Deriving Minimal Conflict Sets by CS-trees with Mark Set in Diagnosis from First Principles

Benjamin Han and Shie-Jue Lee

***Abstract*—To discriminate among all possible diagnoses using Hou's theory of measurement in diagnosis from first principles [5], one has to derive all minimal conflict sets from a known conflict set. However, the result derived from Hou's method depends on the order of node generation in CS-trees. We develop a derivation method with mark set to overcome this drawback of Hou's method. We also show that our method is more efficient in the sense that no redundant tests have to be done. An enhancement to our method with the aid of extra information is presented. Finally, a discussion on top-down and bottom-up derivations is given.**

***Index Terms*—Measurement, minimal hitting set, observation, system description.**

I. Introduction

Under the assumption that the description of a digital system is consistent, if the inputs and the outputs, i.e., the observation, of the system conflict with the way the system is meant to behave, the diagnostic problem is to pinpoint the possible diagnoses, i.e., the possible sets of faulty components that cause the problem. Many researchers have proposed various kinds of approaches to tackle the problem [2], [8]; among them, Reiter [7] has built and formalized the major theorems of diagnosis from first principles upon the work of de Kleer [6] and Genesereth [4]. In [7], Reiter gave the following definitions:

Definition 1.1 [7, Definition 2.1]: A *system* is a pair (SD, COMP) where

1) SD, the *system description,* is a set of first-order sentences describing

   a) the functionality of a component within the system. For example, let A be an AND gate with two inputs, the sentence describing it is:

   $$\text{AB}(A) \rightarrow \text{out}(A) = \text{and}(\text{in1}(A), \text{in2}(A))$$

   where the predicate AB stands for *abnormal,* which is true iff gate A is malfunctioning, "and" is a function returning true iff both inputs of gate A are true, and "$\rightarrow$" stands for *implication.*

   b) the connections among the components of the system. For example, out(A) = in 1(B) means that the output of component A is connected with the first input of component B.

2) COMP is a finite set of constants denoting the collection of components of the system.

Real world diagnostic settings involve observations. Observations allow us to determine whether something is wrong and hence whether a diagnosis is called for. An *observation* of a system is a finite set of first-order sentences describing the values observed at the inputs/outputs of the whole system. We shall write (SD, COMP, OBS) for a system (SD, COMP) with observation OBS.

Manuscript received March 4, 1997; revised June 21, 1998. This work was supported in part by the National Science Council under Grant NSC-85-2213-E-110-037. This paper was recommended by Associate Editor L. O. Hall.

The authors are with the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 80424, R.O.C. (e-mail: leesj@ee.nsysu.edu.tw).

Publisher Item Identifier S 1083-4419(99)00771-2.

Definition 1.2 [7, Prop. 3.4]: Δ is a *diagnosis* for (SD, COMP, OBS) iff Δ is a minimal set such that SD $\cup$ OBS $\cup \{\neg\,\text{AB}\,(c) | c \in$ COMP $- \Delta\}$ is consistent.

Definition 1.3 [7, Definition 4.1]: $C \subseteq$ COMP is a *conflict set* (CS) for (SD, COMP, OBS) iff SD $\cup$ OBS $\cup$ $\{\neg\,\text{AB}\,(c) | c \in C\}$ is inconsistent. A *minimal conflict set* (MCS) is a CS such that none of its subsets is a CS.

Note that we need to do consistency checking to test whether a given set is a conflict set. An inference engine such as de Kleer's ATMS [2] or Davis–Putnam's procedure [1] can do the job. In this paper, we shall not tie ourselves to a particular inference mechanism. Consistency checking is an NP-complete problem [3] and shall be considered expensive and be avoided whenever possible.

Definition 1.4 [7, Definition 4.3]: H is a *hitting set* (HS) for a collection of sets S iff $H \subseteq \bigcup_{C \in S} C$ such that $H \cap C \neq \emptyset$ for each $C \in S$. A *minimal hitting set* (MHS) is an HS such that none of its subsets is an HS.

Theorem 1.5 [7, Corollary 4.5]: $\Delta \subseteq$ COMP is a diagnosis for (SD, COMP, OBS) iff Δ is an MHS for the collection of MCS's for (SD, COMP, OBS).

To locate all possible diagnoses, we have to find all MHS's for the collection of MCS's.

A set of diagnoses for a system can be refined by taking a measurement, Π, from the system. By taking a measurement we mean to probe the input or the output of some component within a system. Usually it requires "opening the hood" of the system, and thus is considered at least inconvenient. In [5], Hou develops an approach to refine the diagnoses incrementally on every Π [5, Theorem 3.13]. To refine the diagnoses, one has to keep those diagnoses predicting Π and recompute those predicting $\neg\Pi$. For a diagnosis Δ_i predicting $\neg\Pi$, one determines new diagnoses from the collection of candidates $\Delta_i \cup h_j$, where h_j is an MHS for the collection of MCS's derived from COMP-Δ_i. To determine which candidate is accepted, one needs to check if it is minimal against other diagnoses and candidates.

Example 1.6: Consider the circuit shown in Fig. 1. Obviously, the output of O_1 shows inconsistency if we assume that each gate of $\{A_1, A_2, A_3, O_1, O_2\}$ is normal, therefore we have $\{A_1, A_2, A_3, O_1, O_2\}$ as the initial CS. To calculate all possible diagnoses, we have to somehow derive all MCS's from the initial CS. The most straightforward way to do it is to test each and every proper subset of the initial CS to see if it is also a CS. This most straightforward way shall be proved too inefficient, and both Hou's method and our approach aim to enhance the efficiency of the derivation. A moment of reflection shall give us all MCS's$\{A_1, O_1\}$, $\{A_2, O_1\}$, $\{A_3, O_1\}$, and $\{O_1, O_2\}$. The corresponding MHS's are then $\{O_1\}$ and $\{A_1, A_2, A_3, O_2\}$, which are all the possible diagnoses of the circuit given the information so far. □

To construct a practical diagnostic system, developing a complete and efficient method for deriving MCS's from a CS is a key issue. In such a method the number of tests of consistency should be minimized without the risk of missing any potential MCS. Hou [5] proposed an approach to do the job. However, the result derived from Hou's method depends on the order of node generation in CS-trees. In Section II, we describe Hou's method for the derivation and show two examples in which some MCS's are missing from the derivation. In Section III, we develop a method with mark set for

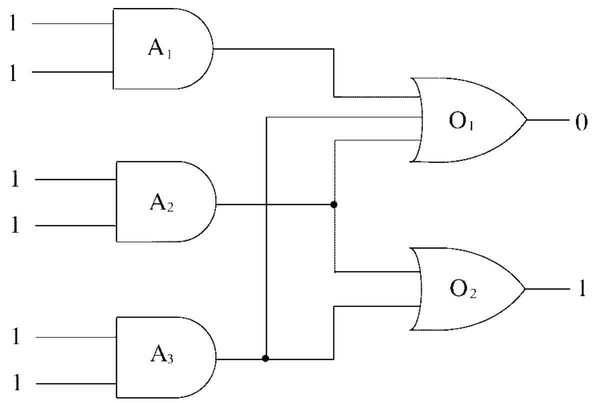

Fig. 1. An example circuit about MCS's and MHS's.

deriving MCS's from a CS, illustrate the method with an example, and prove that the method is correct and more efficient in the sense that no redundant subsets are tested. In Section IV, we show one possible enhancement to our derivation method by exploiting extra information about the system being diagnosed. Finally, in Section V we describe a bottom-up derivation strategy, compare it with the top-down method described in Section III, and conclude this paper with discussions on possible extensions to our method in Section VI.

II. Hou's Method for Deriving Minimal Conflict Sets

Definition 2.1 [5, Definition 5.1]: A CS-tree T rooted in a CS C is defined as follows.

1) Its root is labeled by C.
2) Each node n of T is labeled by a set $S \subseteq C$. For each $c \in S$, node n has a descendant n_c such that node n_c is labeled by a nonempty set $S - \{c\}$.

Note that the definition here is slightly different from [5, Definition 5.1] in that we view the closing of a node labeled by a non-CS as a pruning rule [in pruning rule (c), Procedure 2.2] instead of part of the definition of a CS-tree. This viewpoint facilitates the comparison between Hou's method and our approach. Also, we preclude empty sets from being used for labeling any node in a CS-tree. To derive all MCS's from C, a pruned CS-tree T' rooted in C is generated by the following procedure.

Procedure 2.2 [5, p. 312]: Generate a pruned CS-tree T' rooted in a CS C by the following rules.

1) Generate T' depth-first, i.e., generate the descendants of a node before generating its brothers.
2) Pruning rules:
   a) Let node n be some node already generated in T' with the label set S. If node n' is a new node such that n' will be labeled by S, then we close node n'.
   b) If a nonroot node n will be labeled by a set S and S is a proper superset of some MCS's already used for labeling some node in T', then we close node n.
   c) If node n is labeled by a non-CS, then we close node n.

It turns out that in some occasions pruning rule b) closes some nodes too soon, and may introduce errors to a diagnostic system. Consider the following example.

Example 2.3: Let $C = \{c_1, c_2, c_3, c_4\}$ and suppose that all the MCS's we can derive from C are $\{c_1\}$, $\{c_2, c_3\}$, and $\{c_2, c_4\}$. Fig. 2 shows the derivation of the MCS's from C by Hou's method. Note that we use $\langle\cdot\rangle$ to denote a CS and $(\cdot)$ to denote a non-CS in the figure. Also, we underline a set to denote an MCS. The mark $*$ on a node n indicates that n is closed due to the use of pruning rule a),

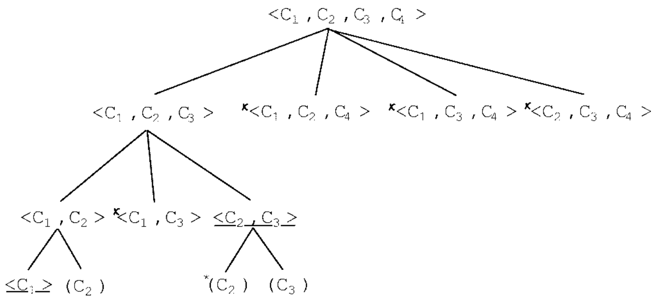

Fig. 2. Deriving the MCS's from C, using Hou's method.

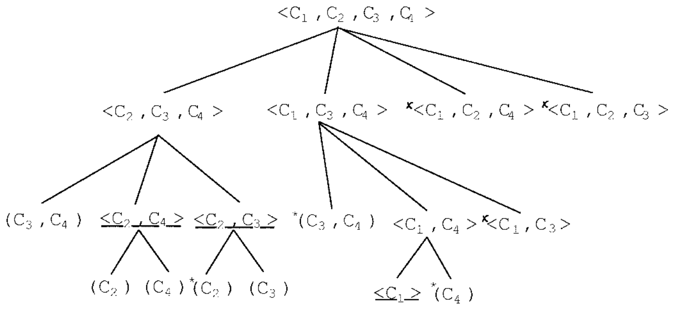

Fig. 3. Deriving the MCS's from C (alternative order), using Hou's method.

and the mark x on a node n shows that n is closed due to the use of pruning rule b). Note that $\{c_2, c_4\}$ is lost during the derivation due to the use of pruning rule b).

Note that if we change the order of generating the nodes in the CS-tree as shown in Fig. 3, the MCS $\{c_2, c_4\}$ would be derived. Obviously, the result obtained from Hou's method depends on the order of node generation in CS-trees. □

Example 2.4: Consider Example 1.6. A correct method should derive four MCS's: $\{A_1, O_1\}$, $\{A_2, O_1\}$, $\{A_3, O_1\}$, and $\{O_1, O_2\}$. However, as shown in Fig. 4, $\{A_3, O_1\}$ and $\{O_1, O_2\}$ are lost using Hou's method. The diagnoses found according to the wrong result are $\{O_1\}$ and $\{A_1, A_2\}$, where $\{A_1, A_2\}$ is obviously an erroneous diagnosis. □

III. A Method with Mark Set for Deriving Minimal Conflict Sets

Hou's method closes a node whenever the label of it is a superset of some MCS's already discovered. This results in closing a node too soon and making some MCS's lost in some cases. We propose a method with mark set to overcome this disadvantage. In addition, we show that our method is more efficient than Hou's approach in the sense that no redundant tests are made. First of all we define a CS-tree with mark set as follows.

Definition 3.1: A *CS-tree with mark set* T_M rooted in a CS C is defined as follows.

1) Its root is labeled by $[C, \emptyset]$.
2) Each node n of T_M is labeled by $[S_n, S_{M,n}]$, where $S_n \subseteq C$ is the *label set* of node n, and $S_{M,n} \subseteq S_n$ is the *mark set* of node n. If $S_{M,n} = S_n$ or $|S_n| = 1$ (hence we preclude empty sets from being the label sets of any node in a CS-tree with mark set), then node n has no descendants. Otherwise for each $c \in S_n - S_{M,n}$, node n_c is a descendant of node n such that node $n_{c'}$ is the *immediate* left brother of node n_c and node n_c is labeled by $[S_{M,n} \cup (S_n - S_{M,n} - \{c\}), S_{M,n_{c'}} \cup \{c'\}]$.
3) For the leftmost node n in every subtree of T_M, $S_{M,n} = S_{M,n_p}$, where n_p is the parent node of node n.

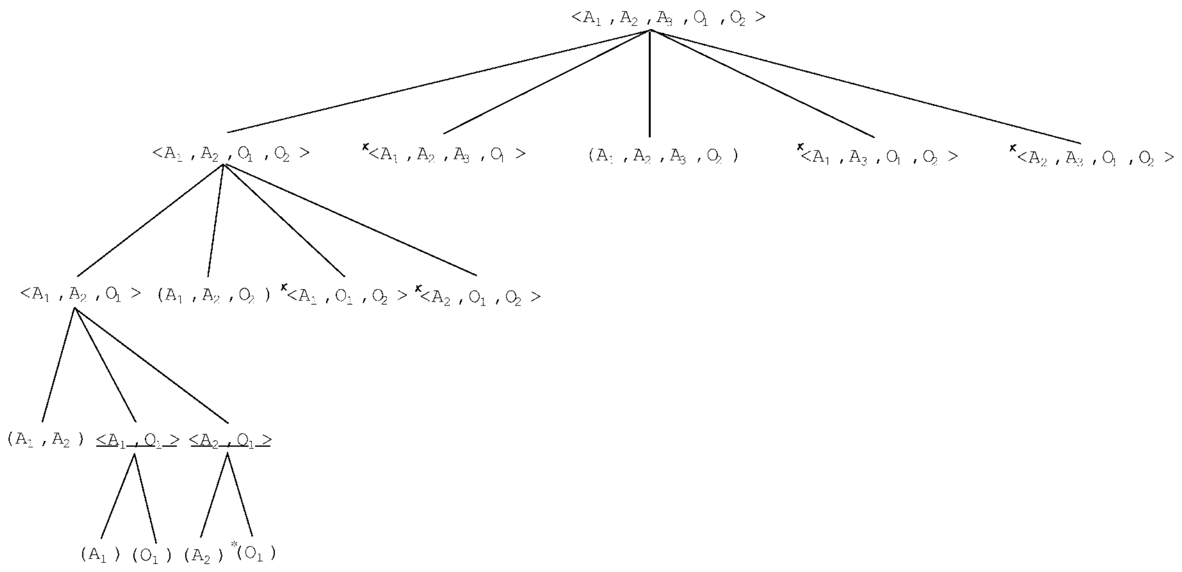

Fig. 4. Deriving all MCS's for the circuit in Fig. 1 using Hou's method.

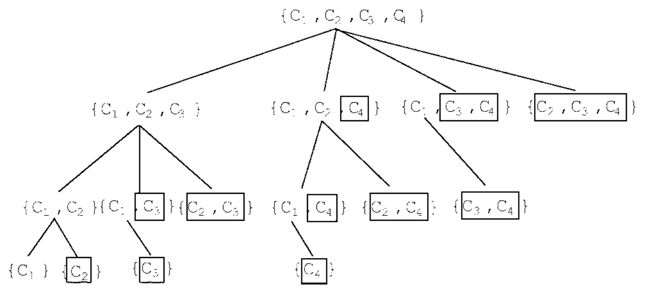

Fig. 5. A CS-tree with mark set rooted in $\{c_1, c_2, c_3, c_4\}$.

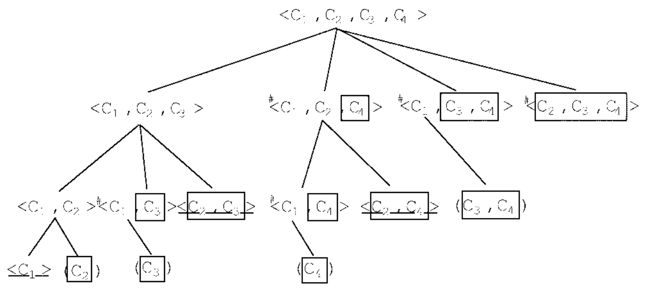

Fig. 6. Deriving the MCS's from C, using our method.

The intuition of the use of "mark sets" is straightforward. In a T_M with the root node labeled by $[C, \emptyset]$, for the leftmost descendant node n_1 labeled by $[C - \{c_1\}, \emptyset]$ we can derive all proper subsets of $C - \{c_1\}$ beneath the node. For the immediate right adjacent node n_2 of n_1, whose label set is $C - \{c_2\}$, to avoid generating duplicate sets when generating the proper subsets of $C - \{c_2\}$, i.e., to avoid generating some proper subset S' of $C - \{c_2\}$ such that $S' \subset C - \{c_1\}$, we mark c_1 and include c_1 first in every subset we generate. This way we are ensured that no node beneath node n_2 will be labeled by a set already used for labeling some node beneath node n_1.

Example 3.2: Let $C = \{c_1, c_2, c_3, c_4\}$. The CS-tree with mark set for C is shown in Fig. 5. The mark set of each node is denoted by putting a box around the elements it contains. For example, the label set of the node (c_1, c_3, c_4) in Fig. 5 is $\{c_1, c_3, c_4\}$, and the mark set of the node (c_1, c_3, c_4) is $\{c_3, c_4\}$. In fact, the mark set of a node includes the mark set of the immediate left brother of the node, and the element which is not contained in the label set of the immediate left brother of the node. For instance, the mark set of the node (c_2, c_4) includes c_4 which is the element in the mark set of the node (c_1, c_4), and c_2 which is not contained in the label set of the node (c_1, c_4).

To derive all MCS's from C, we generate a pruned CS-tree with mark set T'_M rooted in C by the following procedure.

Procedure 3.3: Generate a pruned CS-tree with mark set T'_M rooted in a CS C by the following rules:

1) Generate T'_M depth-first, i.e., generate the descendants of a node before generating its brothers.

2) Pruning rules:

   a) If $S_{M,n}$ of node n is a superset of some MCS already used as the label set of some node in T'_M, then we close node n and do not generate any right brothers of node n.

   b) If S_n of node n is not a CS, then we close node n.

Note that not every label set requires a call for consistency checking to test if it is a CS because a superset of a known CS is a CS. Also note that for a node n labeled by a CS C, if n does not have any descendant or all of its descendant nodes are labeled by non-CS's, C is not necessarily minimal—a check must be done to see if it is minimal against all MCS's already found.

Let us illustrate Procedure 3.3 by deriving the MCS's from the same set in Example 2.3.

Example 3.4: Let $C = \{c_1, c_2, c_3, c_4\}$ and suppose that all the MCS's we can derive from C are $\{c_1\}$, $\{c_2, c_3\}$, and $\{c_2, c_4\}$ (the same as those in Example 2.3). Fig. 6 shows the derivation using our method, with the same order of node generation as in Fig. 2. Note that our method is complete. The MCS $\{c_2, c_4\}$ is derived, in contrast to the loss in Fig. 2 by Hou's method. During the derivation we generated 15 nodes but only did nine tests (the label set of the root node is a CS without doubt). The "$\sharp$" mark beside some nodes in the tree denotes that the corresponding label set is determined to be a CS without the need of a test.

Fig. 7 shows the derivation using our method, with the same order of node generation as in Fig. 3. Note that our method is not only complete, but also more efficient than Hou's method in the sense

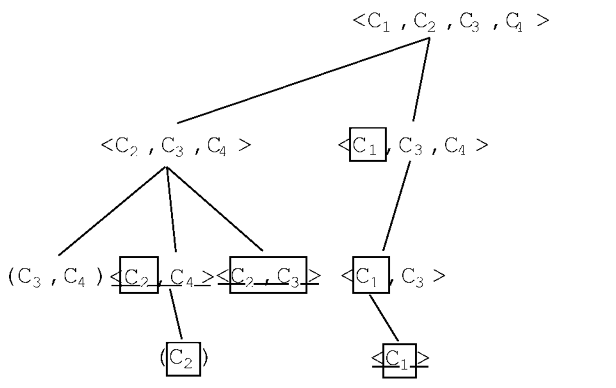

Fig. 7. Deriving the MCS's from C (alternative order), using our method.

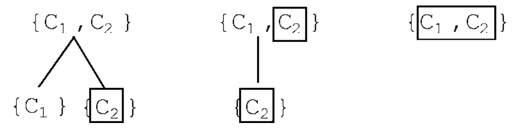

Fig. 8. Three possible subtrees in a CS-tree with mark set rooted in $\{c_1, c_2\}$.

that no duplicate set and no subset of any known non-CS is used for labeling nodes. There are nine nodes generated and eight tests done in Fig. 7, in contrast to 17 nodes and ten tests in Fig. 3.

Since all possibilities of generating the nodes labeled by duplicate sets are implicitly expelled by using the notion of mark sets, no node is labeled by $*$ in Figs. 6 and 7. □

Now we give a formal proof of the correctness of our method. First we give a corollary directly from Definition 3.1.

Corollary 3.5: The label set of each descendant and right brother of node n is a superset of the mark set of node n.

Next we show that all nonempty proper subsets containing set $S_{M,n}$ of a set S_n can be generated in the CS-tree with mark set rooted in a node labeled by $[S_n, S_{M,n}]$.

Theorem 3.6: For a node n labeled by $[S_n, S_{M,n}]$, each and every nonempty proper subset S'_n of S_n such that $S_{M,n} \subseteq S'_n$ is the label set of some node beneath the node n.

Proof: We prove it by induction on $|S_n|$. If $|Sn| = 2$, say $|Sn| = \{c_1, c_2\}$, then Fig. 8 shows that the theorem holds.

Assume that the theorem holds when $|S_n| = k - 1$. Consider the case $|S_n| = k$. If $|S_{M,n}| = 0$, the node n has k descendant nodes, and each node has a label set of size $k - 1$. According to the induction hypothesis, we then know that each and every nonempty proper subset S'_n of S_n is used to label a node in the CS-tree with mark set rooted in the node n. Now consider the case $|S_{M,n}| > 0$. Let $c \in S_{M,n}$. From Corollary 3.5 we know that each and every node in the tree is labeled by a set containing c. Therefore we can strip c from each and every node in the tree and obtain a new tree with the root node n labeled by a set of size $k - 1$. From the induction hypothesis, we know that each and every nonempty proper subset S'_n of S_n such that $S_{M,n} - \{c\} \subseteq S'_n$ must be used to label some node beneath the node n. Adding c back to each and every node completes the proof. □

From the above theorem, if we generate a CS-tree with mark set rooted in a node labeled by $[C, \emptyset]$, we have each and every possible nonempty proper subset of C in the tree. Next, we show that the pruning rules in Procedure 3.3 are correct. Since any subset of a non-CS is not a CS, we can safely prune away those nodes with non-CS's as their label sets, hence pruning rule b) is correct. The correctness of pruning rule a) is a direct result from Corollary 3.5, as stated in the following corollary.

Corollary 3.7: Let T'_M be a pruned CS-tree with mark set. If the mark set of a node, n, is a superset of some MCS P already used for labeling some node in T'_M, then the label set of each descendant and right brother of the node n is a CS.

Corollary 3.5, Theorem 3.6, and Corollary 3.7 together prove the correctness of our method. Now we consider the efficiency of our approach. For comparison, note that Hou's method for the derivation will generate the nodes with duplicate labels. For example, the node labeled by $\{c_2\}$ in Fig. 2 and the nodes labeled by $\{c_2\}$, $\{c_4\}$, and $\{c_3, c_4\}$ in Fig. 3 are duplicates, respectively, in the two derivations. In contrast, our method prevents any node labeled by a duplicate set from being generated by exploiting the notion of mark sets, hence no test for duplicates is necessary in our approach. The following theorem states that no duplicate label set exists in a CS-tree with mark set.

Theorem 3.8: Let T_M be a CS-tree with mark set rooted in a CS C. If $C' \subset C$, then only one node n exists in T_M such that C' is the label set of node n.

Proof: We show it by induction on $|C|$. If $|C| = 2$, say $C = \{c_1, c_2\}$, then by Definition 3.1 each of $\{c_1\}$ and $\{c_2\}$ is used as a label set of one node in T_M, as already shown in Fig. 8 (the left tree).

Now assume that the theorem is valid when $|C| = k - 1$, where C is a CS. When $|C| = k$, $C = \{c_1, \cdots, c_k\}$ is the label set of the root node n. By Definition 3.1 we know that n has k descendant nodes $n_1, \cdots, n_k$ whose label sets are $C_i = C - \{c_i\}$, $i = 1, \cdots, k$, respectively. If $C' \subset C_q$, by the induction hypothesis, the subtree rooted in C_q has only one node with C' as its label set. From Definition 3.1 and Corollary 3.5, we know that the label set of each and every right brother of node n_q must contain c_q. Since $c_q \notin C'$, no node in the subtrees rooted in the right brothers of n_q has C' as its label set.

On the other hand, we need to prove that no node in the subtrees rooted in the left brothers of n_q has C' as its label set. Assume that the subtree rooted in node n_p has a node with C' as its label set, with n_p being a left brother of n_q. Since C_p does not contain c_p, neither does C'. However, since n_q is a right brother of n_p, from Definition 3.1 and Corollary 3.5 C_q and all the label sets of the descendants of n_q must contain c_p, so does C'. We have a contradiction. □

Another improvement to our method is: if C' is not a CS, then no node with $C'' \subseteq C'$ as its label set will be generated in a pruned CS-tree with mark set. In contrast, Hou's pruned CS-tree does have some node labeled by the subset of some known non-CS. For example, the nodes labeled by $\{c_3\}$ and $\{c_4\}$ in Fig. 3 are redundant.

In summary, our method has efficiency edges over Hou's approach in that we do not generate any node with a duplicate set or a subset of a known non-CS as its label set, while Hou generates them and prunes them away afterward.

IV. Derivation with Extra Information

In this section we show that with the aid of extra information about the system being diagnosed, we can further improve the efficiency of our derivation method. First we define an input/output of a component of the system being diagnosed to be *fixed* if it is either in the observation of the system, or it is one of the previous measurements taken.

Example 4.1: Consider the faulty circuit shown in Fig. 9. The actual value of the input/output of each gate is shown in the figure. Also, the shaded gates are the ones causing the system to misbehave. The two inputs of gate A_2 are fixed because they are part of the system observation. If we make a measurement at out(A_2), then both inputs of gate A_4 are fixed as well. □

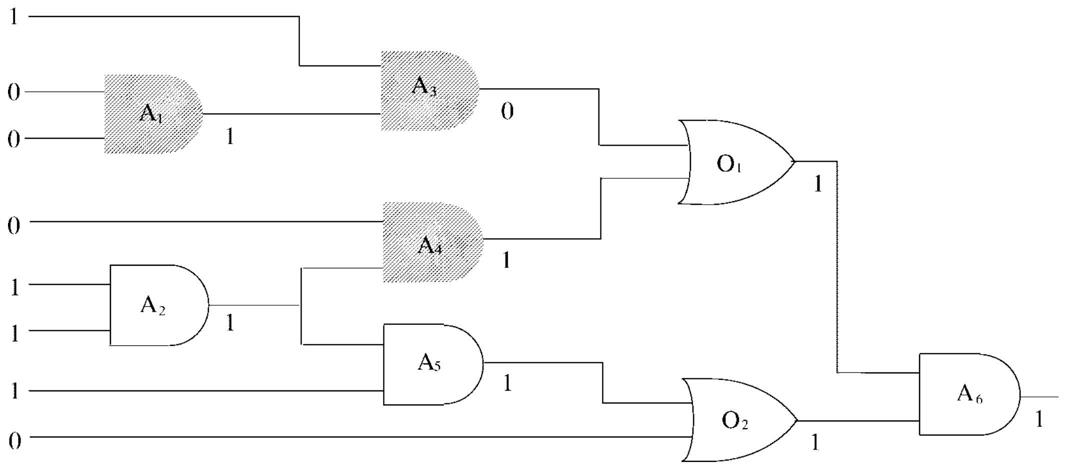

Fig. 9. A faulty circuit.

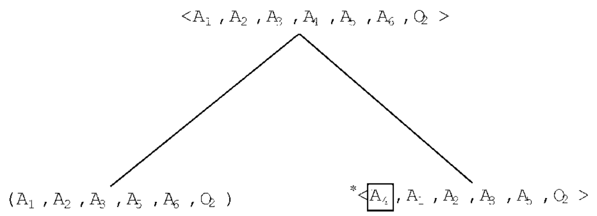
<A1 , A2 , A3 , A4 , A5 , A6 , O2 >
(A1 , A2 , A3 , A5 , A6 , O2)
*<A4 , A1 , A2 , A3 , A5 , O2 >

Fig. 10. Deriving the MCS's from $\{A_1, A_2, A_3, A_4, A_5, A_6, O_2\}$.

Now suppose Π is a measurement at the output of a component c such that all inputs of c are fixed, and Δ is a diagnosis for (SD, COMP, OBS) predicting $\neg\Pi$ such that $c \notin \Delta$, then by Definition 1.1 we know that SD $\cup$ OBS $\cup \{\Pi\} \cup \{\neg\text{AB}(c)\}$ is inconsistent. Formally speaking, we have the following corollary.

Corollary 4.2: Let Π be a measurement at the output of a component c such that all inputs of c are fixed, and let Δ be a diagnosis for (SD, COMP, OBS) predicting $\neg\Pi$ such that $c \notin \Delta$. Then $\{c\}$ is an MCS for (SD, COMP, OBS $\cup \{\Pi\}$) resulting from Π.

The above corollary offers an efficiency improvement as illustrated in the following example.

Example 4.3: Consider the faulty circuit shown in Fig. 9. After considering the observation only, we have five diagnoses: $\{A_1\}$, $\{A_3\}$, $\{A_4\}$, $\{A_6\}$, and $\{O_1\}$. Assume that we have taken a measurement at out(A_2). Now we take a measurement, Π, at out (A_4). It then turns out that diagnosis $\{O_1\}$ predicts $\neg\Pi$, and we have to derive all MCS's from $\{A_1, A_2, A_3, A_4, A_5, A_6, O_2\}$. Now by Corollary 4.2 we know that $\{A_4\}$ is an MCS for (SD, COMP, OB $\cup \{\Pi\}$) resulting from Π without any test to determine if it is an MCS. Moreover, we can arrange the order of generating the nodes in the pruned CS-tree with mark set as shown in Fig. 10. In the derivation, only two nodes are generated in the tree. Without the use of Corollary 4.2, 17 instead of two nodes would be generated. $\square$

Note that despite the great efficiency improvement brought by Corollary 4.2, one has to maintain extra information indicating whether a particular connection is fixed.

V. Top-Down Versus Bottom-Up

The methods described in Sections II and III are top-down approaches. The MCS's of a CS C are derived by creating the CS-tree rooted in C itself. We can also derive the MCS's of C in a bottom-up manner by creating the inverse CS-tree rooted in $\emptyset$.

Definition 5.1: An inverse CS-tree T_I of an ordered CS C is defined as follows.

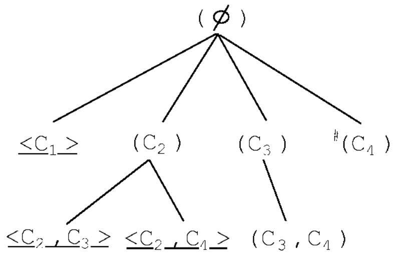

Fig. 11. Deriving the MCS's from C, using an inverse CS-tree.

1) Its root is labeled by $\emptyset$.
2) Let n be a node in T_I, n_p be the parent node of node n and n' be the left brother of node n. Then node n is labeled by an ordered set S_n such that $S_n = S_{n_p} \cup c$, where c is the next adjacent element of c' in C and c' is the last element in $S_{n'}$.
3) For the leftmost node n in every subtree of T_I, $S_n = S_{n_p} \cup c$, where n_p is the parent node of node n, c is the next adjacent element of c_p in C and c_p is the last element in S_{n_p}. If $S_{n_p} = \emptyset$, c is the first element of C.

To derive all MCS's from an ordered CS C, we generate a pruned inverse CS-tree by the following procedure.

Procedure 5.2: Generate a pruned inverse CS-tree T'_I of an ordered CS C by the following rules.

1) Generate T'_I depth-first, i.e., generate the descendants of a node before generating its brothers.
2) Pruning rule: If S_n of node n is a CS, then we close node n.

Note that not every label set requires a call for consistency checking to test if it is a CS because a superset of a known CS is a CS. On the other hand, if a label set is a subset of some known non-CS, it must not be a CS, either.

Example 5.3: Let $C = \{c_1, c_2, c_3, c_4\}$ and suppose that all the MCS's we can derive from C are $\{c_1\}$, $\{c_2, c_3\}$, and $\{c_2, c_4\}$ (the same as those in Example 2.3). Fig. 11 shows the derivation using an inverse CS-tree. The "$\sharp$" mark beside (c_4) denotes that the label set $\{c_4\}$ is determined to be a CS without the need of a test. $\square$

One might get the impression from the above example that the bottom-up approach is more efficient than the top-down approach. However, such conclusion is not necessarily true for all cases. Consider the case in which we have to derive all MCS's from a CS C with n elements, and assume that all we can derive from C is an MCS with m elements ($m \leq n$). Therefore among the 2^n subsets of C we have 2^{n-m} CS's, and the probability of having a

CS as the label set of a node, $P_{\rm CS}$, is $1/2^m$. Since we need a CS to close a node when using the bottom-up method, the lower $P_{\rm CS}$ the worse the efficiency of the bottom-up method. On the other hand, we need a non-CS to close a node when using the top-down method, lower PCS could improve the efficiency of the top-down approach. In general, whether the top-down method has an efficiency edge over the bottom-up approach depends on the particular problem to be solved.

VI. Concluding Remarks

In summary, our method has the following two improvements over Hou's approach.

1) Our method is independent of the order of node generation, while Hou's approach is dependent on the order of node generation due to the use of pruning rule b).
2) By exploiting the concept of mark set, we implicitly discard any possibility of generating the nodes labeled by duplicate sets or the subsets of some known non-CS. The counterpart of the duplicate test is done in pruning rule a) in Hou's method, which requires a search in the CS-tree, and no facility for testing the subset of some known non-CS is available in Hou's approach. Hence our method achieves better efficiency than Hou's approach.

To conclude this paper, we point out one possible extension to this work. As shown in Examples 3.8 and 4.3, by carefully arranging the order of generating nodes in a CS-tree with mark set, the efficiency of the derivation could be dramatically improved. This makes one wonder if one can find an optimal ordering of node generation in a CS-tree with mark set if one knows all MCS's before the derivation. If it is true, then we may use the error rate of each of the components within the system being diagnosed to "approximate" such optimal ordering by first calculating all possible MCS's based on some assigned error threshold. Then we may use the result to guide us to generate nodes in a more efficient order. Moreover, we may discard some node in the CS-tree with mark set if we determine that all possible MCS's found beneath that node are not important enough (again based on some assigned error threshold). In this way, the diagnostic system will not be complete, but its efficiency could be greatly improved.

Acknowledgment

The authors wish to thank the anonymous referees for their useful comments.

References

[1] M. Davis and H. Putnam, "A computing procedure for quantification theory," *J. Assoc. Comput. Mach.*, vol. 7, pp. 201–215, 1960.
[2] K. D. Forbus and J. de Kleer, *Building Problem Solvers.* Cambridge, MA: MIT Press, 1993.
[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Murray Hill, NJ: Bell Labs., 1979.
[4] M. R. Genesereth, "The use of design descriptions in automated diagnosis," *Artif. Intell.*, vol. 24, pp. 411–436, 1984.
[5] A. Hou, "A theory of measurement in diagnosis from first principles," *Artif. Intell.*, vol. 65, pp. 281–328, 1994.
[6] J. de Kleer, "Local methods for localizing faults in electronic circuits," Mass. Inst. Technol, Cambridge, IT AI Memo 394, 1976.
[7] R. Reiter, "A theory of diagnosis from first principles," *Artif. Intell.*, vol. 32, pp. 57–95, 1987.
[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Englewood Cliffs, NJ: Prentice-Hall, 1995.

Postprocessing Statistical Language Models for a Handwritten Chinese Character Recognizer

Pak-Kwong Wong and Chorkin Chan

***Abstract*—Two statistical language models have been investigated on their effectiveness in upgrading the accuracy of a Chinese character recognizer. The baseline model is one of lexical analytic nature which segments a sequence of character images according to the maximum matching of words with consideration of word binding forces. A model of bigram statistics of word-classes is then investigated and compared against the baseline model in terms of recognition rate improvement on the image recognizer. On the average, the baseline language model improves the recognition rate by about 7% while the bigram statistics model upgrades it by about 10%.**

***Index Terms*—Chinese character recognizer, handwritten Chinese character recognizer, language model, statistic language model.**

I. Introduction

An image recognizer of a line of unknown characters can be asked to propose a lattice of degree n of character candidates that are most likely to reveal the true content of the line. The correspondence between a sequence of character candidates and a sequence of words is usually ambiguous because of ambiguous segmentation of the characters into words. A language model as a post-processor, can help selecting among the candidates by evaluating their respective soundness in forming a natural line-of-text of the language because the linguistic information of the characters can provide a useful basis for improving the recognition rate [1]. In this study, a character recognizer [2] is employed to test two statistics based language models as postprocessors. The character recognizer supports a vocabulary of 4616 characters and accepts writer independent off-line handwritten character images (Chinese characters, alphanumeric, and punctuation symbols) from a scanner. It outputs a user-specified number n of candidates for each character image forming a lattice. Because Chinese, unlike Western languages in which words are separated by blanks, has no word markers except the punctuation symbols. If there are m character images lying between a pair of punctuation symbols, the number of possible candidate sequences is m^n which can be extremely large for large m and n. Inevitably, many words can be formed in the lattice just coincidentally. This paper investigates how to select the "best" candidate sequence out of this large number of possible choices efficiently and accurately by means of a post-processing statistical language model.

II. The Lexicon

A lexicon named WORDDATA is acquired from the Institute of Information Science, Academia Sinica, Taiwan. There are 78 410 word entries in WORDDATA covering most, if not all, of the Chinese words actively used in modern texts such as journals, newspapers, and literature in Taiwan. Each word entry is associated with a usage frequency and the membership of at least one syntactic/semantic word-class. A text corpus named "The Selection of Hundred Kinds of the Press in 1994," of over 63 million characters of news lines

Manuscript received November 3, 1996; revised June 21, 1998. This work was supported by the Hong Kong Research Grant Council and the Industrial Support Funds of the Hong Kong Industry Department. This paper was recommended by Associate Editor M. S. Obaidat.

The authors are with the Department of Computer Science, The University of Hong Kong, Hong Kong.

Publisher Item Identifier S 1083-4419(99)00772-4.