Applications Of Abduction in Intelligent Decision Support Systems for Scheduling

Ofelia Vasilescu

National Institute for Research and Development in Informatics
Decision Support Systems Research Laboratory
8-10 Averescu Avenue,
71316 Bucharest
ROMANIA
E-mail: vofelia@u3.ici.ro

Abstract. The paper presents an abduction framework using a single inference procedure (abduction) for implementing various modules of an intelligent decision system for scheduling.

1. Introduction

There is a great deal of scope for the development of packages meant to provide support for decisions by more or less traditional (in comparison with AI techniques) data processing techniques. To be able to provide a comprehensive range of such packages and to integrate them into office automation systems is not a trivial venture. These packages could be improved by more user-friendly interfaces, and comprehensive user guidance facilities. A lot could be done along these lines without having to do with controversial and misunderstood AI techniques, and the results might then be marketed as expert systems.

2. What is IDSS

Historically, researchers have developed systems capable to support decision -making in managerial and organizational matters [1]. Such systems, called decision support systems (DSS), typically consist of software for databases, model base and user interface management.

Early decision support systems were computer systems that used decision rules and models coupled with databases to help decision - makers solve semi-structured and unstructured problems [20]. Keen's definition of a DSS is that of an interactive computer-based aid designed to assist managers in complex tasks requiring human judgment [6].

French and Liang viewed a decision support system as " a computer-based system which helps decision makers form and explore the implications of their judgements and hence to make a decision based upon understanding."[24]

There where the decision- making process depends on the user's initiative, passive forms of decision support are offered by decision support systems as mere computerized assistants. An alternative point of view, expressed by Manheim and Mili [13], suggests active involvement of a DSS in the decision - making process. Manheim [14] defined IDSS "as an active system operating almost independent of explicit direction from the users to provide support that the user finds helpful." Such active involvement is especially needed in complex decision -making environments.

Improved techniques current in computer system development and in Artificial Intelligence (AI) make it feasible to implement active intelligent decision support systems (IDSS). Distributed artificial intelligence techniques, the principles of co-operative distributed problem- solving [11, 20] and cognitive modelling [12] can play a major part in the implementation of intelligent systems.

Menzies [15] defines IDSS as model-based software systems that support *management comfort* in *vague domains*. The main requirements for such a system are the ability:

- to validate models;
- to perform inference over those models using assumptions;
- to manage mutually exclusive assumptions in separate worlds;
- to support domain -specific criteria for finding the best worlds.

He found that a single inference procedure (*abduction*) satisfied all these requirements. According to this definition an IDSS must manage the totality of assumptions of a problem.

The goal of a DSS is *management comfort*, i.e. a subjective impression that all the problems are known and at hand. More specifically, managers need to identify and solve problems, then to install some monitoring routine to check that the fix works. A taxonomy of tasks used in the process is shown in Figure 1.

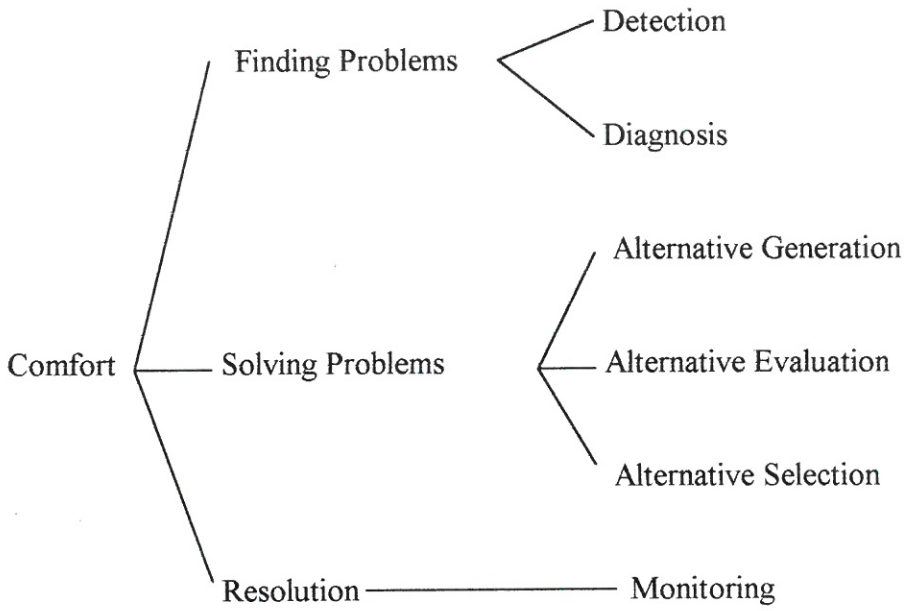

Figure 1. Components of Management Comfort (according to Menzies)

The process of generating and selecting alternatives is the crux matter of most of the decision processes [2].

In a typical business situation, this process occurs in domains containing much guess work. Such *vague domains* [16] are:

- *Poorly measure*: i.e. known data from that domain are insufficient to confirm or deny that some inferred state is valid;
- *Hypothetical*: i.e. the domain lacks an authority that can declare knowledge to be "right" or "wrong"
- *Indeterminate*: i.e. inferencing over a knowledge base could generate numerous, mutually exclusive, outcomes.

Two properties can be found for the models developed for vague domains.

1. Inference requires that guesses or assumptions are made and mutually exclusive assumptions are managed separately.
2. Since vague domains lack an authority, their models may be highly inaccurate. Modelling in vague domains, therefore, requires a validation engine that should not be based on internal syntactic criteria [17].

Menzies believes that abduction provides a comprehensive picture for the declarative knowledge base systems inference. Further, abduction could model certain interesting features of human cognition [18].

In the following abduction as a framework for IDSS will be presented.

3. What is Abduction

It was philosopher Pierce who first introduced the notation of abduction. In [19] he identified three distinguished forms of reasoning:

> *Deduction*, an analytic process based on the application of general rules to particular cases, with the inference of a result.
>
> *Induction*, synthetic reasoning which infers the rule from the case and the result. *Abduction*, another form of synthetic inference, but of the case from a rule and a result.

Peirce further characterized abduction as the "probational adoption of a hypothesis" as an explanation for the observed facts (results), according to known laws. "It is however a weak kind of inference, because we cannot say that

we believe in the truth of explanation, but only that it may be true"[19].

According to Levesque [10]: Given α, β, and the rule R : $\alpha \vdash \beta$, then *deduction* uses the rule and its preconditions to reach a conclusion ($\alpha \wedge R \Rightarrow \beta$); *induction* is learning R, after knowing numerous examples of α and β; *abduction* is using the postcondition and the rule to assume that the precondition could be true ($\beta \wedge R \Rightarrow \alpha$).

According to Toni [23] *abduction* consists of computing explanations for observations. This is a form of non-monotonic reasoning, because explanations which are consistent with one state of a knowledge base may become inconsistent with new information. The existence of *multiple explanations* is a general characteristic of abductive reasoning, and the selection of "preferred" explanations is an important problem.

For IDSS purpose, Toni's explanation is the most intuitive.

Abduction in Logic

Given a set of sentences T (a theory presentation), and a sentence G (an observation), at a first approximation, the abductive task can be characterized as a problem of finding a set of sentences A (abductive explanation for G) such that :

(1) $T \cup A \models G$,

(2) $T \cup A$ is consistent.

This characterization of abduction is independent of the language in which T, G and A have been formulated. The logical implication sign $\models$ in (1) can alternatively be replaced by a deduction operator $\vdash$.

Abduction can be restricted through integrity constraints. Integrity constraints can be used to represent desired properties of a problem. Given a set of integrity constraints, I, the second condition (2) of the semantic definition of abduction, can be replaced by:

(2') $T \cup A$ satisfies I

Abductive Framework

In the sequel an *abductive framework* will be defined as a triplet $< T, A, I >$, where T is a theory, A is a set of assumptions [15] or a set of abducible predicates [23] and I is a set of integrity constraints. The abductive framework was presented in [3].

There are several ways of defining what does it mean for a knowledge base KB ($T \cup A$ in our case) to satisfy an integrity constraint ϕ (in our framework $\phi \in I$). The *consistency view* requires that:

KB satisfies ϕ iff KB $\cup$ ϕ is consistent.

Alternatively, the *theoremhood view* requires that:

KB satisfies ϕ iff KB $\models \phi$.

These definitions have been proposed for the case when the theory is a logic program P written by Kowalski and Sadri [22] and Lloyd and Topor [9] respectively, where KB is the Clark completion of P [15].

A different view of the integrity constraints [5, 8, 21] looks upon these as *epistemic* or *metalevel* statements about the content of database. In this case the integrity constraints are understood as statements at a level other than that of the statements in the knowledge base.

Knowledge Assimilation

Abduction takes place in the context of acquiring new knowledge (information, beliefs or data) for a theory (or knowledge base). There are four possible deductive relationships between the current knowledge base (KB), the new information, and the emerging KB [7].

1. The new information is already deducible from the current KB. The new KB, consequently, is identical to the current one.
2. The current KB = $KB_1 \cup KB_2$ can be decomposed into two parts. One part KB_1 together with the new information can be used to deduce the other part KB_2. The new KB is KB_1 together with the new information.
3. The new information violates the integrity of the current KB. Integrity can be restored by modifying or rejecting one or more of the assumptions which lead to contradiction.

4. The new information is independent of the current KB. The new KB is obtained by adding the new information to the current KB.

4. Production Scheduling Algorithm Using Abduction

The basic decision in a production system is the way how processing is managed, that is the way the resource capacities are operated over the next period of time. The operating mode is decided by the scheduling. The scheme of a simplified production system is shown in Figure 2.

In Figure 2 the (controllable) active elements are: tank1, tank2 and tank3. These tanks function in accordance with the manufacturing (processing) recipes which are inspired by the proportion between the input and output materials in a tank under normal operation; one of these materials is considered to be the main one, usually characteristic of the functioning of all tanks. The tanks are connected (through links) with the other system elements through stocking capacities : stock1, stock2, stock3, stock4 and stock5. The connection with the environment, external to the production system, is performed by the resources (res1, res2 and res3) which are stock supply materials, and by consumers (cons1 and cons2) which are delivered final products.

The principle underlying this scheme is the flow of the materials along the right tracks, in quantities established by the functioning of the tanks. From the point of view of the decision on the functioning conditions, this makes the scheduling.

The conceptual graph of the production scheme shown in Figure 2 is given in Figure 3.

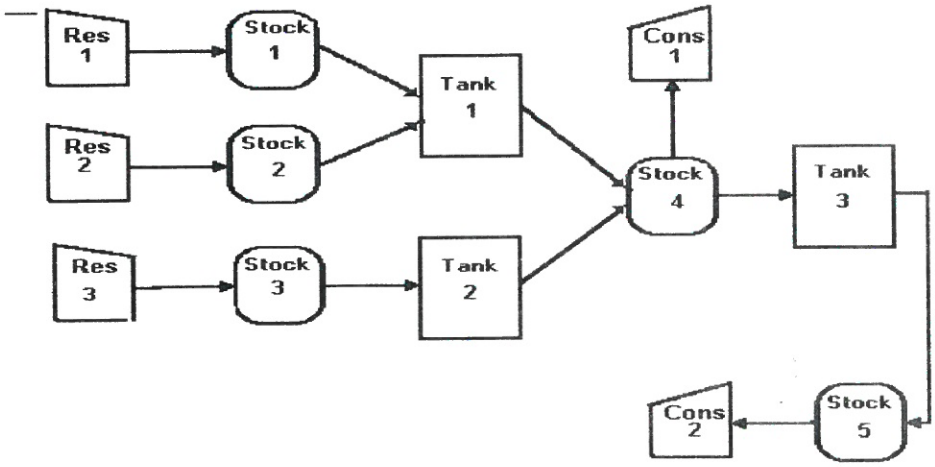

Figure 2. Production System

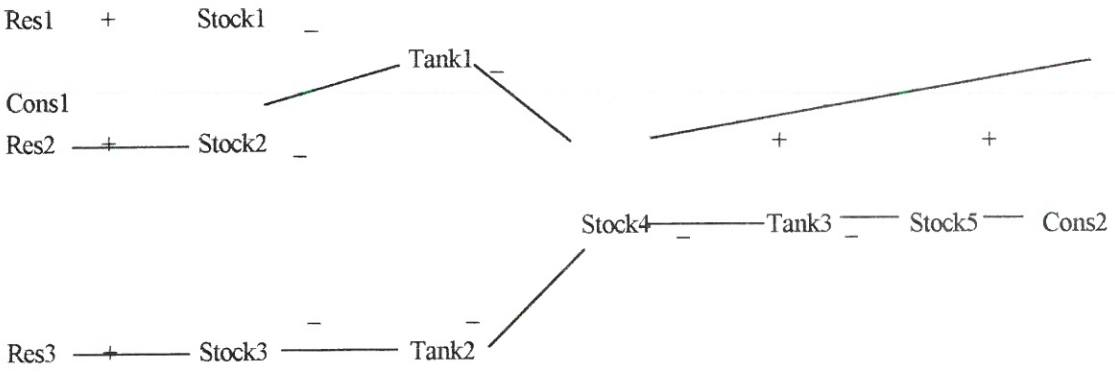

Figure 3. Conceptual Graph of Scheduling

Input nodes are : IN = {Res1, Res2, Res3}

Output nodes are : OUT = {cCons1, Cons2}

Relation :

- I $\overset{+}{\_}$ O denotes that O being UP or DOWN it could be explained by I being UP or DOWN respectively;
- I $\underset{-}{\_}$ O denotes that O being UP or DOWN it

could be explained by I being DOWN or UP respectively.

Each node of that graph can take the value *UP*, *DOWN*, or *STEADY*. The conjunction of an *UP* and *DOWN* can explain a *STEADY*.

The production scheduling algorithm has been developed in three steps :

1. Starting from the final products demanded by consumers cons1 and cons2 the program establishes the values for the functioning of tanks. Can all component tanks function at a nominal value ?

   If *YES* , the set of nominal values will be : $V_n = \{ V_{n,1} , V_{n,2} , V_{n,3} \}$.

   If *NO* , for those tanks that cannot function, new nominal values are calculated; for instance in case of non-functioning of tank1 we will have : $V_n = \{ V_{n,1}^{calc} , V_{n,2} , V_{n,3} \}$ and the deviation from the nominal value will be : $AT_3 = || V_{n,3} - V_{n,3}^{calc} ||$.

2. Stocking capacities will be needed in order to find the nominal values for each tank. Can all component stocking capacities function at a nominal value ?

   If *YES* , the set of nominal values will be : $S_n = \{ S_{n,1} , S_{n,2} , S_{n,3} , S_{n,4} , S_{n,5} \}$

   If *NO* , for unfeasible stocks, a new nominal value is calculated; for instance in case of unfeasibility of stock4 we will have : $S_n = \{ S_{n,1} , S_{n,2} , S_{n,3} , S_{n,4}^{calc} , S_{n,5} \}$ and the deviation from the nominal value will be : $AS_4 = || S_{n,4} - S_{n,4}^{calc} ||$

3. The new nominal values for tanks and stocks help calculate the feasibility of the whole system. Is it done?

   If *YES* , the user selects the criteria on the basis of which alternatives to scheduling are generated.

   If *NO*, *A* assumptions on the system feasibility are added and together with the *T* domain theory they will develop alternatives to scheduling. The mutually exclusive assumptions will be managed in *separate worlds*. For each world there will be user selected criteria based on which alternatives to the production scheduling are generated.

For instance, the author considers unfeasible tank1 with the value

- *DOWN* than the base controversial assumptions for *tank1Down* are : input tank1 UP or output tank1 DOWN, i.e.

  A1 = {stock1UP, stock2UP, stock4DOWN }

  A2 = { stock1UP, stock2Steady, stock4DOWN }

  A3 = { stock1Steady, stock2UP, stock4DOWN }

- *UP* than the base controversial assumptions for *tank1Up* are : inputs tank1 DOWN or output tank1 UP, i.e.

  A1 = {stock1DOWN, stock2DOWN, stock4UP }

  A2 = { stock1DOWN, stock2Steady, stock4UP }

  A3 = { stock1Steady, stock2DOWN, stock4UP }

The system feasibility will be managed in separate worlds. The worlds of this example for *tank*1DOWN are :

W1 = { res1, res2, res3, stock1UP, stock2UP, stock3STEADY, tank1DOWN, tank2STEADY, stock4DOWN, tank3STEADY, stock5STEADY, cons1, cons2 }

W2 = { res1, res2, res3, stock1UP, stock2STEADY, stock3STEADY, tank1DOWN, tank2STEADY, stock4DOWN, tank3STEADY, stock5STEADY, cons1, cons2 }

W3 = { res1, res2, res3, stock1STEADY, stock2UP, stock3STEADY, tank1DOWN, tank2STEADY, stock4DOWN, tank3STEADY, stock5STEADY, cons1, cons2 }

The preferred world is selected according to the user's criteria.

The production management decision provides alternatives to functioning under various conditions when the computed values are not feasible (it leads to a violation of the functioning of tanks and stocking capacities).

4. Conclusion

A single inference procedure (abduction) can support many of the modules required for planning and scheduling in intelligent decision support systems. Abduction can execute in vague and conflicting domains (which, we know, occur very frequently in DSS). We have proposed the use of abduction as a framework for IDSS.

REFERENCES

1. BONCZEK, R.H., HOLSAPPLE, C.W. and WHINSTON, A.B., **Foundations of Decision Support Systems**, ACADEMIC PRESS, 1981.
2. BOOSE, J.H., BRADSHAW, J.M., KOSZAEK, J.L. and SHEMA, D.B., **Knowledge Acquisition Techniques for Group Decision Support**, in B. R. Gaines, M.A. Musen, J.R. Boose (Eds.) Proceedings of the 7th Knowledge Acquisition for Knowledge-Based Systems Workshop, 1992, pp. 2.1-2.22.
3. BUTA, O., **Abduction and Argumentation**, Scientific Session on Action Logic and Argumentation Theory, Faculty of Philosophy, University of Bucharest, 1998 (in Romanian).
4. KAKAS, A.C., KOWALSKI, R.A. and TONI, T., **The Role of Abduction in Logic Programming**.
5. KAKAS, A.C. and MANCARELLA, P., **Generalized Stable Models: A Semantics for Abduction**, Proceedings of the 9th European Conference on Artificial Intelligence, ECAI' 90, Stockholm, 1990, pp. 385-391.
6. KEEN, P.G.W., **Adaptive Design for DSS**, DATABASE, Vol. 12, Nos.1 and 2, 1980, pp. 15-25.
7. KOWALSKI, R.A., **Logic Without Model Theory**, in D. Gabbay (Ed.) What is A Logical System?, OXFORD UNIVERSITY PRESS, 1994.
8. KOWALSKI, R.A., **Problems and Promises of Computational Logic**, Proceedings of the Symposium on Computational Logic, Lecture Notes in Computer Science, SPRINGER- VERLAG, 1990.
9. Lloyd, J.W. and Topor, R.W., **A Basis for Deductive Database System**, Journal of Logic Programming, 2 , 1985, pp. 93-109.
10. LEVESQUE, H. J., **A Knowledge-level Account of Abduction**, Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, ILL, USA, 1989, pp. 1061-1067.
11. LESSER, V.R., and CORKILL, D.D., **Functionally Accurate, Cooperative Distributed Systems**, IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, Vol. 11, January 1981, pp. 81-86.
12. NORMAN, L.K., **Models of the Mind and Machine: Information Flow and Control Between Humans and Computers**, in C. M. Yovitis (Ed.), ADVANCES IN COMPUTERS, Vol. 32, ACADEMIC PRESS, 1991.
13. MILI, F. and MANHEIM, M.L., **Introduction to the DSS Minitrack on Active DSS and Symbiotic Systems**, 22nd Hawaii International Conference on System Science, 1989, pp.24-32.
14. MANHEIM, M.L., **An Architecture for Active DSS**, 21st Hawaii International Conference on System Science, 1988, pp. 356-365.
15. MENZIES, T., **An Overview of Abduction As A General Framework for Knowledge-Based Systems**, 1995, available from http://www.sd.monash.edu.au/~timm/pub/docs/papersonality.html
16. MENZIES, T. and COMPTON, P., **The (Extensive) Implications of Evaluation on the Development of Knowledge-Based Systems**, Proceedings of the 9th AAAI-sponsored Banff Knowledge Acquisition for Knowledge-Based Systems, 1995, available from http://www.sd.monash.edu.au/~timm/pub/docs/papersonality.html
17. MENZIES, T., **Principles for Generalised Testing of Knowledge Bases**, Ph. D Thesis, University of New South Wales, Australia, 1995, available from http://www.sd.monash.edu.au/~timm/pub/docs/papersonality.html
18. MENZIES, T., **Situated Semantics Is A Side-effect of the Computational Complexity of Abduction**, the 3th Conference of Australian Cognitive Science Society, 1995, available from http://www.sd.monash.edu.au/~timm/pub/docs/papersonality.html
19. Hartshorn et al (Eds.) **Collected Papers of Charles Sanders Peirce**, Vol. 2, 1931-1958, HARVARD UNIVERSITY PRESS.
20. RAO, H.R., SRIDHAR, R. and NARAIN, S., An Active Intelligent Decision Support System - Arhitecture and Simulation, DSS Vol. 12, 1994, pp. 79-91.
21. REITER, R., **On Integrity Constraints**, in M. Y. Vardi (Ed.) Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning About Knowledge, Pacific Grove, CA, USA, 1988.
22. SADRI, F. and KOWALSKI, R.A., **An Application of General Purpose Theorem-proving to Database Integrity**, in Minker (Ed.) Foundations of Deductive Databases and Logic Programming, MORGAN KAUFMANN PUBLISHERS, Palo Alto, CA, USA, 1987, pp.313-362.
23. TONI, F., **Abductive Logic Programming**, Ph. D Thesis, Imperial College of London, 1995.