Assessing Diagnosis Algorithms: Of Sampling, Baselines, Metrics and Oracles

Ingo Pill
Institute of Software Engineering and Artificial Intelligence, TU Graz, Austria

Johan de Kleer
c-infinity, Mountain View, CA, USA

Abstract

Assessing and comparing diagnosis algorithms is a surprisingly complex challenge. We have to make decisions ranging from identifying the implications of the chosen baseline, via defining and ensuring a representative sampling strategy, to the choice of metric best suited to capture the computational, probing, or repair costs as well as the deviations from the baseline. We discuss several aspects of the overall challenge, identify related issues, and evaluate a special economic metric.

2012 ACM Subject Classification Computing methodologies → Causal reasoning and diagnostics

Keywords and phrases Model-based Diagnosis, Diagnosis, Algorithms

Digital Object Identifier 10.4230/OASIcs.DX.2025.5

1 Introduction

Given a symbolic or subsymbolic description of a system's structure along with a set of observations, the task of a diagnostic algorithm is to identify which system components might be faulty so that their failure explains the observed behavior. In this paper, we address the question of how to evaluate the *quality* of such an algorithm's conclusions and its actions.

At first glance, this might seem to be a straightforward task, but it is surprisingly difficult. As a thought experiment, imagine that we have all the faulty exemplars of a system that might ever exist their actual faults. For this exhaustive sample set, we then employ our diagnostic algorithm and determine the *costs* that the algorithm incurs while locating the *actual* fault(s).

There are several inherent difficulties with this simple concept: (1) it is impossible to implement in practice due to the required exhaustive set of faulty exemplars. (2) We need to specify what we mean by "costs." That is, whether we are interested in computational, probing, or economic costs. If we are interested in a combination, we need to define a weighted portfolio metric. In practice, the ideal cost function depends on the actual application scenario, so that it is (3) impossible to anticipate a specific user's needs and preferences in terms of costs. (4) A user will most certainly not have the resources to perform their own detailed evaluation of all combinations of available algorithms, potential cost functions, and faulty exemplars. Consequently, an evaluation will suffer from limitations in terms of sampling and quantification of costs and quality. Those limitations affect, in turn, the representativeness of the conclusions that we can draw from the experiments.

In practice, diagnostic algorithms are usually evaluated with simulated data. In particular, we take a simulation model of one or more systems, inject some fault(s) [22, 20], run the simulation for a (set of) specific input scenarios [22, 14], and then deploy the diagnostic algorithm to determine the incurred costs; we repeat this until we decide to have *enough* samples. That is, until we gained enough confidence in the representativeness of the results. We assume implicitly that the diagnostic algorithm is deterministic. Otherwise, we would need to run the algorithm multiple times for each diagnosis problem.

Further issues are related to the observability of signals, to uncontrollable contingencies such as noise that we face in the real world, or to an algorithm's individual characteristics such as whether it is based on a precise [4, 19] or probabilistic framework [12]. These issues most certainly affect the incurred costs, but it is their effect on the algorithm's results' quality that is most noticeable. A particularly intriguing challenge in this direction is that of defining the baseline against which we compare an algorithm's results. Related to these issues, we

- maybe can diagnose some fault(s) only after a delay [3] that the fault(s) need(s) to manifest [26] on observable signals.
- have to take into account that an algorithm can trade precision with resource expenditure [1]. This can certainly result in incomplete or over-approximated sets of diagnoses that individually over- or under-approximate the fault situation. This requires us to decide how to individually penalize delays, approximations, erroneous, and absent diagnoses.
- might not be able to isolate the actual fault(s) from the available data [11]. For example, if the fault is not triggered by the observed input scenarios. A fault may also remain hidden in observed signals while manifesting only in unobserved ones – a phenomenon that can be either triggered or suppressed by interactions among multiple faults. In extreme cases, some specific fault combinations could even result in an equivalent mutant. That is, a mutated system that is indistinguishable in its I/O behavior from the original one [7].
- have to define the desired baseline: Should the fault(s) injected into a simulation model (or the faults present in a real system) be considered as the ideal result which is obviously problematic for equivalent mutants (see Sec. 2.4), or should we seek to define a well-founded gold standard of diagnoses? To obtain the latter, we would take into account all the limitations of the available data and specify exactly what is theoretically diagnosable from the available (limited) observations *OBS* and the structural model *SD* [17].

In this paper, we isolate and discuss a variety of challenges that we face when aiming to assess diagnostic algorithms. Motivated by those discussions, we formulate definitions and outline metrics that allow us to systematically address the evaluation of diagnosis algorithms. Our ulterior aim is to support educated and quantitatively comparable answers to the fundamental question *Did a specific algorithm return the correct result, and if not, how good are the results?*. Our discussions aim to address evaluation needs that range from simple single-algorithm assessment to multi-algorithm comparisons for specific purposes, and finally to comprehensive evaluations in general contexts like competitions with multiple benchmarks.

We do not aim at a qualitative analysis that focuses on algorithmic concepts. We rather aim at an empirical evaluation of an algorithm's results that is (1) mostly agnostic of the implemented symbolic or subsymbolic concepts and (2) supports a quantitative assessment of an algorithm's results in terms of the quality of the conclusions and their costs. When we say mostly agnostic, we refer to the requirement of having to know how to interpret an algorithm's results. For example, algorithms like [4, 25, 19] report ambiguity groups formed by a subset-minimal diagnosis and its supersets implicitly in the results.

We organize our paper as follows. We analyze a variety of issues to illustrate the complexity of the task in Section 2. We then raise the discussion to a more formal level in Section 3, isolating appropriate notions and definitions that allow us to define and tackle two specific subtasks. Before we conclude in Section 5, we discuss in Section 4 a special economic metric that captures the effectiveness of the repair process and has been used in previous DX competitions [8].

2 Why is the representative evaluation/comparison of a diagnosis algorithm not as straightforward as it seems?

In this section, we seek to identify and illustrate a variety of issues that make the tasks of evaluating and comparing diagnosis algorithms not as straightforward and simple as they would seem at first glance. In the following subsections, we walk through a set of illustrating examples to expose individual challenges that we have to address. The observations we make in these sections will support us in specifying precise definitions in Section 3 that will allow us to lead formal discussions of the concepts envisaged in Sections 3.2 and 4.

2.1 Metrics based on sampling and the notion of ambiguity groups

Suppose that we want to design a good metric for single fault diagnosis algorithms. Assume that we have perfect measurements, which is really only possible in digital circuits. Let the components of the system be $X = \{x_1, ..., x_n\}$. Determining the samples to use to fairly test a diagnostic engine for a competition is a very complex problem that deserves a separate treatment, so let us assume for now that the set of *samples* is given. Let a simple diagnostic algorithm A return only one fault for a specific diagnosis problem s defined by an exemplar SD (with a known fault) and some observations OBS. If A returns the actual fault, the score $r(A, s)$ achieved for sample s is 1 and otherwise it is 0. An overall score $S(A)$ of algorithm A can, in turn, be defined as:

$$S(A) = \frac{\Sigma_{s \in samples} r(A, s)}{|samples|}, \tag{1}$$

such that we would sample over a set of diagnosis problems. Intuitively, the overall score estimates the probability of A producing the correct diagnosis for a random sample $s \in samples$. This simple assessment has numerous shortcomings:

1. *Probabilities:* How to take into account that different components fail at different rates?
2. *Ambiguity Groups:* Sometimes the same symptoms can arise from very different system failures and combinations of faults.
3. *Multiple Faults:* How to take into account that a system can suffer from multiple simultaneous faults?
4. *False Positives:* The sampling approach obviously penalizes false negatives (missed diagnoses), but does not adequately penalize false positives (reporting invalid diagnoses).

It is easy to see that the way we sample for computing $S(A)$ has a great impact on the overall score. A diagnostic algorithm is evaluated against a set of benchmark samples. Imagine a simple system of 4 components $\{x_1, x_2, x_3, x_4\}$ with priors $p(x_1) = 0.001, p(x_2) = 0.01, p(x_3) = 0.1, p(x_4) = 0.889$. If we sampled the fault scenarios representatively in the set of components, *i.e.*, considering an equal distribution, an algorithm A_1 that always scored correctly for components x_1, x_2, x_3 but not for x_4 would achieve a much better overall score $S(A_1)$ than A_2 that scored correctly on x_4, but not for x_1, x_2, x_3. And it does so, even though it is much less useful in practice, *i.e.*, since it scores well, s.t. $r(A_2, s) = 1$ for rare samples only. A possible solution to circumvent this would be to draw samples according to the prior distribution, but this requires far too many samples (*i.e.*, expensive simulations) and the score $S(A)$ would not be generalizable for varying priors.

An exponentially more efficient approach is to sample over each of the possible faults and weigh each sample with its prior. This greatly reduces the number of samples needed to obtain a reasonable score $S(A)$:

$$S(A) = \sum_x p(x) \frac{\Sigma_{s \in samples(x)} r(A,s)}{|samples(x)|}, \tag{2}$$

Due to $\sum_x p(x) = 1$, we see that $S(A)$ will be 1 iff $r(A,s)$ is always correct and 0 if it is always wrong. But this is a poor approach, since ambiguity groups introduce considerably more complexity. An *ambiguity group* is a set of diagnoses among which a diagnoser cannot distinguish given a specific set of observations. For example, consider diagnosing a sequence of n digital buffers where we can observe the input of $buffer_1/x_1$ and the output of $buffer_n/x_n$. If the input is $0/low/\bot$, and the output is $1/high/\top$, we know that one of the buffers is faulted, but not which one. So all n buffers form an ambiguity group. A similar challenge arises in the analog case where there are n resistors in a row, or the two inverter example from Section 2.4 illustrated in Fig. 1. Let us assume for now that the fault is in x_1. Considering the effects that an ambiguity group entails, we need to modify $r(A,s)$. That is, requiring A to return x_1 to achieve $r(A,s) = 1$ makes no sense. Indeed, a diagnoser should not be penalized because it returns a different element of the ambiguity group. We need to redefine what $r(A,s)$ returns. Let $G(s) = \{x | x \text{ is a diagnosis of } s\}$. If we require $r(A,s)$ to return x, it will be right only $\frac{1}{|G(s)|}$ of the time. If we require $r(A,s)$ to return one element of $G(s)$, a diagnoser may always pick the same element of the ambiguity group, making it useless for diagnosing other faults in that ambiguity group. If we require $r(A,s)$ to return the entire $G(s)$, then the diagnoser never gets partial credit. One approach is to give the diagnoser credit for the fraction of the ambiguity group it returns. So, for example, if there are two members in the ambiguity group and the diagnoser returns only one member of the ambiguity group, it would receive a score of $\frac{1}{2}$ for that sample s. Rewriting the prior equation to capture this intuition, we obtain

$$S(A) = \frac{\sum_{s \in allsamples} p(G(s)) \frac{|r(A,s)|}{|G(s)|}}{\sum_{s \in allsamples} p(G(s))} \tag{3}$$

where we assume the diagnoser is sound. This reduces the prior equation in the case there are no interesting ambiguity groups (of size greater than 1.)

In the multiple fault case, these challenges become only more difficult. Assume that we have a priori probability distribution over the set of diagnoses $\Delta_i \in 2^X$. Following the single fault case, we would like to write (where the Δ_i can be of any cardinality):

$$S(A) = \sum_{\Delta_i} p(\Delta_i) \frac{\Sigma_{s \in samples(\Delta_i)} r(A,s)}{|samples(\Delta_i)|} \tag{4}$$

But this has the same kinds of problems with ambiguity groups as discussed earlier. Consider the $n = 10$ buffer example. It has a large number of multiple faults in the ambiguity group as well: $\binom{10}{3} + \binom{10}{5} + ...$ A worse challenge arises in the analog case where there are n resistors in a row where the ambiguity group can be of size $2^n - 1$ because any subset of the resistors can be a diagnosis.

Contributing to this confusion is the fact that we have been mixing together two distinct ideas: (1) the construction of a fair benchmark and (2) the fair evaluation of a diagnostic algorithm on a benchmark. There are two important questions: (1) how to construct a benchmark that truely reflects actual occurring faults, and (2) given a set of samples, how do we fairly evaluate a diagnostic algorithm on this benchmark. Designers of a benchmark must concern themselves with the first question. This paper is primarily focused on the second question. So what we start with is the benchmark which is simply a set of samples

(input-output pairs). Let $G(s) = \{\Delta_i | \Delta_i \text{ is a diagnosis of } s\}$. This defines the ambiguity group of a sample. The prior $p(G(s)) = \sum_{\Delta_i \in G(s)} p(s)$. The prior single fault equation remains unchanged with the updated definition of $G(s)$

Some diagnostic algorithms return a posterior probability $V(\Delta)$ with each diagnosis. This requires a new approach, which we will discuss later.

So far we have penalized a diagnoser for false negatives, but not directly for false positives. But what should happen if the diagnoser returns a false positive, *i.e.,* a diagnoser returns a non-diagnosis. An approach for this has been well studied in machine learning. One can construct a confusion matrix with the diagnoser's results on the vertical axis and the actual faults on the horizontal. (This applies for both single faults and multiple faults.) Then accuracy, true positive rate, false positive rate, and precision can be defined in the usual way as in ML. This also needs to be adapted for ambiguity groups. In ML techniques like Top-k-accuracy can be used.

An approach which avoids the problems we have seen thus far is to base a metric on the difference between the true distribution of faults and the one determined by the diagnostic algorithm.

Let $Q(x)$ be the gold standard – the probability distribution which best approximates the state of affairs the single fault diagnoser encounters. Let $P(x)$ be the distribution the diagnoser reports. The most familiar metric is cross-entropy – which is used widely in ML to compare the predicted probability distributions. In our case, we want to compare how close the calculated probability distribution $P(x)$ is to the perfect one $Q(x)$. For this case, KL-divergence is a better metric:

$$D_{KL}(P||Q) = \Sigma_{x \in \mathcal{X}} log \frac{P(x)}{Q(x)} \tag{5}$$

If $P(x)$ and $Q(x)$ are identical, KL-divergence is 0, unlike cross-entropy. One problem is that if the diagnoser assigns non-zero probability to a possibility which never occurs, then KL-divergence is indeterminate. So this is problematic as well.

We hope we have convinced you that all the sampling based metrics described in this section are problematic measures of a diagnoser's capabilities. Although these metrics are often used because they appear to make sense on the surface, none of them is a decent scoring mechanism for a diagnostic algorithm. Instead, we argue diagnostic algorithms need to be scored based on the costs they incur in use.

Complementing an understanding of the problems associated with the metrics, we hope that we have convinced you also that the sampling strategy can have a strong impact on how we perceive some algorithms' performances. When designing benchmarks, we thus also need to anticipate the issues discussed in this section and define a set of samples that keeps those issues within certain and well-understood limits. In this respect, we would like to point out also that there are diagnosis algorithms that consider multiple samples in one diagnostic process [24], such that *SD* would stay the same for all s but *OBS* would change. As Pill and Wotawa mused in [23], supposedly representative concepts for generating test suites like combinatorial testing could help with defining representative sample sets for such algorithms. Ideally, their employment would allow us in turn to explain not only some occurred faults, but to isolate all the faults present in a system via an integrated approach at constructing representative I/O scenarios and diagnosing the corresponding observations. Such related work could help us not only in exploring options to assess multi-scenario diagnosis algorithms, but also when facing the challenge of designing benchmarks for diagnosis algorithms in general.

2.2 Metrics estimating repair costs

As we concluded in Sec. 2.1, sampling has a major impact on how an algorithm performs. But there are also a variety of concepts for assessing the quality of a diagnostic process, and our choice can significantly influence how we perceive an algorithm to perform for a specific sample s (and in general). In this subsection, our focus is on exploring the effects that a diagnosis algorithm's results have on the repair process.

Let us start with single fault diagnosis scenarios, and this time, the algorithm returns a probability distribution $p(x_1), ..., p(x_n)$ over the system's components $x_i \in X$. These are the posterior probabilities given some specific observations. For example, a system of three components $\{x_1, x_2, x_3\}$ might have posterior probabilities $p(x_1) = 0.1, p(x_2) = 0.6, p(x_3) = 0.3$. Assume for the moment that these posterior probabilities are the true ones. A simple repair approach might now replace x_i in decreasing posterior probability order. We first replace x_2 (due to the highest posterior probability), and if the symptoms are not gone, we consider x_3, and lastly x_1. Assuming the cost of a replacement $rep(x_i)$ was unity, the estimated repair costs C for these posterior probabilities are $rep(x_2) + rep(x_3) * (1 - p(x_2)) + rep(x_1) * (1 - p(x_2) - p(x_3)) = 1 + 0.4 + 0.1$. More formally, we can express C via the following equation for a given list L of diagnoses Δ_i that were sorted with decreasing probability.

$$C = \Sigma_{0<i\leq|L|}(rep(x_i) * (1 - \Sigma_{0<j<i}p(x_j))) \tag{6}$$

This simple repair process takes advantage of the fact that repairing a single component and making all repairs suggested by a diagnosis Δ_i is the same in a single-fault scenario. Obviously, we need to distinguish between the two for multi-fault scenarios though, which, in turn, requires us to adapt the repair strategy. Before we reason about such an update, let us first explore a sometimes hidden assumption though. That is, the assumption of whether we have *Perfect Fault Understanding (PFU)* when inspecting a component, or not.

In particular, similar to *perfect bug understanding* from the software engineering community (as it is relevant in a diagnostic context when assessing spectrum-based fault localization metrics), we have to decide whether or not we can assume the ability to recognize faults with complete certainty when inspecting some component. A simple illustrating scenario would be that of when we would inspect a software program by looking at a part of the code (the component), recognize the faulty code in it, and then fix the component's code accordingly. In the single fault case, we can observe whether the problematic behavior is gone when executing the system after the repair of some component x_j (as suggested by a single-fault diagnosis Δ). So, without requiring PFU, we could just replace x_j and avoid a detailed inspection and fault-oriented repair of x_j.
PFU would be an important assumption in multi-fault scenarios insofar, since we would need to replace all x_j in a diagnosis Δ to supposedly make the observed symptoms disappear. So, when checking the quality of the individual repairs via observing the system's behavior, we would first have to repair *all* $x_i \in \Delta$, and then we would observe executions of the supposedly fault-free system. Consequently, judging the situation after inspecting/repairing a single component x_i from Δ would require PFU of x_i.

A repair strategy for multi-fault scenarios could exploit PFU as follows. Choose a component x_i that appears in many diagnoses and then investigate and, if needed, repair x_i. If x_i was found to be correct in the inspection (due to PFU), all diagnoses Δ_j containing x_i can be discarded s.t. they get a probability of 0, and the probabilities $P(\Theta)$ for all diagnoses that are still possible (with search space $\Theta = 2^X$) should be updated accordingly. We repeat this until the symptoms have gone, there is no diagnosis left, or until there is no component left to repair. Obviously, estimating the costs C of this repair process is a bit

more complicated than for a single fault scenario. That is, we need to constantly update $P(2^X)$ after investigating some x_i, and we have to statistically approximate the costs incurred when estimating C.

If we do not have PFU (like when we just swap components without inspection), we could simply walk through the list of diagnoses Δ_i according to their probabilities and repair for each investigated Δ_i all components $x_j \in \Delta_i$. Extending our Eq. 6, we can approximate the average repair costs in this case as

$$C = \Sigma_{0<i\leq|L|}((\Sigma_{x_k\in\Delta_i} rep(x_k)) * (1 - \Sigma_{0<j<i} p(\Delta_j))), \tag{7}$$

with $rep(x_k)$ referring to the repair costs of component x_k. When following this process, one specific downside is that a certain component x_k might get repaired more than once. That is, if it is contained in more than one Δ_i that we had to investigate before arriving at the actual diagnosis (so that the system is finally repaired).

A *supposedly* better repair process would keep track of whether individual components have been repaired before, so as to avoid unnecessary repairs. However, this requires that a repaired component is not destroyed again during the repair process. That is, before all x_i in the actual diagnosis have been repaired. In practice, this can easily happen, when the interaction of faults leads to the immediate destruction of a repaired component x_j by some x_k that has not yet been repaired. A simple example would be that of a fuse. The fuse could be immediately destroyed again if the problem that triggered it has not yet been taken care of. Considering such dependencies, the more naive and also supposedly more costly process could even be better suited (and cheaper) in practice for many applications.

Please note that the metrics we considered in this subsection are simplistic variants of the economic metric discussed in Section 4.

2.3 Metrics tailoring to specific evaluation requirements

When researchers present a new algorithm, they often evaluate it in the context of a family of relevant algorithms and use specific metrics that allow them to show the validity and effectiveness of the proposed improvements over the state-of-the-art. This approach enables a clear and focused presentation of these improvements and allows the authors to adhere to the page limits of a conference at the same time. However, it comes with certain limitations.

Let us illustrate them with the example of RC-Tree [19]. RC-Tree is a diagnosis algorithm that implements a conflict-driven computation as first proposed by Reiter [25] as well as de Kleer and Williams [4] in their seminal papers. The computational concept behind RC-Tree is close to that of HS-DAG [10], but due to some additional reasoning in terms of a divide-and-conquer exploration of the search space, RC-Tree can avoid all redundancy in the search. Intuitively, any diagnosis candidate gets generated only via one tree-based exploration sequence (in contrast to HS-DAG that allows permutations), which results in a narrower search without losing important properties like completeness, soundness, being an anytime algorithm, or supporting an on-the-fly calculation of the conflicts. When evaluating RC-Tree, the authors proved (1) that it returns diagnoses adhering to Reiter's theory in a sound and complete manner (see Defs. 4 and 5 in Section 3.1), and they conducted (2) an empirical evaluation that isolated the performance advantages over HS-DAG as encountered in practice. The algorithm's run-time and the number of evaluated diagnosis candidates (which relate to tree nodes and thus memory consumption) served as metrics in those experiments. The resulting evaluation certainly meets one's expectations in terms of elucidating the authors' contributions. But it does not provide us with a holistic picture on how this algorithm

compares to the entire algorithmic landscape. That is, in comparison to completely orthogonal concepts – like subsymbolic algorithms that would approximate probability distributions (see Section 2.1).

The use of metrics that focus, *e.g.*, on repair costs (see Sec. 2.2) could provide a clearer picture, but those metrics are not suited for highlighting RC-Tree's algorithmic improvements over the state-of-the-art. Experts can draw on their expertise to maintain their own educated picture of the algorithmic landscape. For example, since RC-Tree returns the same diagnoses as HS-DAG (which has been around for about four decades), we can infer that the repair costs and other aspects can be considered to be the same and that there is only a difference in the computational costs. Consequently, this allows an expert to put RC-Tree into perspective. A non-expert would hardly have the knowledge to make such deductions. Metrics and evaluations that are more informative from a global perspective could thus help them in establishing a holistic picture as quickly as possible. Only such a holistic picture would allow a diagnostician or engineer to make an educated decision when faced with the challenge of having to select the most suitable diagnosis algorithm for a specific diagnosis problem.

2.4 Defining the baseline: On the ideal results of a diagnosis algorithm

When we assess the performance of a diagnosis algorithm, we need to think not only about sampling and metrics, but we also need to know how the ideal results should look like. In this subsection, we focus on the latter, and we hope to convince you that this supposedly simple task is not as straightforward to address as it seems.

In order to illustrate some of the issues, let us have a look at a simple example in the form of the circuit illustrated in Fig. 1. It is a simplified variant of the n-buffer example from Sec. 2.1 and contains. two inverters. Assume that we can observe the input in_1 and the output out_2 and let us consider three scenarios for an input value of $in_1 = \top/1/high$.

It is easy to see that the nominal output (such that both inverters work correctly) is $out_2 = \top$. If we observe this expected output in practice, we must recall though that this would be the case also for some fault scenarios. That is, observability limitations (like that can't observe out_1) might prohibit us to distinguish the nominal case from some fault scenarios. For our example, this would be the case for Fault Scenario 1 where Inverter 1 suffers from a stuck-at-zero fault so that out_1 is always $\bot$. The same is true for for Fault Scenario 2 where Inverter 2 suffers from a stuck-at-one fault. In both cases, the fault does not manifest in the observations, but this would require the input to be$in_1 = \bot/0/low$. Consequently, the nominal case, Fault Scenario 1 and Fault Scenario 2 form an ambiguity group (see Sec. 2.1). A consistency-based diagnosis algorithm like RC-Tree [19], HS-DAG [10], or GDE [4] would report from this ambiguity group only the empty diagnosis which represents nominal behavior. In fact, the underlying computation concept of those algorithms entails that when using a weak fault model, any superset of a diagnosis Δ is also an explanation. Thus they form sort of an ambiguity group with Δ by default and are, in turn, not explicitly

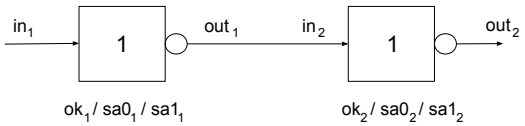

Figure 1 A simple electronic circuit with two inverters, taken from [21].

reported. From this example, we can easily see that we need to know *how to interpret a specific algorithm's results in terms of aspects like implicit ambiguity groups, and we need to decide how we would formalize this in the baseline.*

Let us now consider Fault Scenario 3 which illustrates a special case of an ambiguity group. In this fault scenario, both inverters malfunction so that they act as buffers and pass their input values along instead of inverting them. For this double fault, we can easily observe that there is no input that can reveal it. That is, unless we would improve the observability. Either via monitoring the intermediate signal $out_1 (= in_2)$, or by adding a connection from out_1 to some circuit part that is connected to a different observable output. In fact, we can observe that the system behaves *exactly* as intended on its observable interfaces not only for Fault Scenario 3, but for all I/O scenarios. In particular, the two faults cancel each other's negative effects on the system's output so that there *cannot* be any evidence of this double fault on the observable signals. Since the mutated system is I/O-conformant, we can refer to it as *equivalent mutant* [7].
From a designer's perspective, such an equivalent mutant is among equivalent design choices for implementing the desired I/O behavior. This presents us with quite interesting challenges from a diagnostic perspective. That is,*we need to explore the question of whether a diagnosis algorithm is supposed to report equivalent mutants as diagnoses or not, and, potentially, we need to identify the set of equivalent mutants.*

For elucidating further issues, let us consider two general concepts for defining a baseline, *i.e.*, (1) using the ground truth, and (2) defining/computing a golden set of diagnoses.

For the first option, we would use ground truth knowledge about the faults that are present in a system and would expect an algorithm to report it. Assume that we, *e.g.*, have a simulation-based setup like we discussed it in the Introduction. Since we know which faults we injected and when, we could simply consider this ground truth as gold standard for the diagnosis algorithm's results. This choice would be certainly intuitive and natural, but it comes with several problems, as we can illustrate using the above fault scenarios.
Consider Fault Scenario 3, where, no matter the input, a diagnosis algorithm is likely to completely fail if we do not consider implicit ambiguity groups. That is, since it will most probably report the empty diagnosis instead of the double fault.
But there are more issues. So, what if the faults have not manifested *yet* and there is potential subsequent behavior for nominal and faulty cases? That is, despite the fault being present we could have (a) for a stateless system that we might not *yet* have seen an input combination that manifests the fault(s) and (b) for a stateful system that we did not *yet* experience the delay that is necessary for the manifestation. While the baseline would indeed require the faults' detection, the question is on which basis an algorithm would justify that it reports a related diagnosis? Would we consider the algorithm then to be (a) clairvoyant (see Sec. 3), does it simply report (b) a statistic assessment with corresponding probabilities for all the possible continuations and their fault combinations/diagnoses, or (c) should the diagnosis actually be considered spurious – despite being defined in the baseline?

Reflecting on these issues, we might thus rather choose to define the baseline via computing a well-founded set of diagnoses that takes into account limitations in terms of observability and diagnosability. Although this approach would allow us to address some of the issues mentioned above, it comes with *increased computational costs* and causes its own problems. Let us consider first the diagnosis of a stateless system like the two-inverter circuit. If a system is stateless, its behavior depends only on the current input which means in a diagnostic context that we can diagnose each time step individually. The same holds then also for the computation of the baseline for which we can either use a proven algorithm or manual

analysis. The split into independent computations allows us to address a potential scalability issue that could otherwise arise with larger systems and long computation sequences and their traces. With the focus on individual time steps, the sizes of *SD* and *OBS* are minimized for each individual computation. Still, *we need either an algorithm that can compute the baseline, or we need to manually inspect the diagnosis problem(s).*
For a stateful system, the computational demands might be even higher. That is, we have to consider the entire history of observations and can't split the execution into smaller problems for each time step. The simple reason is that this history is reflected in an internal (potentially unobservable) state [18] that is relevant for the system's I/O relation. Like the authors did for [18, 9], we can still use one health state variable for the entire execution. And this allows us to constrain the exponential diagnosis search space to 2^X instead of $2^{X \times T}$ (for a weak fault model and a temporal horizon of length T). We might also not loose timing information when a fault manifests since this information might be present in the underlying computation (like in the conflicts when using RC-Tree).But the models for *SD* and *OBS* will still grow linearly with T.
It is important to note that these scalability issues are further exacerbated when we evaluate diagnosis algorithms that consider not only one but a set of executions [24] – like when a diagnostic algorithm examines *all* failed tests after executing a test suite [23].

For our final argument, assume that we are investigating a live scenario and that we managed to compute the gold standard of diagnoses for all the individual prefixes. As we suggested in the introduction, we now *have to decide how to penalize the various types of deviation from the baseline.* We have, for instance, that the diagnoses reported could be over- or under-approximations of diagnoses in the baseline. Or there could be a delay in detecting a diagnosis, which would require us to compare the diagnoses reported for time step t_i with the baseline for time step $t_j \neq t_i$. Accommodating these and a variety of other deviations entails another increase in the computational costs and it illustrates that we must not consider metrics and baselines in isolation. In Section 3.1, we will define the notions of near-soundness and near-completeness that will allow us to discuss this in more detail and to support automated reasoning in this context.

3 A more formal take on the problem at hand

In this section, we move our discussion to a more formal level. We start by developing some formal definitions in Section 3.1. Based on these definitions, in Section 3.2, we look at how we can answer the questions of whether *a specific diagnosis algorithm returned the correct result for some diagnosis problem, and if not, how good the results would be.* In particular, we propose to implement an *oracle* for providing a yes/no answer, and we design a *variety of metrics* for quantifying how far off an algorithm is from reporting the ideal results. These metrics can be used to derive an answer for the second (part of the) question, but also in the isolation of a yes/no answer to the first (part).

3.1 Some definitions

Note that the notation for some of our definitions deviates from what the reader might be familiar with from the literature and papers like [4, 25, 19]. The purpose of these deviations is to accommodate the results from as many diagnosis algorithms/approaches as possible, including results like diagnosis probability distributions that were computed under the consideration of strong fault models.

Assume that a system SD consists of components $x_i \in X$, and that we have some observations OBS about the behavior of SD[1]. As indicated in Section 2.1 and following the literature [4, 25], we can define from SD, X and OBS a diagnosis problem (SD, X, OBS) where we, *e.g.*, sampled over such diagnosis problems in Eq. 1. A diagnosis $\Delta \subseteq X$ for a diagnosis problem points us to a subset of faulty components that can explain OBS. To this end, a diagnosis assigns each component $x_i \in X$ a mode $m_j \in M_i$, where M_i is the finite set of modes in which a component x_i can be.
M_i must contain the nominal mode m_{nom} and at least one fault mode. This can be the general fault mode m_{gen}, but also one or more concrete fault modes m_k. The general fault mode m_{gen} implements a *weak fault model* [4, 25] that does not place any restrictions on the behavior of x_i in the faulty case. In contrast, concrete fault modes pose concrete constraints on the behavior of x_i also in the faulty case. Technically, there is no requirement to describe the exact behavior of a *strong fault model* but only some constraint. In practice, though, we tend to do so and define concrete faulty behavior – like a stuck-at-0 fault for a logic gate. As the respective authors outlined in [5, 25], exact strong fault models have advantages and disadvantages. For example, they indicate to a user what exactly happened. This information is certainly welcome during the inspection and repair of the system. When diagnosing specifications or design ideas, using a strong fault model allows us to even suggest repairs [18], which we can exploit in *generative model-based diagnosis* for design purposes as suggested in [17]. But the advantages are secured at the cost of an increased search space, since we now have not only two modes per component, but multiple ones. On an algorithmic level, we are then also limited by the chosen fault modes in our diagnostic search.
We consider nominal mode assignments in a diagnosis Δ to be optional, so that we require only those tuples (x_i, m_j) to be specified in Δ that assign a non-nominal mode $m_j \in M_i \setminus \{m_{nom}\}$ to some component x_i. All components $x_i \in X$ for which Δ does not contain an assignment tuple are assumed to be in nominal node. If we state that a diagnosis Δ is a subset of X, like the definitions in [4, 25] suggest, we refer to the observation that Δ defines a subset of components that are faulty.

▶ **Definition 1** (diagnosis problem, diagnosis, subset-minimal diagnosis, diagnosis candidate)**.** *Given a system SD, a set OBS of observations about SD's behavior, a set of components X, and for each $x_i \in X$ a set of modes M_i that x_i can take and where M_i contains x_i's nominal mode m_{nom} and at least one fault mode. A diagnosis Δ for a* diagnosis problem *(SD, X, OBS) is defined as a set of mode assignments (x_j, m_k), such that component x_j is assigned mode $m_k \in M_j$ and there are no two assignments (x_j, m_k) and (x_l, m_m) that refer to the same component (s.t. $x_j = x_l$). We allow nominal mode assignments to be optional in Δ, so that any $x_j \in X$ where Δ contains no mode assignment $(x_j, *)$ is assigned m_{nom} by default. A set of mode assignments, a.k.a.* diagnosis candidate *Δ becomes a diagnosis if, and only if the diagnosis algorithm assigns it a probability higher than zero (s.t. $p(\Delta) > 0$) such that it supposedly can explain OBS. A diagnosis Δ is subset-minimal, if and only if there is no diagnosis Δ' s.t. $\Delta' \subset \Delta$.*

Like in Section 2.1, we assume that the results of a diagnosis algorithm are given as a probability distribution $P(\Theta)$ over the diagnosis space Θ. The gold standard in terms of the expected results of an algorithm will be referred to as $Q(\Theta)$, where we discuss in Section 2.4 several concepts to define this baseline. If an algorithm algorithm does not report

[1] As usual, we will also formally refer to a system model with SD and it will be clear from the context whether we refer to the system in general or a model

probabilities but only a set of solutions, a naive concept to create a distribution is to assign all *diagnoses* the same probability and non-diagnoses a probability of zero. More elaborate approaches would consider the components' priors and a diagnosis Δ's cardinality when computing $p(\Delta)$, but let us continue this discussion later in this section.
Based on the individual M_i from Def. 1, the search space for diagnoses Θ varies in its size and structure. Let us first consider the cases where $\forall x_i \in X : M_i = \{m_{nom}, m_{gen}\}$, so that we implement a weak fault model. In this case, $P(\Theta)$ and $Q(\Theta)$ are distributions over the diagnosis space $\Theta = 2^X$. This space grows to M^X if we implement strong fault models such that M is a general set of modes for all $x \in X$. If the fault sets vary between individual components (which is likely to be the case in practice), we can use individual mode sets M_i and $\Theta = \Pi_{x_i \in X} M_i$ for a more precise characterization.

▶ **Definition 2** (diagnosis probability distribution)**.** *Given a diagnosis problem* (SD, X, OBS) *as of Def. 1, the diagnosis search space* Θ *is defined by the individual mode sets* M_i *for the components* $x_i \in X$ *as their cross product. A* diagnosis probability distribution $P(\Theta)$ *assigns each diagnosis candidate* $\Delta' \in \Theta$ *a probability* $0 \leq p(\Delta') \leq 1$ *s.t.* $\Sigma_{\Delta' \in \Theta} p(\Delta') = 1$*.*

▶ **Corollary 3.** *As per Def. 1, a diagnosis candidate* $\Delta' \in \Theta$ *from a probability distribution* $P(\Theta)$ *as of Def. 2 is a diagnosis iff its probability is higher than zero such that* $p(\Delta') > 0$*.*

Soundness and *completeness* are important properties of any diagnosis algorithm A, since they capture whether (1) the diagnoses reported by A are indeed diagnoses (in that they follow a formal definition like our Def. 1), and whether (2) the algorithm will return *all* diagnoses if it runs long enough. There's an abundance of reasons why algorithms could be incomplete or unsound. Obvious examples are approximating concepts like subsymbolic approaches, but there are also well-founded analytic concepts. For example, if a diagnosis algorithm is tasked to derive *one* minimum-cardinality diagnosis, we usually employ optimizations that result in an incomplete search but where we can still guarantee that at least one minimum-cardinality diagnosis survives and that A reports sound results [2, 6]. Another example would be that of computing all minimal conflicts between SD and OBS (see [4, 25] for the underlying theory), and where we subsequently employ an approximating algorithm like Staccato [1] for computing diagnoses as the minimal hitting sets of the conflicts.

Assessing traits like completeness and soundness should thus be an integral part of any approach that assesses the quality of a diagnosis algorithm and its results, especially if we do not have detailed knowledge of the algorithm (or its implementation) but have to evaluate it anyway. Having white-box access to an algorithm and its implementation provides us with a powerful and intuitive approach to do so, *i.e.*, by deriving formal proofs. In some contexts, such as competitions, we may have only limited knowledge about the diagnosis engine rather than detailed information about its implementation. But even if we would know the algorithm's details, we need to be aware that a proof would most likely target the *algorithm*'s computational concept and seldomly its *implementation*. In this paper, we are thus interested in assessing soundness and completeness in the context of a diagnosis algorithm's results, rather than proving these traits for the algorithm's computational concept.
In the context of $P(\Theta)$ and $Q(\Theta)$, intuitively, soundness refers to the question whether a diagnosis Δ reported in $P(\Theta)$ (s.t. $p(\Delta) > 0$) is also a diagnosis in $Q(\Theta)$. When evaluating completeness, we need to verify that all diagnoses in $Q(\Theta)$ are also diagnoses in $P(\Theta)$.

▶ **Definition 4** (soundness)**.** *Given the diagnosis search space* Θ*, a diagnosis* $\Delta \in \Theta$ *reported in* $P(\Theta)$ *is* sound*, iff it is indeed a diagnosis s.t.* Δ *is a diagnosis in* $Q(\Theta)$*. The results of a diagnosis algorithm are sound, iff all diagnoses* Δ_i *reported in* $P(\Theta)$ *are sound.*

▶ **Definition 5** (completeness). *Given the diagnosis search space Θ, the results of a diagnosis algorithm A are* complete*, iff there is no $\Delta_i \in \Theta$ that is a diagnosis in $Q(\Theta)$ but not a diagnosis in $P(\Theta)$.*

It is easy to see that the two definitions support any diagnosis search space, but they do not take into account how the algorithms encode their results. In particular, as we observed in Sec. 2.4, some algorithms mention some ambiguity groups *implicitly*. That is, when using a weak fault model, algorithms like [25, 4, 19] report only Δ_i for an ambiguity group defined by a subset-minimal diagnosis Δ_i and all its supersets. That is, since all of Δ_i's supersets would qualify as diagnosis according to our Def. 1 by default.
Continuing our discussion on how to generate $P(\Theta)$, we thus might have to enrich the reported set of diagnoses, like we suggested in Section 2.4. In the current case, we would derive all the missing probabilities for diagnoses that are entailed by the reported ones. All non-diagnoses are then finally assigned a probability of 0. Please note that other algorithms might call for a much more elaborate post-processing.

It is intuitive that Definitions 5 and 4 leave no room for deviations between $P(\Theta)$ and $Q(\Theta)$ when we look at the respective sets of diagnoses. When considering our discussion in Section 2.4, the choices we make for defining the gold standard $Q(\Theta)$ could, however, result in the situation that even the most precise algorithm could not achieve completeness and soundness. Potential delays in diagnosability could be one of these reasons, *i.e.*, when we consider the known ground truth about injected faults to define $Q(\Theta)$ and thus do not consider observability-related issues that would cause the faults to manifest on the observable signals (and thus in *OBS*) only after some delay (see Sec. 2.4).

To this end, let us introduce temporal and cardinality-oriented error bounds. Via those bounds, we can then define some maximum deviations in terms of temporal deviations or over- and under-approximations of a diagnosis. Within those bounds, we will consider the results to be *near-complete* and *near-sound*.

▶ **Definition 6** (near-completeness). *The results of a diagnosis algorithm A are* near-complete *with respect to bounds $\epsilon_C \in \mathbb{N}_0$ and $\epsilon_T \in \mathbb{N}_0$, iff for every diagnosis Δ_i in $Q(\Theta)$ for timestep t_j, there is some diagnosis Δ_k for timestep t_l in $P(\Theta)$ such that*

- *Δ_k is a sub- or superset of Δ_i,*
- *$||\Delta_i| - |\Delta_k|| \leq \epsilon_C$, and*
- *$|l - j| \leq \epsilon_T$.*

Near-completeness allows us to reason within a temporal scope for situations where we compute diagnoses for each individual time step t_i of a stateful system's computation (see our discussion in Sec. 2.4). For every diagnosis in $Q(\Theta)$ for time step t_i, we then search for matching diagnoses in $P(\Theta)$ for any time step $t_{j-\epsilon_T} \leq t_l \leq t_{j+\epsilon_T}$. This allows us to deal with delayed observability as well as with the potential clairvoyance (see Def. 14) of, *e.g.*, subsymbolic algorithms. If we do not want to consider temporal deviations, or for the usual case of diagnosing the entire computation after it finished, we just have to set ϵ_T to 0. Via parameter ϵ_C, we can control the acceptance of over- or under-approximations of diagnoses Δ_i from Q, such that P would contain only super- or subsets of Δ_i. ϵ_C allows us to define a limit on the difference in cardinality.

▶ **Corollary 7.** *The results of a diagnosis algorithm A are complete as of Def. 5 if and only if they are* near-complete *as of Def. 6 with respect to bounds $\epsilon_C = \epsilon_T = 0$.*

▶ **Definition 8** (strict near-completeness). *The results of a diagnosis algorithm A are strictly near-complete iff the results are incomplete, but where there are some bounds ϵ_C and ϵ_T for which they are near-complete.*

▶ **Corollary 9.** *Let the results of a diagnosis algorithm A be near-complete, and let $\epsilon_{C_{min}}$ be the minimum value for ϵ_C enabling near-completeness (ϵ_T being a free variable) and $\epsilon_{T_{min}}$ be the corresponding minimum value for ϵ_T (ϵ_C being a free variable). Then the results of A are* not *necessarily near-complete for bounds $\epsilon_{C_{min}}$ and $\epsilon_{T_{min}}$.*

Similar to considering completeness within temporal and cardinality-oriented bounds, let us consider also soundness within such well-defined bounds.

▶ **Definition 10** (near-soundness)**.** *A diagnosis Δ_k from $P(\Theta)$ for time step t_l is* near-sound *with respect to bounds $\epsilon_C \in \mathbb{N}_0$ and $\epsilon_T \in \mathbb{N}_0$ iff there is a diagnosis Δ_i for time-step t_j in $Q(\Theta)$ such that*

- *Δ_i is a sub-or superset of Δ_k,*
- *$||\Delta_i| - |\Delta_k|| \leq \epsilon_C$, and*
- *$|j - l| \leq \epsilon_T$.*

The results of an algorithm for time step t_l are near-sound, iff all of its diagnoses Δ_k from $P(\Theta)$ for time step t_l are near-sound.

▶ **Corollary 11.** *The results of a diagnosis algorithm A are sound as of Def. 4 if and only if all reported diagnoses are* near-sound *as of Def. 10 with respect to bounds $\epsilon_C = \epsilon_T = 0$.*

▶ **Definition 12** (strict near-soundness)**.** *The results of a diagnosis algorithm A are strictly near-sound iff there is at least one reported diagnosis Δ_k from $P(\Theta)$ for time-step t_l that is not sound as of Def. 4, and every such diagnosis is near-sound for some bounds ϵ_C and ϵ_T as of Def. 10.*

▶ **Corollary 13.** *Let the results of a diagnosis algoritm A be near-sound, and let $\epsilon_{C_{min}}$ be the minimum value for ϵ_C that enables near completeness (s.t. there is some ϵ_T) and $\epsilon_{T_{min}}$ be the corresponding minimum value for ϵ_T (s.t. there is some ϵ_C). Then the results of A are* not *necessarily near-sound for bounds $\epsilon_{C_{min}}$ and $\epsilon_{T_{min}}$.*

When we discussed near-completeness as of Def. 6, we briefly mentioned that an algorithm might be clairvoyant. Let us briefly formalize such clairvoyance and let us connect it to strictly near-sound algorithms.

▶ **Definition 14** (clairvoyance, clairvoyant diagnosis)**.** *A near-sound diagnosis Δ_k from $P(\Theta)$ reported by algorithm A for time step t_l is called a* clairvoyant diagnosis*, if and only if there is a diagnosis Δ_i in $Q(\Theta)$ for some time-step $t_{l<j\leq l+\epsilon_T}$ s.t. Δ_i is a sub- or superset of Δ_k with $||\Delta_i| - |\Delta_k|| < \epsilon_C$, but there is no such Δ_i for $t_{l-\epsilon_T \leq j \leq l}$.*

It is easy to see that a clairvoyant diagnosis means that the algorithm is by definition unsound, *i.e.*, since there was no matching diagnosis for the same time step.

▶ **Corollary 15.** *If the results of a diagnosis algorithm A contain a clairvoyant diagnosis Δ, then the algorithm's results are not sound. The results are strictly near-sound iff there exist appropriate bounds ϵ_T and ϵ_C such that the results are near-sound with respect to those bounds.*

It is evident that the notions of near-completeness and near-soundness allow us to classify and compare some algorithms' performance for a diagnosis problem via inspecting ϵ_T and ϵ_C. That is, via their respective minimum parameters ϵ_T and ϵ_C that are required for achieving near-soundness and/or near-completeness. The lower the minimum values for the parameters, the better the algorithm. Via ϵ_T and ϵ_C we can thus establish partial orders in the sense that

when we fix one of these parameters, the minimum values for the other parameter defines a natural order regarding their performance. For obtaining a total order, we would need a weighted metric considering $\epsilon_{T_{min}}$ *and* $\epsilon_{C_{min}}$, or we could take into account the entire set of all (minimal) combinations. For a representative verdict, some (weighted) average over a set of samples might need to be computed, implementing the discussion offered in Section 2.1.

Please note that the values for parameters ϵ_T and ϵ_C depend on each other, as can be seen from Corollaries 9 and 13. A consequence that we can easily observe is that we might need values for ϵ_T that are higher than $\epsilon_{T_{min}}$ for enabling near-completeness and/or near-soundness for some specific ϵ_C (and vice versa).

3.2 The challenge: of oracles and metrics

We motivated our work in the Introduction with the complexity of finding correct and informative answers to the natural question of whether *a specific diagnosis algorithm returned the correct result for some diagnosis problem, and if not, how good the results are.* As we hope to have convinced you with our discussion in Section 2, providing the right answers is a complex task that requires us to take into account a wide variety of aspects.

Let us now discuss potential solutions to the problem at hand. In principle, we can observe that any solution needs to tackle two distinct problems:

- Task 1: We need to implement an *oracle* that judges whether the results are OK or not.
- Task 2: We need to quantify the quality of the results with a corresponding measure.

T1 and T2 *can* be addressed and answered independently, but we can also exploit synergies and tackle both tasks together. So we could use a quantitative metric for T2 and implement the oracle O for T1 by a qualitative interpretation of the value obtained by the metric.

An intuitive and straightforward solution for the latter would be to consider $P(\Theta)$ and $Q(\Theta)$ and to sum up the absolute value of the differences between $p(\Delta_i)$ and $q(\Delta_i)$ when iterating through the diagnosis space Θ. We could adopt KL-convergence as of Eq. 5 in Section 2.1, we could sum up the squared absolute value such as to penalize larger differences, and we could adopt a large a variety of similar ideas with individual twists. That is, as long as we ensure that the final value is 0 iff there is no deviation at all, we have a solution with the desired property. Obviously, this concept is not very robust against the influence of noisy computations. If we add some error bound $\epsilon \in \mathbb{R}^+$ to mitigate some of the effects (see Eq. 8 for such an oracle O), we would still not be able to address the disadvantage that the metric (the sum in Eq. 8) does not distinguish between problems with different severity levels.

$$O(P, Q, \epsilon) = \top \textbf{ iff } \Sigma_{\Delta_i \in \Theta} |p(\Delta_i) - q(\Delta_i)| \leq \epsilon \textbf{ and } \bot \textbf{ otherwise} \tag{8}$$

Let us elucidate the issue by considering the two following cases. For Case 1, assume that there is a very small deviation $\delta_i = |p(\Delta_i) - q(\Delta_i)|$ in the probabilities for some $\Delta_i : (p(\Delta_i) > 0 \ \wedge q(\Delta_i) > 0$. Case 1 refers thus to a scenario in which Δ_i is a diagnosis in both $P(\Theta)$ *and* $Q(\Theta)$, but in which we observe a slight deviation between $p(\Delta_i)$ and $q(\Delta_i)$. In the context of a repair procedure, this difference might cause a change in the rank of Δ_i, which could in turn affect repair cost estimates (see Section 2.2). But we can easily see that A neither missed to report diagnosis Δ_i in $Q(\Theta)$, nor is Δ_i a spurious diagnosis. Mathematically, we thus observe that the completeness and soundness of the results is not negatively affected by the difference in probabilities. Now let us look at Case 2, in which the probability for the same Δ_i is 0 for either $P(\Theta)$ *or* $Q(\Theta)$ and is δ_i for the other. In Case 2, while the difference is indeed the same as in Case 1, we can easily observe that the results $P(\Theta)$ are either incomplete or unsound, *i.e.*, since A either failed to report a diagnosis (if

$p(\Delta_i) = 0$), or it reported a spurious and thus unsound diagnosis (if $q(\Delta_i = 0)$). The metric can, however, not distinguish Case 2 from Case 1 so that Δ_i contributes in either case with the same δ_i to the computation.

An intuitive way to address the issue would be to employ a combination of such a simple quantitative metric and a formal assessment of completeness and soundness. Following up on our discussions in Sections 2.4 and 3.1, this might also call for inspecting near-soundness and near-completeness as of Definitions 10 and 6 instead, since this would allow us, *e.g.*, to accommodate delays in the diagnosability of some faults.

In combination with this formal assessment, we could replace the metric in Eq. 8 with a much simpler one. First, we rank the diagnoses for $P(\Theta)$ and $Q(\Theta)$ respectively, and then subsequently count the Δ_is for which there is a difference in ranking (taking into account ambiguity groups). This simplification would make the quantitative measure more robust to noise and slight deviations in the probabilities that do not change the ranking, while it already takes the effects on a repair process into account. There is an abundance of similar metrics, and in Section 4 we discuss with the economic metric a much more elaborate approach to quantify effects on the repair costs, and thus a metric that is closer in line with our repair-cost estimate discussion from Section 2.2.

It is easy to see that the combination of a formal assessment with some quantitative ranking could be realized in a *combined* metric that allows us to tackle T1 and T2 together, but also in individual solutions for T1 and T2. We could use the assessment of soundness and completeness for addressing T1, and employ a quantitative metric for T2. Approaches for T1 following the concept described above require us to know $Q(\Theta)$. In reality, this assumption is indeed a strong one and we may not be able to fulfill it in practice. Let us thus explore whether we might be able to assess completeness and soundness without knowing $Q(\Theta)$.

So, we can verify the soundness of a single diagnosis Δ_k, for example, by checking if the symptoms disappear after repairing all (necessary) x_i as suggested by Δ_k. Connecting to our discussions around PFU in Sec. 2.2), such a Δ_k is indeed sound. Furthermore, considering Def. 1, we do not need to verify whether Δ is subset-minimal, since it is not required by our definition. For these two reasons, in general, Δ_k should also be part of $Q(\Theta)$). That is, unless we break the hidden assumption that we consider the ambiguity groups of diagnoses and their supersets in a sensible way such that $Q(\Theta)$ would miss to contain also the relevant supersets of some diagnosis as diagnoses.
Please note that if we would desire subset-minimality in the diagnoses for a weak fault model computation, it would not suffice to simply check whether $P(\Theta)$ contains another diagnosis $\Delta_l : \Delta_l \subset \Delta_k$ – like some diagnosis algorithms do. That is, this check works out for algorithms like RC-Tree [18] since they are sound *and* complete. But while completeness is ensured in those algorithm's construction, it is not necessarily the case in our context.

Verifying completeness is also sometimes possible without knowing the expected results. For example, in cases where we have access to a precise system model that is suitable for working with a SAT, SMT or constraint solver. Assuming that we have, *e.g.*, a behavioral model in some temporal language [18, 9] or a simple combinational circuit [4], we first compile *SD* and *OBS* into a SAT problem as suggested in those papers (see also [13]), or into an SMT problem/constraint problem [16]. We then add blocking clauses for all diagnoses from $P(\Theta)$ and finally ask the solver to search *exhaustively* for a new diagnosis [13, 15]. If one is found, the results $P(\Theta)$ are incomplete, and otherwise they are complete.

We hope that we could convince you that there are many ways to tackle the problem at hand, despite all the challenges that we have to face. To illustrate this, we, indeed, discussed several options to implement the desired oracle (T1), as well as to quantify the quality of the results reported by a diagnosis algorithm (T2).

4 The Economic Metric

The intuition behind economic metrics for diagnostic algorithms is to evaluate their results in the context of their use. Here we will focus on the task of a repair person who's task it is to return the system to its full functioning. (Many other analogous diagnostic metrics are possible.) If diagnosis is completely accurate the diagnostician can simply replace the components listed in the diagnosis. But if the diagnosis is not accurate the diagnostician may replace components which are unfaulted, and in addition incur extra diagnostic expense to restore the system to complete functioning. Thus they incur two types of wasted effort. The report of DXC 2010 [8] outlined one approach which we first summarize and then augment according to the ideas outlined earlier in this paper.

The diagnostic algorithm returns a set of diagnoses: $\Omega = \{\omega_1, \ldots, \omega_k\}$. In addition it provides a weight $V(\omega)$ for each diagnosis. The weights typically represent probabilities. We assume

$$\sum_{\omega \in \Omega} V(\omega) = 1$$

The normalized utility including both types of wasted effort is (see [8] for details):

$$m_{utl}(\omega, \omega^*) = 1 - \frac{n(N+1)}{f(n+1)} - \frac{\bar{n}(\bar{N}+1)}{f(\bar{n}+1)}$$

where f is the count of all components, ω is a diagnosis, N is the number of healthy components, n is the number of false negatives ($|\omega^* - \omega|$), ω^* is the inserted fault, $\bar{n}$ is the number of false positives ($|\omega - \omega^*|$). From this definition we can see that $f = N + \bar{N}$ and $n + \bar{n} = |\omega^* \triangle \omega|$ where $\triangle$ indicates symmetric difference. m_{utl} intuitively has the desired properties. If there are no false positives or false negatives, $m_{utl} = 1$. A system with 10 components for which the diagnoser finds the incorrect single fault, $m_{utl} = 0.4$. If the diagnoser returns multiple diagnoses, the utility for the diagnoser on that task is:

$$M_{utl}(\omega^*) = \sum_{\omega \in \Omega} V(\omega) m_{utl}(\omega^*, \omega)$$

This was the exact metric used in prior DXC competitions. However, it does not accommodate ambiguity groups. Therefore, to obtain a decent score at all, the diagnosers in prior competitions needed to return as many elements of the ambiguity group each with its own $V(\omega)$. The final scores were thus much lower than expected. This was an unfortunate state of affairs as at the point this was discovered the metrics had been fixed for the competitors.

The full discussion of the utility metric in the presence of ambiguity groups is outside the scope of this paper. Generalizing the approach we used for the sampling approach we obtain:

$$S(A) = \frac{\sum_{s \in allsamples} p(G(s)) M_{utl}(s)}{\sum_{s \in allsamples} p(G(s))} \tag{9}$$

5 Conclusions

We hope that we have convinced you that choosing a metric for the evaluation of a diagnostic algorithm is very complex. In addition, we argue that the best metrics for evaluating the algorithms' performance could be those that are based on the costs incurred by the diagnostic algorithm in its context of use, like the economic metric. In particular, we observe that this

family of metrics provides us with an assessment that is agnostic to the algorithm itself and connects directly to how we perceive its performance by considering the practical effects of the derived results.

A complementing way to assess the results of a diagnostic algorithm is to look at their completeness and soundness. In practice, we would want to consider some error bounds in this context, as suggested by our notions of near-completeness and near-soundness. While we discussed the intuitiveness of computations that take advantage of knowing the gold standard, in specific circumstances, we can implement approaches that are able to circumvent the strong requirement of having to know the ideal results.

Following up on our discussions of issues that have to be addressed and our suggestions for concepts to solve the problems, future work will have to empirically evaluate available and new metrics to provide us with a clear picture of the algorithmic landscape.

References

1 R. Abreu and A. J. C. 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*, 2009. URL: `http://www.aaai.org/ocs/index.php/SARA/SARA09/paper/view/834`.

2 J. Biteus, M. Nyberg, and E. Frisk. An algorithm for computing the diagnoses with minimal cardinality in a distributed system. *Engineering Applications of Artificial Intelligence*, 21(2): 269–276, 2008. `doi:10.1016/j.engappai.2007.03.006`.

3 A. Boussif and M. Ghazel. Diagnosability analysis of input/output discrete-event systems using model-checking. *IFAC-PapersOnLine*, 48(7):71–78, 2015. `doi:10.1016/j.ifacol.2015.06.475`.

4 J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987. `doi:10.1016/0004-3702(87)90063-4`.

5 J. de Kleer and B. C. Williams. Diagnosis with Behavioral Modes. In *11th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1324–1330, 1989.

6 Johan de Kleer. Hitting set algorithms for model-based diagnosis. In *22nd International Workshop on Principles of Diagnosis (DX'11)*, 2011.

7 P. Delgado-Pérez and F. Chicano. An experimental and practical study on the equivalent mutant connection: An evolutionary approach. *Information and Software Technology*, 124:106317, 2020. `doi:10.1016/j.infsof.2020.106317`.

8 A. Feldman, T. Kurtoglu, S. Narasimhan, S. Poll, D. Garcia, J. de Kleer, L. Kuhn, and A. van Gemund. Empirical evaluation of diagnostic algorithm performance using a generic framework. *International Journal of Prognostics and Health Management*, pages 1–28, 2010.

9 A. Feldman, I. Pill, F. Wotawa, I. Matei, and J. de Kleer. Efficient model-based diagnosis of sequential circuits. In *34th AAAI Conference on Artificial Intelligence (AAAI'20)*, pages 2814–2821. AAAI Press, 2020. `doi:10.1609/AAAI.V34I03.5670`.

10 R. Greiner, B. A. Smith, and R. W. Wilkerson. A Correction to the Algorithm in Reiter's Theory of Diagnosis. *Artificial Intelligence*, 41(1):79–88, 1989. `doi:10.1016/0004-3702(89)90079-9`.

11 A. Hou. A theory of measurement in diagnosis from first principles. *Artificial Intelligence*, 65(2):281–328, February 1994. `doi:10.1016/0004-3702(94)90019-1`.

12 I. Matei, M. Zhenirovskyy, J. de Kleer, and A. Feldman. Classification-based Diagnosis Using Synthetic Data from Uncertain Models. *Annual Conference of the PHM Society*, 10(1), 2018.

13 A. Metodi, R. Stern, M. Kalech, and M. Codish. Compiling Model-Based Diagnosis to Boolean Satisfaction. In *26th AAAI Conference on Artificial Intelligence*, pages 793–799, 2012.

14 E. Muškardin, I. Pill, and F. Wotawa. CatIO - A Framework for Model-Based Diagnosis of Cyber-Physical Systems. In D. Helic, G. Leitner, M. Stettinger, A. Felfernig, and Z. W. Raś, editors, *Foundations of Intelligent Systems*, pages 267–276, 2020.

15 I. Nica, I. Pill, T. Quaritsch, and F. Wotawa. The Route to Success - A Performance Comparison of Diagnosis Algorithms. In *23rd International Joint Conference on Artificial Intelligence*, pages 1039–1045, 2013.

16 I.-D. Nica and F. Wotawa. ConDiag - Computing minimal diagnoses using a constraint solver. In *23rd International Workshop on Principles of Diagnosis*, 2012.

17 I. Pill and J. de Kleer. Challenges for Model-Based Diagnosis. In *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASIcs)*, pages 6:1–6:20, 2024. doi:10.4230/OASIcs.DX.2024.6.

18 I. Pill and T. Quaritsch. Behavioral diagnosis of LTL specifications at operator level. In *23rd Int. Joint Conf. on Artificial Intelligence*, pages 1053–1059, 2013.

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

20 I. Pill, I. Rubil, F. Wotawa, and M. Nica. SIMULTATE: A Toolset for Fault Injection and Mutation Testing of Simulink Models. In *IEEE 9th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 168–173, 2016.

21 I. Pill and F. Wotawa. Spectrum-Based Fault Localization for Logic-Based Reasoning . In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 192–199, 2018. doi:10.1109/ISSREW.2018.00006.

22 I. Pill and F. Wotawa. On Using an I/O Model for Creating an Abductive Diagnosis Model via Combinatorial Exploration, Fault Injection, and Simulation. In *29th International Workshop on Principles of Diagnosis (DX'18)*, 2018.

23 I. Pill and F. Wotawa. Exploiting observations from combinatorial testing for diagnostic reasoning. In *30th Int. Workshop on Principles of Diagnosis*, 2019.

24 I. Pill and F. Wotawa. Computing Multi-Scenario Diagnoses. In *31st International Workshop on Principles of Diagnosis, DX ; Conference date: 26-09-2020*, 2020. URL: http://dx-2020.org/.

25 R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.

26 L. Ye, P. Dague, D. Longuet, L. B. Briones, and A. Madalinski. Fault manifestability verification for discrete event systems. In *22nd European Conf. on Artificial Intelligence*, pages 1718–1719, 2016. doi:10.3233/978-1-61499-672-9-1718.