Proof Planning and Industrial Configuration

Michal Pechoucek

Czech Technical University of Prague - Artificial Intelligence Group,
Technicka 2, 166 27, Prague 6, Czech Republic,
University of Edinburgh - Department of Artificial Intelligence,
80 South Bridge Edinburgh, United Kingdom,
{pechouc@labe.felk.cvut.cz}

Helen Lowe

Napier University - Department of
Computer Science, Craiglockhart campus
Edinburgh, United Kingdom
{h.lowe@dcs.napier.ac.uk}

Alan Bundy

University of Edinburgh - Department
of Artificial Intelligence, 80 South Bridge
Edinburgh, United Kingdom
{bundy@dai.ed.ac.uk}

The problem of configuration is a complex industrial issue. It has been shown in the past that considerable amounts of labour were spent on problems connected with manufacturing based decisions. Planning and scheduling, design and configuration, logistics and resource management are representative examples. In the case where the complexity of products to be manufactured is getting more flexible, some sort of computer supported decision making is desired. There are a number of examples of Knowledge Based Systems, implemented using various approaches and various methodologies, that have managed to save a considerable amount of resources when supporting computer system configuration, on-board chips design or industrial *"engineer-and-made-to-order"* type configuration.

I introduce Proof Planning, declarative programming paradigm, in order to address main bottlenecks of Knowledge Based Systems design. Commercially successful application needs to consider the problems of enhancability and maintainability as well as clear separation of knowledge and inference engine - weak solver. The proposed methodology is supposed to offer possibility of fast prototyping and reduce overall development times then. The selection of an appropriate methodology shall correspond clearly to the complexity of a problem. In my judgement Proof Planning is very suitable methodology for medium sized Knowledge Based Systems.

1. Introduction

The position I intend to advocate concerns appropriateness of Proof Planning, the theorem proving methodology, for the Knowledge Based System analysis and design. Alan Bundy in [96 Bundy] claims that main features of system based on Proof Planning are important for any Knowledge Based System and thus may be used for development of any Knowledge Based System.

In the first part of the paper Proof Planning is described. The case study of the compressors production problem together with ICON - Industrial Configurator is introduced then. The paper is concluded with few notions on the KADS knowledge analysis phase, the comparative study of the knowledge orientation in both methodologies.

2. Proof planning

Proof Planning is a technique for the global control of search in theorem proving systems. The main features of proof planning were developed for the domain of theorem proving. The proving of a mathematical theorem is just applying a set of logical rules from the theory in question. The hope to come up with a proof of the desired conjecture is the main motivation of this activity. The main difficulty that will be encountered is the combinatorial explosion of the state space. There are a vast number of steps that it is possible to carry out at each step of the theorem proving process. The branching factor is used in order to indicate the overall, worst case complexity of the given problem. We might say that the branching factor of the tree representing the solution space, is considerable. It is obvious that neither depth-first search, which is a very time expensive approach, nor breath-first search, which consumes considerable memory and time, will guarantee to find a solution in reasonable time whilst using a reasonable amounts of resource.

Whilst allowing a sufficient degree of flexibility and adaptability to prove a large variety of different kinds of theorems, the proving strategies themselves are expressed as proof plans by describing Tactics, Preconditions and Effects. Preconditions are decoratively specified pieces of information, under which Tactics, mainly procedurally stated pieces of code, are applicable. Effects describe changes or progression of the solution if the Tactics are successfully applied.

The specifications of Tactics in terms of Preconditions and Effects are called Methods. The proof planning process itself is then searching for a proof of the desired conjecture by means of finding a sequence of tactics that if successfully applied will lead to the conjecture in question. If such a sequence, a proof plan, is found, tactics recorded are carried out and thus the solution itself is specified. Consequently all the time consuming and highly branched reasoning is carried out just once along the path through the tree that was specified by planning. Describing tactics in terms of preconditions and effects can be understood as a kind of meta-level reasoning about further specified pieces of quite expensive computation.

Proof plans try to capture various commonalties between families of proofs and consequently specify the more general direction of the proof, where to head to, instead of some local decision. The problem solver then searches through the space of declaratively stated methods. If an attempt to apply a method fails it does not say anything about the global failure of the problem. Another method is taken instead. This approach describes exactly the way a mathematician behaves. Instead of trying and failing various little pieces of inference, he makes a general plan first with respect to the nature of the problem and then carries out the reasoning process guided by the plan.

Another nice feature of proof planning with respect to its use for Knowledge Based Systems is the separation of factual knowledge and control knowledge. Factual knowledge simply represents the domain theory in terms of a set of axioms and a set of inference rules. Control knowledge describes the inferences in the process of meta-level reasoning about the problem. This fact precisely suits the needs of Knowledge Engineering.

Bundy in [96 Bundy] stated the following specific features, that are typical for meta-level reasoning in terms of proof planning guiding a search through solution space:

Efficiency: If proof plans are well designed, the space complexity is considerably eliminated.

Generality: A given proof plan may be applied to a large number of tasks, especially when hierarchical plans are concerned.

Maintainability: The separation of object knowledge and meta knowledge eases the process of maintainability.

Explanatory power: Meta-level explanation can be of a much higher understandability and use in comparison with long chains of low-level object theory inferences.

3. Case Study

CompAir Reavell, Siebe plc., the leading UK compressor manufacturer, successfully operates on the world-wide market of the industrial compressors. In order to reduce time of the entire **quotation process**, the process of specifying a legal solution with respect to customers desires, they needed a global planning tool that shall help with the product configuration. The system was expected to present an appropriate and logically sequenced series of questions, complemented with a set of all the legal options, in order to facilitate the global product specification. The system should price the particular solution, create the construction description number and set up the final quotation document.

The global knowledge acquisition phase at CompAir Reavell was carried out in order to acquire complete understanding of the field theory, and in order to grasp the expert problem-solving process when creating a quotation. From the very beginning the knowledge were classified as

- Inference knowledge
- Object knowledge

4. ICON - Industrial Configurator

ICON - Industrial Configurator, see [Pech], [96 Pech], application meets all the software requirement specifications briefly mentioned in the Section 3. The system has been programmed in LPA Prolog 3.1 in the MS-Windows environment, running on at least the 386 PC platform. Ladder logic was used for formalising object level knowledge about components of a compressor and attributes of a solution. Proof Planning methods capture the inference knowledge. An ordered set of methods was used for expressing the decision process carried out by an expert in the field. The tactics of a method were used for storing the information about how to create the particular product number.

Proof planning introduced three phases of inference: The Planning Stage, the Validation Stage and the Execution Stage. In the planning stage a user is asked to give as much information as possible in order to give a direction to the search for possible solutions. In the validation stage the system offers the best found solution with a complete set of attributes. The user can either return back to the planning stage and redo some of his/her decisions or let the solution proceed further to the execution stage. In the execution stage the system creates the product number and the final quotation document.

4.1 Planing stage

The language of methods, the domain theory and the system of higher level predicates facilitates creating of an arbitrary planner and thus various planning behaviour. There were two completely different planners implemented within ICON in order to illustrate generality and flexibility of the system.

The User Assisted Planner fulfils all the requirements mentioned in Section 3. It navigates the user through the space of possible attributes and prompts him/her for a value when necessary. This planner is a nice tool that speeds up the process of quotation using all the formalised knowledge. There is no optimisation or build-in decision process incorporated. The User Assisted Planner simulates in fact the behaviour of a quotation expert in the field.

Whereas the Advanced Planner can handle just partially configured solutions. In such case the system allows the user to specify the attributes and optimisation constraints he/she likes and then let the configurator to check the legality of the solution presented and to search through the attribute space in order to create the quotation automatically. It is notable that the rough version of the Advanced Planner was implemented just in a single day.

4.2 Validation stage

The user is supposed to specify whether he/she likes the solution found. Backtracking to the planning stage is an option. In the case of the advanced planning the user may ask for another solution fulfilling the specification in question (Validation Backtracking). Otherwise the program run proceeds further to the execution stage.

Here lies the major limitation of the Advanced Planner. The set of methods is ordered with respect to descending importance. Validation Backtracking is implemented by means of the built-in Prologian backtracking. The last decision is thus respecified first. When the user searches for some lightweight application for less then GBP 5000 he would like to see all possibilities with respect to drive and method of starting first rather than "never ending" plugging on and off of all the optionals.

4.3 Execution stage

The system is engaged with the job of creating the product number and the final quotation document. There is the separated database of rules how to create a product number and the open output template. All of the text is obviously editable and can be either printed out or saved using the conventional Save-Box and .qt extension.

4.4 Object Level Formalisation

The object knowledge has been formalised by means of ladder logic, a well known industrial representation. I decided on this formalism mainly because of its simplicity, being a desired feature when updating a rulebase by non-expert people. I have built a parsing mechanism, that understands an arbitrary ladder logic expression. Consequently the knowledge engineer may use as messy an expression as he likes. This sort of freedom is a substantial virtue of the system since the updating is not just refining of the knowledge-base. Pieces of new knowledge elicitation need to be carried out. The bridge between the natural language rule and the knowledge base formalism needs to be as clear as possible. The other advantage is the efficiency issue, since the Prologian prime key, when searching a database is the first argument of the predicate.

4.5 Meta Level Formalisation

As already mentioned the proof planning methodology introduces a slightly unusual but very efficient meta level knowledge orientation.

The ordered set of methods represents, at the meta level, the decision process in question. When configuring the compressor, the entire quoting process can be viewed as the more or less structured ordering of the decisions to be made. Each particular decision is represented by a single method. A method in proof planning is in fact a meta logical description of what could have happened, under which circumstances such a course of action may happen and finally what will result from such an action. Global run of the program is nothing but (1) an appropriate instatiation of applicable methods (automated or user assisted), e.i.: decisions to be taken and subsequent computation, that shall create the production number[1].

From the maintenance point of view the virtue of proof planning at the meta level is the same as the virtue of the ladder logic at the factual level. The stage between the acquisition of a particular piece of knowledge and the refining of the method language is fairly easy.

Preconditions of a method are intended to record all actions that need to be carried out before deciding whether a particular branch of a sub tree suits the properties of a given sub-solution. An unusual and non-standard action, the user dialogue, needs to be in theory prompted any time when making a decision which way to head. In the case that the domain constraints are that restrictive or the appropriate subspace is getting limited in the right way, the system does not need user assistance and thus does not prompt a question.

Apart from this standard behaviour there are four ways of failing a method implemented within ICON in order to facilitate exhaustive, efficient and comfortable browsing through the space of attributes.

Failing a precondition If one of preconditions that determine the applicability of the method fail, another one is taken.

Postponing a decision If the user is not ready to answer the question presented, applying the method fails and another one is then taken.

Backtracking to redo the last decision may be requested. (explained in the next paragraph).

Complete abort The decision process could be aborted at the level of the user interface.

[1] Any other process, such as CAD drawing or global database update, could be carried out at this place.

4.6 Complexity of Planners

4.6.1 User Assisted Planner

The complexity of the User Assisted Planner is given by the behaviour of the `dialog_list/3` predicate, that is supposed to create a list of legal attributes. Time needed to carry out this predicate is proportional to the number of facts in the rule base, thus to the number of attributes times their possible value. In the worst case each fact would need to parse the entire sub-solution, thus the complete time complexity function needs to be multiplied by the number of attributes.

We may claim that the time complexity of the decision process when configuring an attribute is of a polynomial nature:

$$O(att,val) = att^3 + val.att = \max\{att^3, val.att\} \quad (1)$$

where *att* stands for the number of attributes and *val* is the biggest number of values an arbitrary attribute can take.

The first addendum of the term (1) expresses the time necessary to find an appropriate question to be asked. In the worst case all the database of **methods**, of which there are as many as attributes, needs to be searched through, where all **preconditions** of a method, of which could be as many as **attributes** at worst, needs to be checked. Carrying out a precondition is in fact checking for membership in the solution configured so far, a list of the maximum length equal to the number of attributes.

The second addendum of the term (1) expresses the time needed for creating a list of all possible options. For each value of an **attribute**, set of **conditions** need to be checked. Carrying out a condition is again just checking for membership in the solution configured so far, a list of the maximum length equal to the number of **attributes**. Number of condition specifying applicability of a certain value is of negligible with respect to the number of attributes and number of values. It certainly does not depend on any of these.

4.6.2 Advanced Planner

The time complexity of the Advanced Planner is in fact a general search issue. Apart from the time needed in order to carry out the `dialog_list/3` predicate, that is stated in the previous paragraph, expression (1), we consider the conventional depth first search with the branching factor of val (as defined above) in the worst case and the depth of the search tree given by att (as defined above).

We may claim that the time complexity of the search process when configuring a compressor automatically is of the exponential nature with respect to the number of attributes to be specified:

$$O(att,val) = val^{att.\max\{attt^3, vall.att\}} \quad (2)$$

The space complexity is not a complicated issue. Because of the depth first search used the space needed depends either on the length of the solution or on the biggest number of values an arbitrary attribute can take. Hence the space complexity may be described by the linear function:

$$O(att,val) = \max\{vall, att\} \quad (3)$$

In spite of the fact that the time complexity seems to be of quite explosive combinatorial nature, the system behaves very reasonably as because of clear separation of the independent pieces of inference represented by methods.

5. Comparison with KADS

It is quite ludicrous to claim that one particular approach is completely irrelevant to a given application. The development methodologies are flexible and some sort of engineering approach is required when deciding for one particular approach. There is no specific methodology for comparing methodologies available. Consequently below mentioned notions are based on knowledge analysis I have carried out within the Knowledge Analysis and Design Structuring (KADS) methodology in order to compare the knowledge orientation of Proof Planning and KADS.

The knowledge expertise of KADS proves to be far too complicated in comparison with the approach I have taken. In my judgement, KADS as the global methodology is suitable for projects of larger scale than the one defined above. Knowledge orientation within proof planning is considerably more natural for people maintaining and enhancing the knowledge-base than the complex model layering in KADS.

Nevertheless it is possible (and highly recommended) to use any of the KADS techniques, such as knowledge acquisition techniques or knowledge analysis, in separation from the context of KADS.

6. Results

ICON was warmly welcomed and well appreciated for meeting all specifications in the case of User Assisted Planner and for a worthwhile initiative in the case of the Advanced Planner. The knowledge-base was tested and after a small series of refinements it seemed to behave quite like the experts. As a test case there was used a set of 20 customer specifications, that were representative enough to confirm the accuracy of the configurer.

ICON addresses the topic of maintainability and the ease of further enhancement, the main bottleneck of automated configuration. The entire system was designed carefully in this respect. After a short Prolog syntax tutorial and precise explanation of the system maintenance, the staff of the IT department in CompAir managed successfully to enhance the system in the direction of attributes as well as in the direction of product types.

The system code is well structured and self-explanatory. The openness and flexibility of the object level formalism, the language of methods and the lower-level predicates are illustrated by how fast and straightforward was the development of the Advanced Planner.

Due to all of the substantial field testing, user friendliness, easy maintainability and enhancability and overall system flexibility, ICON is successfully used these days in CompAir Reavel and makes the quotation process considerably easier and faster.

Following CLEM [95 Lowe], automatic configurer, engaged with HP workstation configuration and using the same methodology, ICON is considered as another practical application of proof planning, the theorem proving methodology.

7. Conclusion

The whole Information Technology sector and its industrial applications evolve rapidly. Not only the quality and the price of a software package are crucial factors on the market, but delivery times and meting deadlines are getting more important these days. This is why a knowledge engineer needs to be very careful about the scope of the solution and selecting the appropriate methodology to use.

Proof Planning has shown to be the appropriate methodology for designing a Knowledge Based Systems solving the *engineer-and-made-to-order* kind of configuration problem.

8. Acknowledgements

This paper could not have been written without the financial the support of Excalibur Scholarship Scheme, that facilitated studies of an author in UK.

9. Bibliography

[90 Bundy] *A Science of Reasoning*, Alan Bundy , in Computational Logic, edited by Lassez, J-L. and Plotkin,G, MIT Press, 1990

[95 Lowe] *The application of proof plans to computer configuration problems*, Helen Lowe, Ph.D. Thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1995

[96 Pech] *Proof Planning and Industrial Configuration*, Michal Pechoucek, M.Sc. Thesis, Dept. of Artificial Intelligence, University of Edinburgh, 1996

[Pech] *Icon - Industrial Configurator*, Michal Pechoucek, System Manuals, written for CompAir Reavel, 1996

***Author**: Michal Pechoucek, M.Sc., Dipl. Ing. is a PhD student in artificial intelligence at Czech Technical University of Prague. His field of study is production planning and scheduling. In 1996 he has graduated from University of Edinburgh, Department of Artificial Intelligence and he has obtained M.Sc. in IT - KBS degree. In 1995 he has graduated from Czech Technical University of Prague and he obtained Dipl. Ing. degree in Technical Cybernetics. Paper on Industrial Applications of Proof Planning by Dr. Helen Lowe, Michal Pechoucek and Prof. Alan Bundy was published and presented in INAP'97.*