Experiments in the Hypothesis Space Approach to Model-based Diagnosis of Discrete-event Systems via AI Planning

Luca Ceriani and **Marina Zanella**
Department of Information Engineering, University of Brescia, Brescia, Italy
e-mail: luca.ceriani@ing.unibs.it, marina.zanella@unibs.it

Abstract

This document presents some experiments carried out within the context of a research on model-based diagnosis of discrete-event systems (DESs) partially described in [Ceriani and Zanella, 2014]. The task is performed according to the hypothesis space approach, as proposed by other authors in the literature [Grastien *et al.*, 2011; 2012]. More specifically, the experiments refer to hypothesis space H_{set}, that is, each hypothesis is a set of faults. First the model of the DESs adopted as a case study, which is based on a component model that was introduced in [Grastien *et al.*, 2007a; 2007b], is given. Then the pseudocode of the algorithms that have been run in the experiments is provided. Finally, the experimental results are displayed and commented.

1 System

Experiments have been carried out by considering two distributed non-diagnosable DESs, each consisting in a grid of $m \times n$ identical components (e.g. computers), where the values of m and n are different for either system. The model of every component was introduced in [Grastien *et al.*, 2007a; 2007b] and adopted also in [Sohrabi *et al.*, 2010].

Each component has four neighbors; corner and border components are neighbors to components on their opposite sides. A finite automaton representation of the behavior space of one component is provided in Figure 1, which is taken from the quoted papers where it is described this way: the initial state is O. When the component fails, its state changes from O to F and sends a message *reboot!* to its four neighbors. The neighbors receive the message *reboot?* and change their state from O to W, from WW to W, from R to R or from FF to FF, depending on their current state. There is (only) a faulty event per component, thus the set Σ_f of faulty events of either DES globally includes $n * m$ events. Events *IReboot* and *IAmBack* are the only observable ones.

The sender component of each observed event is assumed to be known, thus the set Σ_o of observable events of either DES globally includes $n * m * 2$ events.

The automaton in Figure 1 is not complying with the notions introduced in [Ceriani and Zanella, 2014] (Section 2.1), according to which message exchange between components is represented by means of synchronous events. Given a component C whose four neighbors are components U, R, D, and L, respectively, the model of C, corresponding to that in Figure 1 and compliant with [Ceriani and Zanella, 2014], is represented in Figure 2. The message *reboot!* sent by a component has become the synchronous event *reboot!C*, C being the name of the sender component. Analogously, the observable events have become *IRebootC* and *IAmBackC*, and event $rebooting$ has become $rebootingC$. The event in red is both faulty and synchronous, purple events are synchronous, blue events are observable.

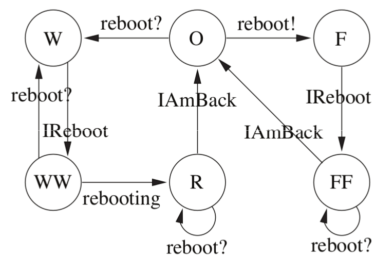

Figure 1: Component model from the literature

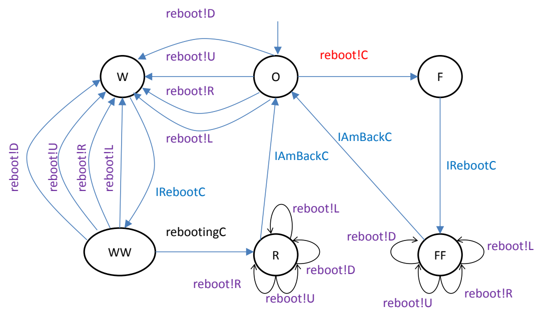

Figure 2: Component model

2 Observation

Each observation considered in the experiments includes N observable events, where N is 30% of the number of the components of the relevant DES.

The temporal uncertainty percentage of an observation that includes N observable events may vary from 0% (the N observable events are totally temporally ordered by $N *$

$(N-1)/2$ precedences) to 100% (all N events are temporally unrelated, since all $N*(N-1)/2$ precedences have been removed). Thus, there are $1+N*(N-1)/2$ distinct observations O, each of which includes the same observed events. Only three distinct (physically possible[1]) observations have been considered for each DES taken into account in the experiments. Each distinct pair (D,O), where D is a considered DES, that is, each distinct diagnosis problem, provides a test case.

3 Algorithms

The algorithm presented in [Ceriani and Zanella, 2014] has two versions, differing just in line 12. The first version, whose pseudocode is given here below, assumes that the domain of the values returned by the solver called at line 9 is $\{pass, fail\}$. This means that such a solver is just able to detect whether hypothesis h is a candidate (*pass*) or not (*fail*).

In this version, given the *current* queue of the poset hypothesis space, the created *next* queue includes all and only the immediate successors of all the non-candidate hypotheses in *current*.

Since the solver is assumed to be able just to process a single hypothesis, without exploring any portion of the hypothesis space, this version is referred to as the *basic* one.

```
1.   algorithm preferred_diagnosis(D,O)
2.     $h \leftarrow$ most preferred hypothesis
3.     $current \leftarrow \langle\ h\ \rangle$
4.     $\Delta_{\preceq} \leftarrow \emptyset$
5.     repeat
6.       $next \leftarrow$ empty queue
7.       repeat
8.         $h \leftarrow dequeue(current)$
9.         $result \leftarrow check(h, D, O)$
10.        if $result = pass$
11.          then $\Delta_{\preceq} \leftarrow \Delta_{\preceq} \cup \{h\}$
12.        if $result = pass$
13.          then remove any hypothesis in $succ_{ng}(h)$ from $next$
14.          else if $result = fail$
15.            then $\forall\ h' \in succ_g(h)$ do
16.              if all immediate predecessors (distinct from $h$)
                   of $h'$ are in $current$
17.                then $enqueue(next, h')$
18.      until $current$ is empty
19.      $current \leftarrow next$
20.    until $current$ is empty
21.    return $\Delta_{\preceq}$
```

The second version instead assumes that the domain of the values returned by the solver called at line 9 is $\{pass, fail, fail\_all\}$. Value *pass* is returned if h is a candidate, *fail* is returned if h is not a candidate but some (unknown) successor of its certainly is, and *fail_all* is returned in case neither h nor any of its (immediate or non immediate) successors is a candidate.

In this version, given the *current* queue of the poset hypothesis space, the created *next* queue includes all and only the immediate successors of all the hypotheses in *current* that are neither a candidate nor the root of a *complete* hierarchy that does not include any candidate. By *complete* hierarchy we mean that the considered hierarchy includes, at any level, all the successors of a hypothesis, be they generated directly from it ($succ_g$) or not ($succ_{ng}$).

Since the solver is assumed to be able to explore the complete hierarchy of any hypothesis that is passed to it as a parameter, we will refer to this version as that performing a *full look ahead* (FLA).

```
1.   algorithm preferred_diagnosis(D,O)
2.     $h \leftarrow$ most preferred hypothesis
3.     $current \leftarrow \langle\ h\ \rangle$
4.     $\Delta_{\preceq} \leftarrow \emptyset$
5.     repeat
6.       $next \leftarrow$ empty queue
7.       repeat
8.         $h \leftarrow dequeue(current)$
9.         $result \leftarrow check(h, D, O)$
10.        if $result = pass$
11.          then $\Delta_{\preceq} \leftarrow \Delta_{\preceq} \cup \{h\}$
12.        if $result = pass$ or $result = fail\_all$
13.          then remove any hypothesis in $succ_{ng}(h)$ from $next$
14.          else if $result = fail$
15.            then $\forall\ h' \in succ_g(h)$ do
16.              if all immediate predecessors (distinct from $h$)
                   of $h'$ are in $current$
17.                then $enqueue(next, h')$
18.      until $current$ is empty
19.      $current \leftarrow next$
20.    until $current$ is empty
21.    return $\Delta_{\preceq}$
```

A third version of the algorithm, slightly different from the one presented in [Ceriani and Zanella, 2014], assumes that the domain of the values returned by the solver called at line 9 is the same as for the second version, that is, $\{pass, fail, fail\_all\}$, but the meaning of *fail_all* is different: such a value is returned if and only if neither h nor any of its (immediate or non immediate) *generated* successors is a candidate.

In this version, given the *current* queue of the poset hypothesis space, the created *next* queue includes all and only the immediate successors of all the hypotheses in *current* that are neither a candidate nor the root of a *generated* hierarchy that does not include any candidate. By *generated* hierarchy we mean that the considered hierarchy rooted in h includes, at any level, only the successors of a hypothesis that are generated directly starting from it, that is the successors that belong to $succ_g(h)$. Since the solver is assumed to be able to explore the generated hierarchy of any hypothesis that is passed to it as a parameter, we will refer to this version as that performing a *right look ahead* (RLA), where *right* points out that the look ahead is limited to the successors of a given hypothesis that are shared with other hypotheses that are on the same level on the right of it.

The corresponding pseudocode is provided in the following.

```
1.   algorithm preferred_diagnosis(D,O)
2.     $h \leftarrow$ most preferred hypothesis
3.     $current \leftarrow \langle\ h\ \rangle$
```

[1]Since a randomly generated observation may be physically impossible, the consistency of each generated observation with the behavioral model of the DES at hand was checked before conducting any experiment by using a complete planner. In case an inconsistency were detected, a new observation having the same uncertainty degree was randomly generated and tested for consistency, etc.

```
4.     $\Delta_{\preceq} \leftarrow \emptyset$
5.     **repeat**
6.       $next \leftarrow$ empty queue
7.       **repeat**
8.         $h \leftarrow dequeue(current)$
9.         $result \leftarrow check(h, D, O)$
10.        **if** $result = pass$
11.          **then** $\Delta_{\preceq} \leftarrow \Delta_{\preceq} \cup \{h\}$
12.        **if** $result = pass$
13.          **then** remove any hypothesis in $succ_{ng}(h)$ from $next$
14.          **else if** $result = fail$ /* **and not**($result = fail\_all$)
15.            **then** $\forall\, h' \in succ_g(h)$ **do**
16.                 $enqueue(next, h')$
17.      **until** $current$ is empty
18.      $current \leftarrow next$
19.    **until** $current$ is empty
20.    **return** $\Delta_{\preceq}$
```

Note that, if the solver is not able to check whether all the successors of h are not candidates, in case *fail_all* is returned, we cannot remove from $next$ any of its successors belonging to $succ_{ng}(h)$, as any of them (or, recursively, of its successors) might be a candidate: this is the reason for line 12 has changed in the third version w.r.t. the second one. Another change has affected the body of the cycle at line 15: if h is not a candidate but some of its generated (immediate or non immediate) successor is a candidate, then all the immediate successors h' of h has to be appended to *next*, independently on whether all the predecessors of h' (distinct from h) are in $current$ or not. In fact, some predecessors of h' (of which h' is a non generated successor) might be missing as none of their generated successors is a candidate. However, some of their non generated successors, h' among them, might be a candidate, then h' has to be added to *next*.

4 Experiments

The experiments refer to hypothesis space H_{set}, this being the power set 2^{Σ_f} of the alphabet of faulty events Σ_f of each considered DES. Two DESs are taken into account, one being a grid of 5 $\times$ 4 components (system 1) and the other (system 2) a grid of 5 $\times$ 6 components. The obtained experimental results are shown in Table 1 for both system 1 and system 2, which are referred to as S1 and S2, respectively.

The table consists for three distinct horizontal parts, each showing the results relevant to each distinct version of the *preferred_diagnosis* algorithm (BASIC, FLA, and RLA). Each row in each horizontal part is relevant to a test case. Three observations, featuring three distinct uncertainty percentages ($\%U$), are considered: value 0% for no uncertainty, value 25% representing a situation when 25% of the temporal constraints between observed events have been removed, and value 50% representing a situation when half the temporal constraints between observed events have been removed. This leads to three distinct test cases for each system. Each observation is compiled into the planning problem the way presented in [Grastien *et al.*, 2007a; 2007b; Sohrabi *et al.*, 2010].

Column *NCOMPS* simply recalls the number of components in each system (20 components in S1 and 30 in S2).

Column *Time (sec)* reports the time (in seconds) taken by algorithm *preferred_diagnosis* to run the test relevant to the considered row. Of course, such a time includes the run times of all the calls to the solver. A timeout of 30 minutes was set for running each version of the algorithm on each test case. In case the run took more, the process was killed.

Column *Complete* contains value *yes* if the complete preferred diagnosis was produced (that is, all the preferred candidates were computed), *no* otherwise. Column *NC* gives the number of preferred candidates that were computed, while columns *Min L.* and *Min L.* represent the minimum and maximum cardinality of the computed preferred candidates (if any), respectively. Column *NH* represents the number of hypotheses of H_{set} checked by each algorithm.

The exploited hardware platform is a 64-bit PC, endowed with a single core AMD NEO 2,2 GHz microprocessor and a 4 GB RAM. The operating system environment is Linux Ubuntu 12.04 LTS.

The complete planner run in our experiments is *MFF* [Hoffmann, 2003], whose input is given in PDDL 2.1 level 2.

The experimental evidence shows that, whichever the test case, no complete preferred diagnosis could be computed within the time limit by the basic version of the algorithm. When the process was halted (after a 30 minute run), just one preferred candidate, whose cardinality is 3, had been computed for system 1 while no preferred candidate had been computed for system 2. Instead, the complete preferred diagnosis could be computed below the time limit by the other two versions of the algorithm for whichever test case.

The experiments relevant to versions FLA and RLA show that, as expected, the time needed to compute the preferred diagnosis increases with the number of components in the system and with the temporal uncertainty of the observation.

The experimental evidence shows that version FLA is much more efficient than the basic version, and that version RLA outperforms version FLA.

References

[Ceriani and Zanella, 2014] L. Ceriani and M. Zanella. Model-based diagnosis and generation of hypothesis space via ai planning. In *25th International Workshop on Principles of Diagnosis – DX'14*, Graz, A, 2014.

[Grastien *et al.*, 2007a] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *22nd National Conference on Artificial Intelligence – AAAI'07*, pages 305–310, Vancouver, BC, 2007.

[Grastien *et al.*, 2007b] A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Modeling and solving diagnosis of discrete-event systems via satisfiability. In *18th International Workshop on Principles of Diagnosis – DX'07*, pages 114–121, Nashville, TN, 2007.

[Grastien *et al.*, 2011] A. Grastien, P. Haslum, and S. Thiébaux. Exhaustive diagnosis of discrete event systems through exploration of the hypothesis space. In *22nd International Workshop on Principles of Diagnosis – DX'11*, pages 60–67, Murnau, D, 2011.

[Grastien *et al.*, 2012] A. Grastien, P. Haslum, and S. Thiébaux. Conflict-based diagnosis of discrete event systems: Theory and practice. In *13th International Conference on Principles of Knowledge Representation and Reasoning – KR 2012*, pages 4989–4996, Rome, I, 2012.

BASIC	**NC**	**Min L.**	**Max L.**	**NH**	**NCOMPS**	**%U**	**Complete**	**Time (sec)**
S1	1	3	3	2022	20	0	No	> 30min
S1	1	3	3	2022	20	25	No	> 30min
S1	1	3	3	2022	20	50	No	> 30min
S2	0	-	-	3908	30	0	No	> 30min
S2	0	-	-	4489	30	25	No	> 30min
S2	0	-	-	4489	30	50	No	> 30min
FLA	**NC**	**Min L.**	**Max L.**	**NH**	**NCOMPS**	**%U**	**Complete**	**Time (sec)**
S1	1	3	3	38	20	0	Yes	7
S1	1	3	3	38	20	25	Yes	9
S1	1	3	3	38	20	50	Yes	18
S2	3	4	5	132	30	0	Yes	302
S2	3	4	5	242	30	25	Yes	1062
S2	6	4	5	242	30	50	No	>30 min
RLA	**NC**	**Min L.**	**Max L.**	**NH**	**NCOMPS**	**%U**	**Complete**	**Time (sec)**
S1	1	3	3	43	20	0	Yes	4
S1	1	3	3	43	20	25	Yes	5
S1	1	3	3	43	20	50	Yes	8
S2	3	4	5	170	30	0	Yes	141
S2	3	4	5	348	30	25	Yes	605
S2	6	4	5	348	30	50	No	>30 min

Table 1: Experimental evidence

[Hoffmann, 2003] J. Hoffmann. The metric-ff planning system: Translating ignoring delete lists to numeric state variables. *Journal of Artificial Intelligence Research*, (20):291–341, 2003.

[Sohrabi *et al.*, 2010] S. Sohrabi, J.A. Baier, and S. McIlraith. Diagnosis as planning revisited. In *12th International Conference on Knowledge Representation and Reasoning – KR 2010*, pages 26–36, Toronto, Canada, 2010.