Planning to discriminate diagnoses

Jérôme Lang
IRIT - Université Paul Sabatier - 31062 Toulouse Cedex (France)
e-mail: lang@irit.fr

Abstract

We propose a logical formalisation of tests for hypothesis discrimination. A discrimination problem is defined as a triple consisting of a knowledge base, a test base and a set of hypotheses. We characterize discrimination problems in abductive terms and we propose an algorithm for building conditional test plans, assuming that the set of available tests is fixed. We focus on the application of our framework to fault isolation in consistency-based diagnosis.

Introduction

The diagnosis community has paid much attention to tests in order to discriminate among competing hypotheses (each of which represents a plausible diagnosis). Practically, the possible outcomes of a test are possible values of a variable which can be observed through a measurement. As to specific applications, test generation for diagnosis has been studied for a long time, especially testing digital circuits. In a more general setting (and probabilistic), (de Kleer & Williams 1987) give a approach to selecting the *next* test to perform (*one-step lookahead*). Their purpose being to isolate quickly (in terms of number of tests) the actual fault configuration, they choose the next test according to the expected gain in information it promises (by entropy minimization). Some approaches to diagnosis and repair consider a phase where it is chosen whether it is preferable to test or to repair, taking account of the probabilities of competing diagnoses and the costs of test and repair actions, e.g. (Friedrich & Nejdl 1992), (Sun & Weld 1992).

These approaches consider the choice of the next test as a part of the diagnosis process. A few other ones consider test generation *for itself* and in a more general setting: (McIlraith & Reiter 1992) and (McIlraith 1994) propose a purely logical approach to test generation. Their goal is to generate tests in order to discriminate a set of hypotheses. Tests are classified in function of their qualitative potentiality to confirm or discard one or several hypotheses. Then, the search for "good" tests is characterized in abductive terms and consequently, test generation can be automated using results and algorithms for the computation of abductive explanations. In (Struss 1994a) and (Struss 1994b), formal approaches of model-based testing are proposed; Struss defines a concept of discrimination (a set of test inputs discriminates a set of possible behaviours if any of its possible results entails one behaviour) and he characterizes discriminating inputs, being those which uncover the distinctions between any two possible behaviours. He also gives a probabilistic modelling of test selection which generalizes GDE's selection strategy.

What we propose in this paper is first a logical modelling of discrimination problems and then an algorithm for constructing *conditional test plans* (or test policies, by analogy with decision processes). Contrarily to one-step lookahead strategies, a test policy specifies an entire test sequence (conditioned by tests results) and thus allows to compile a declarative, logical model of a physical system into a simple decision tree which can be placed in the conventional control system of some real time system.

For this purpose, we will first establish a link between tests and prime implicants, in the same spirit as (McIlraith 1994) but under a different perspective: McIlraith is interested in *test generation* rather than in test planning, and studies the properties of tests considered separately (typically, "search for a test whose at least one / all possible result(s) confirm/refute an hypothesis"), in order to find *the next test* to perform, rather than on the set of available tests *considered in its globality*. In this article, we are interested in properties of *test bases*: we assume that we know exhaustively all available tests and we are interested in questions such as "find a subset of the test base whose different possible results will enable us to find out which hypothesis is true", etc.

A logical framework for hypothesis discrimination

We propose here a rather general logical framework for hypothesis discrimination, where the possible results of a test are propositional formulas. Although we devote this paper to hypothesis discrimination in model-based diagnosis, there are other applications, for instance decision under incomplete knowledge or source verification.

Definition 1 (tests, discrimination problems)

- *A **test** t_i is defined by finite set of propositional formulas $\{r_i^j, j = 1..m\}$ (with $m \geq 2$), called **possible results** (or outcomes). We use the notations $t_i = Test(r_i^1, \ldots, r_i^m)$ and $Res(t) = \{r_i^1, \ldots, r_i^n\}$.*
- *A **test base** is a finite set of tests: $TB = \{t_1, \ldots t_n\}$.*
- *A **discrimination problem** consists in a propositional knowledge base Σ, a test base TB and a set of **hypotheses** $HYP = \{H_1, \ldots H_p\}$ where each H_i is a propositional formula.*

Tests are considered *reliable* (their result is always true in the actual world), *non-intrusive* (their execution never changes the world), *idempotent* (the repeated application of a test in the same world always gives the same result) and *well-defined*: for each test t_i we have $\Sigma \models r_i^1 \vee r_i^2 \ldots \vee r_i^m$. Under the reliability assumption, well-definedness guarantees that whatever the actual world in which the test is performed, *there will be an outcome.* Other specificities of test bases are considered in (Lang 1996).

This model recovers *truth tests* (McIlraith 1994) of the form $Test(\varphi, \neg\varphi)$ and *value tests* (McIlraith 1994) of the form $Test(X = x_1, \ldots X = x_n)$.

The goal of the agent is to *find out a hypothesis H_k which is true in the real world.* We assume that the hypothesis set is exhaustive: $\models_\Sigma \vee_{H_i \in HYP} H_i$. If it is not then we add an extra hypothesis $H^* = \neg(\wedge_k H_k)$, and that *hypotheses are mutually exclusive.* This means that exactly one of the H_i's is true, the problem being to find out which one. In model-based diagnosis, hypotheses correspond to minimal diagnoses; they are generally exclusive.

Definition 2 (test branches)
*A **test branch** $\tau = \{t_i : r_i^j | t_i \in T \subseteq TB\}$ is a set of test results containing at most one result of each test t_i of TB. The **support** of τ is $\{t_i | t_i : r_i^j \in \tau\}$.*
*A test branch τ is **complete** if its support is TB and **consistent** iff it is consistent with Σ. A consistent test branch τ **proves** H_k (denoted $\tau \models_\Sigma H_k$) iff $\wedge_{t_i : r_i^j \in \tau} r_i^j \models_\Sigma H_k$, and **proves minimally** H_k iff it proves H_k and no proper subbranch of τ proves H_k*

A test branch corresponds to what is obtained after the execution of some of the tests (all of them if the branch is complete). For instance, if $t_1 = Test(a, \neg a)$, $t_2 = Test(a \wedge b, \neg b, \top)$, $t_3 = (a \rightarrow c, a \rightarrow \neg c)$, then $\tau = (t_1 : a, t_3 : a \rightarrow \neg c)$ is a test branch whose support is $\{t_1, t_3\}$ – meaning that t_2 has not been executed yet.

Definition 3 (discriminability)
*HYP is **discriminable** with respect to Σ and TB iff for any complete consistent test branch τ, there exists a $H_i \in HYP$ such that $\tau \models_\Sigma H_i$*

Clearly, discriminability means that if we execute all tests then it is guaranteed that we will find out a true hypothesis. It can be proved that hypothesis discriminability is coNP-complete (proof omitted).

Definition 4 *t_i is a **relevant test** for discriminating HYP iff there is a minimal consistent test branch proving some H_i needing t_i, i.e. containing a $(t_i\ r_i^j)$. t_i is a **necessary test** for discriminating HYP iff for every H_i, every minimal consistent test branch proving H_i needs t_i.*

Note that the definitions of relevant and necessary tests are respectively very weak and very strong. It will not be surprising that on a given problem, initially all tests are potentially relevant and none is necessary. However, this becomes less and less true when more and more tests have been executed – first, it may often happen that a given result of a test t makes some other test t' irrelevant; second, when a number of tests have been executed and thus retracted from the test base, the latter shrinks and thus it is more likely that some of the remaining tests become necessary.

Computing test plans

We first give a abductive characterization discriminable problems, namely by means of ATMS labels. The link between tests and ATMS is not new, since (McIlraith & Reiter 1992) and (McIlraith 1994) already characterize conforming and discriminating tests in abductive terms, and formulate thus the test generation problem as an abduction problem. We associate ATMS assumptions to each elementary result of a test, and then associate to each hypothesis its label. We recall that an ATMS label is the set of all minimal reasons to believe in a formula, where a reason is a set of assumptions (environment). Here assumptions are test results and thus each environment in the label of H_k correspondds to a Σ-consistent minimal test branch proving H_k.

Definition 5 *Let $\langle \Sigma, TB, HYP \rangle$ be a discrimination problem. To each test result r_i^j we associate an ATMS assumption $Res(t_i) = r_i^j$, or for short R_i^j. The new* *knowledge base is defined by*

$$\begin{aligned}\Sigma^* = \quad & \Sigma \cup \{R_i^j \rightarrow r_i^j | i = 1 \ldots |TB|, j = 1 \ldots n_i\} \\ & \cup \{\neg(R_i^j \wedge R_i^k) | i = 1 \ldots |TB|, j \neq k\} \\ & \cup \{\vee_j R_i^j | i = 1 \ldots |TB|\}\end{aligned}$$

When tests have mutually exclusive results, it can be easily checked this can be equivalently rewritten to $\Sigma^* = \Sigma \cup \{r_i^j \rightarrow R_i^j | i, j\}$. If tests are furthermore binary (2 possible results), this is even simpler (only $2.|TB|$) assumptions are needed).

Clearly, a consistent environment contains *at most* one assumption R_i^j for each i and corresponds to a Σ-consistent test branch. The set of all minimal consistent environments of H_i is its *label*, denoted by $Label(H_k)$. We can use an ATMS for computing the label of each hypothesis H_k; these labels (together with the nogood base) will be useful for characterizing relevant and necessary tests, discriminable problems, and then to compute a discriminating test plan.

This abductive characterization will be useful for building test plans; first, it enables us characterizing relevant and necessary tests:

Proposition 1
• *t_i is a relevant test iff $\exists j, \exists k$ such that $R_i^j \in Label(H_k)$.*
• *t_i is a necessary test iff $\forall k, \forall E \in Label(H_k), \exists j$ such that $R_i^j \in E$.*

Definition 6 (conditional test plans)
*Let $\langle \Sigma, TB, HYP \rangle$ be a discrimination problem. A **conditional test plan** π (or test policy) is a finite tree whose nodes are labelled either with a test t_i, or with a hypothesis H_k, or with "failure"; if node n is labelled with t_i then it has one child for each possible result of t_i; nodes labelled with a hypothesis or with "failure" are terminal nodes. For each node n, let τ_n be the test branch associated with n in the obvious way.*

Definition 7 (solution plans) *A test plan π is a solution of a discrimination problem $\langle \Sigma, TB, HYP \rangle$ iff it does not contain any failure node and it is* sound*: for each terminal node n labelled with H_k then $\tau_n \models_\Sigma H_k$.*

When the problem is not fully discriminable (i.e., the hypothesis set can not always be discriminated), it can still be useful to define a test plan, which will discriminate HYP each time it is possible. When the execution of this test plan fails, it is guaranteed that no other policy would have been discriminating. This will be called a *maximal sub-solution plans.*

We propose the following recursive algorithm for computing solution plans or maximal sub-solution plans. It contains a compilation phase where all labels of hypotheses are computed, as well as the nogood base (using an ATMS). This list is called `Labels`. Once this list is built, irredundant tests can be eliminated from TB before calling the function `Test-plan`.

The function `Test-plan` calls a function `Update` which updates the $Label(H_k)$'s as well as the nogood base by the test result R_i^j newly obtained:

- when an environment of $Label(H_k)$ contains R_i^j, then the latter is deleted from this environment;
- when an environment of $Label(H_k)$ contains R_i^q with $q \neq j$, the whole environment is deleted.

After these deletions have been made, the updated labels may not be minimal nor consistent any longer; they should then be minimized, and made consistent by deleting all environments containing a nogood.

Function `Test-plan` (T, HYP,`Labels`):Tree;

If $Label(H_k) = \{\emptyset\}$ for some k [2]
then Return(H_k)
else
  Eliminate irrelevant tests from T; [3]
  If $T = \emptyset$ [4]
  then Return "failure"
  else
   Pick a new test t_i in T; [5]
   Return a tree with root labelled t_i and for each consistent [6] result R_i^j, the subtree
   `Test-plan` ($T \setminus \{t_i\}$, HYP, `Update`(`Labels`,R_i^j))
  end {if then else}
end {if then else}

end

Some steps of the algorithm deserve more comments. (The reader may have a look on the detailed example in the last Section). At [1], the computation of the labels and of the nogood base are done once for all (they will only be updated); this list makes Σ ne no longer needed. At [2], the presence of the empty environment in `Label`(H_k) means that the current branch (consisting in the results of all previously executed tests) proves H_k. Step [3] is done by using the `Label`(H_k) and Proposition 2; note that newly irrelevant tests may appear after a test result is taken into account. At [4], all relevant tests have been performed without proving any hypothesis, hence the failure.

Step [5] should use a heuristics ranking tests according to their expected ability to discriminate further. At this point, a minimum entropy criterion (de Kleer & Williams 1987) (assuming that a probability distribution is given) could be useful. In any case, *necessary tests should be selected in the first place.*

At step [6], a subtree is created for each possible result R_i^j of t_i, *except if this result is known to be impossible given the previous test results*, which means that the updated nogood base contains $\{R_i^j\}$.

Proposition 2
The algorithm returns a solution plan iff $\langle \Sigma, TB, HYP\rangle$ is discriminable. Otherwise, it returns a sound and maximal sub-solution plan.

The proof is based on the following invariant, which is proved by upwards induction on the tree: at each node n, corresponding to test branch τ_n, and T_n being the set of tests which have not been executed yet, then the policy assigned to n is a sound and maximal test policy for $\langle \Sigma \wedge \tau, T, HYP\rangle$. Irrelevant tests do not play any role in this invariant (they are just added in the algorithm to speed it up). Applying the invariant to the root proves the correction of the algorithm.

Application to fault isolation in consistency-based diagnosis

Model-based diagnosis mainly focuses on the characterization and the computation of preferred (i.e., most plausible) diagnoses . Some authors propose one-step-lookahead principles (de Kleer et al. 1992) in order to select the *next test* (or measurement) to perform, generally according to a probabilistic criterion (de Kleer & Williams 1987). This can be roughly sketched by the following general algorithm:

While there are several competing diagnoses **do**
choose next test t_i;
execute t_i; {result $= r_i^j$}
$OBS := OBS \cup \{r_i^j\}$
end

What we propose here is the explicit construction of a *test plan* from a purely qualitative (logical) description of the system. We aim at characterizing off-line (before execution) tests bases which enable us isolating the actual system state[1]. We choose the consistency-based diagnosis setting (Reiter 1987) but however things would be similar in the abductive diagnosis setting (Poole 1992).

Definition 8 (minimal diagnoses) *(Reiter 1987) Let $(SD, COMPS, OBS)$ be a diagnosis problem. A minimal diagnosis is a minimal subset Δ of $COMPS$ such that $SD \cup OBS \cup Ab(\Delta)$ is consistent, where $AB(\Delta) = \{Ab\_c \mid c \in \Delta\} \cup \{\neg Ab\_c \mid c \in \bar{\Delta}\}$.*

[1](de Kleer et al. 1992) show that one-step lookahead is sufficient in practice because it generally gives nearly optimal solutions. We however think that in some contexts it is worth building full test plans, for instance, if the number of available tests is not prohibitive, or else, if the time sent off-line computing the plan is much less expensive than the time spent on-line looking for the next best measurement.

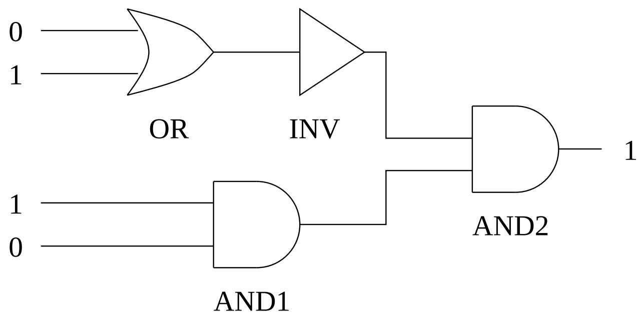

Figure 1: a circuit

Now, the above formalism can be exploited in order to characterize systems where faults can be isolated and to construct test plans for fault isolation.

Definition 9 (fault isolation)
*Let $(SD, COMPS, OBS)$ be a diagnosis problem, and $TB = \{t_1, \ldots t_n\}$ a test base over some of the variables of the system (set of available measures). The fault isolation problem $(SD, COMPS, OBS, TB)$ is the discrimination problem defined by $\Sigma = SD \cup OBS$, TB, and $HYP = \{AB(\Delta) \mid \Delta$ minimal diagnosis for $(SD, COMPS, OBS)\}$. We say that $(SD, COMPS, OBS, TB)$ satisfies **isolability** iff $\langle \Sigma, TB, HYP\rangle$ is discriminable.*

From their construction, hypotheses are mutually exclusive. Furthermore the hypothesis set is exhaustive because it is assumed that the actual diagnosis is among the minimal diagnoses (which is a classical assumption). Generally, tests of TB concern any of the variables of the system. A particular case is when *tests are performed directly on components*:
$COMPS = COMP_T \cup \overline{COMP_T}$ where $c \in COMPS_T$ means that c is directly testable;
$TB = \{test(c) \mid c \in COMP_T\}$ where $test(c) = Test(Ab\_c, \neg Ab\_c)$.

The following result tells that if all components except one are directly testable then faults can be isolated. The proof is easy although not trivial.

Proposition 3 *Assume that tests are performed directly on components. If $|COMPS_T| \geq |COMPS| - 1$ then $(SD, COMPS, OBS, TB)$ is isolable.*

Example

We consider the following circuit (depicted on Figure 1), composed of two AND gates, one OR gate and an inverter.

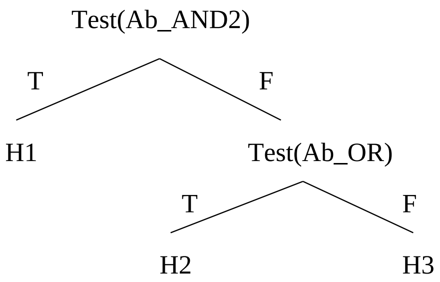

Figure 2: A test plan

It can be checked that the minimal diagnoses are $H_1 = Ab\_AND2$, $H_2 = Ab\_AND1 \wedge Ab\_OR$, $H_3 = Ab\_AND1 \wedge Ab\_INV$. Assume that all components are testable. The labels associated to the hypotheses, and the nogood base, are initially

$$Label(H_1) = \{ \{Ab\_AND2\}, \{\neg Ab\_AND1\}, \{\neg Ab\_OR), \neg Ab\_INV)\}\}$$ [2]

$$Label(H_2) = \{\{Ab\_OR\}, \{\neg Ab\_AND2, \neg Ab\_INV\}\}$$

$$Label(H_3) = \{\{Ab\_INV\}, \{\neg Ab\_AND2, \neg Ab\_OR\}\}$$

$$Nogoods = \{ \{\neg Ab\_AND1, \neg Ab\_AND2\}, \{\neg Ab\_OR, \neg Ab\_INV, \neg Ab\_AND2\}\}$$

For instance, H_1 is proved if $AND2$ is shown to be faulty, or if $AND1$ is shown to be not faulty: in the latter case, both diagnoses H_2 and H_3 are discarded since they include Ab_AND1, hence the only remaining (minimal) diagnosis is H_1.

Now let us perform the policy construction algorithm. Initially, there is no necessary or irrelevant test. We start for instance by testing $AND2$.

If the result is Ab_AND2: updating $Label(H_1)$ by Ab_AND2 gives $NewLabel(H_1) = \{\emptyset\}$, which enables us to conclude that H_1 is the actual diagnosis.

If the result is $\neg Ab\_AND2$, updating the labels and the nogood base give

$$Nogoods = \{\{\neg Ab\_AND1\}, \{\neg Ab\_OR, \neg Ab\_INV\}$$

$$NewLabel(H_1) = \{ \{\neg Ab\_AND1\}, \{\neg Ab\_OR, \neg Ab\_INV\}\}_{min} = \emptyset$$

(because all environments supporting H_1 contain a nogood – hence H_1 is now discarded);

$$NewLabel(H_2) = \{\{Ab\_OR\}, \{\neg Ab\_INV\}\}$$

$$NewLabel(H_3) = \{\{Ab\_INV\}, \{\neg Ab\_OR\}\}$$

Now, it is enough to test either Ab_OR or Ab_INV to conclude; hence, the problem is discriminable. Note that if only Ab_AND1 and Ab_AND2 were testable, discriminability would be lost. The test tree associated with the above policy (where we choose to test Ab_OR at the last step) is depicted on Figure 2.

[2] More formally, we should write

$$Label(H_1) = \{ \{Test(Ab\_AND2) : T\}, \{Test(Ab\_AND1) : F\}, \{Test(Ab\_OR) : F, Test(Ab\_INV) : F\}\}.$$

Concluding remarks

The original contributions of this paper are a logical formalisation of the test phase in model-based diagnosis, a characterization of dicriminable problems with ATMS and an algorithm for the construction of test policies. Among the perspectives for the continuation of this work, we first think of experimenting our algorithm for building test plans and compare the obtained efficiency with one-step lookahead strategies. We also think of enriching our model towards decision-theoretic testing, by considering probabilities and utilities (test costs and "informative" utilities measuring how the agent is happy with a given informational state); this could then be expressed in the framework of partially observable Markov Decision Processes.

References

G. Friedrich and W. Nejdl. 1992. Choosing observations and actions in model-based diagnosis-repair systems. *Proceedings of KR-92*, 489-498.

J. de Kleer and B. C. Williams. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32, 97-130.

J. de Kleer, O. Raiman and M. Shirley, 1992. One-step lookahead is pretty good. *Readings in model-based diagnosis* (W. Hamscher, L. Console, J. de Kleer, eds.), Morgan Kaufman, 138-142.

J. Lang. 1996. A propos des tests : une ébauche de classification et une formalisation logique. *Proceedings of RFIA-96* (in French), 347-356.

S. McIlraith. 1994. Generating tests using abduction. *Proceedings of KR-94*, 449-460.

S. McIlraith and R. Reiter. 1992. On tests for hypothetical reasoning. *Readings in model-based diagnosis* (W. Hamscher, L. Console, J. de Kleer, eds.), Morgan Kaufman, 89-96.

A. Meerwijk and C. Priest. 1992. Using multiple tests for model-based diagnosis. *Proc. of DX-92*, 30-39.

D. Poole. 1992. Normality and faults in logic-based diagnosis. *Readings in model-based diagnosis* (W. Hamscher, L. Console, J. de Kleer, eds.), Morgan Kaufman, 71-77.

R. Reiter, 1987. A theory of diagnosis from first principles. *Artificial Intelligence*, 32 (1), 57-96.

P. Struss. 1994. Testing physical systems. *Proceedings of AAAI'94*, 251-256.

P. Struss. 1994. Testing for the discrimination of diagnoses. *Proceedings of DX-94*, 312-320.

Y. Sun and D. Weld. 1992. Beyond simple observation: Planning to diagnose. *Proc. of DX-92*, 67–75.