Self-Healing as a Combination of Consistency Checks and Conformant Planning Problems

Alban Grastien
Optimisation Research Group, NICTA
Artificial Intelligence Group, The Australian National University
Canberra Research Laboratory, Australia

Abstract

We introduce the problem of self healing, in which a system is asked to self diagnose and self repair. The two problems of computing the diagnosis and the repair are often solved separately. We show in this paper how to tie these two tasks together: a planner searches a prospective plan on a sample of the belief state; a diagnoser verifies the applicability of the plan and returns a state of the belief state (added to the sample) in which the plan is not applicable. This decomposition of the self healing process avoids the explicit computation of the belief state. Our experiments demonstrate that it scales much better than the traditional approach.

1 Introduction

Autonomous systems are subject to faults and require regular repair actions; systems capable of performing such tasks are called self healing. Finding the optimal repair involves solving a diagnosis problem (what may the current system state be?) together with a planning problem (what optimal/near optimal course of actions, applicable in all of the possible states, leads to an acceptable state?). In large, partially observable, systems computing an explicit "belief state" can be intractable; finding a plan applicable in all elements of this belief state can be also intractable.

In this paper we propose a method that avoids these two intractable problems. This method relies on the intuition that the full belief state is not necessary to find the appropriate repair. For instance, if a self-healing problem requires to make sure that n given machines are turned off and if the status (on or off) of these machines is unknown, then the belief state is comprised of 2^n states. However the optimal plan (*press the stop button on every machine*) happens to be the optimal plan of the state where none of the machines has been shut: this single state is "representative" of all the states in the belief state.

Our approach uses a planner to compute an optimal plan for a small sample of the belief state (at most dozens of elements); the plan is applicable in all these states and leads to the goal state. In order to validate the plan for the full belief state we search for an element of the belief state in which the plan is not applicable. To this end we define a new type of diagnoser that solves the following problem: find a possible behaviour of the system (that agrees with the model and the observations) that ends up in a state q in which the plan is not correct; this state q is added to the sample of the belief state so that the planner finds a more suitable repair plan at the next iteration. Failure on the part of the diagnoser to find such a behaviour proves that the plan is indeed correct. In practice the problem of verifying the correctness of a plan is reduced to a propositional satisfiability (SAT) problem that is unsatisfiable iff the plan is applicable in all states and that returns a counterexample if not.

The contributions of this paper are i) a formal definition of the self-healing problem, ii) the solving of self-healing as a combination of diagnosis and planning steps, and iii) the reduction of each step to SAT.

This work is performed in the context of discrete event systems [Cassandras and Lafortune, 1999]. As opposed to supervisory control, where actions (either active or passive, such as forbidding some events) are performed while the system is running, we follow the work from Cordier *et al.* [2007] and assume that the repair is being performed whilst the system is inactive.

The paper is divided as follows. Next section defines the self-healing problem formally. Section 3 presents the proposed algorithm with a set-based perspective. The SAT implementation is presented in Section 4. Experimental validation is given in Section 5. A comparison with other problems and approaches is given in Section 6.

2 Problem Definition

The problem we are addressing is illustrated on Figure 1. We are concerned with finding the most appropriate repair for a partially observed system that has been running freely.

We assume that the system can run in two different modes: the "active" (and useful) mode in which the system is free to operate (left half of the figure) and the "repair" mode in which the system state is being re-adjusted (right half). The system behaves quite differently in the two modes. In the active mode, the system is partially observable but uncontrolled. In the repair mode, the system is not observed albeit controlled; the state changes only through explicit application of actions; and special attention must be made to their

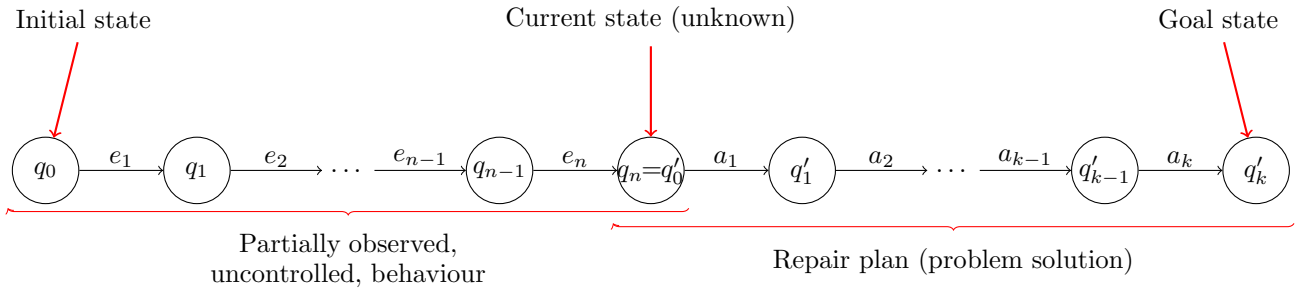

Figure 1: Schematic description of the self-healing problem: find a repair plan that returns the state in the goal set.

applicability/effects. One reason for assuming that the system does not run freely in the repair mode is that we do not want to consider scenarios where faults can occur during the repair, which would increase the overall complexity of the problem. We believe that this limitation, essentially the fact that the repair actions have deterministic effects, can be lifted.

2.1 Explicit Model

We are considering discrete event systems (DES, [Cassandras and Lafortune, 1999]). The system is modeled as a finite state machine, i.e., a finite set Q of states together with a set T of transitions labeled with finitely-many events/actions.

Definition 1 *An explicit* self-healing system model *is a tuple* $M = \langle Q, I, \Sigma, \Sigma_o, \Sigma_a, T, G, U\rangle$ *where*

- Q *is a finite set of states,* $I \subseteq Q$ *is a set of initial states,* $G \subseteq Q$ *is a set of goal states,* $U \subseteq Q$ *is a set of unstable states,*
- Σ *is a finite set of events,* $\Sigma_o \subseteq \Sigma$ *is the set of observable events,* $\Sigma_a \subseteq \Sigma$ *is the set of actions, and*
- $T \subseteq (Q \times \Sigma \times Q)$ *is the set of transitions* $\langle q, e, q'\rangle$ *also denoted* $q \xrightarrow{e} q'$.

In the active mode the system takes a path $\rho = q_0 \xrightarrow{e_1} \ldots \xrightarrow{e_n} q_n$ such that $\{e_1, \ldots, e_n\} \subseteq \Sigma \setminus \Sigma_a$, $q_0 \in I$ and $q_n \notin U$. This last condition is used to prevent situations where a fault happened right before the repair is applied, i.e., before any observation of this fault was made. This assumption is similar to the one made, e.g., by Lamperti and Zanella that the system is *quiescent* (no more event is about to happen) when diagnosis is performed [Lamperti and Zanella, 2003]. This assumption can be removed by assuming $U = Q$. Finally the observation $O = \text{obs}(\rho)$ of this path is the projection of $e_1, \ldots, e_n$ on the observable events Σ_o (i.e., all non-observable events are eliminated from the sequence).

In the repair mode a sequence of actions, called a plan $\pi = a_1, \ldots, a_k$, is applied ($\{a_1, \ldots, a_k\} \subseteq \Sigma_a$). From state $q'_0 \in Q$, the application of π leads to the (single) state $q'_k = \pi(q'_0)$ such that $q'_0 \xrightarrow{a_1} \ldots \xrightarrow{a_k} q'_k$. We assume that every action is applicable in every state (if this is not the case a non-goal sink state can be created where all inapplicable actions lead to) and have deterministic effects. If π leads q'_0 to a goal state, we say that π is correct for q'_0.

Notice that a plan is a simple sequence: we do not assume that additional observations are available at runtime. There is no probing action available. After non deterministic action effects, the use of conditional plans is a second natural extension of this work.

Definition 2 *The* self-healing problem *is a pair* $P = \langle M, O\rangle$ *where* M *is a model and* O *is an observation. A* repair plan *for* P *is a plan that is guaranteed to be correct in the current state. Formally a repair plan is a plan* π *such that*

$$\forall \rho = q_0 \xrightarrow{e_1} \ldots \xrightarrow{e_n} q_n. \\ (q_0 \in I \ \wedge \ \text{obs}(\rho) = O \ \wedge q_n \notin U) \Rightarrow \pi(q_n) \in G. \quad (1)$$

The set of repair plans is denoted $\Pi(M, O)$ or simply Π.

Given a cost function on sequences of actions, the objective of the self-healing problem is to find a cost-minimal repair plan (for simplicity we assume that such a plan exists):

$$\pi^\star = \arg\min_{\pi \in \Pi} \text{cost}(\pi).$$

This definition assumes a cost function that provides a total order on the plans. In practice we will try to minimise the number of actions (all actions have the same cost, the cost is cumulative) and break ties at random.

We see two main categories of self-healing problems, namely i) a recurring situation where the system is stopped regularly, which provides a good opportunity to perform corrective actions on the system; ii) a situation where a diagnoser/monitor detects an anomaly on the system and triggers a self-healing procedure. The present work is independent from how the problem was prompted.

2.2 Solving the Problem Explicitly

This paper works under the assumption that the system model is very large and that it is impractical to manipulate sets of states. We discuss this issue here and present some notations.

The simplest way to solve the self-healing problem is to compute the belief state and then compute the optimal plan for this set of states.

Given a model M and the observation O, the belief state $\mathcal{B}^O$ is defined as the set of states that the system

could be in:

$$\mathcal{B}^O = \{q \in Q \mid \exists \rho = q_0 \xrightarrow{e_1} \ldots \xrightarrow{e_n} q_n. \; q_0 \in I \;\wedge\; \text{obs}(\rho) = O \;\wedge\; q_n \notin U \;\wedge\; q = q_n\}.$$

Notice that the definition of the belief state matches the first part of Equation (1).

A *conformant plan* for the set of states $\mathcal{B}^O$ is a plan π that is correct for all states of $\mathcal{B}^O$: $\forall q \in \mathcal{B}^O.\ \pi(q) \in G$ (cf. Figure 2). Compared to the general definition of a conformant plan (a more detailled comparison is given in Section 6) we only deal with uncertainty on the initial state and we assume that actions have deterministic effects. Conformant planning is provably PSPACE-hard for explicit models.

We consider the conformant planning problem from the initial set of states $\mathcal{B}^O$ and use $b = |\mathcal{B}^O|$ to denote the size of $\mathcal{B}^O$. The problem can be solved by considering the finite state machine M' where each state of M' is a set of states of the original model and each transition from state S labeled by action a leads to $S' = \{q' \in Q \mid \exists q \in S.\ \langle q, a, q' \rangle \in T\}$. The initial state of M' is $\mathcal{B}^O$; a state S of M' is a goal state if it satisfies $S \subseteq G$. A plan π is a sequence of actions such that $\pi(\mathcal{B}^O)$ (in M') is a goal state. Because the original model is deterministic the transition $\langle S, a, S' \rangle$ is such that the size of S' is smaller than S. The number of states in M' is bounded by the sum of binomial coefficients $\binom{|Q|}{1} + \cdots + \binom{|Q|}{b}$.

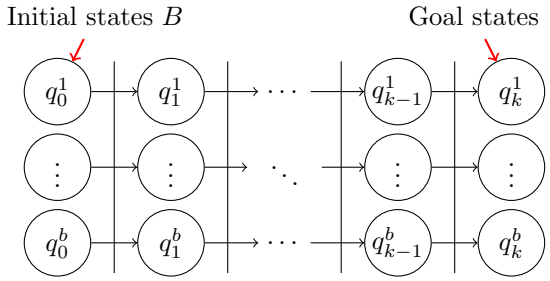

Figure 2: Solving conformant problems; the vertical lines mean that the transitions are labeled by the same action.

The model M' presented before cannot be easily expressed in planning modeling languages such as STRIPS or PDDL, or implemented in SAT. Another reduction, to M'', can be introduced whose states are tuples (with b elements) of states from the original model: $Q'' = Q^b$. A tuple state is a goal state if all its elements are in the goal: $G'' = G^b$. The transitions in M'' correspond to the parallel execution of the same action in each state of the tuple (represented by the vertical lines on Figure 2).

In general M'' is larger than M'. The model also contains symmetries that efficient implementations might need to address explicitely: for instance in model M'' states $\langle q_1, q_2 \rangle$ and $\langle q_2, q_1 \rangle$ are different while they would be the same in M': $\{q_1, q_2\} = \{q_2, q_1\}$.

Clearly this type of approach is only applicable if $\mathcal{B}^O$ comprises no more than a few dozen elements.

Finally we look at a formulation of the planning problem that is complementary to the computation of the belief state. Assume that a plan π is given and we want to compute the set of states $\mathcal{B}^\pi$ in which the plan π is correct: $\mathcal{B}^\pi = \{q \in Q \mid \pi(q) \in G\}$.

Lemma 1 *Plan π is a correct plan iff $\mathcal{B}^O \subseteq \mathcal{B}^\pi$.*

Writing $\overline{\mathcal{B}^\pi} \stackrel{\text{def}}{=} Q \setminus \mathcal{B}^\pi$ the set of states for which π is not correct, plan π is a correct plan iff $\mathcal{B}^O \cap \overline{\mathcal{B}^\pi} = \emptyset$.

3 Set Formulation of Self-Healing

We first present a formulation of our solution that is based on sets and that does not consider implementation issues (presented in the next section).

We propose a lazy approach to self-healing. In this approach we search a correct plan for a sample of the belief state (a "belief sample") and then search for a state of the belief state in which the plan is not applicable; this state is added to the sample and the procedure is iterated again until a robust plan has been found.

We first give the theoretical results that justify the algorithm presented at the end of the section.

In the following we use the notations $\mathcal{B}^O$ and B to represent sets of states such that $B \subseteq \mathcal{B}^O$. $\mathcal{B}^O$ will represent the belief state and B a small subset (a few elements) of $\mathcal{B}^O$. S, S' will represent any set of states.

Let $\Pi(q)$ be the set of repair plans that are correct for state q. Let $\Pi(S)$ be the set of repair plans that are correct whichever is the current state from S. Then $\Pi(S) = \bigcap_{q \in S} \Pi(q)$. Notice that $\Pi = \Pi(\mathcal{B}^O)$.

A trivial result is:

$$S \subseteq S' \Rightarrow \Pi(S) \supseteq \Pi(S').$$

A consequence of this proposition is that the optimal repair for $\mathcal{B}^O$ is a correct plan for B. Computing the optimal repair plan for the latter may therefore yield the optimal plan for the former. Let $\pi^*(S)$ be the optimal plan for a set of states. The next proposition determines how to characterize that an optimal plan was found:

$$S \subseteq S' \;\wedge\; (\pi^*(S) \in \Pi(S')) \Rightarrow \pi^*(S) = \pi^*(S').$$

This result can be derived from the previous proposition. $\pi^*(S')$ belongs to $\Pi(S)$ since $S \subseteq S'$; therefore $\pi^*(S)$ is better than (or equal to) $\pi^*(S')$. However, if $\pi^*(S) \in \Pi(S')$ and yet $\pi^*(S') \neq \pi^*(S)$, then $\pi^*(S')$ must be strictly better than $\pi^*(S)$, which contradicts what was just said.

Applied to $S = B$ and $S' = \mathcal{B}^O \supseteq B$, this means that $\pi^*(B) \in \Pi(\mathcal{B}^O)$ implies $\pi^*(B) = \pi^*(\mathcal{B}^O)$.

We reuse the notation $\mathcal{B}^\pi$ for the set of states in which the plan π is correct, and $\overline{\mathcal{B}^\pi} = Q \setminus \mathcal{B}^\pi$ for the set of states in which it is not. With this notation, $\pi^*(B) \in \Pi(\mathcal{B}^O)$ is equivalent to $\mathcal{B}^O \cap \overline{\mathcal{B}^{\pi^*(B)}} = \emptyset$.

Assume that there exists a procedure *verify_applicability*(S, π) that extracts a state $q \in S \cap \overline{\mathcal{B}^\pi}$ if such a state exists, and returns $\perp$ otherwise. Then, for $S \subseteq S'$, the following results are trivial:

- *verify_applicability*$(S', \pi^*(S)) = \perp \Rightarrow \pi^*(S) = \pi^*(S')$;

- let $q = \mathit{verify\_applicability}(S', \pi^*(S)) \neq \bot$ be a state where $\pi^*(S)$ is not applicable, then $q \notin S$ and $\pi^*(S \cup \{q\}) \neq \pi^*(S)$ (and $\mathit{cost}(\pi^*(S \cup \{q\})) > \mathit{cost}(\pi^*(S))$)[1].

The first proposition shows that *verify_applicability* can be used to check whether the plan $\pi^*(B)$ is correct for $\mathcal{B}^O$. The second proposition indicates how a better prospective plan can be computed if $\pi^*(B)$ is not correct: the addition of q to S guarantees that a different plan will be generated.

These results lead to the procedure presented in Algorithm 1. In this procedure, $\mathit{find\_plan}(B)$ is a method that computes a conformant plan from B as defined at the end of the previous section (and described next section). The procedure computes the optimal plan for a belief sample B. If *verify_applicability* finds a state $q \in \mathcal{B}^O$ in which this plan is not correct, then this state is added to the belief sample and a new optimal plan is generated and tested.

Algorithm 1 Diagnosis algorithm for the self-healing problem without enumerating the belief state $\mathcal{B}^O$

```
B := ∅
loop
  π := find_plan(B)
  q := verify_applicability(B^O, π)
  if q = ⊥ then
    return π
  else
    B := B ∪ {q}
  end if
end loop
```

Because i) each loop iteration adds an element to B and ii) $\mathcal{B}^O$ is finite, this procedure is guaranteed to terminate. The number of iteration is, in the worst case, the size of $\mathcal{B}^O$; we expect however that a handful of calls to $\mathit{find\_plan}(\cdot)$ will be sufficient to find the optimal plan.

Example

We illustrate Algorithm 1 with the example of Figure 3. Assume that the observations are $O = [o_1, o_2]$. According to the model, the belief state is $\mathcal{B}^O = \{A, D, F, H\}$ (state B is unstable, so the system cannot be in this state). The state needs to be returned to a subset of $\{A, G\}$.

Since the belief sample B_0 is initially empty, Algorithm 1 first generates the empty plan $\pi_0 = \varepsilon$. The procedure *verify_applicability* exhibits state F such that $B \xrightarrow{u} D \xrightarrow{o_1} E \xrightarrow{o_2} F$ could explain O and such that plan π_0 does not lead to a goal state when applied from F. The optimal plan for $B_1 = \{F\}$ is $\pi_1 = a_1$. This time *verify_applicability* extracts state H which also belongs to the belief state and for which the application of a_1 leads to sink state I. The belief sample B_2 now equals $\{F, H\}$ and the optimal conformant plan for B_2 is $\pi_2 = a_2, a_1$ (remember that unobservable transition $F \xrightarrow{u} H$ cannot trigger after the execution of a_2). This plan is correct for all elements in the belief state. Notice that neither A nor D from $\mathcal{B}^O$ were explicitly generated during the procedure.

[1]Remember that no two plans have the same cost.

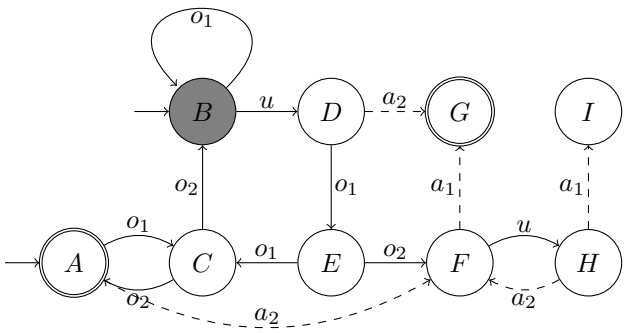

Figure 3: System example with two initial states (A and B), two goal states (A and G), one unstable state (B), two observable events (o_1 and o_2), and two actions (a_1 and a_2; an action affects the system state only if there is a transition).

4 SAT Formulation of Self-Healing

In this section we show how Algorithm 1 can be implemented using SAT. This implementation assumes a symbolic representation of the model, i.e., a representation where states and transitions are not enumerated but are, instead, implicitly defined by a set V of Boolean state variables (aka fluents) as can be found, e.g., in a STRIPS model.

4.1 Computing a Conformant Plan for B

The procedure we use to compute the optimal plan for a belief sample relies on a SAT solver and follows the schematic representation of Figure 2. In planning by SAT [Kautz and Selman, 1996], given a horizon k and a planning problem a propositional formula Φ is defined that is satisfiable iff there exists a sequence of actions of length k that solves the planning problem.[2] Furthermore Φ is defined over $k+1$ copies of the state variables (the state SAT variables p_0 to p_k where p is a state variable) and k copies of the actions (the action SAT variables a_0 to a_{k-1} where a is an action). Φ is defined such that a solution to the planning problem can be trivially extracted from the satisfying assignment (for instance, if a_i evaluates to *true*, then the ith action of the plan is a). If, for instance, action a sets state variable p to *false*, Φ will be defined such that for all $i \in \{1, \ldots, k\}$

$$\Phi \quad \equiv \quad (a_{i-1} \rightarrow \neg p_i) \wedge \cdots$$

We refer the reader to the literature on planning by SAT for more details on this reduction.

Given a sample B of b states we create b copies of the state SAT variables: $p_i^1, \ldots, p_i^b$; the variables p_i^ℓ model the effects of applying the plan on the state $q_\ell \in B$. We stick to a single set of action SAT variables and each copy of the state SAT variables is linked to this

[2]The value of k is initialized to 0 and incremented until Φ becomes satisfiable.

set. The formula Φ presented in the example above will therefore now translate as

$$\Phi \quad \equiv \quad \left(\left(a_{i-1} \rightarrow \neg p_i^1\right) \wedge \cdots \wedge \left(a_{i-1} \rightarrow \neg p_i^b\right)\right) \wedge \cdots$$

4.2 Verifying Correctness of a Plan

Like the plan generation, plan correctness is implemented in SAT. This time it matches the representation of Figure 1.

A plan is proved incorrect if an explanation of the observations can be found in which the application of the plan leads to a non final goal (remember that all plans are applicable).

Once again a propositional formula is defined that is satisfiable iff such an explanation exists. This formula contains two parts: SAT variables $p_{i\in\{0,\ldots,n\}}$ represent the state of the system in the active mode while variables $p'_{i\in\{0,\ldots,k\}}$ represent the state in the repair mode.[3] The formula is the conjunction of the formulas:

- Φ_{active} a propositional formula that is satisfiable iff there exists an explanation to the observations (whose final state is represented by the variables p_n); this type of reduction is quite standard [Grastien and Anbulagan, 2013];
- Φ'_{repair} a propositional formula that is satisfiable iff there exists a state in which the proposed plan is not correct (this state is represented by the variables p'_0);
- $\bigwedge_{p\in V}(p_n \leftrightarrow p'_0)$, where p ranges over the state variables, the formula that links the final state of the active phase and the initial state of the repair phase.

Intuitively, the assignments of the variables p_n that are consistent with Φ_{active} are a symbolic representation of $\mathcal{B}^O$. Formally let $\mathcal{V}$ be the set of variables that appear in Φ_{active}; then $\exists(\mathcal{V} \setminus \{p_n \mid p \in V\})$. Φ_{active} is logically equivalent to the symbolic representation of $\mathcal{B}^O$. Similarly the variables p'_0 of Φ'_{repair} represent $\overline{\mathcal{B}^\pi}$.

As a consequence any other representation of $\mathcal{B}^O$ or $\overline{\mathcal{B}^\pi}$ could be used if such representations are more convenient (e.g., if they are more compact or if they help the SAT solver).

Difference Between the Two Reductions

The first reduction aims at finding a plan of length k that is applicable in b states. Therefore it includes $b \times k$ copies of the state variables and k copies of the action variables.

The second reduction aims at finding a plan composed of two parts: a trajectory in the active space and a trajectory in the repair space. Therefore it includes $n + k$ copies of the state variables and n copies of the events (there could be k copies of the actions but the value of these variables is known in advance since the plan is an input of this reduction).

An interesting difference between the two reductions is that the trajectories of the former should lead to goal states while the trajectory of the latter should lead to a *non* goal state. As a consequence when the repair plan is finally computed the conformant planning reduction to SAT is satisfiable while the reduction of the applicability function is not.

[3] It is assumed that the length of the explanation can be bounded by a known value n; k is the length of the plan being tested.

5 Experiments

We ran some experimental evaluation of the approach presented in this paper.

Since the problem presented here is new, we had to build new benchmarks. We propose a variant of the benchmark presented by Grastien *et al.* [2007] which will be made available to the community. The system comprises 20 components interconnected in a torus shape. Each component contains eight states, including two unstable states and one goal state. The behaviour on each component can affect its neighbour and the local observations cannot allow to determine anything about the local behaviour: the full system needs to be monitored in order to understand the state system. Repair actions can also be local or affect several components.

We built 100 problem instances on this system. We restricted ourselves to totally ordered observations, but notice that one of the benefits of using diagnostic techniques is to be able to handle partially-ordered observations (observations where the order of the observed events is only partially known because the delay between their reception is small compared to the transmission/processing delay).

We compare our approach to a symbolic approach that uses BDDs (specifically the `buddy` package) to track the belief state and then uses A* to find the optimal repair plan. The heuristic used by A* is implemented as follows: a state of the system is extracted from the BDD and the optimal repair is computed for this state using SAT; the length of this optimal repair is used as a lower bound for the optimal repair from the current search node.

Our belief sample method uses `glucose_static 4.0` [Audemard and Simon, 2009]. `glucose` is heavily based on the `minisat` solver [Eén and Sörensson, 2003].

The experiments were run on 4-core 2.5GHz cpu with 4GB RAM, with GNU/Lunix Mint 16 "petra". A ten minutes (600s) timeout was provided.

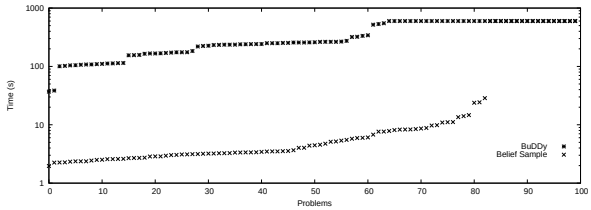

Figure 4: Runtime in seconds required to solve self-healing problem instances; sorted.

The results are summarized in Figure 4. The instances are sorted in increasing runtime, meaning that the instance at position x for one implementation may be different from the instance at the same position for the other. The approach based on the generation of the belief state only saw 64 instances solved before timeout, against 83 for our approach. In general our approach

is two orders of magnitude faster than A*, although we would need more benchmarks and comparisons to understand better the strength of this approach.

Out of the 87 instances instances solved by the Belief Sample method, 82 could be solved by exhibiting only one element of the belief state. Another three instances could be solved with a sample of two elements, and two required a sample of three elements to generate a conformant plan.

6 Discussion

The objective of connecting the diagnostic and planning tasks is quite ambitious. From the diagnostic perspective, and since the seminal work from Sampath *et al.* [1995] the problem has generally been the detection of specific events or patterns of events [Jéron *et al.*, 2006]. The main inspiration of the present work is the self-heability question asked by Cordier *et al.* [2007]; the aforementioned work is one of the first attempt to frame diagnosis as the problem of finding the optimal repair plan, although the complexity of computing the plan is not addressed. In static contexts similar questions have been asked where the problem was framed as finding the optimal balance between increasing the cost of gathering information (observations) and improving the precision of diagnosis (and, consequently, reducing the cost of planning) [Torta *et al.*, 2008].

Supervisory control [Ramadge and Wonham, 1989] is a problem very similar to self-healing. The goal is to control some actions (forbid their occurrence) in order to meet some specification. The main difference with our work is the fact that control applies continuously while we assume that self-healing is performed when the system is not active (either because the repair process is expensive—it might require to stop the system for instance—or because it can only be performed at some time—every night for instance). Furthermore control tries to be as unobtrusive as possible: it merely forbids some transitions and generally does not choose actions to perform.

Conformant planning [Smith and Weld, 1998] is the problem of finding a sequence of actions that is guaranteed to lead to the specified goal, despite uncertainty on the initial state and nondeterministic action effects. Solutions to conformant planning have been proposed that compute the belief state and run heuristic search [Bonet and Geffner, 2000] or that represent the belief state symbolically [Cimatti and Roveri, 2000]. More similar to our work Hoffmann and Brafman [2006] proposed Conformant-FF in which the belief state is represented implicitly by the set of initial states and the sequence of actions leading to the current state; at every time step, a SAT solver is used to determine the state variable values that can be inferred with certainty. This approach is similar to ours in the way it avoids computing belief states. More generally, we would like to adapt our method to solve conformant planning problems.

The combination of planning and diagnosis has also been studied in the context of *plan repair*. There, a (possibly conformant) plan is computed that assumes that contigencies are unlikely to happen. The plan execution is then monitored and if the outcome of execution does not match the predictions, a new plan is generated [Micalizio, 2014].

7 Conclusion and Extensions

In this paper we presented a method to solve the self-healing problem. The problem consists in finding a repair plan that can lead back to a goal state a system whose execution has been partially observed. We avoid computing the belief state. Instead we propose a method whereby plans are computed on a sample of the belief state whilst a diagnoser verifies their correctness and generates an element of the belief state (added to the sample) if the plan is not correct. Both the planning and the diagnosis problems are reduced to SAT problems. We show that non trivial problems can be easily solved by this approach.

There are many possible extensions to this work. One issue is that enforcing a conformant plan may be too restrictive. We want to avoid prohibitive repairs in situations where the system is healthy. This is a common problem in diagnosis of dynamic systems: the state of the system can never be precisely determined at the current time; it is often not unconceivable that a fault just happened on the system and has not had time to develop into a visible faulty trace. The issue here is that conformant plans must provide for such contingencies even when there is no evidence for them. An implicit assumption of our work is that unhealthy system behaviours can be detected to a large extend. The set of unstable states serves this purpose: they are useful to model the fact that any "failure" in the system will lead to abnormal observations before a repair action is performed.

We see two avenues to handle situations where the unstability feature cannot address the problem presented before. First probabilities can be incorporated into the model, which allows for chance-constrained planning [Santana and Williams, 2014]. Issues with this approach include the problem of building large models with meaningful probabilities and the problem of extending the SAT reduction to deal with probabilities (as well as scaling up to large models). A second, qualitative, possibility is to ignore contingencies that are supported by no strong evidence. For instance failures that are not part of a minimal diagnosis might be ignored.

Another restriction of the current approach is that the goal G is assumed to be known explicitly. Specification of goal states may however be more complex: Ciré and Botea [2008] have proposed to define goals as properties of states defined in linear temporal logic (LTL). Other relevant goal properties is diagnosability [Sampath *et al.*, 1995], i.e, the property that the observations on the system will allow to detect/identify the important system failures. A related issue is the incremental aspect: how to handle a repair after an active period following a first repair. A simple solution is to assume that the initial state after the repair is the goal state.

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the

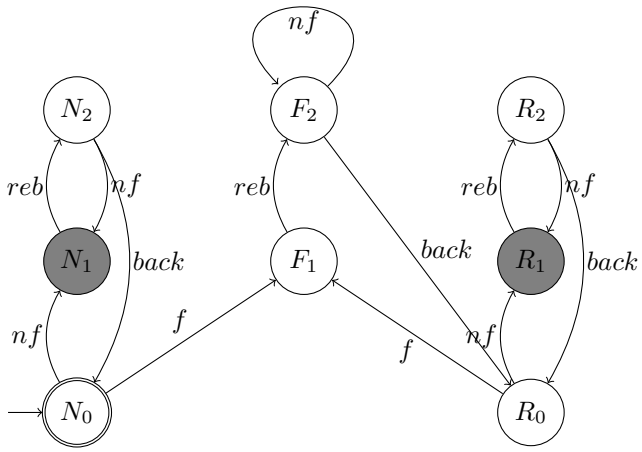

Figure 5: Active model for one component (observable events are *reb* and *back*).

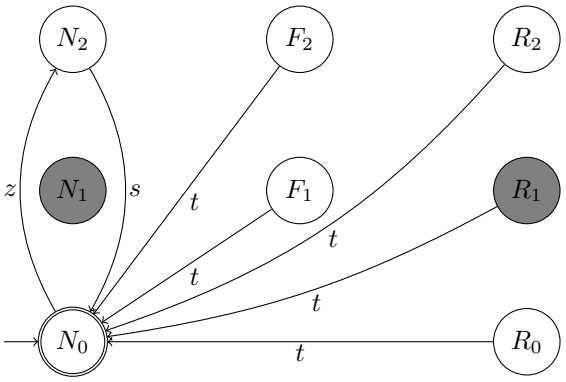

Figure 6: Repair model for one component (no transition means that the state is not affected by action).

Australian Research Council through the ICT Centre of Excellence Program.

A Problem Benchmark

We now present the system we used in the experiments.[4]

The system includes 20 components $c_{i,j}$ where i ranges between 0 and 3 and j between 0 and 4. The component $c_{i,j}$ is connected to $c_{i',j'}$ iff the total different $|i-i'|+|j-j'|$ is at most one (where i and j are taken modulo 3 and 4). For instance, $c_{0,1}$ is connected to four components $c_{0,0}$, $c_{0,2}$, $c_{3,1}$, and $c_{1,1}$.

The model of one component for the active mode is given in Figure 5 and the model for the repair mode is given in Figure 6. The connections between components implies forced transitions when some events occur; these are summarised in Table 1 For instance, when event f occurs on component $c_{0,1}$, event nf occurs on every one of its four neighbours.

A component state contains two types of information: whether a failure occurred on the component and whether it is run. The first part of the state is initially

[4]The benchmark is available at this address: `http://www.grastien.net/ban/data/bench-dx15.tar.gz`.

event/action	neighbour event/action
f	nf
t	z

Table 1: Synchronised events

N (no fault); it moves to F when a fault occurs and R when it recovers. The second part of the state is generally 0 (the component is running) and moves to 1 when it needs to reboot and to 2 when it is rebooting. A fault on a component forces its neighbours to reboot. One difficulty of diagnosis for this type of system is that the observations (*reb* and *back*) do not point precisely to the faulty component.

The repair consists in returning to state N_0. Most states require action t to return to state N_0 but this action can move the neighbours of the component to state N_2. Therefore finding the optimal repair requires to order the actions carefully.

References

[Audemard and Simon, 2009] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solver. In *21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[Bonet and Geffner, 2000] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Fifth International Conference on AI Planning and Scheduling (AIPS-00)*, pages 52–61, 2000.

[Cassandras and Lafortune, 1999] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Publishers, 1999.

[Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research (JAIR)*, 13:305–338, 2000.

[Ciré and Botea, 2008] A. Ciré and A. Botea. Learning in planning with temporally extended goals and uncontrollable events. In *Eighteenth European Conference on Artificial Intelligence (ECAI-08)*, pages 578–582, 2008.

[Cordier *et al.*, 2007] M.-O. Cordier, Y. Pencolé, L. Travé-Massuyès, and T. Vidal. Self-healability = diagnosability + repairability. In *Eighteenth International Workshop on Principles of Diagnosis (DX-07)*, pages 251–258, 2007.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *Sixth Conference on Theory and Applications of Satisfiability Testing (SAT-03)*, pages 333–336, 2003.

[Grastien and Anbulagan, 2013] A. Grastien and A. Anbulagan. Diagnosis of discrete event systems using satisfiability algorithms: a theoretical and empirical study. *IEEE Transactions on Automatic Control (TAC)*, 58(12):3070–3083, 2013.

[Grastien *et al.*, 2007] A. Grastien, A. Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In

22nd Conference on Artificial Intelligence (AAAI-07), pages 305–310, 2007.

[Hoffmann and Brafman, 2006] J. Hoffmann and R. Brafman. Conformant planning via heuristic forward search: a new approach. *Artificial Intelligence (AIJ)*, 170:507–541, 2006.

[Jéron *et al.*, 2006] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete-event systems diagnosis. In *Seventeenth International Workshop on Principles of Diagnosis (DX-06)*, pages 117–124, 2006.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope : planning, propositional logic, and stochastic search. In *Thirteenth Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, 1996.

[Lamperti and Zanella, 2003] G. Lamperti and M. Zanella. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.

[Micalizio, 2014] R. Micalizio. Plan repair driven by model-based agent diagnosis. *Intelligenza Artificiale*, 8(1):71–85, 2014.

[Ramadge and Wonham, 1989] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE: special issue on Dynamics of Discrete Event Systems*, 77(1):81–98, 1989.

[Sampath *et al.*, 1995] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control (TAC)*, 40(9):1555–1575, 1995.

[Santana and Williams, 2014] P. Santana and B. Williams. Chance-constrained consistency for probabilistic temporal plan networks. In *24th International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[Smith and Weld, 1998] D. Smith and D. Weld. Conformant graphplan. In *Fifteenth Conference on Artificial Intelligence (AAAI-98)*, pages 889–896, 1998.

[Torta *et al.*, 2008] G. Torta, D. Theseider Dupré, and L. Anselma. Hypothesis discrimination with abstractions based on observation and action costs. In *Nineteenth International Workshop on Principles of Diagnosis (DX-08)*, pages 189–196, 2008.