HEWLETT PACKARD

A Hybrid Approach to Fault Diagnosis in Network and System Management

Along Lin
Internet Business Management Department
HP Laboratories Bristol
HPL-98-20
February, 1998

E-mail: alin@hplb.hpl.hp.com

diagnosis,
model-based reasoning,
logic programming

In this paper, several Artificial Intelligence (AI) techniques such as Rule-Based Reasoning (RBR), Bayesian Networks (BNs), Neural Networks (NNs), Case-Based Reasoning (CBR), Qualitative Reasoning (QR), and Model-Based Reasoning (MBR) are described. Then an automated management system prototype is presented. Finally, a hybrid approach to automated network and system management is proposed. However, we focus on fault diagnosis.

Internal Accession Date Only

© Copyright Hewlett-Packard Company 1998

A Hybrid Approach to Fault Diagnosis in Network and System Management

Along Lin

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS12 6QZ, U.K.
Email:alin@hplb.hpl.hp.com

Abstract

In this paper, several Artificial Intelligence (AI) techniques such as Rule-Based Reasoning (RBR), Bayesian Networks (BNs), Neural Networks (NNs), Case-Based Reasoning (CBR), Qualitative Reasoning (QR), and Model-Based Reasoning (MBR) are described. Then an automated management system prototype is presented. Finally, a hybrid approach to automated network and system management is proposed. However, we focus on fault diagnosis.

Keywords: diagnosis, model-based reasoning, logic programming.

Introduction

Because fault diagnosis requires much expertise and is knowledge intensive, it has been one of the key research topics of AI and Expert Systems (ESs) since 1960s. Although there have been various kinds of diagnostic systems, most of them were mainly designed for medical diagnosis, electronic circuit analysis and nuclear power station maintenance (Reiter 1987, Weber 1993). Only a little progress has been made on the fault diagnosis of networked systems and applications. The management of systems and networks is getting increasingly complex and crucial with the explosive growth of the Internet and the worldwide acceptance of the Web.

Traditional fault diagnostic techniques such as RBR and BN are not flexible enough to deal with the dynamic evolving of a diagnosed system. The complete diagnostic knowledge about a system must be acquired and fully represented in advance, which has proven to be hard and sometimes even impossible if people have not got enough experience on the diagnosed system. Particularly, much of the diagnostic knowledge has to be modified in case of the changes to managed components or the connections among them. Even though NN and CBR systems are capable of learning from examples or new cases, they still lack of flexibility and extensibility as most RBR and BN systems do. This difficulty stems from the fact that those systems diagnose faults based on people's experiences rather than the basic principles about how the components in the diagnosed system behave and their causalities.

Model-Based fault diagnosis seems to be more promising than other AI approaches (Chen and Srihari 1989, Davis 1984, de Kleer 1991, de Kleer, Mackworth, and Reiter 1992, de Kleer and Williams 1987, Hofmann 1993). It is based on the models of a diagnosed system, which describe the structure of the system and its behaviors. The faults in a managed component are diagnosed by comparing its observations with the predictions about its intended behavior. However, reasoning from first principles is inherently inefficient and obtaining models may be either very difficult or too complicated. Therefore, a hybrid fault diagnosis approach is required.

In the following, several AI techniques are described first. In section 3, we present a goal-driven model-based automated management system prototype. A hybrid approach to automated network and system management is proposed in section 4. Finally, some conclusions are given.

Knowledge-Based Fault Diagnosis Approaches

Rule-Based Reasoning

Rule-Based Expert Systems (RBESs) have been rather successful in a variety of fields since the early 1960s. A RBES includes an inference engine (IE) and a rule-base (RB), in which the diagnostic knowledge from domain experts has been coded in advance. It works by chaining the rules in its RB given a query goal. Some RBESs have additional modules for explanation and knowledge acquisition. The success of RBES is due to the separation of RB from IE, thus increasing its flexibility significantly. However, this approach has several serious drawbacks for its application to network and system management. Firstly, all of the relationships between failure symptoms and underlying faults must be pre-determined completely and correctly. Secondly, eliciting diagnostic knowledge from experts is hard. Thirdly, if managed components or the connections among them change, its RB has to be modified dramatically. Finally, it may become increasingly slow with the expanding of RB.

Bayesian Networks

A BN consists of nodes representing uncertain assertions and directed arcs representing cause and effect dependencies among the nodes. It reasons from effects to causes. The arcs are expressed by a probability distribution for each effect node, based on the probabilities of its cause nodes. Due to the ability to handle uncertainty and its special knowledge representation, a BN is especially useful in the fields in which people have incomplete knowledge (Charniak 1991). Furthermore, it can provide some guidance in diagnosis by calculating nodes probabilities, which is helpful for choosing a most likely fault candidate when integrated with MBR. However, its shortcomings are very similar to those with RBESs.

Neural Networks

NNs have found many applications in the fields of diagnosis, image processing, classification, and pattern recognition. A feed-forward NN is very suitable for fault diagnosis and event correlation because of its ability to recognize given patterns efficiently, approximate any functions given enough neurons, and learn from input/output pairs by adjusting its connection weights automatically without the deep understanding of a domain. More importantly, as with a BN, an NN can handle incomplete, ambiguous, and imperfect data. A problem with this approach is that an NN must be trained by large amounts of example data consisting of input/output pairs in advance, which may be unavailable in some domains. Its another disadvantage is that it is unable to explain why a specific solution is given.

Case-Based Reasoning

CBR has been proven very successful in a few of domains. A CBR system uses a library of previous cases with known solutions to obtain the most appropriate solution to present problem, which usually involves in case retrieval, adaptation, evaluation, and learning. In order to retrieve cases efficiently, it is crucial to decide what the key attributes of a case are and on which attributes the cases should be indexed. It has the following advantages over ESs and BNs (Aamodt & Olaza 1994, Barletta 1991, Kolodner 1993, Slade 1991, Voss 1997, Watson & Marir 1994).

- It can solve problems within partially understood domains.
- It can reason by analogy efficiently.
- It can learn from new cases.
- It requires little maintenance.
- It is easier to build a case library than a RB/BN.
- Its knowledge representation is less restrictive.
- It allows faster knowledge acquisition.
- It can evaluate a proposed solution.

The drawbacks of a CBR system are that it is unable to provide a solution to the given problem if there are no matches for the current situation in its case library. As with NNs, CBR systems are also opaque, unable to provide deep-level explanations for the proposed solutions. When the cases are unavailable, it will be time-consuming to build a case library.

Model-Based Reasoning

MBR reasons from the normal behavior of a system of which the structure is known. A Model-Based diagnostic system consists of two parts, the fault detection mechanism and the diagnostic engine. Because MBR can diagnose the faults which never occurred before, it is a more promising fault diagnosis technique than any of the approaches described above. There are several modeling techniques, of which device-centered models are the most commonly used modeling techniques.

Production-rules. A system is modeled by a set of rules representing causal relationships, based on first principles. The rules model the behavior of the system by using the inputs as antecedents in them and then firing the appropriate rules to change the values for the state variables in the system.

Device-centered models. It is assumed that a component is the smallest replaceable device which can fail. Each component is described in terms of its behavior and relationship to other components and modeled as an independent object whose functionality is determined by the methods in the object.

Qualitative reasoning. This technique allows to model a system in a more abstract way. In the case of network and system management, it is hard to model the behavior of a system quantitatively. The modeled system is described by a set of physical parameters and qualitative differential equations (QDEs) describing how the parameters relate to each other. The qualitative state of a parameter consists of a pair of values representing its ordinal relations with its landmark values, which mark the borders of operating regions, and its changing direction (de Kleer & Brown 1984, Forbus 1981, 1984, Kuipers 1986, 1993).

In MBR, it is assumed that the structure of a system and its normal behavior are well understood and the basic principles about how its components behave and their causalities have been captured in the models, based on which we diagnose the faults in a component by comparing its observations with the predictions about its intended behaviors. However, reasoning from the actual structure and the normal behavior of a system is inherently inefficient whereas the methods based on heuristics or experiences are quite effective.

An Automated Management System Prototype

Object-Oriented Logic Programming (OOLP)

Logic Programming (LP), a well-known knowledge-based programming paradigm, has found a wide range of industrial applications of Artificial Intelligence and Expert Systems since the early 1970s. LP has the following advantages:

- It is declarative. A problem can be solved by describing what has been known about it.
- It also provides a procedural reading so that it can solve problems.
- It is based on First Order Logic (FOL), which provides a solid theoretical foundation.
- It supports backtracking, thus being capable of solving non-deterministic problems.
- It is an assignment-free programming language, making it much easier to debug a program.
- It is very suitable for software prototyping.
- It supports incremental development and refinement of software.
- It supports goal-directed reasoning and meta-level reasoning.
- It supports recursive programming, infinite, recursive data structure and automatic garbage collection and memory allocation.
- All of its data structures can be uniformly expressed as structures.

On the other hand, Object-Oriented Methodology (OOM) offers an innovative software development model, which is crucial to large-scale software development and supports encapsulation and inheritance, making it possible to achieve greater productivity than would otherwise have been possible. OOLP, resulted from the application of OOM to LP, has several advantages. First, methods have the ability of reasoning about the attributes of modeled objects. Secondly, methods are declarative and able to backtrack. Thirdly, it provides a mechanism for keeping information as private as possible. Fourth, it supports the software reuse, enabled by inheritance.

An Automated Management System Prototype

FLIPPER is a goal-driven model-based prototype automated management system, which manages devices like workstations, UNIX, Netware servers, PC's, printers, routers. It adopts OOLP as its knowledge representation language, which supports type definition and an inheritance mechanism that allows the methods in a super object type to be overridden by those in subtypes. It uses the Linear resolution on Definite clauses with Selection function and Negation by Failure (SLDNF) as its inference system. The architecture of an automated management system is described in Figure 1.

There are two special types of goals. Monitored goals are always watched and used to trigger an event to signal some system state change to other applications, agents or managers, thus event filtering, event correlation and fault-detection can further be accomplished based on monitored goals. Resident goals need to be maintained true, which involves monitoring and fixing. As a resident goal becomes false, a sequence of executable primitive actions will be produced automatically by a planner. After executing them, the system will be in a consistent state in which the failed goal is true.

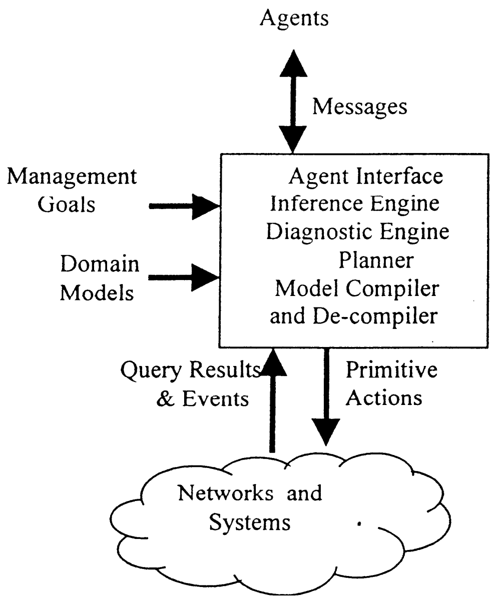

Figure 1: An Architecture of goal-driven model based automated management system

A model defines the object types in a managed domain, each of which is described in terms of its attributes and behaviors. The attributes represent its current state. Object behaviors are defined by the methods consisting of query rules, access methods and primitive actions. A relationship among several managed objects can be defined by a set of rules which have the same relation signature. Each rule is expressed by a Typed Guarded Horn Clause (TGHC). Access methods link to the real world to obtain or check the information about managed components. The mapping of an access method to its external function is achieved by a particular Dynamic Linking Library (DLL). Although there is a difference in the way they are defined, query rules and access methods are dealt with uniformly by the underlying inference engine. Primitive actions are used to change the states of the managed objects. There has been seven predefined core types: Character, CoreRoot, Integer, Real, String, Set and List, where CoreRoot is the super type of all core types. Users can have new core types by defining their external functions for hashing, printing and equality checking. They can also add several methods for those user-defined core types by providing an external access module. In addition to this, users can define their own object types.

Each FLIPPER agent manages its local environment autonomously based on its management goals and the models of managed components. It can accomplish various kinds of management tasks, such as configuration, performance monitoring, event filtering, event correlation, fault diagnosis, fault repairing and so on. With the help of Inter-Agent Protocol (IAP), an agent can exchange information with other agents, thus achieving the scalability easily. Because the intelligence of a FLIPPER agent comes from the input models, which can be transmitted across networks, we can manage a system remotely and securely by adding a layer of security management on top of it, which usually involves in authentication, authorization, and encryption/decryption. To keep the transmitted data as little as possible, the models can be compiled before they are transmitted, in which case either a de-compiler or a virtual machine must be provided.

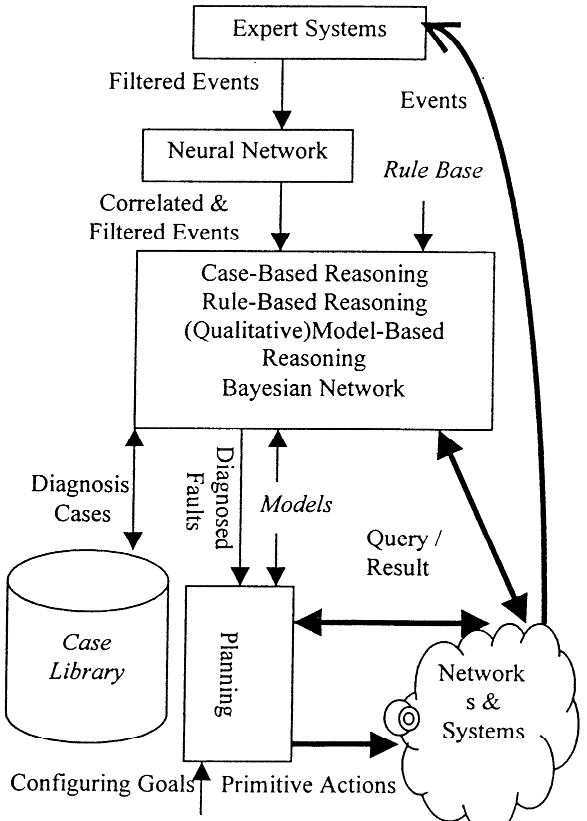

Figure 2: An architecture for automated network and system management.

Fault Correction

For the time being, we fix the diagnosed faults by executing some pre-written scripts, which is inflexible and time consuming to develop them for all the possible situations. In the case of policy-driven management, because the policies for each component in a managed system must be enforced, those hard-coded scripts will have to be modified when there are some changes to the policies, thus resulting in enormous amount of work. To repair faults automatically, a planner will be utilized to generate a sequence of primitive actions, which execute some commands to manipulate the managed objects in the real world or to reflect some changes to its dynamic store. Each action describes a precondition, which must be true before it can be invoked, and a post-condition, which will be true after it is executed. Its syntax is defined as follows:

```
<action>    ::=<relation>'PRE'<pre>'POST'<post>'.'
<goal>      ::=<relation>|'¬'<goal>|<goal>{'&'<goal>}*|
                <goal>{'|'<goal>}*
<non-code>::=<var1>'≠'<var2>{'&'<var1>'≠'<var2>}*
<pre>       ::=<goal>['&'<noncode>]|∀x∈T('<pre>')'
<effect>    ::=<relation>|'¬'<relation>|
                <effect>{'&'<effect>}*
<post>      ::=<effect>|<goal>'→'<effect>|
                ∀x∈T('<post>')'|<post>{'&'<post>}*
```

In <post>, <goal> refers to the world before the action is executed while <effect> refers to the world after execution. Furthermore, $\forall x \in t\ Q(x)$ can be expressed as:

[Set *t*] forAll [Type x] goal Q(x)

For example, [Set staff] forAll [Employee e] goal ([e] authorisedToPrintOn [Printer p]). $\forall x \in t Q(x)$ is replaced by its universal base $Q(c1)\&\ldots\&Q(cs)$ if $T = \{c1,\ldots, cs\}$, providing the world is static and making the Closed World Assumption (CWA). In addition, the variables in goals are implicitly existentially quantified. It is assumed that the initial state contains no variables and all the variables in the effects of an action must appear in its preconditions.

A Hybrid Approach to Fault Diagnosis

Due to the diverse nature of the tasks in network and system management such as configuration, performance monitoring, fault diagnosis, and repairing, it is impossible to perform all of them with one technique. A hybrid AI approach to automated network and system management is shown in Figure 2, where planning is used to configure a system and to rectify the diagnosed faults. Even though a hybrid method can be utilized for each of those tasks, we focus only on fault diagnosis here.

The diagnosis is driven by data instead of goals. The RB in RBR consists of two parts, one describes the most commonly occurring symptom/fault associations, the other one contains meta-rules, giving the control knowledge about when and how to use different diagnostic strategies. After receiving the filtered and correlated events, the diagnostic system works as follows. It first uses RBR to diagnose the underlying faults based on the first part of its RB. If this fails, it uses the meta-knowledge described in RB to provide some clues for choosing CBR system or MBR system to identify the faults. If the CBR system is chosen and the result is not satisfactory because there are no cases resembling the current situation in the case library or the proposed solution is bad, it will then try MBR system to diagnose the faults. The MBR system uses BNs to choose the components for testing based on their probabilities after it has obtained the suspect set from the fault detection process. During the candidate elimination, MBR may use QR to reason about the normal and fault behaviors of some components of which quantitative models are difficult to acquire or too complicated to describe. Thus, in this hybrid approach, the behaviors of managed components, of which quantitative models are difficult to acquire, are described qualitatively. After the faults have been diagnosed, the solution will be evaluated, and then stored as a new case together with its solution steps and the current contextual information.

Conclusions

Model-Based approach to fault diagnosis has been recognized as more appropriate than other knowledge-based methods such as RBR, BNs, NNs, CBR since it can precisely diagnose the faults never known before. However, obtaining complete and correct models could be either difficult or too complicated in the field of network and system management. In those cases, we can diagnose faults based on the typical symptom/fault associations, described in a rule base, BN, NN or case library. The hybrid approach we proposed utilizes various kinds of knowledge, including both quantitative and qualitative knowledge and even heuristics, to overcome the drawbacks of a pure model-based approach. Even though it is time-consuming to acquire different kinds of knowledge for a hybrid approach, it provides more flexibility and capability of diagnosing the faults in network and system management.

References

Aamodt, A., & Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(1):39-59.

Barletta, R. 1991. An Introduction to Case-Based Reasoning. *AI EXPERT* 6(8).

Charniak, E. 1991. Bayesian networks without tears. *AI Magazine* 12:50-63.

Chen, J.S. & Srihari, S.N. 1989. Candidate Ordering and Elimination in Model-Based Fault Diagnosis. In *Proc. of the 11th IJCAI*, 1363-1368.

Cunningham, P. & Smyth B. 1994. A Comparison of Model-Based And Incremental Case-Based Approaches To Electronic Fault Diagnosis. In *Case-Based Reasoning Workshop, Twelfth National Conference on Artificial Intelligence.*

Cunningham, P. & Brady, M. 1987. Qualitative Reasoning In Electronic Fault Diagnosis. In *Proc. of the 10th IJCAI*, 443-445.

Davis, R. 1984. Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence* 24(1): 347-410.

de Kleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28(2):127-162.

de Kleer, J. 1991. Focusing on Probable Diagnoses. In *Proc. 9th National Conf. On Artificial Intelligence,* 842-848.

de Kleer, J., Mackworth, A.K. & Reiter, R. 1992. Characterizing Diagnosis and Systems. *Artificial Intelligence* 56:197-222.

de Kleer, J. & Williams, B.C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97-130.

de Kleer, J. & Brown, J. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24:7-83.

Forbus, K. 1981. Qualitative reasoning about physical processes. In *Proc. of the 7th IJCAI.*

Forbus, K. 1984. Qualitative process theory, *Artificial Intelligence* 24(3):85-168.

Gebhardt, F. 1997. Survey on structure-based case retrieval. *The Knowledge Engineering Review* 12(1):41-58.

Hofmann, M.O. 1993. Model-Based Diagnosis Directed by Heuristic Search. *The Ninth Conference on Artificial Intelligence for Applications*, 197-203.

Kolodner, J. L. 1993. *Case-Based Reasoning*, San Mateo, CA: Morgan Kaufman.

Kuipers, B.J. 1986. Qualitative Simulation. *Artificial Intelligence* 29:289-338.

Kuipers, B.J. 1993. Qualitative simulation: then and now. *Artificial Intelligence* 59:133-140.

Reiter, R. 1987. A Theory of Diagnosis From First Principles. *Artificial Intelligence* 32(1):57-96.

Slade, S. 1991. Case-Based Reasoning: A Research Paradigm. *AI Magazine* 12(1):42-55.

Voss, A. 1997. Case reusing systems-survey, framework and guidelines. *The Knowledge Engineering Review* 12(1):59-89.

Watson, I. & Marir, F. 1994. Case-based reasoning: a review. *The Knowledge Engineering Review* 9: 327-354.

Weber, G. 1993. ELM: Case-based diagnosis of program code in a knowledge-based help system. In proc. of the *1st European Workshop on Case-Based Reasoning, Posters and Presentations.*

Zeleznikow, J., Hunter, D. & Vossos, G. 1993. Integrating rule-based and case-based reasoning with information retrieval: the IKBALS project. In *Proc. of the 1st European Workshop on Case-Based Reasoning*, 341-346.