Minimal Hitting Set Computation via Hypothesis Exploration

Marina Zanella[1] and **Ingo Pill**[2]
[1]Department of Information Engineering, University of Brescia, Brescia, Italy
e-mail: marina.zanella@unibs.it
[2]Institute for Software Technology, Graz University of Technology, Graz, Austria
e-mail: ipill@ist.tugraz.at

Abstract

Minimal hitting set (MHS) computation is a challenging problem in conflict-oriented model-based diagnosis. This paper is a first attempt to face the problem by searching the powerset $\mathcal{H}$ of the conflicts' domain, which exhibits a partial order in respect of subset inclusion. Our idea is to enhance this partial order by ordering $H \in \mathcal{H}$, called a hypothesis, in each (cardinality) layer of the powerset. The overall regularity can then be exploited to process a layer at a time, according to an ascending cardinality of the hypotheses, checking whether each hypothesis is valid, and pruning all its supersets (that necessarily belong to the next layers) in one shot if it is. This is a high-level description of a template anytime algorithm that can host different methods of checking whether a given hypothesis is valid, and that is the basis not only for a monolithic computation of MHSs but also for a distributed one.

1 Introduction

Minimal hitting sets (MHSs) are a very interesting structural aspect that can help with isolating solutions in several contexts [Eiter *et al.*, 2008]. They can be used, for instance, for converting the disjunctive normal form of a Boolean formula into its conjunctive counterpart and vice versa. Some fundamental contributions [de Kleer and Williams, 1987; Reiter, 1987; Greiner *et al.*, 1989] showed the worth of MHSs also for diagnostic reasoning when a system model is taken into account. In this case, we can add special health state variables encoding our assumptions whether the individual system components work correctly. Then, we can compute the desired diagnoses as MHSs of the conflicts between our assumptions that the components work correctly and the observed behavior. A MHS contains at least one element of each conflict (minimal or not) and thus, if we assume the corresponding components to work abnormally, the conflicting assumptions in the minimal conflicts are resolved (and in turn also in the non-minimal ones) by such a *diagnosis*.

Reiter's approach intermingles conflict detection and MHS computation, whereas de Kleer and Williams' variant first computes all the conflicts. But for both, and some other conflict-oriented model-based diagnosis (MBD) approaches (like when using HST [Wotawa, 2001] or RC-Tree [Pill and Quaritsch, 2015]), the exploration of the diagnosis search space via MHS computation is based on the conflicts rather than the *hypothesis space* as defined by 2^M (M being the set of health state variables). In this paper, we will explore an option for MHS computation that focuses on the latter.

Our motivation for this originates in the attractive performance [Nica *et al.*, 2013] of methods that use a general purpose solver, such as a SAT solver, to derive diagnoses directly from a satisfiability encoding of a MBD problem [Metodi *et al.*, 2012]. Such methods iteratively search for satisfying assignments with a limited number of violated health state assumptions, and add blocking clauses to the encoding whenever a solution is found (for avoiding this solution and its supersets in the future). If we now start with a cardinality limit of one and increase it if, and only if, there is no solution anymore for the current cardinality, we can indeed explore the whole search space.

Internally, a solver identifies a satisfying assignment to the problem encoding by Boolean constraint propagation [Moskewicz *et al.*, 2001], such that some variable assignments can be deduced from currently available data via the clauses in the encoding. If there is no further knowledge to propagate, the solver has to *decide* about *some* unassigned variable values in order to be able to proceed (such decisions might however have to be retracted later on, if they prohibit a satisfying assignment) [Moskewicz *et al.*, 2001]. Obviously, the resulting search for diagnoses leans more on the hypothesis space, rather than the conflicts.

In order to implement a similar approach for MHS computation, we thus propose in this paper a method implementing a not-so-brute-force strategy for conquering the hypothesis space, and show also how this approach can be used to compute MHSs. Exploiting a general purpose SAT solver for computing MHSs did not show attractive performance in earlier tests (in contrast to diagnosis computation), so that we deem an investigation of the options for a hypothesis-oriented MHS computation a rather interesting one.

In the remainder of this paper, Section 2 provides a background about the notions of MHS problem and hypothesis space; Section 3 proposes an additional order for the hypothesis space and a strategy to explore it; finally, Section 4 draws some conclusions.

2 Preliminaries

Let us start with a formal definition of a MHS.

Definition 1. *Let $\mathcal{N}$ be a finite collection of finite sets N_i that contain elements in the domain M. A set $H \subseteq M$ is a* hitting set *for $\mathcal{N}$ if and only if, for all $N_i \in \mathcal{N}$, $H \cap N_i \neq \emptyset$.*

Such a hitting set H is minimal *if there is no $H' \subset H$ s.t. H' is a hitting set of $\mathcal{N}$. A hitting set H of $\mathcal{N}$ can only contain elements in $M_{\mathcal{N}} \subseteq M$, which restricts M to the finite domain of elements occurring in $\mathcal{N}$ (that is, $M_{\mathcal{N}}$ is the union of all $N_i \in \mathcal{N}$). The problem of finding* all *MHSs for some $\mathcal{N}$ is referred to as the* MHS problem*.*

In our definition, we consider minimality in the context of subset containment. In the literature, there are further options including cardinality (a hitting set of minimum cardinality is referred to as a *minimum hitting set*), or also probabilities [de Kleer and Williams, 1987] and similar weight/cost functions.

The exponential search space for solutions to a MHS problem instance is the powerset $2^{M_{\mathcal{N}}}$, denoted $\mathcal{H}$ and called the *hypothesis space*. Since the relation of subset inclusion ($\subseteq$) is reflexive, antisymmetric and transitive, $\mathcal{H}$ is a poset and can be displayed as a Hasse diagram, like the one shown in Fig. 1 for $M_{\mathcal{N}} = \{a, b, c, d\}$. Following the idea of subset inclusion, a hypothesis H_1 is preferable to H_2 if, and only if, $H_1 \subset H_2$. Looking at the Hasse diagram example, this is a directed (acyclic) graph where nodes are hypotheses and edges (that are implicitly downward) implement such a preference relation in that all nodes in the subgraph rooted in some hypothesis H represent hypotheses H' that are less preferable than H. Thanks to the partial order induced by subset inclusion, $\mathcal{H}$ exhibits a hierarchical structure, where each layer includes all and only the hypotheses having the same cardinality.

Definition 2. *Given some hypothesis space $\mathcal{H}$, the* successors *H' of some hypothesis H (denoted as $succ^*(H)$) are those hypotheses that are less preferable than H as $H \subset H'$. Consequently, those hypotheses H' are in the subgraph rooted in H in the Hasse diagram for $\mathcal{H}$, and we call H a* predecessor *of H'. Those successors H' such that $|H'| - |H| = 1$ are the* immediate successors *of H (denoted as $succ(H)$), and a hypothesis H is an* immediate predecessor *of any of its immediate successors.*

Intuitively, a hypothesis H can be encoded as a binary word $bin(H)$ with length $|M_{\mathcal{N}}|$ such that each bit in $bin(H)$ tells us, for a given order in $M_{\mathcal{N}}$, whether a specific $m \in M_{\mathcal{N}}$ is included in H or not. The binary representation of any immediate successor H_s of some hypothesis H has exactly one bit assigned 1 which is 0 for $bin(H)$, while the binary representation of any immediate predecessor H_p of H has exactly one bit assigned 0 which is 1 for $bin(H)$, with the other bits unchanged. This leaves us with the following properties:

Corollary 1. *The Hamming distance between hypothesis H and its immediate predecessors as well as immediate successors is 1 considering their binary representations.*

Corollary 2. *Hypotheses H_1 and H_2, with $H_1 \neq H_2$ and such that $|H_1| = |H_2|$ (they are at the same level in the Hasse diagram), are incomparable in respect of the chosen preference relation (subset containment), and share at most one immediate successor due to Corollary 1.*

Corollary 3. *For any given pair of immediate predecessors H_{p1} and H_{p2} of hypothesis H, we have $H = H_{p1} \cup H_{p2}$, and $bin(H) = bin(H_{p1})$ OR $bin(H_{p2})$ for a bitwise logic OR. Consequently, the Hamming distance between the binary representations of H_{p1} and H_{p2} is 2.*

3 Traversing the Hypothesis Space

The layers in the hypothesis space are top-down ordered by ascending cardinality. Since no hypothesis in a layer is less preferable than any hypothesis in the lower layers, a top-down exploration of the hypothesis space that takes into account one layer at a time guarantees to single out the valid hypotheses (e.g. the hitting sets) that are the most preferred (i.e. the MHSs) first. However, the hypotheses on each layer are incomparable by subset inclusion. For a structured approach at traversing $\mathcal{H}$, let us introduce an additional order for the hypotheses with the same cardinality, so as to achieve a total order on the hypothesis space. To this end, we can use the binary numbers represented by $bin(H)$ such that the highest number associated with some H at level i in the Hasse diagram is at the very left, it is progressively decreasing from left to right, so as the lowest number is on the very right. This means that, for a given cardinality $i = |H|$, in the binary representation of the left-most hypothesis the left-most i bits are assigned 1 and the others 0, while in that of the right-most hypothesis the right-most i bits are assigned 1 and the others 0. The binary representations of all the hypotheses on the same layer include i instances of 1.

The following partition of the set of immediate successors of a hypothesis will allow us to traverse $\mathcal{H}$ without producing a hypothesis twice, as it will become clear in Corollary 6. An idea for avoiding the generation of a hypothesis twice, that is similar to the one explained here but focuses on conflict-oriented MHS computation, was shown in [Pill and Quaritsch, 2015].

Definition 3. *Let $succ_L(H) \subseteq succ(H)$ be the subset of immediate successors H' (Def. 2) of H such that the other possible immediate predecessors H^* of H' are only to the left of H according to the additional order for the hypotheses with cardinality $|H| = |H^*|$, i.e. $bin(H^*) > bin(H)$. Let $succ_R(H) = succ(H) \setminus succ_L(H)$.*

In the Hasse diagram in Fig. 1, the hypotheses in $succ_L(H)$ are connected to H with solid lines, while those in $succ_R(H)$ are connected via dashed lines.

Theorem 1. *Let H' be a hypothesis in $succ_L(H)$ for some hypothesis H (as of Def. 3). The binary representation $bin(H')$ of such a H' can only be obtained by switching a bit with index $k > j$ in $bin(H)$, such that j is the index of the left-most bit assigned 1 in $bin(H)$, counting starting with 0 at the least significant bit (LSB), the most significant bit (MSB) having index $|M_{\mathcal{N}}| - 1$.*

Proof. (Sketch) We know that the binary representation of any immediate successor of H is obtained by switching a bit assigned 0 in $bin(H)$ (as $|H'| = |H| + 1$ and $H \subset H'$). Now let i be the index of the right-most bit assigned 1 in $bin(H)$, and k the index of the bit (assigned 0) we want to switch. Since the bits i and j in $bin(H)$ are assigned 1, we obviously have $k \neq i$ and $k \neq j$ (switching those bits cannot lead to any H' such that $|H'| = |H| + 1$).

Now, if we switch the bit indexed k for some $k < i$, then there is some hypothesis H^* to the right of H which is also an immediate predecessor of H'. This follows from Corollaries 1 and 3, and is in contradiction with the definition of $succ_L(H)$.

If we switch some bit assigned 0 such that $i < k < j$, then there is also some hypothesis H^* to the right of H that is an immediate predecessor of H'. While it is more difficult to see, this follows from the same corollaries. In particular,

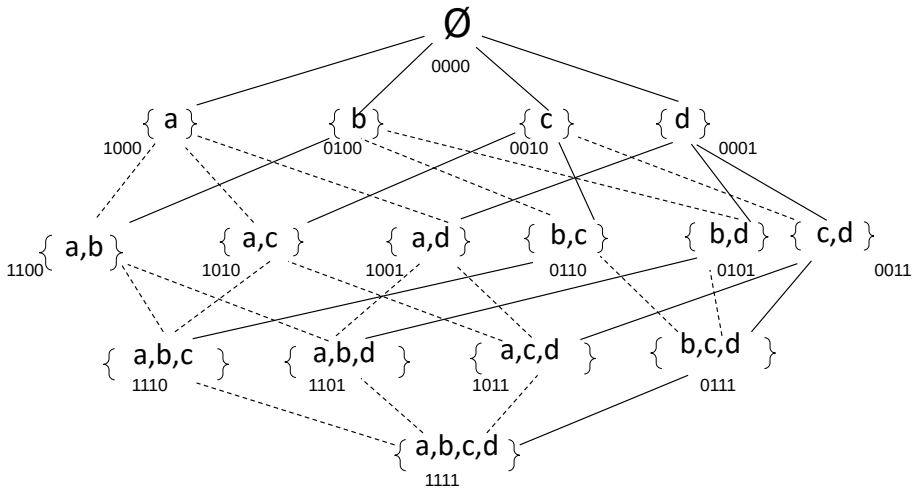

Figure 1: Hasse diagram for $\mathcal{H}$ for $M_{\mathcal{N}} = \{a, b, c, d\}$

there is some hypothesis H^* such that bit k is 1 but bit j is 0. We then have $bin(H^*) < bin(H)$, which would contradict the definition of $succ_L(H)$.

Now, if we switch a bit of index $k > j$, no hypothesis H^+ to the right of H ($bin(H^+) < bin(H)$) can have H' as an immediate successor, since any immediate predecessor of H' other than H is bound to have the bit with index k assigned 1 (due to Corollary 3). Then, by definition, any immediate predecessor H^* of H' other than H, where $bin(H')$ has been obtained by switching a bit with index k in $bin(H)$, is necessarily to the left of H, that is, $bin(H^*) > bin(H)$. □

Corollary 4. *From the proof of Theorem 1, it directly follows that* $succ_L(H) = \emptyset$ *if* $j = |M_N| - 1$.

Corollary 5. *Among the predecessors of a hypothesis* H'*, there is exactly one immediate predecessor* H *such that* $H' \in succ_L(H)$.

Corollary 6. *Due to Corollary 5,* $\mathcal{H}$ *can be constructed via iteratively computing* $succ_L(H)$ *starting with the* $\emptyset$ *hypothesis. Furthermore, from the same corollary it follows that there is exactly one (construction) path via* $succ_L$ *for any* $H \in \mathcal{H}$.

Theorem 2. *The binary representation of each hypothesis* H' *to the left of* H *such that* $succ(H') \cap succ_L(H) \neq \emptyset$ *is obtained by switching some bit assigned* 0 *(with index* j*) to the left of the left-most bit assigned* 1 *in* $bin(H)$ *(with index* k*) and switching one of the bits originally assigned* 1 *in* $bin(H)$ *with index* $l \leq k$ *(counting starting at the LSB with* 0*).*

Proof. It is easy to see that $|H'| = |H|$, and that the Hamming distance between the two hypotheses is 2 (only the bits indexed j and l are different). In the spirit of Corollary 3, H and H' thus share a common immediate successor $H^* = H' \cup H$, where $bin(H^*) = bin(H)$ *OR* $bin(H')$, such that, compared to $bin(H)$, $bin(H^*)$ has assigned 1 to some bit of index $j > k$ and thus is in $succ_L(H)$, as of Theorem 1. □

Let us then assume that we traverse $\mathcal{H}$ iteratively, layer by layer, in order of ascending cardinality, and let us check the individual hypotheses for their validity. A hypothesis H is valid if an oracle (invoked as *check(H)*) states so (by returning value *true*). At any moment we assume that there are two sequences, *current* and *next*, that are meant to include hypotheses belonging to the current and next layer of the hypothesis space, respectively, where such hypotheses are ordered based on their binary representations, as explained above. Let us assume that the current layer is that inherent to cardinality $i > 0$, hence sequence *current* includes the hypotheses that have been generated for this layer in the previous iteration (at the first iteration *current* includes just the top layer hypothesis, the only one with cardinality 0). At the beginning of the new iteration, sequence *next* is empty. The new iteration has to process all the hypotheses in *current*, from left to right, in order to isolate among them the valid ones and generate the hypotheses relevant to the next layer. Each hypothesis H is checked in order to find out whether it is valid or not. If it is, it is added to the (initially empty) set of most preferred valid hypotheses, $\mathcal{H}_+$, and removed from *current*.

If H is invalid, the relevant hypotheses in $succ_L(H)$ are generated (in the usual order based on their binary representations) and merged in sequence *next* in such a way as to preserve its descending order of the binary representations. Since we want to prune the hypothesis space so as to avoid exploring portions of $\mathcal{H}$ that include only hypotheses that are less preferable than those already saved in $\mathcal{H}_+$, if hypothesis H is invalid, only its immediate successors in $succ_L(H)$ that have all invalid immediate predecessors are generated (via Theorem 1). This way we ensure that we obtain the most preferred valid hypotheses only, without the need of having to perform any subset-checks in the set $\mathcal{H}_+$ of already obtained ones. Ascertaining whether all the immediate predecessors of a hypothesis $H' \in succ_L(H)$ are invalid is easy as such immediate predecessors are necessarily to the left of H; hence, they have been processed before H and, in case they are invalid, they have not been removed from *current*. Via Theorem 2, we can produce their binary representations, therefore we can look for them in *current*, starting from the hypothesis preceding H and going left.

Algorithm 1 above implements these concepts and uses the function given by Algorithm 2 for generating the rel-

Algorithm 1 Exploring $\mathcal{H}$ and generating $\mathcal{H}_+$.

Requires: *check*(H) — a function to check the validity of hypothesis H
Requires: D — the domain of the hypothesis space $\mathcal{H}$

```
1:  procedure EXPLOREH(D):
2:      H+ ← ∅
3:      current ← < ∅ > — top layer hypothesis
4:      while current ≠<> do
5:          next ← <> — empty sequence
6:          for all H ∈ current from left to right do
7:              if check(H) then
8:                  H+ ← H+ ∪ {H}
9:                  remove H from current
10:             else
11:                 succ'_L ← genChildren(H, current, D)
12:                 next ← next ⊕ succ'_L
13:         current ← next — next layer
        return H+
```

evant $H' \in succ_L(H)$. (Dyadic operator '$\oplus$' merges the operand sequences, which are in descending order, into one sequence, in descending order. Dyadic operator '+', as applied to sequences, appends the second operand sequence to the first operand one.)

Theorem 3. *Algorithm 1 for traversing $\mathcal{H}$ and generating $\mathcal{H}_+ \subset \mathcal{H}$, the set of most preferred valid hypotheses as of Def. 2, is sound and complete.*

Proof. (sketch) Alg. 1 starts the exploration from the top layer, containing just the empty set hypothesis (line 3), and it processes a layer of $\mathcal{H}$ at each iteration of the *while* loop (line 4). It deals differently with a valid hypothesis H (lines 8-9) and with an invalid one (lines 11-12). Any valid hypothesis is ruled out for further exploration, and none of its successors is generated. Since each hypothesis added to the (initially empty) set of solutions $\mathcal{H}_+$ is valid, as it has been stated by the oracle (line 7), and it is subset minimal thanks to the exploration order, this ensures that the algorithm is sound. For each invalid hypothesis, the sequence of hypotheses $succ'_L$ is generated by Alg. 2. Let us assume that the call of Alg. 2 at line 11 of Alg. 1 returns the (properly ordered) sequence containing all and only the hypotheses in $succ_L(H)$ whose immediate predecessors are all invalid. That is, the content of $succ'_L$ is a subset of $succ_L(H)$ that does not contain any hypothesis $H' \in succ_L(H)$ that is less preferable than some hypothesis $\hat{H} \in \mathcal{H}_+$. We do not need to traverse this space, it cannot lead to any $\tilde{H} \in \mathcal{H}_+$ since this $\tilde{H}$ would be less preferable than some other $\hat{H} \in \mathcal{H}_+$, which is a contradiction. Hence, the traversal of $\mathcal{H}$ for generating $\mathcal{H}_+$, which terminates when no next level hypothesis has been produced, is complete.

The proof above relevant to Alg. 1 assumes that Alg. 2 is in turn sound and complete. This is what is (sketchily) proven in the following. The expected output of Alg. 2 is the (initially empty) sequence $succ'_L$ of all and only the immediate successors in $succ_L(H)$ (where H is an invalid hypothesis) whose immediate predecessors are all invalid. Lines 3-7 manage the case when H is the empty hypothesis, where all the immediate successors have to be generated since all of them fall in $succ_L(H)$ and their only immediate predecessor is H. Each iteration of the loop at line 4 generates a new immediate successor H', according to the order from right to

Algorithm 2 Generating immediate successors.

Requires: H — a hypothesis
Requires: *current* — the sequence of hypotheses on the current layer of $\mathcal{H}$ as it is after H has been checked
Requires: D — the domain of the hypothesis space $\mathcal{H}$

```
1:  procedure GENCHILDREN(H, current, D):
2:      succ'_L ← <> — empty sequence
3:      if H = ∅ then
4:          for j ← 0 to |D| − 1 do
5:              bin(H') ← bin(H)
6:              bin(H')[j] ← 1
7:              succ'_L ← < H' > +succ'_L
8:      else
9:          pred ← pred2(H, current)
10:         k ← index of left-most 1 in bin(H)
11:         h ← index of right-most 1 in bin(H)
12:         for l ← k + 1 to |D| − 1 do
13:             if pred ≠ NIL then
14:                 bin(H') ← bin(H)
15:                 bin(H')[l] ← 1 — successor in succ_L
16:                 allfound ← true
17:                 for n ← k downto h do
18:                     if bin(H)[n] = 1 then
19:                         bin(H*) ← bin(H)
20:                         bin(H*)[n] ← 0
21:                         if pred ≠ H* then
22:                             allfound ← false
23:                             bin(H_F) ← bin(H')
24:                             bin(H_F)[h] ← 0
25:                             while pred ≠ NIL and
26:                                 bin(pred) ≤ bin(H_F)) do
27:                                 pred ←
28:                                     pred2(H, current)
29:                             break
30:                         else
31:                             pred ← pred2(H, current)
32:                 if allfound then
33:                     succ'_L ← < H' > +succ'_L
        return succ'_L
```

left: this is the reason for H' becomes the head of sequence $succ'_L$ (line 7). Since each H' is an immediate successor of H (according to Definition 2) and all immediate successors are encompassed, this case is sound and complete.

Lines 8-33 manage the case when H is an (invalid) hypothesis on a layer that is not the top one. Function *pred2(H, current)* is assumed to return the closer hypothesis to the left of the 'cursor' position in *current* (which is initially the position of H) whose Hamming distance from H is 2 (since, based on Corollary 3, two hypotheses can share an immediate successor only if the Hamming distance between them is 2). Such hypothesis is assigned to variable *pred*. The reason for identifying a hypothesis that is placed to the left of H is that all the hypotheses that share with H a successor in $succ_L(H)$ are to the left of H (according to Definition 3). Each iteration in the *for* loop at line 12 makes something just in case the value of variable *pred* is not *NIL*, that is, there is still a hypothesis to the left of H that can share a successor with H (line 13). Such an iteration generates (by exploiting Theorem 1) a hypothesis $H' \in succ_L(H)$, according to the order from right to left.

Then, it sets to true the Boolean variable *allfound*, which is expected to keep on being true only in case all the immediate predecessors of H' are in *current*, to the left of H. Based on Theorem 2, the inner *for* loop at line 17, at each iteration generates (according to an order from right to left) a hypothesis H^* to the left of H that is an immediate predecessor of H'. If (line 21) hypothesis *pred* does not equal H^*, this means that H^* is missing in *current*. In this case the value of variable *allfound* is changed to false (line 22); any immediate predecessors of H' in *current* is skipped, and variable *pred* is updated to the closer hypothesis (moving the cursor on the left) that is not a predecessor of H', if any (*while* loop at lines 25-28); then, the inner loop (line 29) is exited. If, instead, hypothesis *pred* equals H^* (line 30), *pred* is simply updated to the closer hypothesis to the left of the cursor whose Hamming distance w.r.t. H is 2, and a new iteration of the loop at line 17 is performed. When either this loop has been exited forcibly (line 29) or all its iterations have finished, if *allfound* is still true, hypothesis H' becomes the head of sequence $succ'_L$. Since H' is certainly an immediate successors in $succ_L(H)$ whose immediate predecessors are all invalid, the algorithm is sound. The algorithm terminates when all the hypotheses in $succ_L(H)$ have been encompassed by the cycle at line 12, therefore the algorithm is complete. □

Algorithm 1 is anytime in that, at whichever time the running process is halted, set $\mathcal{H}_+ \subseteq \mathcal{H}$, computed so far, includes indeed only MHSs. If the algorithm is not halted, termination is guaranteed as the hypothesis space is finite, and it returns all (and only) the valid hypotheses that are preferable under subset inclusion. This is indeed a template algorithm (or engine) that can be exploited for solving a variety of problems that require the exploration of the hypothesis space under the considered preference relation. Specifically, it can be adopted to solve any instance of the MHS problem.

Corollary 7. *Given a MHS problem instance as of Def. 1, Algorithm 1 can be used to compute the MHSs with the following inputs:*

- *function check(H) verifying whether H is indeed a hitting set;*
- *the domain of $\mathcal{H}$ for the considered instance, this being $D = M_{\mathcal{N}}$.*

Algorithm 1 can provide the means also for solving instances of a distributed MHS problem, as defined here below.

Definition 4. *Let $\mathcal{C}$ be a finite collection of finite sets C_i, each including all (and only) the MHSs relevant to a (non-given) finite collection $\mathcal{N}_i$ of finite sets that contain elements in the domain M. The problem of finding* all *the MHSs for the (non-given) collection $\mathcal{N} = \bigcup_i \mathcal{N}_i$ is referred to as the* distributed MHS problem. *Each MHS of $\mathcal{N}$ can include only elements occurring in $\mathcal{N}$, hence the domain can be restricted to $M_{\mathcal{N}} = \bigcup_i \bigcup_j N_j$ where $N_j \in \mathcal{N}_i$.*

It is easy to see that the search space of the distributed MHS problem is the same as for the (monolithic) MHS problem, and that the preference relation that has been adopted (subset containment) is appropriate also for the new problem. However, while it is easy to fancy an oracle for the MHS problem (for instance, given the hypothesis H to check, the oracle can compute the intersection of H with each set in the considered collection and return true if no intersection is empty), it is not straightforward to envisage one for the distributed MHS problem since $\mathcal{N}$ is unknown. A deeper insight in the distributed MHS problem is therefore needed.

Theorem 4. *Given a distributed MHS problem, a hypothesis $H \in \mathcal{H}$ is a hitting set of (the non-given) collection $\mathcal{N} = \bigcup_i \mathcal{N}_i$ if, and only if, for each $C_i \in \mathcal{C}$, there exists a (MHS of the non-given collection $\mathcal{N}_i$) $H_{si} \in C_i$ such that $H \in \{H_{si}\} \cup succ^*(H_{si})$.*

Proof. (sketch) Let us assume that collection $\mathcal{N}$ includes k elements.

($\rightarrow$) Let us assume that for each $i \in [1..k]$, there exists a (MHS of collection $\mathcal{N}_i$) $H_{si} \in C_i$ such that $H \in \{H_{si}\} \cup succ^*(H_{si})$. Then, for each $\mathcal{N}_i$, H is either a MHS ($H = H_{si}$) or a superset of a MHS ($H \in succ^*(H_{si})$), hence H hits all the sets of collection $\mathcal{N}_i$. Since it is so for all the k collections $\mathcal{N}_i$, H is a hitting set of their union $\mathcal{N}$.

($\leftarrow$) Let us assume that H is a hitting set of $\mathcal{N}$, hence it is a hitting set of each collection $\mathcal{N}_i \in \mathcal{N}$. This means that, for each $i \in [1..k]$, H is either a MHS of $\mathcal{N}_i$ or a superset of a MHS of $\mathcal{N}_i$, which implies that, for each $i \in [1..k]$, there exists a $H_{si} \in C_i$ s.t. $H \in \{H_{si}\} \cup succ^*(H_{si})$. □

Definition 5. *Assuming that collection $\mathcal{N}$ includes k elements $\mathcal{N}_i$, the sequence of all the H_{si} mentioned in Theorem 4, as ordered according to the value of $i \in [1..k]$, is a* justification *of the hitting set H of $\mathcal{N}$.*

The top-down exploration of the hypothesis space performed by Algorithm 1 guarantees to isolate the most preferred valid hypotheses according to the subset containment preference relation. In the distributed MHS problem, a hypothesis is valid if it is provided with a justification.

Corollary 8. *Given a distributed MHS problem instance as of Def. 4, Algorithm 1 can be used to compute the MHSs with the following inputs:*

- *function check(H) verifying whether H is indeed a hitting set of $\mathcal{N}$; based on Theorem 4, this amounts to finding out whether H is provided with a justification (see Definition 5);*
- *the domain of $\mathcal{H}$ for the considered instance, this being $D = M_{\mathcal{N}}$.*

4 Related Work and Conclusions

Solving the MHS problem is of interest for many applications, which has resulted in the availability of a selection of corresponding algorithms [Eiter *et al.*, 2008]. In a diagnostic context, de Kleer and Williams [de Kleer and Williams, 1987] as well as Reiter [Reiter, 1987] showed the relevance of MHSs and also presented some conflict-based solutions that were extended later on [Greiner *et al.*, 1989; Wotawa, 2001; Pill and Quaritsch, 2015] and complemented by approximative heuristic approaches [Abreu and van Gemund, 2009] and distributed / parallelized ones [Cardoso and Abreu, 2013; Jannach *et al.*, 2015].

Rather than presenting another solution whose search is based on the sets we want to hit (or some corresponding heuristic), in this paper we have proposed a method that is based on the exponential hypothesis space $\mathcal{H}$ as given by the powerset of the union of the elements in the sets that we

want to hit (or some domain). We have shown how to avoid exploring hypotheses that are less preferable than some hypothesis already proven to be valid. We have furthermore shown a way to explore $\mathcal{H}$ efficiently, without constructing or investigating any hypothesis twice, and focusing on relevant data only. Our algorithm ensures the minimality of the reported MHSs by construction, without having to perform subset-checks on the final outputs, which are instead needed by other performant solutions [Pill *et al.*, 2011] like the Boolean approach at MHS computation [Lin and Jiang, 2003; Pill and Quaritsch, 2012].

Our research was inspired by the performance [Nica *et al.*, 2013] of diagnosis methods that compute diagnoses directly from satisfying assignments to some special SAT problem [Metodi *et al.*, 2012; Feldman *et al.*, 2010], and the fact that we did not see such performance for a SAT encoding of MHS problems [Pill *et al.*, 2016]. The algorithm shown in this paper is a first step towards efficiently exploring the MHS search space, based on the hypothesis space rather than the "conflicts". Furthermore, it is also a step towards corresponding diagnosis solutions. In particular, in line with Corollary 7, we can easily search for diagnoses if function *check(H)* verifies whether a diagnosis H makes the observations compliant with the system model; this is actually the point in [Ceriani and Zanella, 2014]. The advantage of such an exploration, if compared to conflict-based searches, is that no solver (SAT-, constraint-, ...) is required, instead we need only a function that is able to judge the hypothesis (similar to a test oracle).

A direction for future work stems from the direct SAT solutions for diagnosis problems tending in their search more towards our exploration strategy, as discussed in the introduction. By exploiting our illustrated strategy, we aim to develop corresponding (e.g. SAT-based) solutions that are specifically tailored towards diagnosis purposes, rather than relying on a general purpose strategy solver.

An important future step for these ideas will be an experimental evaluation of the proposed initial search strategy for both (monolithic and distributed) MHS and MBD problems. Such an evaluation will allow us to optimize the depicted developments.

Acknowledgments

The work reported in this paper was supported in part by the European Commission under FP-7 agreement number 608770 (project "eDAS").

References

[Abreu and van Gemund, 2009] R. Abreu and A. van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *8th Symposium on Abstraction Reformulation, and Approximation (SARA)*, July 2009.

[Cardoso and Abreu, 2013] N. Cardoso and R. Abreu. MHS2: A map-reduce heuristic-driven minimal hitting set search algorithm. In *Multicore Software Engineering, Performance, and Tools - International Conference MUSEPAT*, pages 25–36, 2013.

[Ceriani and Zanella, 2014] L. Ceriani and M. Zanella. Model-based diagnosis and generation of hypothesis space via AI planning. In *25th Int. Workshop on the Principles of Diagnosis (DX-14)*, 2014.

[de Kleer and Williams, 1987] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[Eiter *et al.*, 2008] T. Eiter, K. Makino, and G. Gottlob. Computational aspects of monotone dualization: A brief survey. *Discrete Applied Mathematics*, 156(11):2035–2049, June 2008.

[Feldman *et al.*, 2010] A. Feldman, G. Provan, J. de Kleer, S. Robert, and A. van Gemund. Solving model-based diagnosis problems with Max-SAT solvers and vice versa. In *21st Int. Workshop on Principles of Diagnosis (DX-10)*, 2010.

[Greiner *et al.*, 1989] R. Greiner, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artif. Intelligence*, 41(1):79–88, 1989.

[Jannach *et al.*, 2015] D. Jannach, T. Schmitz, and K. M. Shchekotykhin. Parallelized hitting set computation for model-based diagnosis. In *29th AAAI Conference on Artificial Intelligence*, pages 1503–1510, 2015.

[Lin and Jiang, 2003] L. Lin and Y. Jiang. The computation of hitting sets: review and new algorithms. *Information Processing Letters*, 86:177–184, May 2003.

[Metodi *et al.*, 2012] A. Metodi, R. Stern, M. Kalech, and M. Codish. Compiling model-based diagnosis to Boolean satisfaction. In *AAAI Conference on Artificial Intelligence*, pages 793–799, 2012.

[Moskewicz *et al.*, 2001] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *38th Annual Design Automation Conference (DAC)*, pages 530–535, 2001.

[Nica *et al.*, 2013] I. Nica, I. Pill, T. Quaritsch, and F. Wotawa. The route to success - a performance comparison of diagnosis algorithms. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 1039–1045, 2013.

[Pill and Quaritsch, 2012] I. Pill and T. Quaritsch. Optimizations for the boolean approach to computing minimal hitting sets. In *20th European Conference on Artificial Intelligence (ECAI)*, pages 648–653, 2012.

[Pill and Quaritsch, 2015] I. Pill and T. Quaritsch. RC-tree: A variant avoiding all the redundancy in Reiter's minimal hitting set algorithm. In *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 78–84, 2015.

[Pill *et al.*, 2011] I. Pill, T. Quaritsch, and F. Wotawa. From conflicts to diagnoses: An empirical evaluation of minimal hitting set algorithms. In *22nd Int. Workshop on the Principles of Diagnosis (DX)*, pages 203–210, 2011.

[Pill *et al.*, 2016] Ingo Pill, Thomas Quaritsch, and Franz Wotawa. On the Practical Performance of Minimal Hitting Set Algorithms from a Diagnostic Perspective. *International Journal of Prognostics and Health Management*, 7(2), 2016. ISSN2153-2648, 2016 010.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Wotawa, 2001] F. Wotawa. A variant of Reiter's hitting-set algorithm. *Information Proc. Letters*, 79:45–51, 2001.