NATO ASI Series

Springer-Verlag
Berlin Heidelberg NewYork London Paris Tokyo
HongKong Barcelona Budapest

ISBN 3-540-55754-7 · ISBN 0-387-55754-7

ASI

E. Costa

New Directions for Intelligent Tutoring Systems

F 91

New Directions for Intelligent Tutoring Systems

Edited by Ernesto Costa

NATO ASI Series

Series F: Computer and Systems Sciences, Vol. 91

Table of Contents

1 FOUNDATIONS

COLAPSES: A Modular Architecture and Language for Modelling Meta-Control and Uncertainty[1]

R. López de Mántaras, C. Sierra, J. Agustí
Centre d'Estudis Avançats de Blanes,Camí de Santa Bàrbara. 17300 Blanes, Girona, Spain.
e-mail: mantaras@ceab.es, sierra@ceab.es, agusti@ceab.es

Abstract: In this paper we present the management of uncertainty proposed in the COLAPSES language. Uncertainty is mainly a control parameter of the execution of a knowledge base. This language allows the possibility of defining local logics inside a hierarchic structure of modules containing rules. These local logics interact between them in a user defined fashion that gives to the uncertainty treatment a flexibility not found in other systems. The relation between control (meta-logic) and deduction (logic) is based on a reflexive architecture that uses introspection mechanisms to take decisions from the uncertainty in the facts. These features allow to mimic more closely human problem solving behaviour and particularly when uncertainty is present. We conjecture that this type of architecture may have implications for ITS because it allows to provide explanations involving control knowledge decisions.

Keywords: Meta-level architecture, Modularity, Uncertainty management.

Introduction

Most AI research on reasoning under uncertainty is concerned with global normative methods to propagate and combine certainty values and there is a controversial debate about which methods are most appropiate and why. Disagreement between the proponents of the different methods (Bayesians, Dempster-Shafer, fuzzy logicians) is about the meaning of uncertainty and having a formalism that produces rational conclusions with no claim to mimic human uncertainty management methods. In restricted domains where the uncertainty involved permits, for example, a direct interpretation in the probability theory sense and where the main decision of the expert system involves the computation of such uncertainty if appropiate data are available, then probability calculus would be the best independently of how humans solve problems in the same situation. However, the most interesting aspect of the expert systems research is to gain some insights of human problem solving strategies by trying to emulate them in programs. Although human problem solvers are almost always uncertain about the possible solution, they very often achieve their goals despite uncertainty by using methods to manage uncertainty that are particularized to the type of problem solving that they are performing at a given time. In fact, we believe that managing uncertainty consists in selecting actions that simultaneously achieve solutions and reduce their uncertainty. Since actions are selected not only for their domain effects but also for their effect on uncertainty, problem solving under

[1]Research partially supported by CICYT project SPES nº 880j382

uncertainty is more constrained that problem solving under total certainty as was also noticed by Cohen [2-3].

This view leads to consider uncertainty mainly as a control feature because it helps to constrain the focus of attention (i.e. which part of the problem to work next) and action selection (i.e. how to work on it). The problem solving strategies are implemented in the problem solver's control part. The knowledge engineers translate into control strategies the human problem solving strategies.

Problem solving strategies are a fundamental component of expertise and have to be acquired by first implementing a set of task-level primitives with which experts can describe their strategies. Uncertainty is, in our opinion, a task-level primitive that is used at the implementation level to descriminate alternative control decisions. Furthermore, when large expert systems emulate human problem solving strategies, the organization of their complex knowledge bases makes the propagation and combination of uncertainty a local, context dependent, process. In our opinion, such large domain expert systems draw their problem solving capabilities more from the power of their organizational and problem solving structures than from the particular uncertainty management formalism they use (different formalisms can be adjusted to give similar answers).

This paper discusses these ideas in the framework of COLAPSES a modular language based upon the MILORD language [4]. This expert systems building tool uses uncertainty as a control feature and performs local combination and propagation of uncertainty. A medical diagnosis application is used as example whose potential tutoring capabilities, to perform case studies by advanced medical students, are enhaced by the capability to provide justifications of the problem solving strategies selected at the control level.

Modularity and locality

A knowledge base is a large set of knowledge units that cover a domain of expertise and provide solutions to problems in that domain of expertise.

When faced with a particular case, human experts use only a subset of their knowledge for two reasons: adecuacy of the general knowledge -the theory- to the particular problem and availability and cost of data. For exemple, the suspicion of a bacterian disease will rule out all knowledge referring to virical diseases; and also a patient in coma will make useless all the knowledge units that need patient's answers.

The adequation of general knowledge to a particular problem is done at a certain level of granularity, for instance, the expert uses all the knowledge related to the diagnosis of a colon neoplasy or the knowledge related to the radiological analysis of a chest x-ray.
In particular the structuration of KB's is made in MILORD taking into account this granularity in the use of knowledge.

Each structural unit or theory (module from now on) will define an indivisible set of knowledge units (for example rules and predicates). The control will be responsible for the combination of the modules. The combinations will represent the particularization of general knowledge to the problem that is being solved. The control will determine which combinations are acceptable. For example, a module that determines the dosis of penicillin that has to be given to a patient must not be present in any acceptable combination for a patient allergic to penicillin.

The modularization of KB's leads to the concept of locality in the modules of a KB. It is possible to define the contents of a module independently of the definition of the rest of the modules. This possiblility, methodologicaly desirable, allows the use of different local logics and reasoning mechanisms adapted to the subtasks that the system is performing.

Modularity over MILORD: The COLAPSES language

The basic units of KB's written in our language, COLAPSES, are the modules. These may be hierarchically organized, and consist of an excapsulated set of import, export, rule, meta-rule and submodule declarations. The declaration of submodules in a module is what structures the hierarchy. The declaration of submodules does not differ from the declaration of modules. We shall briefly outline which is the meaning of the primitive components of a module. A complete definition of the language and its semantics can be found in [5].

Import: determines the non-deducible facts needed in the module to apply the rules. These facts are to be obtained from the user at run time.

Export: defines which facts deduced or imported inside a module are visible from the rest of the modules that include the modules as a submodule. The import, submodule declarations and export components define the interface of the module.

Rule: defines the deductive units that relate the import and the export components within a module.

Metarule: defines the meta-logical component of the module. Thus, the meta-rules of a module will control the execution of the rules in the module and the execution of the submodules in the hierarchy underneath the module.

The syntax of a module definition is as follows:

Module *modid* = *modexpr*

where *modid* stands for an identifier of the module and *modexpr* for the body of the definition made out of the components specified above. Let us look at an example of module definition.

```
Module Gram_esputum =
begin
        import Class, Morphology
        export Morphology, Esputum_ok
        deductive knowledge:
                Rules
                R001 if Class > 4 then Esputum_ok is sure
                ...
        end deductive
end
```

There is also the possibility of defining generic modules that represent functional abstractions of several non generic modules.

Local Logics

It is clear that the experts use different approaches to the management of uncertainty depending on the task they are performing. Usually expert system building tools provide a fixed way of dealing with uncertainty proposing a unique and global method for representing and combining evidence. In the COLAPSES language it is possible to define different deduction procedures for each one of the modules. If from a methodological point of view a task is associated with a module then, a different logic can be used depending on the task. The definition of local logics is made by the next primitive in the COLAPSES language:

Inference systems
Truth values = *list of linguistic terms*
Renaming = *morphisms between linguistic terms*
Connectives:
Conjuction = *function definition (or "AND" truth table)*
Disjunction = *function definition (or "OR" truth table)*
Inference patterns:
Modus ponens = *function definition*

This primitive is included as a component of the deductive knowledge of a module. Next, we shall explain each one of the components of the local logic definition.

Truth values. This component defines the set of linguistic terms that will be used in the logical valuation of facts, rules and meta-rules of the module where this logic is to be used. Different modules can have different sets of linguistic terms.

Renaming. Modules in a KB define a hierarchy of tasks. Each of the modules can have a different logic, so it is necessary to define a way of interconnecting these different logics. In MILORD this is done in a declarative way. Each module that contains several submodules has a set of morphism definitions that translate the valuations of predicates in the submodules to valuations in the logic of the module. Let us see an example

```
Module B =
 begin
      Module A =
       begin
            Import C
            Export P
            Deductive knowledge:
            Rules:
                 R1 if C the conclude P is possible
            Inference system:
                 Truth values = (false, possible, true)
            End deductive
       end
      Import D
      Export Q
      Deductive knowledge:
            Rules:
                 R1 if A/P and D then conclude Q is quite_possible
            Inference system:
                 Truth values = (impossible, moderately_possible,
                                 quite_possible, sure)
                 Renaming = A/false ==> impossible
                            A/possible ==> quite_possible
                            A/true ==> sure
      End deductive
 end
```


Notice in the above example that the predicate P exported by the submodule A of B which is used in the rule defined in B will be valuated with one of the three values: false, possible or true. To use this fact in the module B we need to change that value for a different one which is known by the logic defined in B. This is done via the renaming definition.

Connectives. This components defines the function that will be used in the deduction process associated with the module. Different multiple-valued functions can be defined depending on the task defined by the module or different truth tables can be used after their elicitation from the expert.

Inference patterns. This field defines the inference rules to be used along the deductive process. To date the only accepted pattern in COLAPSES in the modus ponens for which a propagating function can be defined.

Meta-reasoning by introspection using uncertainty

Having considered uncertainty as a logical component of the COLAPSES language, i.e. the semantics of formulae, the control of reasoning under the uncertainty must be considered as a meta-logic component. Thus the meta-inference over the uncertainty will determine which will the inference control be at the logic level. This meta-inference acts upon the logic component using mechanisms of introspection, this is, the same language represents the uncertainty of the propositions and provides mechanisms both to look at this uncertainty and to determine the control to be followed.

This meta-control is defined as a component of the modules, allowing a local meta-logic defintion. This control component acts over the deductive knowledge and over the submodule hierarchy. It determines which rules and submodules are useful for the current case. The mechanism of interaction between both components is a reflection mechanism: the deductive component reflects on the control component to know which will be the next strategic step, which submodule to execute next, or which rule to use next.

It is not a full reflection mechanism because we allow the meta-logic to see only the valuation of atomic formulae (facts) and the valuation of strategies (sets of modules that combined can lead the system to the solution of the problem), rules and meta-rules cannot be consulted by the meta-logic.

This general mechanism is used to drive the inference process in different directions; we are going to discuss some of them.

Evidence increasing

The current uncertainty of facts can be used to control the deduction steps in order to increase the evidence of a given hypothesis. So, for example, if we have an alcoholic patient showing a cavitation in the chest x-ray and there is low evidence for tuberculosis, then the Ziehl-Nielssen test to determine more clearly whether he has a tuberculosis should not be done. But if he also presents a risk factor for AIDS then we shall increase our evidence for tuberculosis and the test will be suggested. This is expressed as follows:

If tuberculosis > moderately_possible
then conclude Test Ziehl-Nielssen

If risk_factor_for_AIDS **then conclude** tuberculosis **is** possible

If Alcoholic **and** Cavitation
then tuberculosis **is** almost_impossible

It should be noticed that the first rule is a rule of the meta-logic component of the language whilst the others are rules at the logic level.

Strategy focusing

The uncertainty of facts can determine the set of hypothesis to be followed in the sequel. Example:

If the pneumonia **is** bacterial **with** certainty < quite_possible **and**
the pneumonia **is** atipical **with** certainty > possible
Then focus on
Mycoplasm, Virus, Clamidia, Tuberculosis, Nocardia, Criptococcus,
Pneumocistis-Carinii
with certainty quite possible

This example means that the modules to be used in order to find a solution to the current case are those indicated in the conclusion of the meta-rule and should be considered in the order specified there.

Strategies have a certainty degree attached to them. This is useful to differentiate the strategies generated by very specific data from those generated by general data. As an example consider the case of a patient with AIDS (a kind of immunodepression). If we know that the patient suffers from AIDS, a more specific strategy (and also more certain) can be generated. But if we just know that the patient has an immunodepression a less certain general strategy would be generated. Since we may have several candidate strategies simultaneously, combining different strategies is a matter of great importance in the control of the system. This is also achieved by looking at the uncertainty of the strategies, as shows the next example:

If Strategy (X) **and** Strategy (Y) **and** Certainty (X) > Certainty (Y) **and**
Goals (X) $\cap$ Goals (Y) $\neq \emptyset$
Then Ockham (X,Y)

where Ockham (X,Y) is a combination of the strategies that gives priority to those modules found in the intersection of both strategies: (Goals (X) $\cap$ Goals (Y))

Knowledge adequation

As indicated at the begining of the paper a KB is a set knowledge units that have to be adapted to the current case. For example alcoholism is a useful concept when determining a bacterial pneumonia, but it is useless for non-bacterial diseases. Then, a possible use of the uncertainty of the fact bacterianicity is to decide about the use of a given concept in the whole KB, i.e. to adequate the general knowledge to the particular problem. Example:

If no bacterian disease
then do not use alcoholism in finding the solution

Solution acceptance

The degree of uncertainty of a fact can also be used to stop the execution of the system. For example

If Pneumocistis-carinii **and** tuberculosis < possible **and** Criptococcus < possible
Then stop

The control tasks we have discussed use uncertainty as a control parameter and are tasks of the meta-logic level. They are represented as a local meta-logic component of each module in what is called the control knowledge component of a module.

Metacontrol and locality

The structured definition of KB's helps not only in the definition of safe and maintainable KB's but also gives some new features that where impossible to achieve in the previous generation of systems. Among them the most important is the possibility of defining a local meta-logical component for each one of the modules.

The definition of strategies (ordered set of elementary steps to solve a problem) in the MILORD system was made globally. Only one strategy could be active at any moment. Presently, as many strategies as nodes in the modules graph structure can be active. This flexibility is linked with the fact that each module can have a different treatment of uncertainty. So, uncertainty plays a different role as a control feature depending on the association between module and logic.

Furthermore, given the fact that the system consists of a hierarchy of submodules the meta-logical components act ones upon the others in a pyramidal fashion. This allows us to have as many meta-logic levels as necessary in an application. Further research will be pursued along this line. A richer representation of the logic components in the meta-logic will also be investigated and sound semantics from the logic point of view will be defined. Finally, the potential for tutoring advanced medical students due to the higher explanation capabilities involving control knowledge will be evaluated.

Conclusions

The interesting aspect of building expert systems is to learn something about human problem solving strategies by trying to reproduce them in programs. Human problem solver's are uncertain in many situations and do not use a simple normative method to handle uncertainty. Instead they take advantage of a good organization in the problem solving task to obtain good solutions using qualitative approximations. This suggests to consider uncertainty as playing an important role at the control level by guiding the problem solving strategies. In order to

illustrate these points, we have briefly described a modular architecture and language that extensively exploits uncertainty as a control feature and uses local context dependent combination and propagation uncertainty operators.

Bibliography

1. Agustí J., Sierra C., Sannella D. (1989): Adding generic modules to flat rule-based languages: a low cost approach, in *Methodologies for Intelligent Systems 4*, (Z. Ras, ed.), Elsevier Science Pub., pp. 43-51.

2. Cohen P.R., Day D., De Iisio J., Greenberg M., Kjeldsen R., Suthers D., Berman P. (1987): Management of Uncertainty in Medicine, *International Journal of Approximate Reasoning* 1, pp. 103-116.

3. Cohen P.R. (1987): The Control of Reasoning Under Uncertainty: A Discussion of Some Programs, COINS Technical Report 87-81, University of Massachusetts at Amherst.

4. Godo L., López de Mántaras R., Sierra C., Verdaguer A., (1987): MILORD, the architecture and management of linguistically expressed uncertainty, *International Journal of Intelligent Systems*, 4:4, pp. 471-501.

5. Sierra C., Agustí J. (1990): COLAPSES: Syntax and Semantics, CEAB Research Report 90/8.