A New Model-Based Automated Diagnosis Algorithm

Along Lin
Internet Business Management Department
HP Laboratories Bristol
HPL-98-26
February, 1998

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

model-based, knowledge-based, diagnosis, management and automation

The explosive growth of the Internet and the World Wide Web has made the management tasks such as configuration, monitoring, diagnosis and recovery much more complicated. An automated management system is highly needed to alleviate those tasks of system managers. Some tools use Case-Based Reasoning and Rule-Based Reasoning to do the fault diagnoses. However, they have difficulties in solving problems requiring deep level knowledge. Although some Model-Based diagnostic systems have been developed, they are mainly used to tackle problems in other domains. In this paper, a new model-based automated diagnosis algorithm is proposed.

Internal Accession Date Only

A New Model-Based Automated Diagnosis Algorithm

Along Lin

Extended Enterprise Laboratory
Internet Business Management Department
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS12 6QZ, U.K.
Tel: +44 0117 9229413 Fax: +44 0117 9229250
Email: alin@hplb.hpl.hp.com

Abstract

The explosive growth of the Internet and the World Wide Web has made the management tasks such as configuration, monitoring, diagnosis and recovery much more complicated. An automated management system is highly needed to alleviate those tasks of system managers. Some tools use Case-Based Reasoning and Rule-Based Reasoning to do the fault diagnoses. However, they have difficulties in solving problems requiring deep level knowledge. Although some Model-Based diagnostic systems have been developed, they are mainly used to tackle problems in other domains. In this paper, a new model-based automated diagnosis algorithm is proposed.

Keywords: Model-Based, Knowledge-Based, Diagnosis, Management and Automation.

1 Introduction

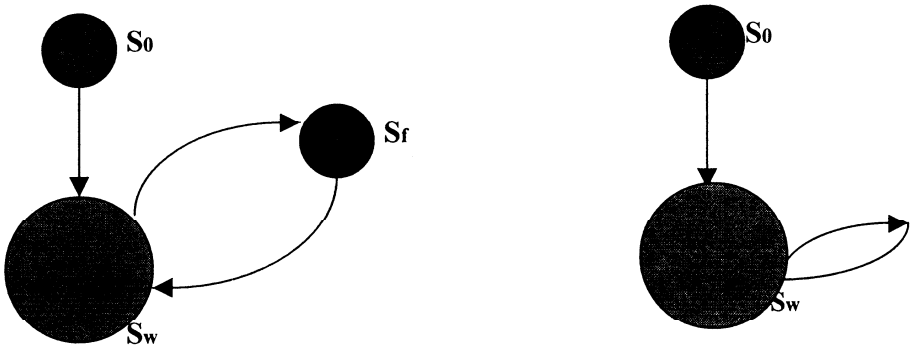
The explosive growth of the Internet and the widespread acceptance of the World Wide Web have made the network and system management much more complex. Some tools are highly needed to alleviate fundamentally the management tasks such as configuration, monitoring, diagnosis and repairing problems. An automated management system plays a very important role in providing customers with services of high quality over networks.

Model-Based Reasoning has been widely accepted as the principal diagnostic technique in several domains [3-7]. However, only little emphasis has been put on its application to network and system management domain. With Model-Based approach, it is assumed that the structure of a system and its correct behaviour have been well understood and the basic principles about how its components behave and their causalities have been given in the models of the diagnosed system. We can use that knowledge to diagnose the faults in a component by comparing its observations with the predictions about its intended behaviours based on its correct models. Because it can diagnose the faults which have not been pre-determined, it has been considered as a more promising technique than other knowledge-based methods such as Rule-Based or Case-Based systems. A Model-Based diagnostic system consists of a fault detection mechanism and a diagnostic engine. Some autonomous systems can also include a recovery mechanism. In Model-Based approach, all what we need to do is to write all of the models of domain components which describe their normal behaviours and relationships to other components.

FLIPPER is a prototype of automated management system of several networked systems which include devices like workstations, UNIX, Netware servers, PC's, printers, routers. It is based on MBR technology and developed by Hewlett- Packard Laboratories, Bristol, U.K. Models in FLIPPER define the types of objects in a managed domain, express the relationships among objects, describe methods linking objects to the real world to access the information about the managed components, which describe how an object operates. In FLIPPER, while an inference engine is used to prove goals, faults are diagnosed by an diagnostic engine, providing only the information which contributes to those faults in a managed system. In the following, we present a managed system's life cycle first. Then, a new Model Based diagnostic algorithm MBDA for our future prototype is proposed in detail in section 3. Finally, some conclusions are given in section 4.

2 The Life Cycle of a Managed System

Systems must be configured or installed correctly by using some tools before it can work. Then, the system will be in a consistent state. Unfortunately, during its run-time, there are some events occurred, causing the system to enter some incorrect states. An automated management system is needed to alleviate system administrator's task of maintaining the system in a correct state more efficiently. Reactive and proactive management are two major types of system management. The reactive one manages a system by fixing faults after they are diagnosed, whereas the proactive management tries to prevent a system from being in a faulty state by monitoring and reconfiguring a system. Both approaches need a mechanism to detect a fault or some tendency to reach the thresholds set for some parameters in advance. In Fig. 1, S_0, S_w, S_f represents the initial state, working state and fault state of the managed system, respectively. The state transitions from S_0 to S_w and from S_f to S_w are accomplished by configuring the managed system initially and by repairing the diagnosed faults, respectively. Therefore, fault diagnosis is a crucial task in the reactive system management. Obviously, it is the recovery that closes the loop and makes it possible to maintain automatically a system in a consistent and desired state.



(a) Reactively managed system which has fault states in its state space. (b) Proactively managed system which has no fault states.

Fig. 1. The two main types of life cycles of a managed system

In FLIPPER, system administrators use goals to express their management intentions. A goal represents the relationship among several managed objects. There are two special

types of goals. Monitored goals are always watched and used to trigger diagnostic engine or generate an event to signal some state change in the system to a system administrator. Therefore, the fault-detection mechanism can 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 resident goal becomes true. Those primitive actions are described as precondition/post-condition pairs in a declarative way. Thus, FLIPPER is also called a goal-directed model-based prototype of automated network and system management.

3 A New Model Based Diagnostic Algorithm MBDA

The rules in the models could be expressed by Guarded Horn Clauses. For an accurate diagnosis, some semantic information is provided in the domain object models. A rule R_i is expressed as: H_i IF $G_i \mid B_i$, where H_i , G_i , and B_i are the head, guard and body of R_i , respectively. Facts are the special rules whose bodies are empty. The sub-goals in B_i can be divided into two categories: diagnosable goals and non-diagnosable goals. Model writers are responsible for attaching the semantic tag "diagnosable" to the rules which they think are meaningful to the real diagnoses. A goal is diagnosable if one of the rules defining it is diagnosable. Furthermore, a rule is also diagnosable if its body contains a diagnosable goal. Evidently, a guarding goal G_i must not contain any diagnosable goals. For a fault diagnosis, G does not contribute to any faults in a system, it is just used to commit to some rule R. Although backtracking is allowed within G, B is deterministic.

In order to describe MBDA more concisely, we introduce following notations:
NT : $Nodes \rightarrow \{and, or, leaf\}$, where *Nodes* is the set of all nodes in a search tree;
T : $Goals \rightarrow \{true, false, unknown, ignored\}$, where *Goals* is the set of all goals, and *true*, *false*, *unknown*, *ignored* are possible goal attributes. If a goal G has been proved true/false, $T(G)$ becomes *true/false*; Otherwise, $T(G)$ remains *unknown* as a default. If $T(G)$= *ignored*, G will not be interested in any more by the underlying reasoning engine. $CP(N)$ is the choice point of the goal represented by node N, pointing to N's candidate rules. If $CP(N)$ becomes *NULL*, N has been fully expanded. Moreover, $PN(N)$ denotes the parent of node N. For a root node N, $PN(N)$ is *NULL*. Nodes in a search tree correspond to goals. MBDA consists of algorithms **3.1~3.5** detailed below.

Algorithm **3.1** **Expand**(*N*, *DE*)
1 Let *next* = *NULL*; If (*NT(N)* == *or*) {*R* = *CP(N)*; go to 7;}
2 *CP(N)* = *NULL*;
3 If *N* is an access method, call its corresponding procedure to get the attributes of the managed objects in the real world. If it fails, return(**Propagate**(*N*, *false*, *DE*)); Otherwise, return(**Propagate**(*N*, *true*, *DE*));
4 If there are no rules defined for *N*, return(**Propagate**(*N*, *false*, *DE*));
5 Let $R_1, \ldots, R_n$ be the n ($n \geq 1$) rules defined for a goal corresponding to *N*, these rules are linked together. Let *R* be the head of the linked list, *expand* = *false*; go to 7;
6 *R* = *R*→ *NEXT*;
If (*R* == *NULL*) {If (*expand* ≠ *true*) return(**Propagate**(*N*, *false*, *DE*)); return(*next*);}
7 Let *H*, *G*, *B* are the head, guard and body of *R*, respectively. If there is no such *mgu* θ that $N \bullet \theta = H \bullet \theta$, go to 6; Otherwise, if *G* is empty, go to 9;
8 If **Prove**($G \bullet \theta$) returns false, go to 6;

9 Let ***expand*** = ***true***; If (***NT(N)*** == ***leaf***) ***NT(N)*** = ***or***;
10 Construct a node ***C***, ***PN(C)*** = ***N***, θ is attached to ***C***;
11 If (***B*** is empty) {if (***next*** == ***NULL***) ***next*** = **Propagate**(***C***, ***true***, ***DE***); } Otherwise,
Assume the sub-goals in ***B*** are b_1, … , b_m , construct leaf nodes t_1, … , t_m consisting of the goal $b_1 \bullet \theta$, … , $b_m \bullet \theta$, respectively.
For $j = 1, \ldots, m$, ***T(tj)*** = *unknown*; ***NT(tj)*** = ***leaf***; ***PN(tj)*** = ***C***; If ($j < m$) ***RB***(t_j) = t_{j+1};
NT(C) = ***and***; ***T(C)*** = ***unknown***; If (***next*** == ***NULL***) ***next*** = t_1;
12 go to 6.

In algorithm **3.1**, **Prove** is the inference engine, which has the backtracking mechanism. For the sake of efficiency, we can expand ***N*** by only one rule at a time, rather than by all of its children, which can be achieved by changing step 12 to 12':
12' If (***R*→ *NEXT*** ≠ ***NULL***) ***CP(N)*** = ***R*→ *NEXT***. Return(***next***);
Actually, for any fully bound goal, only one of the rules defining it can be committed. In that case, after committing to a rule, the choice point of a goal is ***NULL***.

Algorithm **3.2** **Propagate**(***G***, ***value***, ***DE***)
1 ***NT(G)*** = ***leaf***; ***T(G)*** = ***value***;
2 Let ***N*** = ***PN(G)*** and assume ***N*** has ***n*** children $N_1, N_2, \ldots, N_n$;
3 ***T(N)*** = ***unknown***; ***E***= ***T(G)***;
4 If ((***NT(N)*** == ***or*** && ***E*** ≠ ***true***) || (***NT(N)*** == ***and*** && ***E*** == ***true***)) go to 7;
5 *T(N)* =***E***. If (***DE*** == ***diagnose***) and ***G*** is non-diagnosable, let *T(N)* = ***T(G)*** = ***ignored***;
6 **IgnoreChildren**(***N***); go to 9;
7 If (***NT(N)*** == ***or*** && ***CP(N)*** ≠ ***NULL***) return ***N***;
8 If (***T***(N_1) == ***T***(N_2) == … ==***T***(N_n) == ***T(G)***) {***T(N)*** = ***T(G)***; go to 9; }
If (***NT(N)*** == ***and*** || ***DE*** ≠ ***diagnose***) go to 9;
T(N) = ***ignored***; For $j = 1, \ldots, n$, if (***T***(N_j) == ***false***) ***T(N)*** = ***false***;
9 If (***PN(N)*** == ***NULL***) return ***N***;
If (***T(N)*** == ***unknown***) {If (***NT(N)*** == ***or***) return ***N***; return(***RB***(***N***));}
10 Let ***G*** = ***N***, go to 2.

Algorithm **3.3** **IgnoreChildren**(***N***)
1 If (***NT(N)*** == ***leaf***) { if (***T(N)*** == ***unknown***) ***T(N)*** = ***ignored***; return;}
2 Assume ***N*** has ***n*** children $N_1, N_2, \ldots, N_n$; For $j = 1, \ldots, n$, **IgnoreChildren**(N_j); return;

Both the diagnostic tree and the explanation tree of a goal ***G*** can be constructed from its search tree. The produced tree is denoted by ***DT(G)***.

Algorithm **3.4** **Construct**(***G***)
1 If (***T(G)*** == ***ignored***) return ***NULL***;
2 ***G*** is in ***DT(G)***; If (***NT(G)*** == ***leaf***) return ***G***;
3 Let G_i ($i = 1, \ldots, n$) be ***n*** children of ***G*** and ***E*** = ***T***(***G***). If (***NT(G)*** == ***or***) ***E*** = ~***E***;
4 If ***E*** is ***false***, there must be some G_j ($1 \le j \le n$) such that ***T***(G_j)== ***T(G)***, ***DT***(G_j) belongs to ***DT(G)***; Otherwise, for $i = 1, \ldots, n$, if (**Construct**(G_i) ≠ ***NULL***) ***DT***(G_i) is in ***DT(G)***;
5 Return ***DT(G).***

The diagnostic tree of a failed goal ***G*** is returned by **Diagnose**(***G***):
Algorithm **3.5** **Diagnose**(***G***)
1 Construct a leaf node ***N***, corresponding to the failed goal ***G***;

Let ***T(N)*** = ***unknown***; ***PN(N)*** = ***NULL***; ***NT(N)*** = ***leaf***;
2 While (***T(N)*** == ***unknown***) ***N*** = **Expand**(***N***, ***diagnose***);
3 Return **Construct**(***G***).

Thus, the diagnosis of the failed goal ***G*** consists of all the leaf nodes in **Diagnose**(***G***). If there are multiple faults in a system, after repairing the diagnosed fault by executing the sequence of primitive actions produced by a planner, we can diagnose others similarly. MBDA is a goal-directed model-based diagnostic algorithm and can be easily proved to be sound.

4 Conclusions

Fault diagnosis is a very important issue in an automated management system. Model Based approach appears to be a more promising technique than other knowledge-based methods such as Rule-Based and Case-Based solutions because of its ability of solving the problems which have not been pre-determined. Faults in a system can be diagnosed automatically based on the models, which describe the normal behavior of the system. However, because Model-Based approach reasons from the actual structure and behavior of a system, it is not efficient for some simple fault diagnosis which happened frequently in the past and have typical symptom/fault associations. Furthermore, obtaining domain models may be either difficult or too complicated sometimes, whereas some faults can be diagnosed based on the past experience, which can be easily captured in a rule base or case library. A possible solution is a hybrid approach which integrates all of those methods. For the faults in some components which are not well known or too complex to model, they can be diagnosed efficiently by a set of heuristic diagnostic rules or by accessing the case library to adapt the previous solutions to similar cases to the current ones. For the faults which could not be predicted or exhaustively enumerated in advance we have to use the model-based approach to do the diagnosis.

Further researches include:
- develop an efficient planner which can generate a sequence of primitive actions automatically to do the recovery of the diagnosed faults and configurations,
- integrate fault detection, diagnosis and recovery mechanisms into a single system,
- add the security mechanism to FLIPPER so that it can manage systems remotely,
- generate the component models automatically from its descriptions used to design and development by using CAD/CAM technologies.

References

[1] A. Baldwin, C. Bartolini, G.D. Vitantonio, K. Eshghi, A Novel Algorithm for Fault Diagnosis in Internet-based Services, *Workshop on OVUA*, 1997.

[2] J.S. Chen and S.N. Srihari, Candidate Ordering and Elimination in Model-Based Fault Diagnosis, In Proc. Of the 11th IJCAI, 1363-1368, 1989.

[3] R. Davis, Diagnostic reasoning based on structure and behaviour, *Artificial Intelligence,* Vol. 24, No. 1, 347-410, 1984.

[4] M.O. Hofmann, Model-Based Diagnosis Directed by Heuristic Search, *The Ninth*

Conference on Artificial Intelligence for Applications, 197-203, March,1993.

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

[6] J.de Kleer, A.K. Mackworth, R. Reiter, Characterizing Diagnosis and Systems, *Artificial Intelligence,* Vol. 56, 197-222, 1992.

[7] J.de Kleer and B.C. Williams, Diagnosing multiple faults, *Artificial Intelligence,* Vol. 32, No. 1, 97-130, April 1987.

[8] B.J., Kuipers, Qualitative Simulation, *Artificial Intelligence,* Vol. 29, 289-338, 1986.

[9] M.K. Reiter, K.P. Birman and R. van Renesse, A Security Architecture for Fault Tolerant Systems, *ACM Trans. on Computer Systems, Vol.12, No.4, 340-371,* 1994.

[10] R. Reiter, A Theory of Diagnosis From First Principles, *Artificial Intelligence*, Vol. 32 No. 1, 57-96, 1987.

[11] T. Sakao, Y. Umeda, T. Tomiyama and Y. Shimomura, Model-Based Automatic Generation of Sequence-Control Programs from Design Information, *IEEE Expert*, 54-61, 1997.

[12] W. Hamscher, L. Console and J. de Kleer, (Eds.) ***Readings in Model-Based Diagnosis***, Morgan Kaufmann, San Mateo, CA, May 1992.