast domain. Actually it is this application that has led us to the development of the quasi-static approach to the reactive expert systems. The quasi-static approach is quite applicable here as the process leading to an earth quake develops very slowly in comparison to the rate of performance of the inference engine of the Expert System.

The main menu to which the user may address many times (during one session) to choose a problem of interest related to the earthquake forecast is demonstrated in Fig. 5.

********** System SEISMO *************** @AIPRO Moscow ******22:51

```
     Choose subgoal:

  >  estimation of the time of a strong earthquake
     estimation of strength of expected earthquake
     forecast of probability and the power
     monitoring
     possibility of a forecast
     the end of the session (exit)
```

Fig. 5. Starting menu of the seismic forecast system

The result given by the Expert System may be presented in a pseudo-graphical form (see Fig. 6), which is a convenient way to follow changes in the seismic situation in the seismic observation site.

In practice this dynamic expert system never stops. It behaves as an *alive creature*. Automatically it shows new results of forecast in accordance with its internal calendar. Occasionally it asks the user about some new data to replace the obsolete ones from the system point of view. Or else, the user (or a sensor, in the monitoring mode) may interrupt the system and the user may click a new entry in the main menu (see Figure 5).

VIII. Conclusion

A practical way to make a reactive (dynamic) ES is proposed and its limitations and problems are studied. The quasi-static shell differs from conventional static one in the possibility of a frequent restart provided that the static data accumulated in DB from cycle to cycle are remembered. The architecture was

********** System SEISMO *************** @AIPRO Moscow ****23:15**

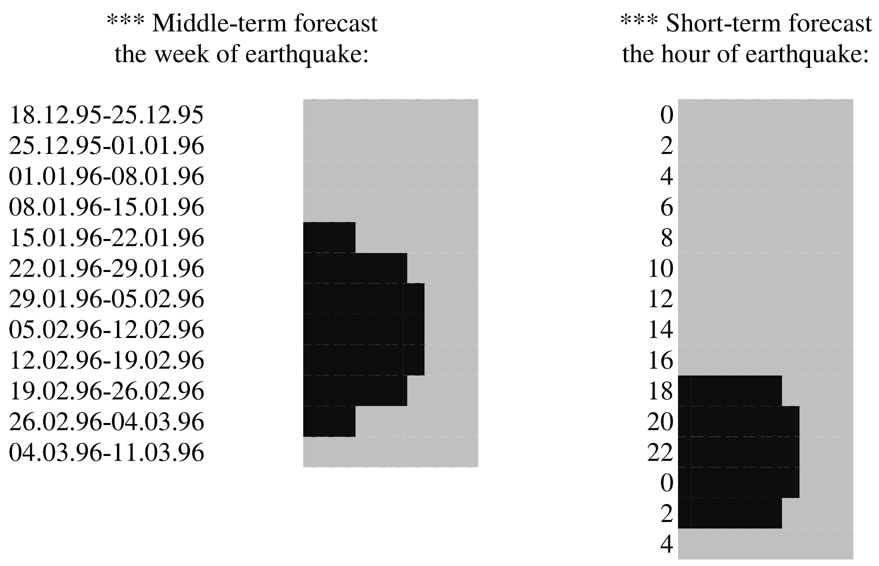


Fig. 6. A copy of screen output: the time of a strong earthquake

implemented in Common Lisp. The earthquake forecast problem was used as a case study [12, 13].

In the described application only data base was subject to changes. However in some other applications the proposed quasi-static approach is expected to be also useful (see [14]). In this respect some future experiments might be illuminating [15].

Obviously, the proposed architecture will be at a loss when the environmental changes are too fast in comparison to the "proper times" of both inference engine and the user reaction [16]. Of course, it is the inference engine throughput that puts a practical limit to the usefulness of the proposed quasi-static approach. Yet our experience showed that the large human reaction time might cause some problems as well [16].

The Dynamic Expert System SEISMO was made for the use in Earth Physics Institute of Russian Academy of Sciences.

ACKNOWLEDGEMENT

The author would like to thank his colleagues, especially Alexander Zhozhikashvili, for many valuable discussions of the basic principles described above. In fact the Dynamic Expert System *SEISMO* [13] was implemented jointly with Alexander Zhozhikashvili.

The research was partially supported by the Russian Fond for Basic Research and by Fundamental Research Program of the Presidium of Russian Academy of Science.

REFERENCES

[1] Shoppers, M., "Universal plans for reactive robots in unpredictable environments," *Proc. of IJCAI-87*, 1987.
[2] Wern, A., "Reactive Abduction," *Proc. of ECAI 92*, pp.159-163, 1992.
[3] Drummond, M., "Situated control rules," *Proc. of KR-89*, 1989.
[4] Laffey, T. et al., "Real-time knowledge-based system," *AI Magazine*, 9(1), pp. 27-45, 1988.
[5] Thomson, B. and Thomson, W., "Inside an Expert System," *Byte*, April 1987.
[6] Stefanuk V., "The design of a dialogue in the Expert System for a Pandemia," *Proceedings of the all Union school and workshop "Problems of Expert System design"*, Part 2. Moscow, USSR Academy of Sciences, pp.255-256, 1988, (in Russian)
[7] Zhozhikashvili, A. and Stefanuk,V. "Programmable expert system shell ZNATOK and the problems of its description in the theory of categories," *Izvestiya AN SSSR. Technicheskaya kibernetika*, No. 5. - pp.134-147, 1990 (in Russian).
[8] Stefanuk, V., "Some aspects of Expert System theory, " *Soviet J. Comp. Sci. (Tech. Kibernet.)*, No.2. - pp. 85-91, 1987.
[9] Johnson, N. and Kotz, S., "Axiomatic approach to formulas for combining likelihoods or evidence," *Journal of Statistical Computation and Simulation*, No. 31, pp.49-54, 1989.
[10] Stefanuk, V., "The use of a graph representation in optimization of variable replacement in LISP in the presence of side-effects," *Machine Intelligence*, No. 9, New York, John Wiley & Sons, pp. 27-40, 1979.
[11] Kandrashina, E., "Representation of temporal knowledge," *Proc. of IJCAI-83*, pp. 346-348, 1983.
[12] Stefanuk, V.L. "Dynamic expert systems," *KYBERNETES, The Interna-tional Journal of Systems & Cybernetics.* V.29, Issue 5, pp. 702-709, MCB University Press: 2000.
[13] Stefanuk V.L. "The Behavior of Quasistatic Shell in Changing Fuzzy Environment," *The Proceedings of the IY National Artificial Intelligence Conference KII'94*, Rybinsk, 1994, V.1, P.199-203 (in Russian).
[14] Perlovsky L.I. *Neural networks and intellect: Using model based concepts*, Oxford University Press, New York (2001), 469 P.
[15] Stefanuk V.L. "Problems of Performance Measurement in Locally-Organized Systems," *International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (KIMAS'03: Modeling, Exploration, and Engineering", September 30 - October 4, 2003), pp. 432-437.
[16] Stefanuk V.L. *Local Organization of Intellectual Systems. Models and Applications.* Moscow: Fizmatlit, 2004, 349 p. (in Russian).