Efficient Generation of Plausible Diagnoses in a Model-Based Diagnostic Engine

Abstract
An important way in order to reduce the complexity of model-based diagnosis is to focus the attention on a subset of ”plausible” diagnoses. Two important methods of focusing on plausible diagnoses have been reported: *(i)* search for the most probable diagnoses (de Kleer); and *(ii)* search for the preferred diagnoses (Dressler and Struss). The first method has the nice property that it allows to focus only on a *small* number of candidates. The second one allows to structure the candidate space such that the discovery of a conflict *prunes* a whole subspace of candidates. We propose a focusing method which combines the advantages of the above mentioned approaches, i.e. we search for *the most probable preferred diagnoses*. Our algorithm relies more on the explicit *construction* of the focus diagnoses rather than on a merely sequential test of *all* candidates. The effect of using a small focus combined with pruning the candidate space provides dramatic reductions of the time required for diagnosis.

Content areas: diagnosis.

1 Introduction

The purpose of model-based diagnosis is to detect and localize faulty components of a given technical system. The only knowledge to be used is the description of the local behavior of the components, and of the way the components are connected to each other. Using this kind of knowledge, a general view on the diagnostic process could be:

1. *Assumption initialization:* Make initial assumptions about the behavior of the components and other kinds of diagnostic assumptions (cf. [17], [19]).
2. *Observation acquirement:* Propose points of the technical system where measurements or observations are to be made or where certain inputs or states have to be set. Add the results of these measurements and settings to the knowledge base.
3. *Inference engine:* Infer consequences of the observations and the assumptions. If this step discovers contradictions, continue with Step 4, otherwise continue with Step 5.
4. *Candidate generation:* Revise the previous assumptions such that consistency is restored and continue with Step 3.
5. *Termination check:* Use a termination criteria that either stops the process or continues with Step 2.

The GDE [3] and its extensions provide a systematic and general approach to tackle the above tasks. For the solution of Step 2, the original GDE proposed an approach based on information theory, which required the consideration of the failure probabilities of each component. Although the concept of the original GDE is very general, it is not directly applicable to real technical systems, because: *(i)* its algorithmic complexity is too high; *(ii)* a significant part of physical behavior cannot be described properly. But nowadays there are several extensions or modifications of the GDE (for examples cf. [1], [5], [10], [12], [18]) that attempt to overcome these problems. Two kinds of extensions are crucial:

1. A *focusing diagnostic engine* restricts the attention to a subset of diagnoses. For practical applications, this subset of *focus diagnoses* cannot be chosen arbitrarily, but has to be defined according to some *plausibility criteria* (for examples cf. [6], [9], [14]). This has two major advantages: First, it corresponds more to the needs of the user, because typically the user is not interested in *all* diagnoses that are physically possible. The second advantage is that restricting the attention to a *small* focus leads to tremendous reductions of the time required for diagnosis [6].
2. A *diagnostic engine with behavioral modes* gives the opportunity or even requires the description of faulty behavior. This considers the fact that technical components usually do not fail in an arbitrary way. Without the use of this knowledge, the diagnostic engine might give diagnoses that are physically impossible [18].

These two extensions benefit from each other: On one hand, the knowledge of behavioral modes facilitates a proper definition of what are *plausible* diagnoses. On the other hand, the introduction of multiple behavioral modes significantly increases the complexity of diagnosis. Thus, the need for focusing is even more urgent.

This paper aims to identify efficient solutions to the problem of finding plausible diagnoses in large technical systems where the consideration of multiple behavioral modes is a requirement (e.g. in electrical systems). The current state-of-the-art with respect to focusing on plausible diagnoses comprises of two notable approaches:

1. Find the *most probable* diagnoses (this is de Kleer's focusing method, cf. [6]). This method requires the consideration of probabilities for each behavioral mode in order to define the probability of a candidate.
2. Find the *preferred* diagnoses (this is Dressler's and Struss' focusing method, cf. [9]). The method defines a partial order for the individual modes of behavior of a component. The order reflects how frequently the modes of behavior usually occur. The preference order among the modes induces a preference relation among candidates. The *preferred* diagnoses are the minimal diagnoses with respect to this ordering.

We think that a combination of these two methods satisfies best the requirements of practical applications: de Kleer's method has the advantage that one can focus on very few – normally a constant number – of diagnoses. Note that there can be a very large set even of *preferred* diagnoses, because the preference relation is just a *partial* ordering. Thus, the search for *all* preferred diagnoses is usually very expensive.

On the other hand, de Kleer's method could supply a diagnosis of which the set of faulty components is a superset of the faulty components of another diagnosis. This can never be the case with preferred diagnoses, if we assume that the normal mode is the most preferred mode of a component. But a much more important advantage of the use of prefernces is the fact that the partial ordering defined by the preferences *structures the candidate space* into a lattice. This structure enable us to *prune* a whole subspace of candidates when a conflict is discovered. Our algorithm relies more on the explicit *construction* of the focus diagnoses directly from the discovered conflicts rather than merely testing all candidates in the intrinsic order defined by their probability[1]. The effect of using a small focus combined with pruning the candidate space leads to order of magnitude reductions of the time required for diagnosis.

Our solution does not depend on the way in which the observation acquirement (i.e. Step 2 of the general algorithm presented at the beginning of the paper) is solved. It is also independent of the inference engine used in Step 3. The algorithms we propose are very simple and do not require the use of default logics or prioritized defaults, used by the previous papers dealing with preferences (cf. [9], [14]). Thus, the results of our work are relevant for a broad spectrum of diagnostic engines. Only minor changes are required in order to accommodate a broader spectrum of definitions for ”plausibility”.

This paper is organized as follows: Section 2 specifies the framework in which the problem discussed in this paper is embedded. Section 3 introduces the basic notations and properties needed for our solution. Section 4 presents a crude solution of the problem. This solution will be correct but slow. Section 5 shows how to improve the crude solution in order to get it much faster. That improvement is the crucial part of this paper. In Section 6, we give some experimental results comparing the crude algorithm with the improved one. In Section 7, we sketch how our results may be used in a more general context than the one described here.

2 A Framework for Focusing Diagnosis

Since our paper deals only with candidate generation, which is only one task (namely Step 4) of the general algorithm presented at the beginning of the paper, we will give here an outline of the top-level control procedure in a way suitable for the presentation in the subsequent sections.

We share the view expressed in [17], that diagnosis can be seen as the process of retracting assumptions. *Flexible* diagnosis requires the manipulation of *several* kinds of diagnostic assumptions (cf. [19]), such as the assumptions that observations are correct, or that the use of simplified models is appropriate. In order to keep the formal presentation as simple as possible, we consider in the following only the assumptions for the behavioral modes assigned to the components. Note, however, that our method can also handle other diagnostic assumptions, provided they have been assigned a measure of probability.

One task of the *inference engine* (Step 3) is to find conflicts. We require that the inference engine finds *at least* the minimal set of conflicts which invalidate the current focus. Of course, we even *suggest* that the inference engine should perform only those inferences (and, thus, only search for those conflicts) which hold in the current focus (cf. [4], [6], [8]), because this leads to significant complexity reductions in Step 3. An architecture based on a *focused* ATMS (cf. [4],[8]) usually ensures this, but our method does *not require* such an architecture.

1. The original GDE [3] and Reiter’s HS–tree algorithm [16] also constructed the *minimal diagnoses* directly from the conflicts.

The top-level control algorithm uses two global sets: *Focus* – containing the current focus candidates; and *Conflicts* – containing the set of conflicts among the assumptions considered so far. Note that we cannot guarantee in general that once a candidate entered the focus it is a diagnosis. In most cases the inference engine must focus the prediction on that candidate. This may lead to the discovery of new conflicts which invalidate the candidate.

```
Controler for Model-Based Diagnosis
  Initialize Focus by the candidate that assumes everything is normal;
  Initialize Conflicts by the empty set
  Repeat
    Set and acquire values and states for selected points of the technical system
    Make inferences based on the acquired observations and considering the current focus:
      Whenever a conflict conf is detected do
        Insert conf into Conflicts
        For all focus diagnoses diag invalidated by conf do
          Invoke Remove-Diagnosis-from-Focus (diag, conf)
        While Focus is not saturated (which is decided by any heuristic) do
          Invoke Insert-Diagnosis-into-Focus
  until it is decided that the diagnoses of Focus are satisfactory
End of Controler
```

The goal of the next sections will be to design the procedures **Remove-Diagnosis-from-Focus** – which takes candidates out of the focus when they are found inconsistent; and **Insert-Diagnosis-into-Focus** – which adds a new candidate to the focus. The criterion which decides how many diagnoses need to enter the focus is usually decided by a heuristic (for an example, cf. [6]) and is again independent of our method.

3 Basic Notations and Properties

Notational conventions about technical systems discussed in this paper:

Let $\mathcal{S}$ *be a technical system with* n *components* $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_n$.
For each component $\mathcal{C}_i$ *define* k_i *behavioral modes.*
Without loss of generality, assume that the modes $m_1, m_2, ..., m_{ki}$ *of a component* $\mathcal{C}_i$ *are ordered with respect to their probability, i.e.* m_1 *is the most probable mode and* m_{ki} *is the least probable mode.*

Definition 1: (component-mode-assignment)
A component-mode-assignment A *is an* n-*tuple* $A = [a_1,a_2,...,a_n]$ *where each* a_i *is an integer number between* 0 *and* k_i.
The interpretation of A *is a set of assumptions defined as follows:*
If $a_i = 0$ *then no assumption is made about the behavioral mode of component* $\mathcal{C}_i$.
If $a_i \neq 0$ *then* A *assumes that the component* $\mathcal{C}_i$ *is in mode* m_{ai} .

Definition 2: (candidate)
A candidate *is a complete component-mode-assignment , i.e.* $a_i \neq 0$, *for all* $i = 1,...,n$.

Both, conflicts and candidates, are denoted by component-mode-assignments. However, a conflict may not be a *complete* component-mode-assignment. On the contrary, the more entries of a conflict are 0, the stronger a restriction (and thus, more information) is expressed by this conflict. Our definition of a component-mode-assignment implicitely entails that the behavioral modes are exclusive, i.e. a component cannot be in two different modes at the same time.

Definition 3:

A candidate $A = [a_1, a_2, ..., a_n]$ contains *a conflict* $C = [c_1, c_2, ..., c_n]$ (*denoted by* $C \subseteq A$), *iff* $(c_i \neq 0 \Rightarrow a_i = c_i)$ *for all* $i = 1, ..., n$.

Considering the interpretation of Definition 1, it is obvious that a candidate *A* contains a conflict *C* if and only if the interpretation of *C* is a subset of the interpretation of *A.*

Definition 4: (preference between candidates)

A candidate $A=[a_1,a_2,...,a_n]$ *is* preferred *to a candidate* $B=[b_1,b_2,...,b_n]$ *(denoted by* $A \leq B$*) iff* $a_i \leq b_i$ *for all* $i=1,...,n$*.*

If a candidate A *is preferred to a candidate* B*, denote* A *to be a* predecessor *of* B *and* B *to be a* successor *of* A.

Denote A *a* direct predecessor *of* B *(resp.* B *a* direct successor *of* A*) iff* $A \neq B$ *and there is no component-mode-assignment* C *such that* $A \neq C \neq B$ *and* $A \leq C \leq B$.

A candidate *A* is preferred to a candidate *B,* if *B* is obtained from *A* by replacing some modes of behavior with less probable (less preferred) modes of behavior. The partial order defined by the preference relation *structures the candidate space* into a *lattice*. Figure 1 shows the candidate lattice for a system containing 3 components, each having 3 modes of behavior.

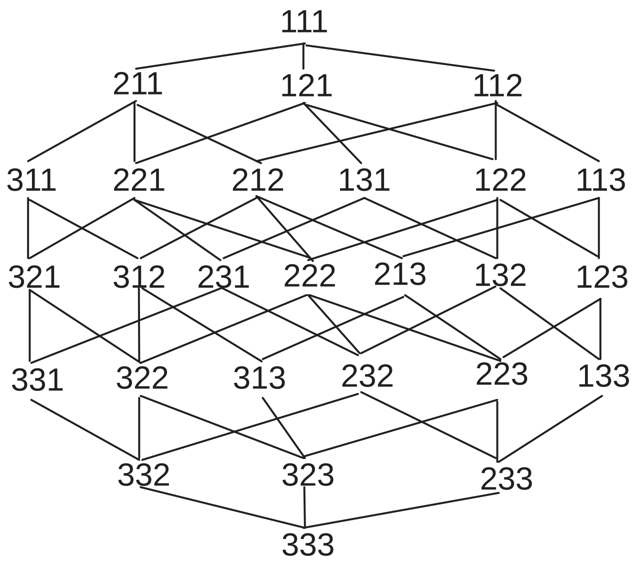

Figure 1: A candidate lattice

Definition 5: (preference w.r.t. a set)

A candidate A *is* preferred with respect to a set of candidates $\mathcal{A}$ *if and only if* $\mathcal{A}$ *contains no candidate that is preferred to* A *other than* A *itself.*

Observation 1:

[1,1,...,1] *is the unique candidate preferred to every other candidate, i.e. all candidates are successors of this candidate.*

Observation 2:

Every successor of a candidate A *is less or equal probable than* A*.*

The most probable successor of a candidate A*, different from* A *itself, is among the direct successors of* A.

Definition 6: (diagnosis)

A candidate is a diagnosis *for $\mathcal{S}$ if and only if it contains no conflicts.*
A candidate is a preferred diagnosis *for $\mathcal{S}$ if and only if it is a diagnosis for $\mathcal{S}$ and it is preferred with respect to the set of all possible diagnoses for $\mathcal{S}$.*

The notion of preferred diagnosis is a natural extension of the notion of minimal diagnosis (cf. [3]) when multiple modes of behavior are considered: Applying the above definitions to a system which uses only the correct mode and an unknown failure mode, the set of preferred diagnoses coincides exactly with the set of minimal diagnoses (provided, of course, that the correct mode is preferred to the failure mode). Also, the set of all preferred diagnoses partially characterizes the set of all possible diagnoses, in that every possible diagnosis can be obtained from a preferred diagnosis by replacing some mode assignments with less probable (less preferred) mode assignments. The most probable diagnosis is always one of the preferred diagnoses.

Note that the preference relation is not a total ordering. Thus, if [1,1,...,1] is not a diagnosis anymore, there are several preferred diagnoses. Our experiments show that the set of preferred diagnoses grows rapidly when the set of conflicts increases. This is another motivation for our focusing method, i.e. to search only for the *most probable preferred diagnoses*.

By Observations 1 and 2, we immediately obtain the following:

Lemma 1:

If the focussing method is to compute the most probable preferred diagnoses and if no conflict has been found yet, then the focus must contain [1,1,...,1] *as the only element, regardless of which heuristic is used for focus saturation.*

4 A Crude Solution of the Problem

In the previous section, we have seen that *(i)* we always have to start with a focus that contains only the candidate [1,1,...,1] assuming that everything behaves normally, and *(ii)* all the candidates can be constructed as successors of this initial candidate. Our algorithms will maintain two sets of candidates: *Focus* and *Candidates*. The following properties of these two sets will be ensured at every moment:

- *Focus* contains only valid candidates. There is no candidate which is simultaneously: *(i)* valid, *(ii)* preferred with respect to the set of all diagnoses, and *(iii)* more probable than an element of *Focus*.
- No element of *Candidates* is preferred to any element from *Focus*.
- All the candidates not containing a known conflict (thus all the diagnoses) are *successors* of at least one element from *Focus* or from *Candidates*.

As long as no conflict is discovered, these properties hold if *Focus* contains only the element [1,1,...,1] and *Candidates* is empty. All the operations we perform on *Focus* and on *Candidates* will preserve the above properties: As long as an element stays in *Focus*, none of its successors needs to be computed. If an element is removed from *Focus*, the above properties are preserved when *Candidates* contains at least all *direct* successors of this element. Some elements of *Candidates* may contain conflicts. It is easy to see that the above properties are still satisfied, if we replace an element of *Candidates* that contains a conflict by all of its direct successors. Further, the above properties are still satisfied if a successor of a *Focus* element is removed from *Candidates*.

These considerations informally prove that the following is a correct (though rather inefficient) solution of our problem:

Crude Algorithm:

- If a conflict is found which invalidates an element of *Focus*, this element is removed from *Focus* and, instead, all of its direct successors are inserted into *Candidates.*
- When a new element has to be added to *Focus*, the most probable element of *Candidates* is removed from *Candidates*. If this element does not contain any conflict, it is added to *Focus*. If it contains a conflict, all its direct successors are inserted into *Candidates* and the search for the next most probable element of *Candidates* continues. If it is a successor of a *Focus* candidate, it is ignored, and the search for the next most probable candidate continues further.

It follows directly from Definitions 1 and 4 that the following procedure inserts all direct successors of a given candidate *pred* into *Candidates*. For the sake of efficiency, the set *Candidates* should be organized as a priority queue ordered by probability.

```
Procedure Compute-Direct-Successors (pred,conf: component-mode-assignment)
pred: candidate [$p_1$,...,$p_n$] of which all direct successors must be inserted into Candidates
conf: not relevant for this version

   For i := 1 to n do
      If $p_i < k_i$ (where $k_i$ is the number of modes for component $\mathcal{C}_i$)
         then
            succ := [$p_1$,..., $p_{i-1}$, $p_i$+1, $p_{i+1}$,..., $p_n$]
            Insert succ into Candidates according to its probability
End of Compute-Direct-Successors
```

In pseudo-code, the crude algorithm looks as follows:

```
Procedure Remove-Diagnosis-from-Focus (old-diag,conf: component-mode-assignment)
old-diag: a focus diagnosis that has been found invalid now
conf: a conflict that invalidates old-diag (i.e., $conf \subseteq diag$)

   Remove old-diag from Focus
   Compute-Direct-Successors (old-diag,conf)
End of Remove-Diagnosis-from-Focus
```

```
Procedure Insert-Diagnosis-into-Focus

  Repeat
    top-cand := the most probable candidate of Candidates
    Remove top-cand from Candidates
    If there is a conflict conf which is a subset of top-cand
      then
        Compute-Direct-Successors (top-cand,conf)
      else
{pref.  If there is no diagnosis diag in Focus that is preferred to top-cand
 check}   then
            Insert top-cand into Focus
  until a new diagnosis has been inserted into Focus or Candidates is empty
End of Insert-Diagnosis-into-Focus
```

The preference check in the last procedure is only done in order to meet the requirements of the focusing principle of this paper. If it is omitted, the above procedures give a solution for de Kleer’s focusing method.

5 Speeding up the Solution

The problem with the solution described in the previous section is that the set *Candidates* grows exponentially very soon. This combinatorial explosion is due to the fact that the crude algorithm constructs *all* candidates more probable than the diagnosis that is searched for, checking the consistency and the preference w.r.t. focus for each of them. Our goal is to improve the crude algorithm such that *Candidates* is kept much smaller and still suffices to solve our problem. In particular, we show that it is not necessary to inspect each point of the search space.

Improvement I: It is easy to see that our focusing method does not require the successors of a candidate to be considered, before the candidate is considered itself. We already took advantage of this property even in the crude algorithm: We invoked **Compute-Direct-Successors** (*cand*,*conf*) only when *cand* was removed from *Focus* or *Candidates*. But we did not consider the fact that a candidate usually has *several* direct predecessors. The procedure **Compute-Direct-Successors** need not insert an element into *Candidates* if that element has at least one predecessor in *Focus* or in *Candidates*. Note that this makes the preference check in the procedure **Insert-Diagnosis-into-Focus** superfluous.

Even with the above improvement, our procedures construct and test *all* candidates in descending order of their probability until enough valid ones are found. In a system with n components, each having k modes of behavior, the discovery of a conflict of size s $(s < n)$, invalidates a subspace of $(n–s)^p$ candidates. One may be tempted not to insert into *Candidates* any element which contains a conflict. However, we must insert inconsistent elements into *Candidates*: A successor of an inconsistent candidate need not necessarily be inconsistent.

Improvement II: Up to this point, we have not made use of the knowledge *which* conflict eliminated a candidate. Note that an element is only constructed in **Compute-Direct-Suc-**

cessors when its direct predecessor has been proven to contain a conflict. So, the information *which* conflict eliminated it, is readily available. Once we already know a conflict *C*, it is obvious that we are only interested in successors of a candidate *A* that do *not* contain *C*. The following lemma gives a convenient characterization of such successors:

Lemma 2:

Let A *be a candidate that contains a conflict* C. *Let* S *be another candidate.Then:*
S *is a successor of* A *and does not contain* C
$\Leftrightarrow$ S *is a successor of a direct successor* D *of* A *such that* D *does not contain* C.

Proof:

Let $A = [a_1, a_2, ..., a_n]$ contain the conflict $C = [c_1, c_2, ..., c_n]$. Thus, by Definition 3:
For all i, either $c_i = 0$ or $c_i = a_i$. Let $S = [s_1, s_2, ..., s_n]$ be another candidate.
$\Rightarrow$: Assume that *S* is a successor of *A*. By Definition 4, for all i, $s_i \geq a_i$. Assume further that *S* does not contain *C*. By Definition 3, there is an index k such that $c_k \neq 0$ and $c_k \neq s_k$. Since $c_k = a_k$ and $s_k \geq a_k$, we conclude that $s_k > a_k$. Then $D := [a_1, a_2, ..., a_{k-1}, a_k+1, a_{k+1}, ..., a_n]$ is a direct successor of *A,* does not contain *C* ($a_k = c_k \neq a_k+1$), and has *S* as a successor.
$\Leftarrow$: Assume that $D = [d_1, d_2, ..., d_n]$ is a direct successor of *A*. By Definition 4, there exists a k such that $a_k = d_k - 1$ and for all $i \neq k$, $a_i = d_i$. Suppose that *S* is a successor of *D*. Then *S* is clearly also a successor of *A*. Now, if *D* does not contain *C*, it must differ from *C* in at least one position j where $c_j \neq 0$. Since *A does* contain *C* and *A* differs from *D* only in position k, $j = k$. Thus, $c_k = d_k - 1$. Since $s_k \geq d_k$, we obtain that $0 \neq c_k \neq s_k$. Thus, *S* does not contain *C*.

Corollary to Lemma 2:

The procedure **Compute-Direct-Successors** *(*A*,*C*) only needs to compute and insert into* Candidates *the direct successors of* A *that do not contain* C.

Lemma 2 proves that ignoring the *direct* successors of *A* which contain the same conflict as *A* achieves a pruning effect: At later stages of our search, we avoid to construct other *non–direct* successors of *A* which contain the same conflict as *A* did.

We summarize the above improvements giving a revised version for procedure **Compute-Direct-Successors**:

Improved (I+II) Procedure Compute-Direct-Successors (*pred*,*conf*: component-mode-assignment)
pred: component-mode-assignment $[p_1,...,p_n]$ of which all direct successors must be computed
conf: conflict $[c_1,...,c_n]$ that invalidates *pred* (thus, $conf \subseteq pred$)

```
  For i := 1 to n do
     If $c_i \neq 0$ and $c_i < k_i$ (where $k_i$ is the number of modes for component $\mathcal{C}_i$)
          {Note that $c_i = p_i$ by assumption since $conf \subseteq pred$ and $c_i \neq 0$}
        then
          $succ := [p_1,..., p_{i-1}, p_i+1, p_{i+1},..., p_n]$
{pref.    If succ is not a successor of any element in Focus or Candidates
 check}     then
               Insert succ into Candidates according to its probability
End of Direct-Successors
```

The other procedures may remain in the way stated in Section 4. As mentioned above, the preference check in procedure **Insert-Diagnosis-into-Focus** can now be omitted.

6 Experimental Results

The improvements presented in the last section lead to tremendous reductions in the time required for diagnosis. The results we show here were obtained while diagnosing a family of circuits described by Figure 2, where M_{ij} are multipliers, and A_i are adders. By size, we denote the number of output pads. Every component was characterized by 7 modes of behavior, as follows: *ok* – the correct mode of behavior, probability 0.75; *s1* – output stuck at 1, probability 0.1; *s0* – output stuck at 0, probability 0.05; *l* – output equal with the left input, probability 0.04; *r* – output equal with the right input, probability 0.04; *sh* – output equal with the correct result shifted with one bit to the left, probability 0.018; *u* – unknown failure, probability 0.002.

The *inference engine* used was based on a focused ATMS and only attempted to find those conflicts which invalidate the current focus. The heuristic for focus saturation, used for the experiments, required *Focus* to contain the preferred diagnoses with maximal probability only (so, if the maximum was unique, *Focus* had to contain just one diagnosis). Note that the set of diagnoses with maximal probability coincides with the set of preferred diagnoses with maximal probability.

The following table compares the results of the crude algorithm presented in section 4 with its successive improvements (I and I+II).

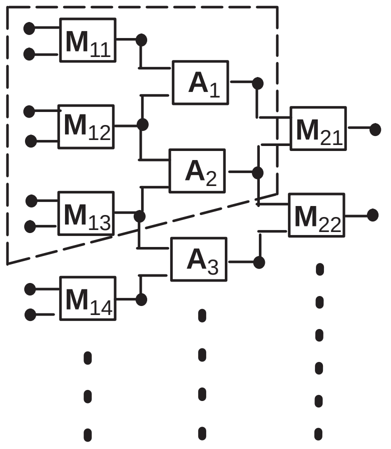

Figure 2: Circuit family (size shown: 2)

The first two rows show results obtained for the circuit enclosed in the dashed region in Figure 2, which corresponds to a well known example in the literature. Since our improvement rates grow tremendously with increasing circuit size, we gave some results for the circuit of size 4. Columns 1, 2 and 3 of the table indicate the complexity of the problem. Column 4 compares the total time required for diagnosis on a 386 PC. Column 5 compares the number of candidates which were extracted from *Candidates* during the search. Column 6 gives the total size of *Candidates* at the end of diagnosis.

components	conflicts	diagnosis	TIME crude / I / I+II	Extracted Cand. crude / I / I+II	*Candidates* size crude / I / I+II
5	27	{u(M_{12})}	20 / 4 / 0.5	223 / 211 / 45	480 / 46 / 15
5	106	{u(M_{12}),sh(A_2)}	725 / 68 / 3	1873 / 1626 / 212	2216 / 170 / 42
15	9	{r(M_{14})}	37.5 / 8 / 0.3	61 / 61 / 13	735 / 119 / 19
15	17	{l(M_{15}),s1(M_{11})}	– / 414 / 1	– / 810 / 27	– / 784 / 82
15	238	{s1(A_5),s1(A_3),u(A_4)}	– / – / 80	– / – / 1068	– / – / 490

7 Discussion and Comparison with Other Work

In [5] and [6], a scheme for updating the probability of a candidate was proposed. The update can only lower the probability of a focus candidate if the candidate does not explain some observations. The method of handling posterior probabilities could, in principle, be incorporated in our algorithms, too. Dressler and Struss use default logics to characterize and compute preferred diagnoses (cf. [9]). An extended ATMS (namely the NM-ATMS, [7]) constructed the set of all preferred diagnoses as a label of a special node. Although the NM–ATMS offers a nice solution, it must use a double number of assumptions than we need (it needs also *out*-assumptions) and cannot compute only a *subset* of the preferred diagnoses. There are other papers in which candidate generation is discussed. Reiter (cf. [16]) introduced the HS-tree algorithm which has been improved afterwards (cf. [11]). But his algorithm only applies to systems without modes for faulty behavior. His focusing method is to find all *minimal* diagnoses. Mozetic (cf. [15]) even gave a polynomial algorithm, but his algorithm also only applies to systems without modes for faulty behavior. The focusing method of Mozetic's algorithm is to find just *any* feasible diagnosis, so no notion of plausibility was used.

In this paper, we proposed to focus on the most probable preferred diagnoses, an approach which combines the use of probabilities [6], with the use of preferences [9]. The method allows to focus on *small* numbers of candidates, and furthermore provides a mechanism for *pruning* the candidate space.

Since the heuristic for focus saturation of the top-level control procedure from Section 2 may be arbitrary for our focusing method, an admissible heuristic would also be to find *all* preferred diagnoses. For this purpose the heuristic requires that, as long as the set *Candidates* is not empty, another element must enter *Focus*. Thus, our focusing method is a generalization of Dressler's and Struss' focusing method. However, our tests showed that even for small systems like those presented in the preceding section, such a focusing principle is extremely expensive.

Note that our focusing method is *not* a generalization of de Kleer's focusing method. The set of the first k most probable preferred diagnoses does not coincide with the set of the first k most probable diagnoses, unless $k = 1$. This is because some of the k most probable diagnoses can be successors of some of the $k–1, k–2, ..., 1$ most probable diagnoses, and so they are not preferred diagnoses. The algorithms presented in Section 4 could be used to search for the most probable diagnoses, provided that the preference check w.r.t. *Focus* is removed from the procedure **Insert-Diagnosis-into-Focus**. But the preference check w.r.t. *Candidates* may also be done when the focusing principle is to search for the actual next probable diagnosis. Then the set *Candidates* would just consist of fewer elements and could still be used to yield the same result. Thus, the preference check with respect to *Candidates* is a real speed-up compared to a method that has no notion of preferences.

But the crucial speed-up is obtained by the consequence of Lemma 2 which allows the *pruning* effect when searching the candidate space. If we did not consider it, the candidate generator would still inspect sequentially all the candidates, according to the probability order, until enough valid candidates are found. The procedure would have an exponential *any case complexity*. Consider the case in which all the discovered conflicts have size one. The size of the set *Candidates* grows exponentially in the crude algorithm, while it will always be *zero* in the improved one – in such a case there will be only one preferred diagnosis. In the *worst* case, i.e. when all the discovered conflicts have the size of a candidate, the behavior of the improved algorithms coincide with that of the crude algorithms, since no pruning is possible. Hopefully, this worst case is extremely unlikely to appear in practical applications. Our empirical results showed reductions of the time required for diagnosis up to a factor of several hundreds due to the pruning effect *only*.

With respect to the problem of *plausible* candidate generation, the described algorithms are more general, and only minor changes must be added to accommodate a broader *spectrum of definitions for plausibility*, which need not refer to the notion of probability at all. In the above presentation we *only* needed the probabilities for ordering the set *Candidates*. Different criteria for ordering *Candidates* could be considered for different definitions of ”plausibility”. The only requirement is that the plausibility should be defined *on top of the preference relation*, in that the successors of a candidate should be less plausible than the candidate itself. Such a restriction is sufficient to allow the *pruning* effect and the other improvements described above. A different definition of plausibility, which fulfills this demand, could be based on the number of faults assumed by a candidate, e.g. *all the candidates which assume* p *defective components are more plausible than the candidates assuming* p+1 *faults*. An admissible heuristic for this definition of plausibility would be to search for *all preferred diagnoses with less than* k *faults*.

References

[1] DAGUE, Philippe / DEVES, Philippe / LUCIANI, Pierre / TAILLIBERT, Patrick: Analog Systems Diagnosis. *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI '90)*, pp. 173-178, Stockholm (Sweden) 1990.
Also in: [13], pp. 229-234.

[2] DE KLEER, Johan: An Assumption Based Truth Maintenance System. *Artificial Intelligence* **28**, pp. 127-162, 1986.

[3] DE KLEER, Johan / WILLIAMS, Brian: Diagnosing Multiple Faults. *Artificial Intelligence* **32** (1), pp. 97-130, 1987.
Also in: [13], pp. 100-117.

[4] DE KLEER, Johan / FORBUS, Ken: Focusing the ATMS. *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI '88)*, pp. 193-198, Saint Paul (MN) 1988

[5] DE KLEER, Johan / WILLIAMS, Brian: Diagnosis with Behavioral Modes. *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI '89)*, pp. 1324-1330, Detroit (MI) 1989.
Also in: [13], pp. 124-130.

[6] DE KLEER, Johan: Focusing on Probable Diagnoses. *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI '91)*, pp. 842-848, Anaheim (CA) 1991.

[7] DRESSLER, Oskar: Problem Solving with the NM-ATMS. *Proceedings of the 9th European Conference on Artificial Intelligence (ECAI '90)*, pp. 252-258, Stockholm (Sweden) 1990.

[8] DRESSLER, Oskar / FARQUHAR, Adam: Putting the Problem Solver Back in the Driver's Seat: Contextual Control of the ATMS, *Lecture Notes in AI 515,* Springer Verlag, 1990.

[9] DRESSLER, Oskar / STRUSS, Peter: Back to Defaults: Characterizing and Computing Diagnoses as Coherent Assumption Sets. *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI '92)*, pp. 719-723, Vienna (Austria) 1992

[10] FRIEDRICH, Gerhard / GOTTLOB, Georg / NEIJDL, Wolfgang: Physical Impossibility instead of Fault Models. *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI '90)*, pp. 331-336, Boston (MA) 1990.
Also in: [13], pp. 159-164.

[11] GREINER, Russel / SMITH, Barbara / WILKERSON, Ralph: A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence* **41** (1), pp. 79-88, 1989
Also in: [13], pp. 49-53.

[12] HAMSCHER, Walter: Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence* **51** (1-3), pp. 223-271, 1991.
Also in: [13], pp. 283-308.

[13] HAMSCHER, Walter / CONSOLE, Luca, / DE KLEER, Johan (Editors): *Readings in Model-Based Diagnosis,* Morgan Kaufmann Publishers, San Mateo (CA) 1992

[14] JUNKERS, Ulrich: Generating Diagnoses by Prioritized Defaults. *(A)TMS in Expert Systems, SEKI Report* **SR-92-06**. pp. 19-24, Kaiserslautern (Germany) 1992.

[15] MOZETIC, Igor: A Polynomial-Time Algorithm for Model-Based Diagnosis. *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI '92)*, pp. 729-733, Vienna (Austria) 1992

[16] REITER, Raymond: A Theory of Diagnosis from First Principles. *Artificial Intelligence* **32** (1), pp. 57-96, 1987
Also in: [13], pp. 29-48.

[17] STRUSS, Peter: Diagnosis as a Process. *Working Notes of the Workshop on Model-Based Diagnosis*, Paris (France) 1989.
Also in: [13], pp. 408-418.

[18] STRUSS, Peter / DRESSLER, Oskar: Physical Negation: Integrating Fault Models into the General Diagnostic Engine. *Proceedings of the 11th International Joint Conference on Artificial Inteeligence (IJCAI '89)*, pp. 1318-1323, Detroit (MI) 1989.
Also in: [13], pp. 153-158.

[19] STRUSS, Peter: What's in SD? Towards a Theory of Modeling for Diagnosis. *Working Notes of the 2nd International Workshop on Principles of Diagnosis, Technical Report* **RT/DI/91-10-7**, pp. 41-51, Torino (Italy) 1991.
Also in: [13], pp. 419-449.