

Qualitative Faults Diagnosis Algorithm: Process

He-xuan Hu^{1,2}

Agricultural and Animal Husbandry College of Tibet
University¹
Lin-zhi, Tibet, P.R. China
College of Energy and Electrical Engineering
Hohai University²
Nanjing, Jiangsu Province, P.R. China
e-mail: hexuan_hu@hhu.edu.cn

Hao-hua Li

College of Energy and Electrical Engineering
Hohai University
Nanjing, Jiangsu Province, P.R. China
e-mail: haohuali@163.com

Shi-ping Huang

College of Energy and Electrical Engineering
Hohai University
Nanjing, Jiangsu Province, P.R. China
e-mail: sph2014@126.com

Ye Zhang

College of Computer and Information Engineering
Hohai University
Nanjing, Jiangsu Province, P.R. China
e-mail: silve_fox@hotmail.com

Abstract—This paper presents the process of qualitative multi-faults diagnosis. We propose a new diagnostic process for continuous and dynamic system and its corresponding consistency-checking module. The STRIPS language is used to generate the system model since it integrates the cause-effect information required to process the diagnosis. With the STRIPS language, actions are described in terms of their preconditions and effects and states are formulated as conjunctions of positive literals. A diagnosis is established when assuming particular components to be faulty and others to be normally functioning restore consistency. It reasons about multiple faults or causes for an abnormality by testing an assumption if it leads to an inconsistency. The main contribution is to realize the qualitative multi-faults diagnosis without requiring detail and precise knowledge about faulty components and without the impossible diagnoses.

Keywords—qualitative; multi-faults; diagnosis; cause-effect; reasoning ability

I. INTRODUCTION

In this paper, we continue to present the part of isolation of qualitative multi-fault diagnosis. For the qualitative diagnosis, there are fundamentally two different approaches to search in fault diagnosis [1]: topographic search and symptomatic search. Topographic searches perform malfunction analysis using a template of normal operation, whereas, symptomatic searches look for symptoms to direct the search to the fault location. The Reiter's theory [6] belongs to symptomatic search. It is called consistency-based approach. Basically, consistency-based diagnosis amounts to finding faulty device components that account for a discrepancy between predicted normal behavior of a device and actually observed behavior. The discrepancy is formalized as logical inconsistency; a diagnosis is established when

assuming particular components to be faulty and others to be normally functioning restore consistency. It reasons about multiple faults or causes for an abnormality by testing an assumption if it leads to an inconsistency. From the definition 3 of companion paper: theory and detection, the key step in the diagnosis process is to test the consistency of the formula,

$$SD \cup OBS \cup \{AB(c) | c \in \Delta\} \cup \{\neg AB(c) | c \in COMP - \Delta\}, \quad (1)$$

in which $\Delta \subseteq COMP$ is the diagnosis for system $(SD, COMP, OBS)$ where: SD , the system description, is a set of first-order sentence; $COMP$, the system components, is a finite set of constants; OBS , the observations of a system, is a finite set of first-order sentences and AB is a predicate indicating that a component is abnormal.

As we adopt a different model form with the one adopted by Reiter, we should develop a different consistency-checking module for the formula (1). Moreover we should develop a new diagnostic process for dynamic and continuous system, with respect to the one used by Reiter for digital circuits.

The basic idea consistency-checking module is shown in Fig. 1 The consistency-checking module firstly receives the observations and hypothesis Δ (assuming particular components to be faulty and others to be normally functioning) from the principle diagnostic process. The fault models are built by using the same means for building normal model. The difference is that the fault models use the fault and normal STRIPS (Stanford Research Institute Problem Solver, [7]) actions, but the normal model only uses the normal STRIPS actions. Then the fault models produce the predictions. Finally, these predictions will be compared with the observations. If they match each other, the hypothesis Δ makes the formula (1) consistent, otherwise inconsistent. It should be noted that the fault models are used in our diagnostic process, but Reiter's approach does not use any fault

model. Why? Because there exists the impossible diagnoses. A famous example is that the light of a bulb is on although no voltage is present [2]. Reiter's approach provides an elegant and general framework for multi-fault diagnosis. However, it is lacking an important part of diagnostic reasoning: knowledge about how components may behave when they are faulty. Thus it loses the ability of generating explanations and confirming diagnosis by analyzing whether the malfunctioning of a (set of) component(s) is consistent with the observations.

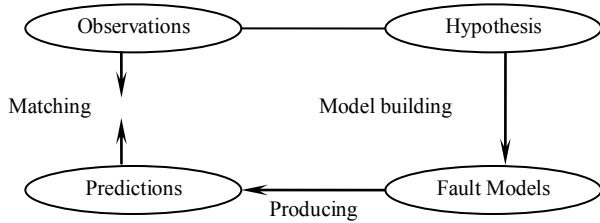


Figure 1. The basic idea consistency-checking module.

The rest of this paper is organized as follows. The fault models are introduced in section II. The new diagnostic process for dynamic and continuous system is presented in section III. The consistency-checking module corresponding to our system model is described in section IV. Section V summarizes the work done and discusses directions for future research.

II. THE FAULT MODELS

As the difference with the Reiter's approach, fault models are used in our diagnostic process. Reiter's approach provides an elegant and general framework for multi-fault diagnosis. However, it is lacking an important part of diagnostic reasoning: knowledge about how components may behave when they are faulty. Thus it loses the ability of generating explanations and confirming diagnosis by analyzing whether the malfunctioning of a (set of) component(s) is consistent with the observations. It may produce the impossible diagnoses as that the light of a bulb is on although no voltage is present [2].

A faulty component has more than one faulty behaviors, for example, a valve may be blocked in the „on“ position (i.e., cannot be closed) or be blocked in the „off“ position (i.e., cannot be opened). These different behaviors cannot be described by an unique single fault model. That is the reason why the notion of fault mode is introduced in our framework. Fault modes and effects analysis is a tool originally developed by reliability engineers. It analyses potential effects caused by simple or aggregated components ceasing to behave as intended, i.e. they stop providing the service designated to the component. A fault modes and effects analysis procedure starts with listing, for each component, in which ways can this component fail. This is referred to as “fault modes”. In this paper, a fault mode describes a same kind of faults and is used to predict effects of a fault as a fault model does.

The fault modes can be defined as a specific set of STRIPS actions. In our modeling, the system's behavior is described as an automaton which is generated by the STRIPS actions. Therefore, the system's faulty behavior

is also generated by the faulty STRIPS actions. The faulty STRIPS actions are those unobservable events which cause the faults, such as “valve stuck-closed” and “sensor short-circuited”, etc. In discrete event systems [3] and [4], a kind of fault in a component has been pre-described in the way of automaton, in which an unconditional transition represents an unobservable fault event and the destination state of this transition represents the effects of this fault. The system's faulty behavior is generated by the synchronous composition of these components' automata. In the similar way, a faulty action is defined as an unconditional STRIPS action without preconditions in which the effects of fault have been described. The difference with respect to discrete event systems is that the system's faulty behavior is progressively generated and is not necessary to generate the whole system's faulty behavior.

Definition 1: (Fault model): Let $L = \{P1, P2, \dots, Pn\}$ be a finite set of logical atoms. With this set, a fault model can be represented as a faulty STRIPS action. $A_F = \{a1, a2, \dots\}$ is a finite or recursively enumerable set of faulty STRIPS actions. Each faulty STRIPS action $a_F \in A_F$ is a multiple of subsets of L , which can be expressed as $a_F = (\text{preconditions}, \text{determin-effects}, \text{nondetermin-effects})$.

As the limitation of paper space, the example is omitted.

III. THE PRINCIPLE DIAGNOSTIC PROCESS

Once a fault symptom is detected, the fault isolation will be realized by our diagnostic process which is based on the Reiter's *HS-Dag* graph presented in section 2 of the companion paper: theory and detection. The principle diagnostic process is an incremental procedure to continuously compute a diagnosis for a system description (*SD*, *COMP*, *OBS*) and its successive new observations $\{New-OBS\}$. The entire process is illustrated in Fig. 2 and the corresponding algorithm is the algorithm 1.

Algorithm 1: The principle diagnostic process

```

Input: the current state  $s_i$ ,
       the previous state  $s_{i-1}$ ,

If HS-DAG does not exist
Then { build the root node  $n_0$  of HS-DAG with COMPONENTS;
      build the child nodes of  $n_0$ ;
      create the set  $N_L$  as the unchecked leaf nodes of the child nodes of  $n_0$ ;
Else { create the set  $N_L$  as all the unchecked leaf nodes of HS-DAG;
End If;

While ( $N_L \neq \emptyset$ ) Do
  Choose  $n_i$  of  $N_L$ ;
   $N_L \leftarrow N_L - n_i$ ;
  If (some pruning rules are available)
  Then { update the HS-DAG as Reiter's algorithm;
        add the new leaf nodes to  $N_L$ ;
  Else { call the consistency-checking module ( $s_i, n_i$ );
        If (some pruning rules are available)
        Then { update the HS-DAG as Reiter's algorithm;
              add the new leaf nodes to  $N_L$ ;
        End If;
  End If;
End While;
Return the Diagnose of this state = all the paths whose leaf node is labeled by “√”;
Clean the related marks of all normal scenario subsets (in algorithm 2);

```

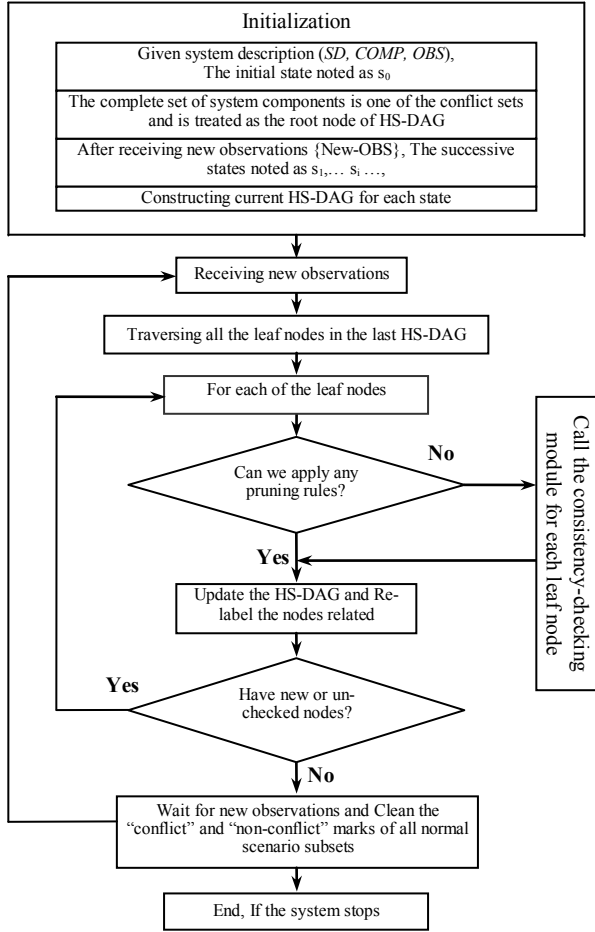


Figure 2. The principle diagnostic process.

The algorithm 1 is called each time when the current state s_i does not belong to the possible successive states $S_{np(i-1)}$ determined at last time.

- (1) In the initialization, $(SD, COMPONENTS, OBS)$ is the system description and the initial state is noted as s_0 . The successive states will be produced as the system receives new observations and they are noted as $s_1, s_2, \dots, s_k, \dots$ (explicated in step (1) of algorithm of detection in the companion paper). When the system detects that a fault has occurred (the result returned by algorithm of detection), the complete set of system components is considered as one of the conflict sets because according to the definition of conflict set, the assumption that all the system components are normal is not consistent with the actual observations. This complete set of system components is used as the root node of the graph *HS-DAG* and the initial *HS-DAG* has only this root node.
- (2) For each of the successive states s_1, \dots, s_i , the current *HS-DAG* is generated based on the last graph *HS-DAG*.
- (3) Open all the leaf nodes in the last *HS-DAG* (mark them as yet to be explored).
- (4) For each of the leaf nodes n_i in the order of increasing depths, check whether there are some pruning rules applied on this leaf node to avoid doing the consistency checking.

- (5) If there is no available pruning rule, make a call to the consistency-checking module but use s_i as the *New-OBS*. Wait until receive the result of the consistency-checking module and update the *HS-DAG* as in Reiter's algorithm. It is noted that some new nodes may be generated in this updating.
- (6) If there are some available pruning rules, update the *HS-DAG* as in Reiter's algorithm. For the new *HS-DAG*, all the new leaf nodes are marked as "unchecked".
- (7) Check whether there are any un-checked nodes. If there is no un-checked node, the diagnoses are all the paths whose leaf nodes are labeled by " \checkmark " in the final *HS-DAG* for this state. The system waits for next new observation and go to step 2 to begin a new cycle. If there are un-checked nodes, go to step 4 to continue constructing *HS-DAG* for this state.
- (8) The process will wait for new observations and will clean the "conflict" and "non-conflict" marks of all normal scenario subsets (explicated in algorithm 2: The consistency-checking module).

IV. THE CONSISTENCY-CHECKING MODULE

In Reiter's theory, a theorem prover serves as a consistency checker, but in this paper, the normal and faulty STRIPS actions can be used as the consistency checker. They seek to maintain a model whose state and faults (if any) reflect the current state of the physical system. Before presenting the algorithm of consistency-checking module, we first introduce two notions.

"Normal scenario subset, ch_i ": As the name suggests that the components in normal scenario subset are treated as the normal components and the rest components in system are assumed to be faulty or normal. A normal scenario subset is a subsets of complementary set of $H(n_i)$, the set of edge labels of node n_i , and record it in the set $CH(n_i)$. $CH(n_i)$ is a set of sets, for example, ch_i is an element of $CH(n_i)$ and itself is a set, a normal scenario subset.

"The un-checked fault mode combination set": As a component has several different fault modes and it can not be in two or more fault modes at the same time, thus each fault component takes one of their fault modes one time. For example, there are three components. The first one has two fault modes (1-1, 1-2) and its normal mode (1-N), the second one has only one fault mode (2-1) and its normal mode (2-N), and the third one has one fault modes (3-1) and its normal mode (3-N). Then there will be totally twelve fault mode combinations for these three components. Finally, we mark these combinations as un-checked and record them in the set *FMC*. We call a fault combination as "a fault scenario". The entire consistency-checking module is illustrated in Fig .3 and the corresponding algorithm is the algorithm 2.

The explanations of steps of the algorithm 2 are the following:

- (0) Clean the set $CH(n_i)$ (explicated in step 2).
- (1) Select a leaf node in a breadth first order and obtain the set of edge labels, $H(n_i)$.
- (2) List all "normal scenario subset" of node n_i in the order of increasing cardinal numbers except for the null set, mark them as un-checked and record them in the set $CH(n_i)$.

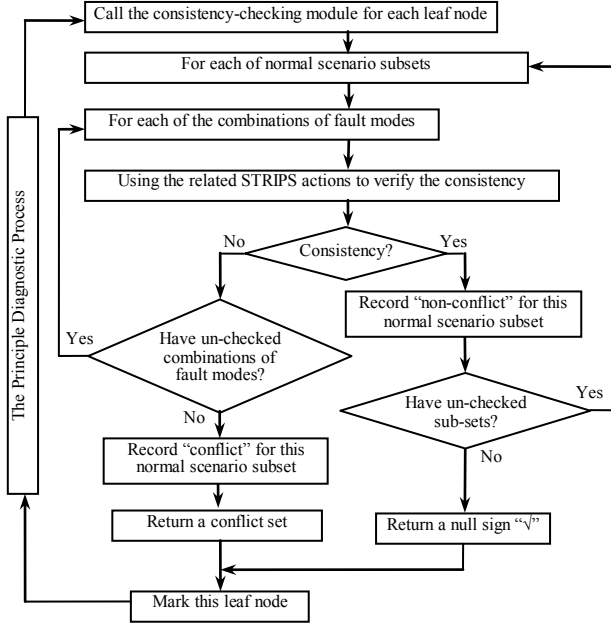


Figure 3. The consistency-checking module

Algorithm 2: The consistency-checking module

Input: the current state s_{cs} ,
the previous state s_{i-1} ,
the current node n_i

$A = false; B = false;$
/*two logic variables to control the conditions of the iteration*/

Clean the set $CH(n_i)$
Calculate the normal scenario set $CH(n_i)$ of n_i in the order of increasing cardinal numbers;
While ($CH(n_i) \neq \emptyset$) and ($A = false$) Do
 Clean the set FMC ;
 Choose an element ch_i of $CH(n_i)$ in order;
 $CH(n_i) \leftarrow CH(n_i) - ch_i$;
 Determine the set $FMC(ch_i)$;
 /* the un-checked fault mode combination set for ch_i */

 While ($FMC(ch_i) \neq \emptyset$) and ($B = false$) Do
 Choose an element fmc_j of $FMC(ch_i)$;
 $FMC(ch_i) \leftarrow FMC(ch_i) - fmc_j$;
 If ($SD \cup OBS \cup \{\neg AB(c_k) \mid c_k \in ch_i\}$ is in-consistency)
 Then { If ($FMC(ch_i) = \emptyset$)
 Then { $ch_i \leftarrow$ Conflict;
 The components in ch_i are returned as
 a conflict set for n_i ;
 $A = True$; }
 Else $j = j + 1$;
 End If;
 Else { $ch_i \leftarrow$ non-conflict;
 $B = True$;
 If ($CH(n_i) = \emptyset$) Then Return a null sign " \sqrt " for n_i ;
 End If; }
 End While;
 End If;
End While;
 $i = i + 1$;
End While

- (3) Choose a normal scenario subset ch_i . An additional operation is to clean the set FMC (the un-checked fault mode combination set, explicated in step 4).
- (4) List all the fault mode combinations of a set of components in ch_i .
- (5) With the normal component set $\{c_1, \dots, c_k\}$ in ch_i chosen in step 3 and the corresponding fault mode combination chosen in step 4, test the consistency of $SD \cup OBS \cup \{\neg AB(c_k) \mid c_k \in ch_i\}$ from the fault predictive observations computed by the actions issued and the fault events in the fault mode combination. If a parameter value takes on two distinct qualitative magnitudes or directions in the

fault predictive observations and the observed observations, an inconsistency is detected, otherwise a sign of consistency is returned. Finally this fault mode combination chosen fmc_j is marked as "checked".

- (6) If an inconsistency is detected, then check whether there are any un-checked fault mode combinations. If there is no un-checked fault mode combination, then return the chosen normal component set (normal scenario subset) ch_i as a conflict set and mark this leaf node by this conflict set. It means that all the fault mode combinations can not explicate the conflict results between the actual observations and the assumption of this normal component set. Moreover, a mark of "conflict" is recorded for this normal scenario subset temporarily. If there are un-checked fault mode combinations, then go to step 4.
- (7) If a consistency is returned, then a mark of "non-conflict" is recorded for this normal scenario subset temporarily and the process will check whether there is any un-checked normal scenario subset in $CH(n_i)$. If there is no un-checked subset, then return a null sign " \sqrt " and mark this leaf node by it. It means that there is no conflict set under this leaf node and the assumption that the components in $H(n_i)$ are faulty can explicate the existing situation. If there are un-checked subsets in $CH(n_i)$, then go to step 3.

As the limitations of the space, we ignore the example of application and some details.

V. CONCLUSION

This paper continues to present the framework of multi-faults diagnosis from the preceding paper. There are two main contributions in this part. One is aimed at developing a new diagnostic process for continuous and dynamic system and its corresponding consistency-checking module. This diagnostic process and its consistency-checking module are all based on the models defined by STRIPS actions. The other is that the fault models are introduced into the consistency-checking module for preventing the impossible diagnoses. The STRIPS can qualitatively define the fault models without requiring detail and precise knowledge about faulty components. Although we have proposed several skills to improve the efficiency of our algorithm, there are still a lot of works to do in the future. The model and its building method that we used here are also used in the reconfiguration [5, 8-10]. We hope that this multi-faults framework and the reconfiguration can be well integrated into a supervisory control system.

ACKNOWLEDGMENT

This work is supported by "The Nature Science Foundation of Tibet", "A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (Coastal Development Conservancy)", "Technology Foundation for Selected Overseas Chinese Scholar, Ministry of Personnel of China", "the Fundamental Research Funds for the Central Universities", and "the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry".

REFERENCES

- [1] J. Rasmussen, *Information processing and human-machine interaction*, North Holland, New York, 1986.
- [2] P. Struss and O. Dressler, "Physical negation: integrating fault models into the general diagnostic engine," In: *Readings in Model-based Diagnosis*. Morgan Kaufmann Publishers, San Mateo, CA, 1992.
- [3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Automatic Control*, Vol. 40, No. 9, pp. 1555–1575, 1995.
- [4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems (Second Version)*, Kluwer Academic Publishers, 2007.
- [5] H. X. Hu, A. L. Gehin, and M. Bayart, "A Formal Framework of Reconfigurable Control Based on Model Checking," in *American Control Conference*, Seattle, Washington, USA, 2008.
- [6] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, Vol.32, No. 1, pp. 57-95, 1987.
- [7] R. E. Fikes and N. J. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, Vol. 2, Issues 3-4, pp. 189-208, 1971.
- [8] H. X. Hu, A. L. Gehin, and M. Bayart, "An extended qualitative multi-faults diagnosis from first principles I: theory and modelling", in *48th IEEE Conference on Decision and Control*, China, 2009.
- [9] H. X. Hu, A. L. Gehin, and M. Bayart, "An extended qualitative multi-faults diagnosis from first principles II: algorithm and case study," in *48th IEEE Conference on Decision and Control*, China, 2009.
- [10] A. L. Gehin, H. X. Hu, and M. Bayart, "A self-updating model for analysing system reconfigurability", *Engineering Applications of Artificial Intelligence*, Vol.25, Issue 1, pp.20-30, 2012.