AN INTELLIGENT ENVIRONMENT FOR MECHANICAL ENGINEERING DESIGN - APPLICATION TO GEAR TRANSMISSIONS

George DOBRE, Bruno PICASSO, Gheorghe RADULESCU, Dorin CARSTOIU, Cosmin GRIGORESCU
Department of Machine Elements and Tribology and Department of Computers and Industrial Informatics, University "POLITEHNICA" of Bucharest, Spl. Independentei 313, 77206 Bucharest, Romania
Department of Mechanical Engineering, University of Cagliary, Piazza d'Armi, 09123 Cagliary, Italy
Email:geo@meca.omtr.pub.ro, cosming@aii.pub.ro, picasso@zot.unica.it

Abstract

This paper presents a description of an Intelligent Design Environment (IDE) based on AI tools. Fundamentally an IDE is based on the coexistence of algorithmic procedures, of common usage in Mechanical Engineering Design methods, and knowledge based systems where non algorithmic knowledge is dealt with. This synthesis leads to the development of a powerful design environment where the two types of knowledge mutually enhance their capabilities. Starting from the results obtained during an EC funded research programme, aimed at the optimisation of gear transmissions using AI tools, the paper presents structures and particularities of an IDE that can be used in any domain of mechanical engineering design. The application to gear transmission design is then described in detail including the data acquisition module, the inference module, reasoning strategy and user interface. The system capabilities are critically discussed, particularly by the controversial point of view of the utility of AI methods and tools in Mechanical Engineering Design.

1. Introduction

Human activities imply using both quantitative and qualitative information. The implementation by computer programs of this quantitative and qualitative information implies using both AI techniques and algorithmic (procedural) programs.

Generally the design domain, particularly the mechanical engineering design, involves both the existence of *empirical (non-formal) knowledge* and *formal knowledge*. From our point of view an IDE is a complex computer program based on AI techniques that can make decisions and has facilities to use procedural programs. Three paradygms exist to model the decisional act in AI: Decision Support Systems, Executive Information Systems and Expert Systems [5]. All these systems use *knowledge* as fundamental information, and are called *knowledge based systems*.

An Expert System is the most complete model that can emulate human expert activity. Thus the most appropriate solution for modeling the human decision (including engineering design) process is an ES with procedural program call facilities. This paper outlines a formal approach and the practical implementation of an IDE applied to gear design which can be easily extended to any engineering design domain.

2. Modeling the design process

Many models of the design process are present in the literature. In [3] an extensive review of historical and modern models is presented. One of the most general models is shown in fig. 1.

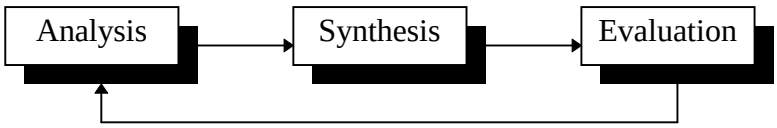

Figure 1: Analysis-synthesis-evaluation model.

In fig. 1 the analysis box represents the definition of design goals and constraints. As all designers know, this set of initial data can be incomplete, its full completion being reached only after a certain number of iterations of the Analysis-synthesis-evaluation cycle. The synthesis box represents all design methods and procedures which lead to the solution (better to multiple solutions) of the design problem. The last step represents the activity of evaluation and rating of design solutions to reach the final design. The analysis-synthesis-evaluation process is iterated until a satisfactory solution is found for

the design problem. What is meant for "satisfactory" is of course problem and context dependent.

A more analytical description of the design process maps the space of design variables (decision space) to the space of performance variables. The decision space is constrained by known conditions acting on design variables (e.g. the maximum number of teeth is limited by the need of avoiding interference).

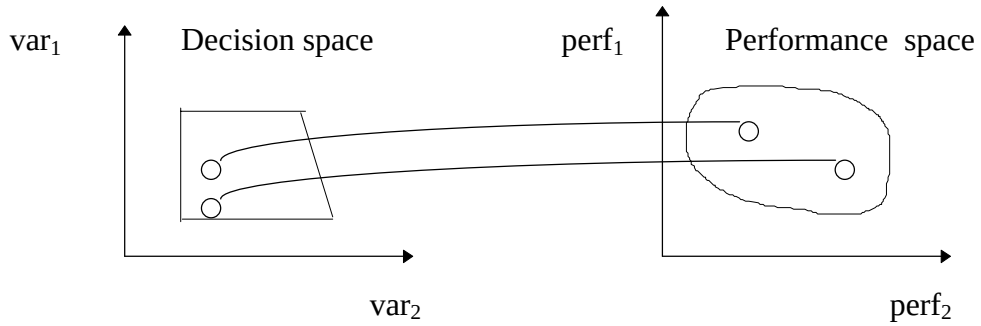

Figure 2: Mapping of the decision space to performance space.

If scanning the whole design space or hyper-space a corresponding region is obtained in the performance space where the different design solutions can be rated and classified using a merit function. This process leads to an optimum design according to the merit function chosen. In all the steps described design knowledge has been implicitly used in all design methods and rules and in performance evaluation. To build a knowledge based design system the knowledge must be made explicit by considering for example that the functions which map the decision space to the performance space are not always algorithmic. Furthermore the inference have to be separate from knowledge and construct a knowledge body, made of different kinds of knowledge which is used and modified by the inference engine. Some examples :

Algorithmic: *Primitive radius and gear mass are linked by a known formula.*
Non-algorithmic: *Plastic materials can be used only for limited power gears.*

The idea of scanning the entire design space to find the best design solution is in most cases unrealistic. Only in computer oriented optimization with a limited number of design variables this process is feasible. In a real design problem the designer uses syntactic knowledge about the domain to go from the initial design requirements to a feasible design solution. This process is mainly deductive but it has been demonstrated that a purely deductive action cannot lead to a good solution of the design problem. The so called systematic design tries to break down the design goals in a manifold of elementary sub-goals. This is equivalent to solve many elementary problems instead of a complex one. However the design activity cannot lack two kinds of logic reasoning, induction i.e. the capability of extracting general rules from known facts, and

abduction which could be defined as a reversal of deduction , going from one fact to another fact by analogy. These concepts are clarified by the following examples:

Deduction: *If the dimensions of the pinion are increased its mass will be higher.*
Induction: *Automotive gear reducers are generally flame or induction hardened.*
Abduction: *Plastic materials are good for gear construction.*

The last statement comes from:

Nylon is good for small gear construction.
Nylon is a plastic material.
Then plastic materials are good for gear construction.

In traditional logic this operation is not permitted, however it is generally accepted that designers use abduction quite often. In plain terms we could say that deduction refers mainly to mathematical and logical reasoning, induction to experience, abduction to inspiration and analogy. A knowledge based model of design could be graphically represented by fig. 3.

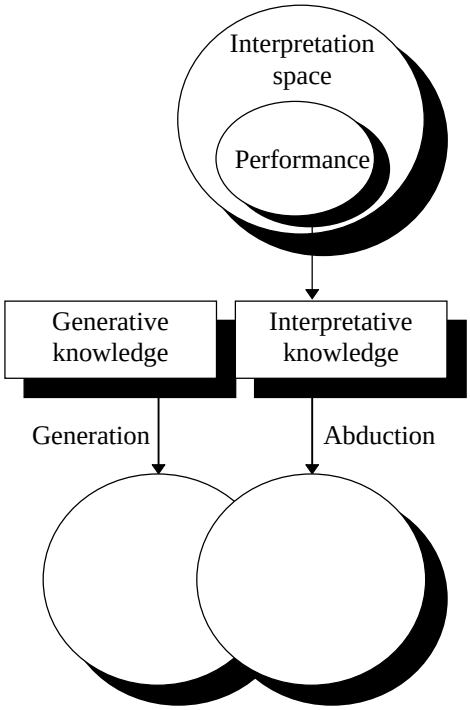

Figure 3: Knowledge based model of design.

In this model a generative knowledge is used, in the discussed case all the methods and procedures which lead to a "syntactically correct" gear dimensioning. The design space is increased through the use of abduction. Again interpretation knowledge is the key to map the design space to the performance space. In the ES which is been described in the sequel only forward chaining inference has been implemented while induction and abduction reasoning are planned for the next version of the design system.

3. IDE concepts

This section contains the basic features of the ES module that is the main component of an IDE. The specific functions of ES for engineering design are also shown.

The implementation of an ES involves the following requirements:

1. The structure of the ES must be consistent with the organization of the knowledge base. From this requirement follows that a deep investigation on the domain of application is needed before the domain knowledge could be formalised.
2. The ES must be able to perform any sort of manipulation on the knowledge atoms, including addition, deletion, data modification and change of the data structure.

About the **structure of ES** multiple data representation types can be found, every kind having advantages and disadvantages. Alternatives for data representation in conjunction with knowledge are shown in the table 1.

Table 1: Knowledge type and representation (from [5])

Knowledge type	**Representation**
Procedural knowledge	Rules Strategies Procedures
Declarative knowledge	Concepts Objects Facts
Meta-knowledge	Knowledge about knowledge or knowledge using knowledge
Euristic knowledge	Experience based knowledge
Structural knowledge	Set of rules Concepts relationships Relations between concepts and objects

For the ES the following representation modes were chosen:

- facts (by an *Object-Attribute-Value* form or variable form);
- rules (simplified IF-THEN form);
- procedures (algorithmic form)

This option is justified by natural data representation (close to human data representation) and by the nature of the design process.
For the **facts** the *object-attribute-value* (OAV) form and a representation form were used. Both form of data representation (OAV and variables) are recommended for the their easy understanding and the capacity to encapsulate all knowledge types (formal and non-formal). Variables belonging to factual representations can have different types and formats, namely numerical, symbolic, logical etc..

For OAV, a human knowledge like a proposition: "*wheel material class is steels*" can be represented as in fig. 4.

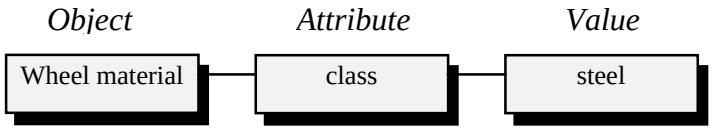

Figure 4: Object-attribute-value data representation.

The attribute can have multiple values (fig. 5).

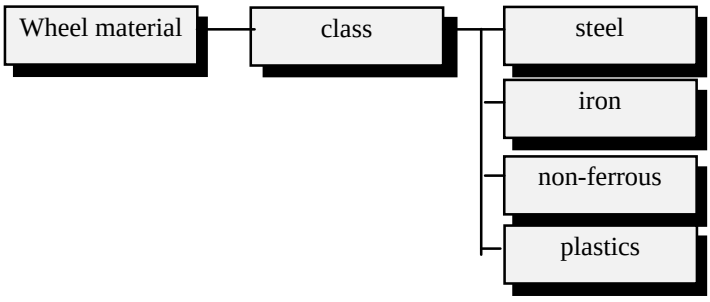

Figure 5: Object-attribute-value with multiple attributes.

Rules manipulates facts using the *IF/THEN* form. The facts can be concatenated with logic connectives like OR, AND and can have the negated form NOT (see, for example, the rule 1 specific for choosing wheel gear materials).

Rule 1
IF
Material class is steels AND
Material subclass of order 1 for steels is surface hardened steels
AND
Material subclass of order 2 for surface hardened steels is case hardened alloy steels
AND
[diameter] < 600 mm AND
[module] < 20 mm
THEN
Symbol is 16 Mn Cr 5 (DIN) OR
Symbol is 527 M 19 (BS) OR
Symbol is 16 MC 5 (NF).

Procedures are algorithmic implementations of mathematical computations that can interact with the ES. Every procedure has input data and give results. The structure of a rule is modified to support data exchange and procedural program call. Example:

Rule 2
IF
[rotational speed] > 10 rot/min AND
[bearing load] is known
THEN
CALL bearing durability ([rotational speed], [bearing load])

In this case the results given by the procedure *bearing durability* are stored and used for further inferences.

4. IDE structure

The IDE will have the structure [2] presented below.
The modules have the following functions:

- *Knowledge Acquisition Module* take data from user in an appropriate form (OAV or variables). This data will be stored to *Knowledge Database*;
- *Knowledge Database* stores the data for further use. This module contains all the system knowledge. The data is static (meaning that can be modified only by *Knowledge Acquisition Module* and not in a dynamic way by the inference process);
- *Context* is the part of the system that stores the knowledge pieces involved in the inference process. This system keeps only facts that are true, the other facts are assumed to be false;

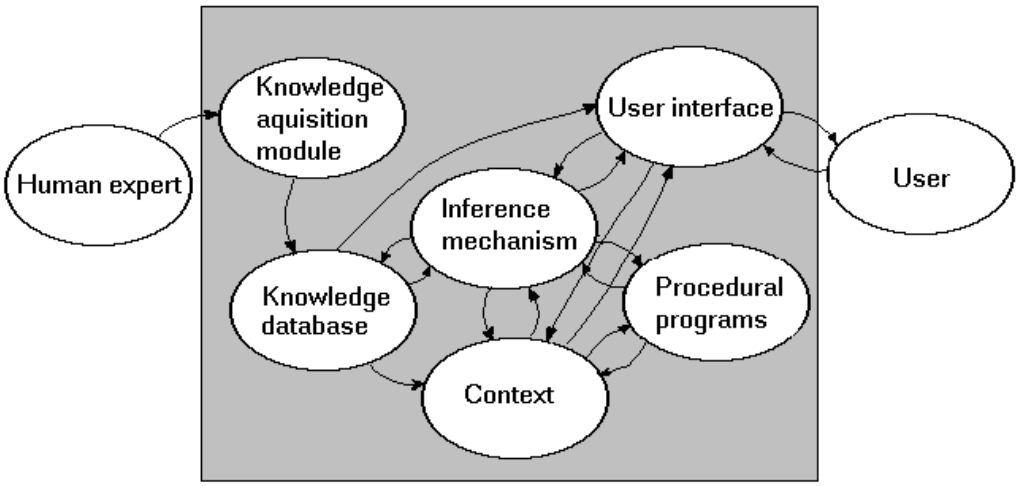

Figure 6: IDE general structure.

- *The Inference Mechanism* is the engine of the system. Data is retrieved from the context and rules are chained using a forward chaining technique;
- *Procedural Programs* are algorithmic programs. A production rule has the possibility to call a procedure and to retrieve results given by procedure;
- *User Interface* is the module that handles user interaction with the IDE.

The possibility of executing a procedural program while a rule is firing is implemented via the mechanism shown in fig. 7. The inference mechanism is responsive for triggering the appropriate procedural program and this program puts its output data (values) in the context in *variable* form data representation.

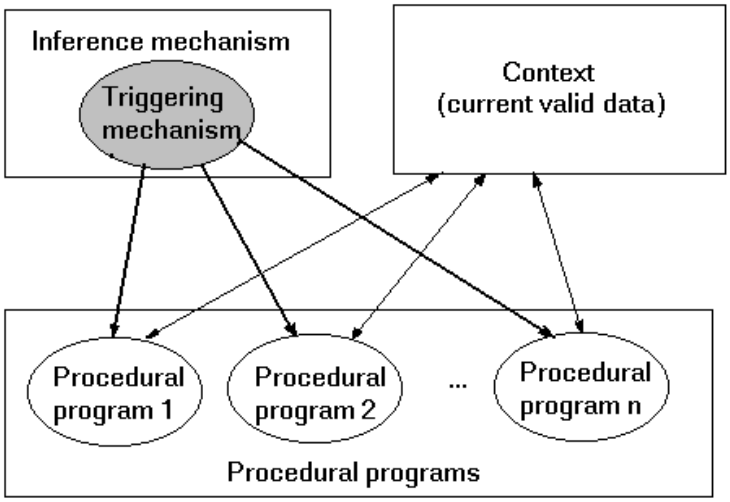

Figure 7: ES with the capacity of calling procedural programs.

From the IDE designer's point of view, the system can be *logically split* in two main phases:

- a phase dedicated to data acquisition and storage
- a phase in which the inference process is running.

5. Rule editor

5.1. General features

For the implementation a high level object oriented language was chosen (Borland DELPHI). This language has special peculiarities compared to other languages (like Visual C++, AI tools: PROLOG, LISP):

1. built-in capabilities to database access;
2. very good support for user interface implementation;
3. short time used for developing the application.

These advantages were the main reason for this choice. The project started with version DELPHI 1.0 under Windows 3.1 but was later migrated to DELPHI 2.0 under Windows 95. In this way, more capabilities were exploited like improved interface, improved running speed, registering application, easy install and uninstall, etc.

In this implementation, the editor handles the human expert data acquisition by a special interface (see fig. 8, 9 and 10) . The human expert or knowledge engineer inputs by this editor:

1. facts as atomic data representation (in both OAV representation and variable representation);
2. rules for combining facts.

5.2 Fact editor

On fig. 8 the facts editor display is shown.

Regarding screen controls of fact editor, the following features appear:

1. in the edit controls the current selected fact is displayed. The OA part is called *Subject of fact* and V is called *Attribute.* In this way, theoretical aspects were hidden. Every subject of the fact can have at most 5 attributes;
2. a table was implemented to display a global view of OAV facts. The facts are automatically alphabetically ordered for easy interaction with the system. This property is useful when the count of facts is large;
3. the user can do an incremental search (using the *Search...* button) for a word in the whole knowledge database fact. For this another window for

typing the word provided with a *Find next* button appears (this window is like almost every text editor find window and is not shown here);
4. a *Help* button assures extra explanation for using the interface. Also, when the mouse is sitting on a control more than 3 seconds, a brief explanation appears;
5. the navigation (and also *new*, *delete*, *edit* operations) for the facts knowledge database is done using a compact bar of buttons.

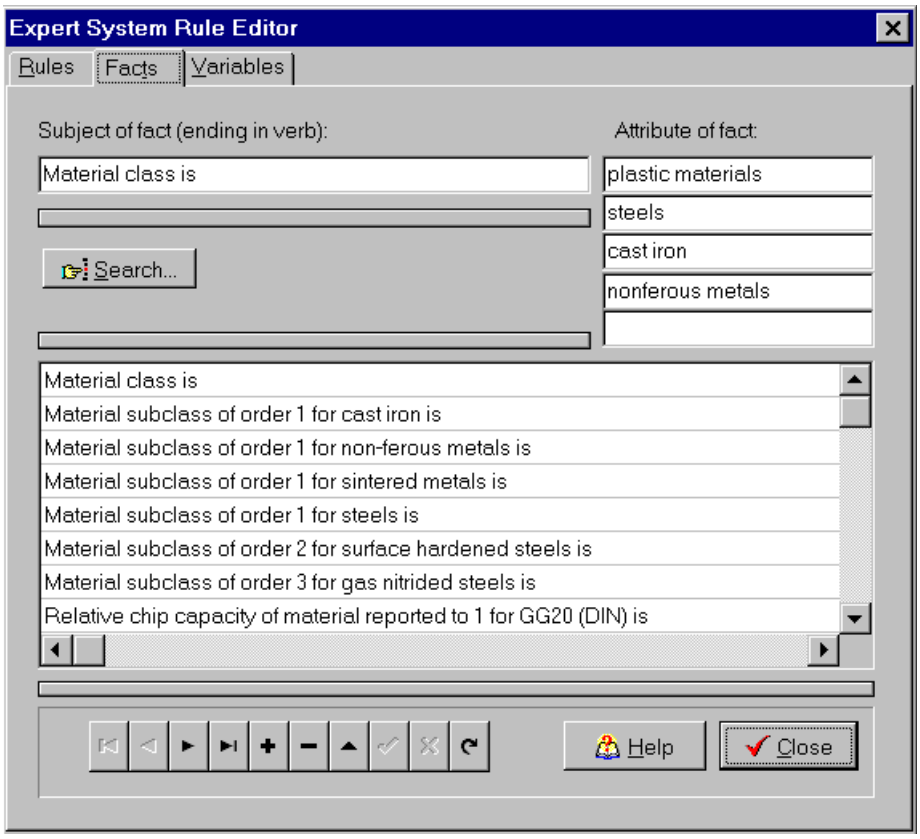

Figure 8: Fact acquisition (fact editor) from human expert.

5.3. Features of rule editor

The rule editor presents a *IF/THEN* rule (rule 1 given below).

The screen control features of rule part are:

1. rule numbering is automatic (see the edit control *Nr. of rule).* This numbering is done only for user information (unused by the inference mechanism);
2. every rule may have a short optional description (see the edit control *Description*). This description gives a general idea about what the rule does;
3. the *IF* and *THEN* parts are displayed using listboxes;

4. at current rule an external procedure can be defined. The pop-up control shows all possible procedures that can be associated with current rule. Also, a supplemental procedure can be defined using the *Define...* button. The input data for the current selected procedure is stored in variable format of rule's *IF* part and the output data is retrieved;
5. the navigation between rules is done using the *Next rule, Previous rule, First rule* and *Last rule* buttons. Also, a new rule can be added (see *Add...* button), the current rule can be modified using the *Edit...* button or deleted by the *Delete* button;
6. for easy finding, a *Search...* button is used. A word (text) can be found in a rule and that rule will be currently displayed; this facility is useful when working with a lot of rules like in engineering design;
7. finally the *Go to rule nr.* button offers the possibility of going at a specific rule when rule number is known; otherwise the *Search...* button can be used.

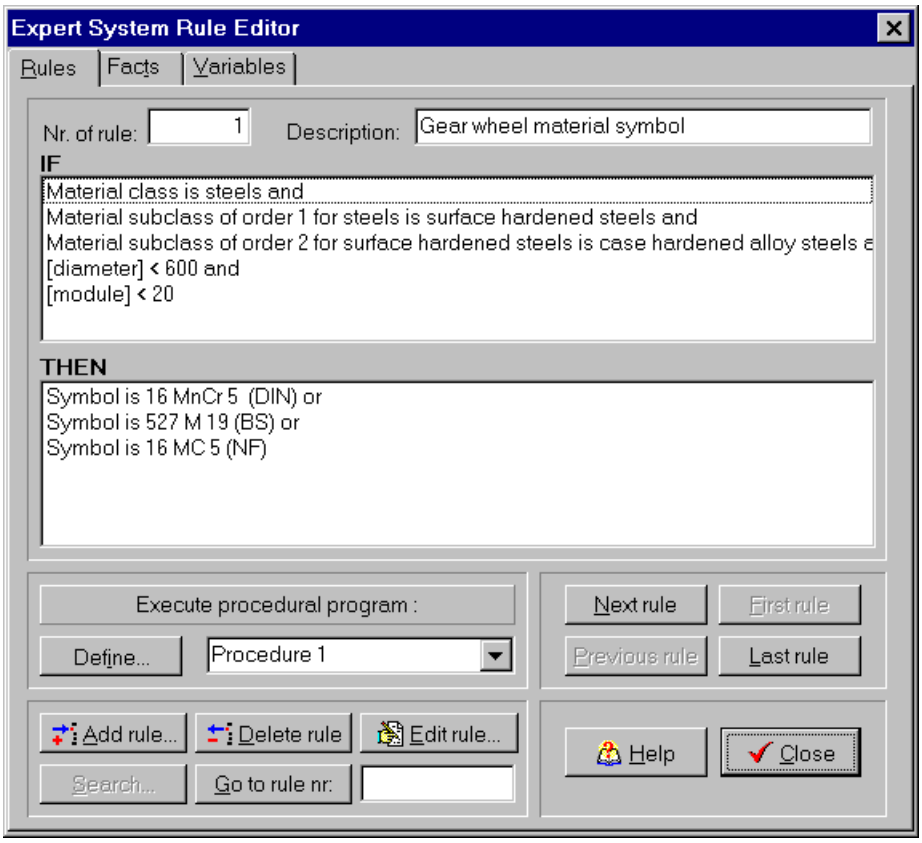

Figure 9: Rule acquisition from human expert.

6. Inference mechanism

This mechanism is designed for the temporal chaining of rules using a forward chaining technique. The inference mechanism is working in steps. A simplified image of the inference process of forward chaining technique used in this application is as follows:

1. the first edited rule is examined. The *IF* part says which of the *THEN* or *ELSE* part will take place and in the *THEN/ELSE* part qualifiers are stored in a workspace inside the *Context*. Also, in another part of the *Context* the variables are stored (see rule 4 where the variable is *load speed factor);*
2. the next step is to determine which are the possible applicable rules (that have the *IF* condition accomplished). One of the rule is selected for execution on the next step.
3. Execution starts with a true premise (the IF part of a rule) and gain a conclusion (the THEN part of the rule). This conclusion is added to context and become use for the next inference step.

The following rules 3 and 4 shown below will be used for further explanation of inference mechanism run.

Rule 3
IF
The type of transmission is hypoid
AND
Maximal speed is under 15 m/s
THEN
The type of gear lubrication material is oil
AND
The oil must contain additives

Rule 4
IF
The lubrication material is oil
AND
The type of transmission is spur, bevel or hypoid
AND
[load speed factor] = *value*
THEN
CALL viscosity procedure([load speed factor])

The inference mechanism will run as follows:

1. rule 3 have the premise "*the type of transmission is hypoid and maximal speed is under 15 m/s*" and gives the conclusion "*the lubrication material is oil AND the oil must contain additives***";**

2. the conclusion facts from rule 3 becomes part of the premise facts for the rule 4. In this rule the computation of *basic cinematic viscosity* by procedural program is done calling value of specific size called *load speed factor* [3].
3. this rule will be triggered and, finally, the basic cinematic viscosity has resulted as a conclusion using a procedural program call.

All the applicable rules are stored in a chained list; the rule that will be "fired" will be erased from this list. If the rule contains a procedural program call (for example for the calculus of basic cinematic viscosity), the result of procedural program run is stored in the *Context*. Expert System will use this variable for next inferences.

Some initial conditions assumed to be true in the forward chaining of rules are specified a special designed screen (fig. 10).

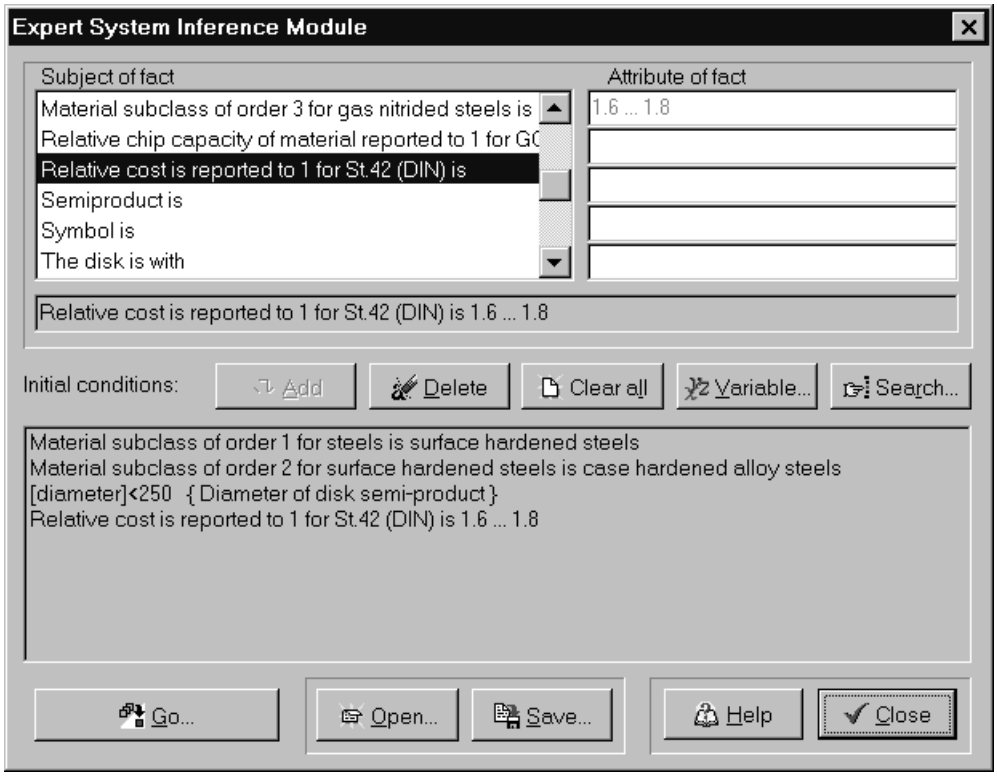

Figure 10: Input of initial conditions assumed to be true in the start of inference process.

In this screen:

1. facts assumed to be true are chosen from the top listbox controls by clicking their subject contents and attribute contents;

2. these facts are added using the *Add* button to the *Context* and the *true* logical value is assigned to them;
3. the *Delete* button erase a facts for the *Context*;
4. *Clear all* button empties the *Context*;
5. a variable can be also introduced in the *Context* using the *Variable...* button;
6. a specific fact can be searched for in the *Context* using the *Search...* button;
7. a *Context* can be saved to disk using the *Save* button (implicit extension for this type of file is *.cnd*);
8. a *Context* can be loaded from the file stored on disk using the *Open* button;
9. the *Go* button starts the inference mechanism.

An important problem that appears when loading a *Context* (when pressing *Open* button) is that of inconsistency that means a new loaded fact in *Context* do not exists in knowledge database. This problem occurs when :

1. a user saves the initial conditions;
2. deletes a fact included in these initial facts from knowledge database
3. try to use again previously saved conditions.

This problem is solved by automatic search for finding facts that are not in knowledge database.

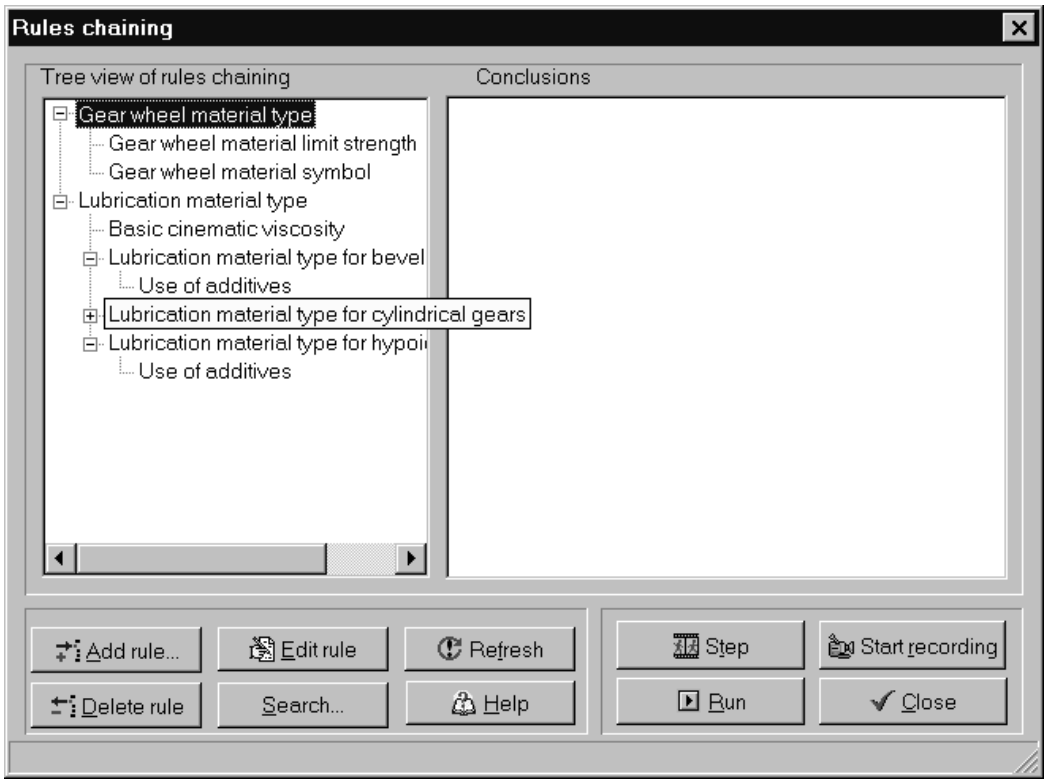

Figure 11: The inference mechanism display.

Rules are chained and displayed in a tree form (fig. 11). A subtree can be expanded and all rules depending of current rule are shown.

The user interface has the following facilities:

1. the tree shape for displaying the chaining of rules is very intuitive. In this tree only the description of rule is shown;
2. a rule can be added on-line in the tree using the *Add rule...* button. In this case the conclusions of highlighted rule automatically appear in the condition part (IF part) of the new input rule;
3. the contents of current highlighted rule can be viewed using the *Edit...* button;
4. current highlighted rule can be deleted from tree using the *Delete* button;
5. the user may ask to refresh the view in case that major changes are done in rules;
6. the *Search...* button looks to specific text in a rule and highlights the rule found;
7. the listbox *Conclusions* display the current of the *Context* (only rule found as true at a well defined moment in inferencing).

The inference mechanism has multiple facilities regarding rule selection for triggering:

1. automatic triggering using first-depth search mechanism. Rules are chained and the user can trigger the whole inference mechanism using the *Run* button. When the *Run* button is pressed, the system makes depth-first search algorithm and puts in *Conclusions* listbox facts that are true at the end of inference process;
2. manual triggering is made when pressing the *Step* button. In this case, step-by-step triggering occurs. Rules that are potential applicable appears with green colour and can be founded anywhere in the chaining tree. In this way, a human expert can easy debug the whole expert system;
3. in the manual triggering style of working, a human inference can be recorded using the *Start recording* button. After pressed, this button becomes a *Stop recording* button; at the end of this recording process a name can be given to the inference made.

Regarding *Conclusions,* some facts are more important the first choice made was to display only important facts. In the next expert system version (currently in progress), the facts will have a special flag set when input to indicate if the fact will be or not displayed in the *Conclusions* listbox.

6. Conclusions

1. The engineering design process involves non-procedural and procedural knowledge.

2. The integration of different knowledge types in an Intelligent Design Environment (IDE) looks promising for engineering design applications.
3. The Authors have elaborated an intelligent environment for mechanical engineering design applied to gear transmissions.
4. The expert system developed can be easily adapted to other design domains provided the knowledge base is changed.
5. The interface with the user is friendly. This interface permits the use of the expert system not only by the knowledge engineer but even by a human expert non specialised in computational science.

Keywords

Intelligent Design Environment (IDE), Artificial Intelligence (AI), Expert System (ES), Knowledge Base (KB), Rule Editor, Inference Mechanism, Mechanical Engineering Design (MED).

References

1. Bunsen, C. *Anwendung von Methoden der Expertsystemtechnik im Maschinenbau.* Diss. TH Aachen, 1991.
2. Carstoiu, D. *Sisteme expert.* ALL, Bucuresti, 1994.
3. Coyne R.,D., et Al. , *Knowledge-Based Design Systems,* Addison Wesley 1990.
4. Dieter, G.E. *Engineering Design - A Material and Processing Approach,* McGraw- Hill International Editors, 1991.
5. Durkin, J. *Expert Systems.* Prentice Hall, New Jersey, 1995.
6. DIN 51509, Teil 1 - 1976. *Auswahl von Schmierstoffen für Zahnradgetriebe.*
7. Ermine, J.L. *Systèmes experts.* Technique et Documentation - Lavoisier, Cachan Cedex,1989.
8. Harbu, I., Picasso, B., *GASS, Gear Assistant. Un sistema esperto per la progettazione di route dentate.* Proceedings XII AIAS Conference, Forli, Bologna, Oct. 1993. (In Italian).
9. Harbu, I., Picasso, B., *Sistema esperto per la progettazione di ruote dentate*. Organi di trasmissione n.3, March, 1995, Tecniche Nuove Ed. (In Italian).
10. Neipp, G. *Artificial Intelligence. A future dimension for industry.* Technische Mitteilungen Krupp, 2/1988, 73-84.
11. Picasso, B., Dobre, G., Radulescu, G., Harbu, I., Carstoiu, D. *About Expert System Utilisation in the Cylindrical gear Geometry.* Proceedings of the Ninth World Congress on the Theory of Machines and Mechanisms, Milano, 1995, vol. 1, 595-599.