Hierarchal Model-Based Diagnosis

Igor Mozetic

MLI 89-1

HIERARCHICAL MODEL-BASED DIAGNOSIS

Igor Mozetic

Department of Computer Science and
Center for Artificial Intelligence
George Mason University
4400 University Drive
Fairfax, VA 22030

MLI 89-1
TR 3-89

April 1989

HIERARCHICAL MODEL-BASED DIAGNOSIS

Abstract

Model-based reasoning about a system requires an explicit representation of the system's components and their connections. Diagnosing such a system consists of locating those components whose abnormal behavior accounts for the faulty system behavior. In order to increase the efficiency of model-based diagnosis, we propose a model representation at several levels of detail, and define three refinement (abstraction) principles. We specify formal conditions that have to be satisfied by the hierarchical representation, and emphasize that the multi-level scheme is independent of any particular single-level model representation. The hierarchical diagnostic algorithm which we define turns out to be very general. We show that it emulates the bisection method, and can be used for hierarchical constraint satisfaction.

We apply the hierarchical modeling principle and diagnostic algorithm to a medium-scale medical problem. The performance of a four-level qualitative model of the heart is compared to other representations in terms of diagnostic efficiency and space requirements. Despite the fact that the hierarchical model does not reach the time/space performance of dedicated diagnostic rules, it speeds up the diagnostic efficiency of one-level model for a factor of 20. Further, hierarchical model representation allows for varying the level of detail at explanation and reasoning under time constraints.

Acknowledgments

The author wishes to thank Claudio Carpineto, Heedong Ko, and Jan Zytkow for valuable discussions and comments.

This research was conducted in the Center for Artificial Intelligence at George Mason University. Research of the Center for Artificial Intelligence is supported in part by the Defense Advanced Research Project Agency under grant, administered by the Office of Naval Research, No. N00014-87-K-0874, in part by the Office of Naval Research under grant No. N00014-88-K-0226, and in part by the Office of Naval Research under grant No. N00014-88-K-0397.

Table of contents

1 Introduction

The diagnosis of a system that behaves abnormally consists of locating those subsystems whose abnormal behavior accounts for the observed behavior. For example, a system being diagnosed might be a mechanical device exhibiting malfunction, or a human patient. There are two fundamentally different approaches to diagnostic reasoning.

In the first, heuristic approach, one attempts to codify diagnostic rules of thumb and past experience of human experts in a given domain. Representatives of this approach are diagnostic expert systems of the first generation, such as MYCIN (Shortliffe 1976). Here, diagnostic reasoning of human experts is being modeled, and diagnostic accuracy depends on the successful encoding of human experience. The structure of the real-world system being diagnosed is not explicitly represented, nor is its behavior being modeled.

The second approach is often called diagnosis from the first principles, or model-based diagnosis, where one starts with a description (a model) of a real-world system, e.g., de Kleer (1976), Genesereth (1984), Reiter (1987). A model explicitly represents the structure of the system, i.e., its constituent components and their connections. The diagnostic problem arises when an observation of the system's actual behavior conflicts with the system's expected behavior. The diagnostic task is to identify those system components which, when assumed to function abnormally, will account for the difference between the observed and expected system behavior. To solve the problem, model-based diagnosis relies solely on the system description and observations of its behavior. In particular, it does not use any heuristic information about the system failures.

This paper deals with model-based diagnosis only. Originality of this research is based on the idea of representing and effectively using a model of the system at several levels of detail, or abstraction (Mozetic, Bratko & Urbancic 1989). The proposed multi-level scheme is independent of any particular single-level model representation. However, certain model design principles have to be followed, and adjacent abstraction levels of the model have to satisfy formal consistency requirements.

In section 2 we relate our approach to model-based diagnosis to other model-based approaches. Usually, diagnostic reasoning is regarded as a form of nonmonotonic (Reiter 1987) or abductive reasoning (Cox & Pietrzykowski 1987). A model entails assumptions about normal states of components, and possible diagnoses are those minimal sets of

assumptions which, if removed, render the model behavior consistent with the observed behavior. In our approach, we treat every component's state as a variable, and the model as defining a mapping from any state (normal or abnormal) to corresponding observations. The diagnostic problem is then to find the inverse mapping, from given observations to possible states.

In section 3 we propose a solution to the reformulated diagnostic problem by representing a model at several levels of detail. Three refinement (abstraction) principles which can be used in the top-down or bottom-up model development are defined. We state formal conditions that must be satisfied by any pair of adjacent abstraction levels in the model representation. These conditions lead to the formulation of the hierarchical diagnostic algorithm, which exploits the hierarchical model representation. With the appropriate hierarchical model representation, the time complexity of the diagnostic algorithm is $O(\log n)$, as opposed to $O(n)$ for the generate-and-test method, where n is the number of possible states of the model. A similar reduction of complexity, from linear to logarithmic, when using abstraction hierarchies in planning was reported by Korf (1987).

It turns out that the algorithm is very general and that it can be used to solve a variety of problems. In section 4 we show how the algorithm emulates the well-known bisection method for numerical equation solving, and how the search space in a typical constraint satisfaction problem (the eight queens) can be reduced. Finally, we apply the hierarchical modeling principle and diagnostic algorithm to a nontrivial medical problem, originating from the KARDIO project (Bratko, Mozetic & Lavrac 1988, 1989). A qualitative model of the heart that simulates its electrical activity is represented at four levels of detail. The diagnostic algorithm is then used to efficiently solve the ECG interpretation problem, i.e., to locate possible heart failures based on symbolic description of electrocardiographic (ECG) data. The most detailed heart model relates 943 heart failures (both single and multiple) to 5240 ECG descriptions altogether.

Experiments and results are described in section 5. First, we outline several attempts at solving the ECG interpretation problem in KARDIO. The detailed level model of the heart was automatically transformed into different types of representation, using deductive and inductive inference techniques. We compare diagnostic efficiency and space requirements of different representations. Four-level hierarchical model falls short of being the best on the time/space tradeoff scale, but the diagnostic efficiency over one-level model is improved by a factor of 20. The hierarchical model also achieves satisfactory performance from the

practical point of view, with the average diagnostic time below 3 seconds. Its performance is very close to the compressed diagnostic rules which appear to be the optimal representation for the ECG interpretation task. Furthermore, hierarchical model representation allows for a focused explanation, and enables a tradeoff between diagnostic certainty and specificity when reasoning under time constraints.

We conclude the paper in section 6 by giving some guidelines for multi-level model representation in order to improve the diagnostic efficiency. Possible directions of further research are discussed, too.

2 Approaches to diagnosis

In order to relate our approach to model-based diagnosis to the work of others, we start this section with an example. Throughout the paper, we define models and algorithms by logic programs. We use standard Edinburgh Prolog syntax, where variables start with capital letters or underscores, and constants start with lowercase letters. All variables are implicitly universally quantified.

2.1 An example

Figure 1 depicts a binary adder, taken from Reiter (1987) and originally used by Genesereth (1984) as an example.

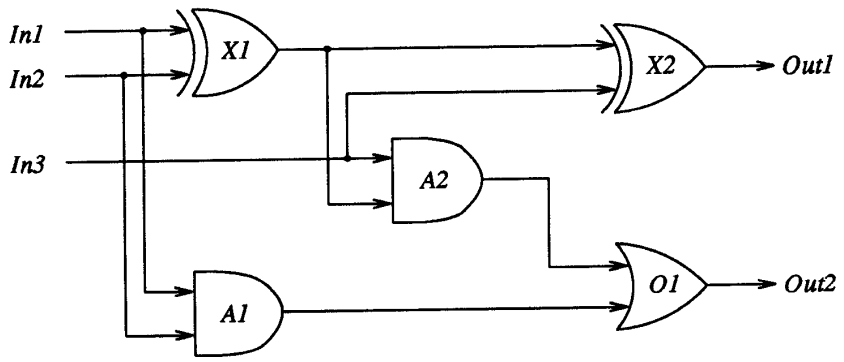

Figure 1. A full binary adder. *X1* and *X2* denote *exclusive-or* gates, *A1* and *A2* are *and* gates, and *O1* is an *or* gate.

In our approach a model relates any state (normal or abnormal) to corresponding input-output observations. The model is specified by its structure (a set of components and their connections) and functions of its components. In the case of a binary adder, its components are *and*, *exclusive-or* and *or* gates, and their functions are defined by Boolean algebra over *{0, 1}*. In a logic program, the model structure may be defined by a single clause. The head of the clause relates the state of the model to its input and output. Atoms in the body represent constituent components, and shared variables denote connections between components. The following clause defines the structure of the adder from Figure 1:

adder(state(X1, X2, A1, A2, O1), in(In1, In2, In3), out(Out1, Out2)) ←
xorg(X1, In1, In2, OutX1),
xorg(X2, In3, OutX1, Out1),
andg(A1, In1, In2, OutA1),
andg(A2, In3, OutX1, OutA2),
org(O1, OutA1, OutA2, Out2).

Normal behavior of the gates is defined by the corresponding Boolean functions:

xorg(normal, In1, In2, Out) ← xor(In1, In2, Out).
andg(normal, In1, In2, Out) ← and(In1, In2, Out).
org(normal, In1, In2, Out) ← or(In1, In2, Out).

xor(1, 1, 0).	*and(1, 1, 1).*	*or(1, 1, 1).*
xor(1, 0, 1).	*and(1, 0, 0).*	*or(1, 0, 1).*
xor(0, 1, 1).	*and(0, 1, 0).*	*or(0, 1, 1).*
xor(0, 0, 0).	*and(0, 0, 0).*	*or(0, 0, 0).*

However, in our approach abnormal behavior has to be defined as well. In the most general case, we may specify as abnormal any behavior that is not normal:

xorg(abnormal, In1, In2, Out) ← ¬xor(In1, In2, Out).
andg(abnormal, In1, In2, Out) ← ¬and(In1, In2, Out).
org(abnormal, In1, In2, Out) ← ¬or(In1, In2, Out).

Here, ¬ denotes the *negation-as-failure* operator. We will assume that the logic program interpreter correctly handles negation-as-failure, either by delaying negative goals, or by making them ground (in our example, the latter can always be done, since all variables are binary valued).

In many domains, especially in medicine, it is interesting and helpful to distinguish between different kinds of abnormal behavior. In our case, for example, we may alternatively define a faulty gate as either *open* (the output is always *0*), or *shorted* (the output is *1* for any nonzero input):

xorg(open, 1, 0, 0).
xorg(open, 0, 1, 0).
xorg(shorted, 1, 1, 1).

andg(open, 1, 1, 0).
andg(shorted, 1, 0, 1).

andg(shorted, 0, 1, 1).

org(open, 1, 1, 0).
org(open, 1, 0, 0).
org(open, 0, 1, 0).

Note that the above specification, in contrast to the original definition, does not account for all possible behaviors. In particular, there is no gate state that produces the output *Out=1* for the inputs *In1=0, In2=0*. In medicine, this would correspond to a physiologically impossible state of a patient that does not need to be considered as a possible diagnosis.

Suppose that a real adder is given the inputs *In1=1, In2=0, In3=1*, and it produces the outputs *Out1=1, Out2=0* in response. Since both outputs are wrong (correct outputs are *Out1=0, Out2=1*), this observation indicates that the adder is faulty. The diagnostic task is to locate components in the adder which, when assumed to behave abnormally, produce the observed outputs. To solve the problem, the model of the adder is used, by submitting the following query to the interpreter:

?– adder(State, in(1, 0, 1), out(1, 0)).

The query asks whether there exists a state of the adder (defined by states of its components) that produces the given input-output observation. Since several such states exist, the interpreter returns (through backtracking) the following set of answers:

```
%              X1          X2          A1          A2          O1
State = state( normal,     abnormal,   normal,     normal,     abnormal );
State = state( normal,     abnormal,   normal,     abnormal,   normal   );
State = state( normal,     abnormal,   abnormal,   normal,     abnormal );
State = state( normal,     abnormal,   abnormal,   abnormal,   abnormal );
State = state( abnormal,   normal,     normal,     normal,     normal   );
State = state( abnormal,   normal,     normal,     abnormal,   abnormal );
State = state( abnormal,   normal,     abnormal,   normal,     abnormal );
State = state( abnormal,   normal,     abnormal,   abnormal,   abnormal )
```

The query, with any of the above answer substitutions is a logical consequence of the model definition, and any answer can be considered as a possible diagnosis.

2.2 Related research

Reiter (1987) defines a system (a model in our terminology) as a pair (SD, COMPONENTS),

where SD is the system description, and COMPONENTS, the system components, is a finite set of constants. A system description is a set of first-order sentences defining how the system components are connected and how they *normally* behave. A distinguished unary predicate AB whose intended meaning is 'abnormal' is used in a system description. An observation OBS of a system is a finite set of first-order sentences. A diagnosis Δ for (SD, COMPONENTS, OBS) is a minimal subset $\Delta \subseteq$ COMPONENTS such that

$$\text{SD} \cup \text{OBS} \cup \{\text{AB}(c) \mid c \in \Delta\} \cup \{\neg\text{AB}(c) \mid c \in \text{COMPONENTS} - \Delta\}$$

is consistent. Direct generate-and-test mechanism that systematically generates subsets of COMPONENTS, with minimal cardinality first, is too inefficient for systems with large numbers of components. Instead, Reiter (1987) proposes a diagnostic method based on the concept of a conflict set, originally due to de Kleer (1976).

Corresponding to Reiter's definition, there are three diagnoses for the faulty adder: *{X1}*, *{X2, O1}*, *{X2, A2}*. In our notation, the last diagnosis *{X2, A2}* corresponds to the following state of the adder: *state(normal, abnormal, normal, abnormal, normal)*. In our representation, a diagnosis is a term, while in Reiter's representation, a diagnosis is essentially a conjunctive statement of the form: AB(*X2*) $\wedge$ AB(*A2*). More importantly, his system description models only normal behavior of the components, while we model both normal and abnormal behavior. Final distinction concerns the definition of a diagnosis. According to Reiter, a diagnosis is a conjecture that some minimal set of components are faulty, such that the consistency to SD and OBS is restored. Our definition is broader, since a diagnosis is any correct answer substitution for the state of the model which is a logical consequence of the model definition, given input-output observations. Notice, for example, that a conjecture where all gates are simultaneously abnormal: *{X1, X2, A1, A2, O1}* always restores the consistency to SD and OBS in Reiter's approach. The corresponding *state(abnormal, abnormal, abnormal, abnormal, abnormal)*, however, is not a logical consequence of our model definition for the given input-output observation.

Cox and Pietrzykowski (1987) regard diagnostic reasoning as a form of abductive inference. They extend the notion of diagnoses to causes, and define a cause as fundamental iff it is minimal, acceptable, nontrivial, and basic. The minimality criterion eliminates overly general causes, acceptability eliminates causes unrelated to the observation, nontriviality eliminates causes which directly imply the observation, and basicness eliminates intermediate causes. They show that for closed diagnostic problems where all gate connections and

observations are uniquely specified, their causes are equivalent to Reiter's diagnoses. However, for extended problems in which some gate inputs or identities of some gates are unknown, their causes contain more useful information than Reiter's diagnoses. Consider, for example, a single *and* gate *A*, with only one specified input *In1=1* and the output *Out=0*. There are two fundamental causes: *In2=0* and AB(*A*) $\wedge$ *In2=1*. In Reiter's terms, however, the diagnosis is empty. Our definition also yields as possible corresponding diagnoses *andg(normal,1,0,0)* and *andg(abnormal,1,1,0)*, since they both logically follow from the *and* gate definition. However, we do not address the problem of finding fundamental causes. We are satisfied, instead, with any logical consequence of the model that satisfies the input-output requirements.

Geffner and Pearl (1987) present an improved constraint-propagation algorithm for diagnosis, based on a probabilistic approach. They propose a diagnostic scheme where every component's state is treated as a variable. As a consequence, normal and abnormal behavior are considered on the same basis, and predictions for any possible behavior of the system can be generated. We take a non-probabilistic approach, but similarly require that the model entails both normal and abnormal (or different kinds of abnormal) behavior. Since we do not make any distinction between what is normal and abnormal, it also does not make sense to define a diagnosis as a minimal or fundamental with respect to abnormal states of components. Treatment of normal and abnormal behavior on the same basis is common in medicine, for example, since a behavior that is considered abnormal under some conditions may be a normal reaction of the body under different, unusual conditions.

3 Hierachical diagnostic algorithm

In this section we define the diagnostic problem and propose a solution by representing a model at several levels of detail. Three refinement or abstraction principles that may be used in the model development are defined, and a formal condition that must be satisfied by the hierarchical model representation is formulated. Finally, we specify a general purpose hierarchical diagnostic algorithm.

3.1 Diagnostic problem

Many approaches to model-based diagnosis rely on a model of the system which describes only normal behavior of its components (de Kleer 1976, Genesereth 1984, Reiter 1987). One may regard such a model as defining a mapping from the input to the output, under the assumption that the system is in a normal state:

$$normal: \ in \rightarrow out$$

In contrast, we consider normal and abnormal states of the system on the same basis, and require that the model describes behavior of the system for any state:

$$state_1: \ in \rightarrow out$$
$$\ldots$$
$$state_n: \ in \rightarrow out$$

Consequently, such a model may be regarded as defining a mapping from any state of the system to corresponding input-output observations:

$$model: \ state_i \rightarrow \langle in, out \rangle, \ \ 1 \leq i \leq n$$

Notice that there is no specific requirements for the model representation. We just assume that a model m is defined by a set of axioms which map a tuple of independent variables x (x denotes states) into a tuple of dependent variables y (y denotes input-output observations):

$$m: \ x \rightarrow y$$

When a system exhibits deterministic behavior (e.g., a binary adder), its model is defined by a many-to-one mapping, i.e., a function. In general, however, a system may behave non-deterministically, and consequently, its model must be defined by a many-to-many mapping.

In both cases, to denote a model, we will use either relational notation $m(x, y)$, or functional notation $y = m(x)$ when we want to emphasize the direction of inference.

Given a model m that maps any state x to the corresponding input-output observations $y = \langle in, out \rangle$, we may formulate three different tasks to be solved by the model:

- Prediction task: given x and *in*, find *out*.
- Control task: given x and *out*, find *in*.
- Diagnostic task: given $y = \langle in, out \rangle$, find x.

The **diagnostic problem**, which is the topic of the paper, is thus effectively reformulated: given mapping $y = m(x)$, find the inverse mapping $x = m^{-1}(y)$ for given values of y.

In order to appreciate the problem and its formulation, consider three cases of general interest:

(1) Equation solving, where m is a real-valued function.

For example, given is a function $y = f(x) = x + tan(x)$ where the inverse function $x = f^{-1}(y) = ?$ does not exist in analytical form. The task, to find an x for a given y, is usually solved by numerical methods.

(2) Constraint satisfaction, where m is a boolean function over discrete variables.

Given constraints, the problem is to find an assignment of values to a tuple of variables x such that the constraints are satisfied, i.e., x is mapped to $y = true$. Efficient solutions are typically based on a generate-and-test approach, where testing is incorporated into the early phases of generation.

(3) Model-based diagnosis, where m is a nondeterministic simulation model.

In technical domains, simulation models describing the behavior of physical or biological systems often exist. Such a model can be readily applied for prediction, since it maps the initial state of the system x (causes) to its final state y (manifestations). However, in general, it is not possible to interpret equations or run simulations 'backwards' in order to infer causes from their manifestations, because causal knowledge often maps different causes onto the same manifestations.

A direct generate-and-test method to diagnosis is not applicable if the domain of x is infinite,

as it is in the case (1). Even if the domain of x is finite, the method may be too inefficient for systems with large number of components, or large number of different states of components (especially when multiple faults are considered), since the domain of x is large.

To solve the diagnostic problem more efficiently, we propose to represent a model at several levels of detail, and to use a diagnostic algorithm that exploits the hierarchical representation. The idea behind the method is to first solve the diagnostic problem at an abstract level, where the model is simpler and the search space smaller. The abstract, coarse solutions are then used to guide the search at more detailed levels, where the model is more complex and the search space larger.

3.2 Three refinement/abstraction principles

In Figure 2, the representation of a model at two adjacent levels of detail is outlined. Recall that any model definition, say m_1 or m_2, may introduce some intermediate variables. However, notice that models m_1 and m_2 in Figure 2 are connected only through the hierarchical relation h between the states x and input-output observations y.

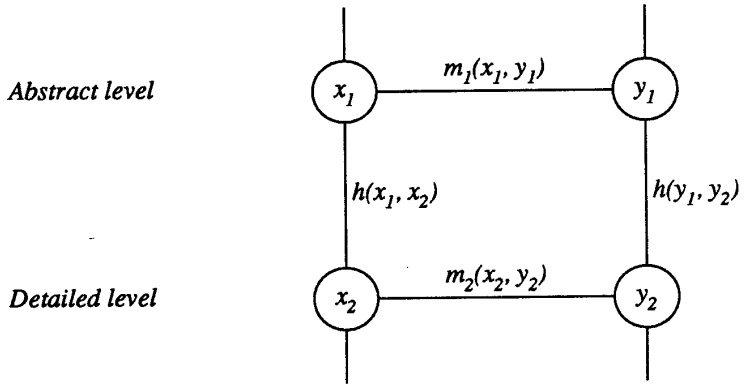

Figure 2. Hierarchical model representation. m denotes a mapping from any state x to input-output observations y, and h a relation between the abstract and detailed level states (left column) and input-output pairs (right column).

Below we define three refinement or abstraction principles that can be used in a multi-level model representation. The principles can be applied either when one refines a model in a *top-down* fashion (from abstract to detailed), or in a *bottom-up* model abstraction (from

detailed to abstract). Each principle is defined in terms of differences it induces between the abstract and detailed level model, and named with respect to the top-down/bottom-up method of model development:

(1) Introduction/supression of variables

Let w_1 be an abstract level tuple of variables (either x_1 or y_1), and w_2 a detailed level tuple (x_2 or y_2):

$$w_1 = \langle w_{11}, ..., w_{1n} \rangle$$
$$w_2 = \langle w_{21}, ..., w_{2n}, w_{2n+1}, ..., w_{2m} \rangle, \quad n \leq m$$

Each abstract level variable w_{1i} must have a detailed level counterpart w_{2i}, $1 \leq i \leq n$. However, new variables w_{2n+1}, ..., w_{2m} that are not relevant at the abstract level may be introducted at the detailed level. The relation h between tuples of variables w_1 and w_2 can be defined by the following clause:

$$h(w_1, w_2) \leftarrow h(w_{11}, w_{21}), ..., h(w_{1n}, w_{2n}).$$

(2) Refinement/abstraction of values

The relation h between individual (non-tuple) variable pairs w_1 and w_2 is defined through relations between elements of their domains (values). For example, a variable w_2 can take some values v_{21}, ..., v_{2i} which all correspond to an abstract value v_1 of w_1. Such hierarchical relations can be defined by a set of unit clauses:

$$h(v_1, v_{21}). \quad ... \quad h(v_1, v_{2i}).$$

Hierarchies of values are not restricted to tree-structured, but must be acyclic.

(3) Elaboration/simplification of mapping – model structure expansion/aggregation

The abstract level model $m_1(x_1, y_1)$ can be defined by a simpler mapping than the detailed model $m_2(x_2, y_2)$, denoted by:

$$m_1 \propto m_2$$

In the case of a component-based model representation, a function of each component c_{11}, ..., c_{1n} is also defined by a mapping. The abstract model m_1 is then defined by a composition of mappings:

$$m_1(x_1, y_1) \leftarrow c_{11}(x_1, z_{11}), ..., c_{1n}(z_{1n-1}, y_1).$$

where z_{11}, ..., z_{1n-1} are intermediate variables. On the detailed level, one can expand the model structure by introducing new components c_{2n+1}, ..., c_{2m}, and consequently define more

elaborate mapping m_2:

$$m_2(x_2, y_2) \leftarrow c_{21}(x_2, z_{21}), \ldots, c_{2n}(z_{2n-1}, z_{2n}), c_{2n+1}(z_{2n}, z_{2n+1}), \ldots, c_{2m}(z_{2m-1}, y_2).$$

Further, a function of each detailed level component can be defined by a more elaborate mapping than the abstract level component:

$$c_{1i} \propto c_{2i}, \quad 1 \le i \le n$$

3.3 Formal requirements for hierarchical representation

The three model development principles allow for a number of ways to refine or abstract the model, thus hopefully covering a large number of real-world situations. However, in order to exploit possible computational advantages of hierarchical representation over one-level representation, different levels of the model have to be mutually consistent. In particular, any pair of adjacent levels in the model representation has to satisfy the following **consistency condition**:

$$\forall x_1, y_2 \ (\exists x_2 \ m_2(x_2, y_2) \wedge h(x_1, x_2)) \Rightarrow (\exists y_1 \ m_1(x_1, y_1) \wedge h(y_1, y_2)) \qquad (1)$$

In order to give an intuitive interpretation of the above condition, let us first consider a special case.

Suppose we have a multi-level model, where the domain of the variable y (denoting observation) remains unchanged across the levels. Consequently, $h(y_1, y_2)$ is the identity relation and the consistency condition can be simplified:

$$\forall x_1, y \ (\exists x_2 \ m_2(x_2, y) \wedge h(x_1, x_2)) \Rightarrow m_1(x_1, y)$$

The simplified condition states that for any pair of variables x_1 and y, if there exists an x_2 that maps to y at the detailed level and has an abstraction x_1, then x_1 should map to y at the abstract level as well.

From the model development viewpoint, the condition prevents inconsistent abstractions and refinements. In particular, when abstracting a detailed model m_2 in a bottom-up approach, one should not oversimplify the abstract level model m_1, without proper abstraction of variable x_2 to x_1. The simplified consistency condition is violated if, given a detailed model m_2, there exists a mapping from x_2 to y, and there is an abstraction x_1 of x_2 that does not map to y at the abstract level model m_1:

$$\exists x_2,y\ m_2(x_2, y) \wedge \exists x_1\ h(x_1, x_2) \wedge \neg m_1(x_1, y)$$

In a top-down approach, when refining a given abstract model m_1, one should avoid those variable refinements of x_1 to x_2 that turn impossible mappings $\neg m_1$ into possible mappings m_2 at the detailed level. Specifically, the simplified consistency condition is violated if the abstract model m_1 does *not* map some x_1 to y, but there is a refinement x_2 of x_1 that does map to y at the detailed level model m_2:

$$\exists x_1,y\ \neg m_1(x_1, y) \wedge \exists x_2\ h(x_1, x_2) \wedge m_2(x_2, y)$$

The original consistency condition (1) is a straightforward extension of the simplified condition, and so are considerations in the bottom-up and top-down approaches.

We turn now to some consequences of the consistency condition (1) to exploit the possibilities of the search space reductions at diagnostic reasoning. This also offers guidelines for the formulation of the hierarchical diagnostic algorithm. Suppose that an input-output observation y_2 at the detailed level is given, and one wants to find the corresponding detailed level diagnosis x_2. First consider the case when y_2 does not have any abstraction y_1. A logical consequence of the consistency condition (1):

$$\forall y_2,x_2\ \neg(\exists y_1\ h(y_1, y_2)) \wedge (\exists x_1\ h(x_1, x_2)) \Rightarrow \neg m_2(x_2, y_2) \qquad (2)$$

states that for any y_2, if there is no abstraction y_1, then *no* x_2 which does have an abstraction x_1 maps to y_2. Consequently, in this rather special case, possible candidates for the diagnosis are only x_2 without any abstraction x_1.

Now consider a more common case when y_2 does have an abstraction y_1. The following logical consequence of the consistency condition (1):

$$\forall y_2,x_2\ (\forall y_1\ \exists x_1\ h(y_1, y_2) \wedge \neg m_1(x_1, y_1) \wedge h(x_1, x_2)) \Rightarrow \neg m_2(x_2, y_2) \qquad (3)$$

states that, if for all abstractions y_1 of y_2 there is an x_1 that does not map to y_1, then all refinements x_2 of x_1 do *not* map to y_2 either. Therefore, possible candidates are only those x_2 that are not refinements of x_1. This reformulation enables a major reduction of the search space at the detailed level, since it basically says that diagnoses which are impossible at the abstract level (where the search space is smaller) are impossible at the detailed level as well.

From the above conclusions it follows that the abstract level model acts as a falsity-preserving filter which can be used early in order to eliminate a number of impossible diagnoses. However, this does not ensure that diagnoses not eliminated by the abstract model are all actually possible at the detailed level. Specifically, the following is *not* a logical consequence of the consistency condition (1):

$$\forall y_2,x_2\ (\forall y_1\ \exists x_1\ h(y_1, y_2) \wedge m_1(x_1, y_1) \wedge h(x_1, x_2)) \Rightarrow m_2(x_2, y_2) \qquad (4)$$

It is not necessarily the case that for all x_1 that map to y_1 at the abstract level, all refinements x_2 of x_1 do map to y_2 at the detailed level. Therefore, it has to be explicitly verified if an individual x_2 actually maps to y_2.

The first refinement principle allows for the introduction of new variables at the detailed level. The expressive power of hierarchical model representation is thus enhanced, since phenomena which cannot be envisioned or are irrelevant at the abstract level can be ignored. However, this also renders the abstract level model **incomplete** with respect to the detailed level when some mappings do not have corresponding abstract counterparts. As a consequence, in the case of the incompleteness, the abstract level model cannot always be used as a falsity-preserving filter. Formally, the following is *not* a logical consequence of the consistency condition (1):

$$\forall y_2,x_2\ (\forall y_1\ \exists x_1\ h(y_1, y_2) \wedge m_1(x_1, y_1) \wedge \neg h(x_1, x_2)) \Rightarrow \neg m_2(x_2, y_2) \qquad (5)$$

It is not necessarily the case that for all x_2, if there is an x_1 that maps to y_1 and x_2 is not a refinement of x_1, then x_2 does *not* map to y_2. Consequently, all x_2 that have no abstraction x_1 have to be verified for the possibility that they map to y_2. This effectively means that the diagnostic algorithm cannot take any advantage of the hierarchical model representation for the parts of the model that do not have any abstractions.

The following summarizes the relationship between the consistency condition (1) and statements (2, ..., 5):

(1) $\supset$ *(2), (3)* and *(1)* $\not\supset$ *(4), (5)*

3.4 Diagnostic algorithm

Suppose that an ordered list of models m_1, ..., m_L satisfying the consistency condition is given, and hierarchical relations between adjacent levels, states, and input-output observations are specified by a binary predicate *h*. The hierarchical diagnostic algorithm is defined by a logic program which implements a depth-first, backtracking search through the space of possible states (diagnoses). The top level predicate *diagnose(L, Y, X)* relates an input-output observation *Y* to the corresponding state *X* of the model, at the level of detail *L*. *L0*, *Y0* and *X0* denote more abstract level, input-output observation, and state, respectively:

```
diagnose( L, Y, X ) ←
    abstract( L, L0 ),
    abstract( Y, Y0 ),
    diagnose( L0, Y0, X0 ),
    detailed( X0, X ),
    verify( L, X, Y ).
diagnose( L, Y, X ) ←
    no_abstract( L, X ),
    verify( L, X, Y ).
```

Normally, the procedure is invoked with a given *Y* at the detailed level *L*, and *X* unknown. The first clause deals with the case when there exists a more abstract model at level *L0*, and the observation *Y* has an abstraction *Y0*. The procedure recursively searches for the corresponding abstract state *X0*, and, if found, verifies if a refinement *X* of *X0* actually maps to the given *Y*. The intended meaning of the predicates *abstract(X, X0)* and *detailed(X0, X)* is that *X0* is an abstraction of *X*:

```
abstract( X, X0 ) ← h( X0, X ).
detailed( X0, X ) ← h( X0, X ).
```

The second clause deals with the diagnosis at the top level when there is no more abstract model, and with instances of states that do not have any corresponding abstractions. It is assumed that at each level *L*, all states *X* without any abstraction *X0* are the intended meaning of the predicate *no_abstract(L, X)*:

```
no_abstract( L, X ) ← ¬(∃ X0) h( X0, X ).
```

According to the consistency condition, if there is no abstraction for the given *Y* it suffices to check only those *X* without any abstraction. Further, all *X* without any abstraction have to be

always verified as potentially possible diagnoses. The predicate *verify(L, X, Y)* checks if the model m_L at the level L really maps X to Y:

$$verify(L, X, Y) \leftarrow m_L(X, Y).$$

Provided that the consistency condition is satisfied, it can be shown that the algorithm is *correct* and *complete* with respect to the model definition. The algorithm is obviously correct since all pairs state-observation are explicitly verified by the model itself. The algorithm is also complete since it finds all possible pairs state-observation that have a mapping according to the model definition. Suppose there is a state-observation mapping for which neither the body of the first nor the second clause can be satisfied. It is straightforward to show that such assumption is either contradictory or that it violates the consistency condition.

The reduction of search space in hierarchical diagnosis is illustrated in Figure 3.

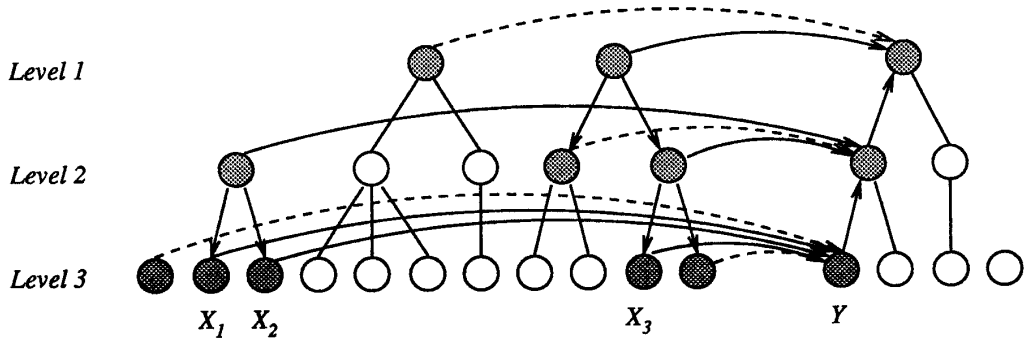

Figure 3. Search space reduction in hierarchical diagnosis.

Given a Y at the detailed level 3, the algorithm first climbs the hierarchies of input-output observations (filled circles on the right-hand side of Figure 3). The algorithm uses the abstract (level 1) model to verify if any abstract state maps to the abstract observation. Verifications are denoted by arcs, where solid arcs denote mappings while dashed arcs denote non-mappings. At the more detailed levels (2 and 3), only states that are refinements of possible abstract states, and states without abstractions are considered (filled circles on the left-hand side of Figure 3). Eventually, all three detailed states that do map to Y are found through backtracking: X_1, X_2, X_3. Now suppose that at the detailed level 3 a Y is given which does not have any abstraction, e.g., the rightmost circle in Figure 3. In this case the

algorithm checks for possible mappings only the states without abstractions, i.e., in Figure 3 only the leftmost state would be verified.

Suppose a model is defined by a one-to-one (i.e., a strictly monotonic function) or one-to-many mapping, and the state values hierarchy has the form of a tree. If there are n distinct states at the detailed level, the time complexity of the hierarchical diagnostic algorithm is $O(\log n)$, a considerable improvement over the $O(n)$ complexity of the generate-and-test method. The same reduction of complexity applies even if the model is defined by a k-to-many mapping, where k is an upper bound of possible diagnoses at each level, fixed in advance and independent of n.

4 Three case studies

In this section we show applications of hierarchical model representation and the diagnostic algorithm to three domains of general interest: equation solving, constraint satisfaction, and qualitative modeling.

4.1 Numerical equation solving: the bisection method

Suppose there is a continuous function $y = f(x)$ which does not have the inverse function f^{-1} in analytical form. To solve the equation $y = f(x)$ means to find an x for a given y_0. Suppose the initial interval $[x_l, x_r]$, $f(x_l) \le y_0 \le f(x_r)$ where f is monotonic is given. For a given error tolerance ε, the task is to narrow the interval $[x_l, x_r]$ until $|x_l - x_r| < \varepsilon$.

The hierarchical diagnostic algorithm can be readily applied to emulate the well-known bisection method. The independent, state variable X is a pair *[Xl, Xr]*, representing the interval $[x_l, x_r]$. The dependent variable Y is a real-valued variable y, and the mapping is defined by the function f. The mapping and the values of Y do not change across the hierarchical levels, while the values of X are defined by a binary tree. Notice that only the refinement/abstraction of values – principle (2) – is used in this hierarchical model specification.

Since there is no hierarchies for Y, the diagnostic algorithm can be slightly simplified:

```
diagnose( L, Y, X ) ←
    abstract( L, L0 ),
    diagnose( L0, Y, X0 ),
    detailed( X0, X ),
    verify( X, Y ).
diagnose( L, Y, X ) ←
    no_abstract( L, X ),
    verify( X, Y ).
```

Let denote abstraction levels by integers *1*, ..., L, and assume that the value for the most abstract X is the initial interval, defined by the predicate *init_solution(X)*:

```
abstract( L, L0 ) ← L > 1, L0 := L -1.

no_abstract( 1, X ) ← init_solution( X ).
```

The binary tree-structured hierarchies for X are defined by the following two clauses, where Xm is the midpoint between the interval boundaries Xl and Xr:

$$detailed(\ [Xl, Xr], [Xl, Xm]\ ) \leftarrow Xm := (Xl+Xr)/2.$$
$$detailed(\ [Xl, Xr], [Xm, Xr]\ ) \leftarrow Xm := (Xl+Xr)/2.$$

The model, unchanged across levels, just verifies if the given value of Y is within the interval $[f(Xl), f(Xr)]$ at the current level of detail:

$$verify(\ [Xl, Xr], Y\ ) \leftarrow$$
$$function(\ Xl, Yl\ ),$$
$$function(\ Xr, Yr\ ),$$
$$Yl \leq Y, Y \leq Yr.$$

Now suppose that one wants to solve the equation $x + tan(x) = 1$. Function f and the initial interval are specified by the following two clauses:

$$function(\ X, Y\ ) \leftarrow Y := X + tan(X).$$

$$init\_solution(\ [0, 1]\ ).$$

Given the error tolerance $\varepsilon = 0.00001$, and by successively increasing the level of detail until $L = 18$, the query:

$$?-\ diagnose(\ 18, 1, X\ ).$$

returns the solution $X = [0.479729, 0.479736]$.

4.2 Hierarchical constraint satisfaction: the eight queens problem

Given constraints over variables, the constraint satisfaction problem is to find an assignment of values to variables such that the constraints are satisfied. Due to a deductive nature of the problem, in principle, straightforward backtracking techniques may be used to solve it. To improve the efficiency and eliminate redundancies exploited by a simple-minded backtracking, a number of intelligent backtracking techniques was proposed, (e.g., Bruynooghe & Pereira 1984). Alternatively, Bibel (1988) proposes a general bottom-up, lazy-evaluation method which transforms a constraint satisfaction problem into the problem of evaluating a database expression. In our approach, we do not address the backtracking redundancies, but rather reduce the search by first satisfying more abstract constraints over smaller search

space.

A typical constraint satisfaction problem is to place eight queens on an empty chessboard so that no queen attacks any other queen (e.g., Bratko 1986). A sample solution on an abstract *4×4*, and a detailed *8×8* board is given in Figure 4.

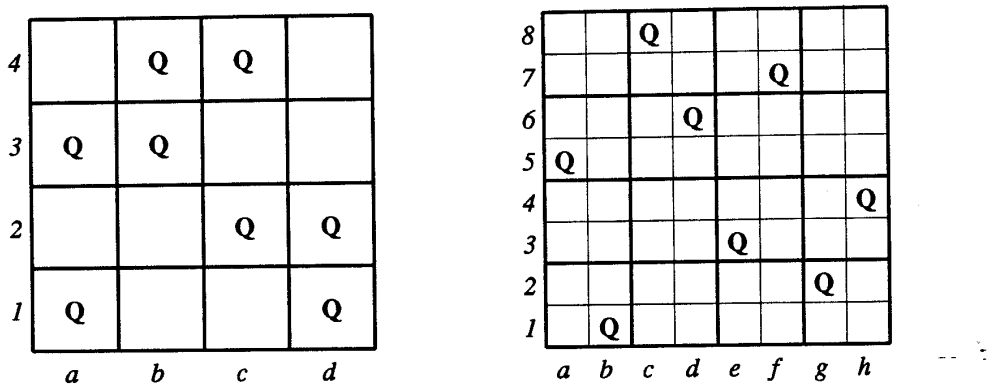

Figure 4. An abstract and detailed solution to the eight queens problem.

In the diagnostic framework, the eight queens problem may be formulated as follows: The independent variable X is an 8-tuple $Board = \langle Q_1, ..., Q_8 \rangle$ of discrete valued variables, representing a position of each queen on the board. The dependent variable Y is binary-valued $\{true, false\}$. The mapping $m(Q_1, ..., Q_8) = \{true, false\}$ is a boolean function that maps *Board* to *true* if constraints are satisfied, and to *false* otherwise. There are two levels of abstraction corresponding to the board dimensions *4×4* and *8×8*. Hierarchies for the values of X are tree structured, while values of Y are the same at both levels. The mapping $m_{4\times4}$ (i.e., constraints) at the abstract level is different from the mapping $m_{8\times8}$ at the detailed level. Notice that in this hierarchical model definition only the refinement/abstraction principles (2) and (3) are used.

We are interested only in solutions where constraints are satisfied, i.e., when *Board* maps to $Y = true$. Therefore, we can omit the dependent variable Y from the algorithm definition:

diagnose(L, Board) ←
    abstract(L, L0),
    diagnose(L0, Board0),
    detailed(Board0, Board),
    verify(L, Board).
diagnose(L, Board) ←
    no_abstract(L, Board),
    verify(L, Board).

The model has only two levels of abstraction:

abstract(8×8, 4×4).

At the abstract *4×4* level, all board positions are without abstraction:

$$no\_abstract(\ 4\times4, Board\ ) \leftarrow (\forall i\ 1\le i\le 8)\ Q_i = CR,\ C \in \{a,b,c,d\}, R \in \{1,2,3,4\}.$$

For each square on the *4×4* board, there are four corresponding squares on the *8×8* board, e.g., *c2* has refinements *e3, e4, f3, f4*. Constraints at the *8×8* board allow to place at most one queen in each row, column and diagonal, while at the *4×4* board they allow up to two queens in the same row or column, and up to three queens in the same diagonal:

verify(4×4, Board) ←
    max_row(Board, 2),
    max_col(Board, 2),
    max_diag(Board, 3).

verify(8×8, Board) ←
    max_row(Board, 1),
    max_col(Board, 1),
    max_diag(Board, 1).

It is obvious that such hierarchical model definition satisfies the consistency condition, since all configurations of eight nonattacking queens also satisfy the abstract constraints. The computational advantage of this representation stems from the fact that configurations not satisfying the abstract constraints do not need to be considered at all at the detailed level, and that the number of possible configurations on the *4×4* board is smaller than on the *8×8* board. A comparison between the one-level (*8×8*) and hierarchical (both *4×4* and *8×8*) constraints is given in Table 1.

Board	Queens per col.	Queens per row	Positions checked by constraints One-level	Hierarchical	Queens per diagonal	Solutions
4×4	2	2	90	N/A	3	73
			90	N/A		45
8×8	1	1	40320	18688	1	92
			3544	2796		92

Table 1. Number of board configurations checked by and satisfying one-level and hierarchical constraints. Lines 2 and 4 corresponds to stronger constraints at the *4×4* level, and to an early test incorporation at the *8×8* level.

In an efficient implementation of the eight queens problem, the *pigeonhole principle* can be used: since there are eight columns and rows, and eight queens to be placed on the board, it follows that in every one of the columns and rows there must be exactly one queen. There are 8! = 40320 distinct positions that satisfy this one-level *8×8* constraint (see column 4, line 3 in table 1). A similar principle can be used when refining 73 abstract level solutions (column 7, line 1), yielding $73*2^4*2^4 = 18688$ distinct positions at the detailed level (column 5, line 3). As a consequence, the hierarchical constraints reduce the number of positions to be checked for a diagonal attack by more than two times. A further improvement may be achieved by an *early test incorporation*. Instead of checking if any two queens are on the same diagonal only after all queens are on the board, we may check for the diagonal attack immediately after placing each queen on the board. This reduces the number of positions considered by one-level constraints to 3544 (column 4, line 4), and to 2796 for hierarchical constraints (column 5, line 4). In this case, constraints at the abstract level were also stronger, limiting the maximum number of queens on adjacent diagonals, and thus yielding only 45 abstract solutions (column 7, line 2).

4.3 Hierarchical qualitative modeling: the heart

The underlying motivation of the KARDIO project (Bratko, Mozetic & Lavrac 1988, 1989) was to solve the ECG interpretation problem: given a symbolic description of the ECG data, find all possible heart failures (cardiac arrhythmias). Several qualitative models which simulate the electrical activity of the heart were developed to solve the problem. In this subsection we concentrate on the hierarchical model, represented at four levels of detail, and the application of hierarchical diagnostic algorithm to efficiently solve the ECG interpretation problem. The model at the most detailed level maps 943 heart failures (both single and multiple) to 5240 ECG descriptions altogether.

In the diagnostic framework, the independent variable X denotes the qualitative state of the heart *Arr*, and the dependent variable Y the output from the heart *ECG*; there is no input. Each state *Arr* is defined as a tuple of states of the heart components (each component state in turn denotes an isolated arrhythmia *A*), and corresponds to a single or multiple cardiac arrhythmia. The *ECG* is defined as a tuple of individual ECG features *E*. There are four levels of detail, *1, 2, 3, 4*, and at each level some new variables are introduced. Specifically:

$Arr_1 = \langle A_1 \rangle$ $\quad ECG_1 = \langle E_1 \rangle$

$Arr_2 = \langle A_1, A_2, A_3 \rangle$ $\quad ECG_2 = \langle E_1, E_2, E_3, E_4 \rangle$

$Arr_3 = \langle A_1, A_2, A_3, A_4, A_5, A_6 \rangle$ $\quad ECG_3 = \langle E_1, E_2, E_3, E_4, E_5, E_6, E_7 \rangle$

$Arr_4 = \langle A_1, A_2, A_3, A_4, A_5, A_6, A_7 \rangle$ $\quad ECG_4 = \langle E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9, E_{10} \rangle$

In the hierarchical model development, all three refinement/abstraction principles were used. Apart to the introduction of new variables, values of the variables are refined at each level of detail. The model also defines different mappings $m_1, ..., m_4$ from *Arr* to *ECG* by introducing new components at each level.

The abstract heart models are usually incomplete with respect to their detailed counterparts, due to the introduction of new variables. The incompleteness prevents the search space reduction at an abstract level, and the algorithm has to resort to the inefficient generate-and-test method for the states *Arr* without abstractions. In order to avoid the repetitive use of the generate-and-test method, a set of all pairs $\langle Arr, ECG \rangle$ for all *Arr* without abstractions was generated in advance from the model at each level *L*. This renders a slightly modified diagnostic algorithm, where the second clause resorts to the predicate *surface(L, Arr, ECG)* defining $\langle Arr, ECG \rangle$ pairs in the extensional form:

```
diagnose( L, ECG, Arr ) ←
    abstract( L, L0 ),
    abstract( ECG, ECG0 ),
    diagnose( L0, ECG0, Arr0 ),
    detailed( Arr0, Arr ),
    verify( L, Arr, ECG ).
diagnose( L, ECG, Arr ) ←
    no_abstract( L, Arr ),
    surface( L, Arr, ECG ).
```

The verification whether an individial heart disorder *Arr* can actually cause a given *ECG* consists of two steps. First, the disorder is checked against constraints which eliminate physiologically impossible and medically uninteresting heart states. Then, the model simulates

the heart activity for the disorder:

$verify(\ L, Arr, ECG\ ) \leftarrow$
$\quad constraints(\ L, Arr\ ),$
$\quad heart(\ L, Arr, ECG\ ).$

At each level L, the simulation model maps a heart disorder Arr to one or more ECG descriptions. The model is defined by its structure (a set of components and their connections) and functions of the constituent components:

$heart(\ L, Arr, ECG\ ) \leftarrow$
$\quad generator(\ A_{STATE},\ Impulse_{OUT}\ ), \ldots$
$\quad conductor(\ A_{STATE},\ Impulse_{IN},\ Impulse_{OUT}\ ), \ldots$
$\quad summator(\ Impulse_{IN},\ Impulse_{IN},\ Impulse_{OUT}\ ), \ldots$
$\quad projector(\ Impulse_{IN},\ E_{OUT}\ ), \ldots$

A model component, in general, relates its qualitative state to the input and output. In the heart, the state of a component corresponds to an isolated arrhythmia A, the input is an electrical impulse *Impulse*, and the output is either an electrical impulse or an individual ECG feature E. There are four types of components in the heart model: impulse generators, conductors of impulses, summators of impulses, and projectors of impulses to the ECG.

5 Experiments and results

In this section, we emphasize the importance of application and experimental evaluation of the multi-level representation and hierarchical diagnosis to a non-toy problem. First we outline transformations between different representations of diagnostic knowledge in KARDIO, with the goal to efficiently solve the ECG interpretation problem. Then we compare diagnostic efficiency and space requirements between different representations and the four-level hierarchical model of the heart.

5.1 Knowledge transformations in KARDIO

In KARDIO (Bratko, Mozetic & Lavrac 1988, 1989), the ECG interpretation problem is formulated as follows: given a symbolic description of the ECG data, find all possible heart disorders (cardiac arrhythmias). There are both single and multiple disorders in the heart, and in the medical literature there is no systematic description of ECG features which correspond to complicated multiple disorders. Further, there is no simple rule yielding ECG features of multiple disorders, given ECG features of the constituent single disorders. These were the two main problems we encountered when attempting to construct the diagnostic knowledge base.

In order to solve the problem of multiple disorders, we took the reverse approach. Instead of constructing diagnostic rules directly, we rather developed a *simulation* model of the heart. The model is *qualitative* in the sense that it does not deal with electrical signals represented numerically as functions of time, but rather by symbolic descriptions. Subsequently, using deductive and inductive inference techniques, the qualitative model (1) was automatically transformed into a set of surface if-then rules (2), and compressed diagnostic rules (3), both representations more suitable for *diagnosis*.

The original model of the heart in KARDIO related over 2400 heart disorders to over 140 000 ECG description. In this paper, however, all experiments described were conducted by a subset of the original model, here referred to as the detailed, one-level model, relating 943 heart failures to 5240 ECG descriptions. A set of rules which reconstruct the original model from the subset is specified in (Bratko, Mozetic & Lavrac 1989).

(1) Qualitative model of the heart

The one-level model of the heart simulates its electrical activity. Specifically, the model maps any arrhythmia (a single or multiple disorder) to all corresponding ECG descriptions. An arrhythmia *Arr* is defined as a 7-tuple of isolated arrhythmias *A*, and an *ECG* as a 10-tuple of individual ECG features *E*:

$$Arr = \langle A_1, ..., A_7 \rangle$$
$$ECG = \langle E_1, ..., E_{10} \rangle$$

The model is defined by a many-to-many mapping, since each arrhythmia *Arr* may have more than one corresponding *ECG*, and several arrhythmias may map to the same ECG description. However, due to the simulation nature of the model *m*, its application in the 'forward' direction can be carried out efficiently, resorting only to *shallow backtracking* when deriving all *ECG* descriptions for a given *Arr*:

$$m(A_1, ..., A_7) = \langle E_1, ..., E_{10} \rangle$$

Since the model *m* is specified by a logic program which defines a relation between *Arr* and *ECG*, it can be used in the 'backward' direction as well:

$$m^{-1}(E_1, ..., E_{10}) = \langle A_1, ..., A_7 \rangle$$

However, the reasoning from *ECG* to *Arr* involves *deep backtracking* where a large number of fruitless paths are explored, and therefore renders the 'backward' application inefficient. The main source of fruitless branching is the model component *summator(X, Y, Z)* which, when applied, requires that for a given impulse *Z*, a pair of impulses *X* and *Y* is to be found, such that their 'sum' yields *Z*. Usually, there is a number of possible decompositions of *Z*, only few of which are consistent with other constraints in the model, and further, those inconsistencies may be found only in late stages of the model application.

(2) Surface if-then rules

Despite the fact that the model cannot be used for efficient diagnosis directly, it can be used *indirectly*. Since the model *m* relates any *Arr* to all corresponding *ECG* descriptions, one can generate an exhaustive set of pairs ⟨*Arr, ECG*⟩:

$$m(Arr, ECG) = \langle A_1, ..., A_7, E_1, ..., E_{10} \rangle$$

Such a table of pairs, properly organized and simplified, can be interpreted as a set of surface if-then rules, directly relating heart disorders to ECG observations. *Prediction rules* of the form:

if $A_1, ..., A_7$ **then** $E_1, ..., E_{10}$

can be used to predict possible ECGs for a given heart disorder, and *diagnostic rules* of the form:

if $E_1, ..., E_{10}$ **then** $A_1, ..., A_7$

can be used for efficient diagnosis.

A problem with such an exhaustive set of if-then rules is a large storage space which may be required, thus rendering it impractical for diagnostic purposes. In the KARDIO project, for example, the original model of the heart was used to generate a set of rules occupaying over 5 Mb when stored as a text file. In many practical applications it might not even be feasible to generate all pairs disorder-observation, but only a small subset. Some inductive generalization techniques must then be applied to the subset in order to extend the coverage to the whole diagnostic space (or at least most of it).

(3) Compressed diagnostic rules

In *inductive learning* (Michalski 1983), one is given a set of learning examples and some background knowledge, and the goal is to find a concept description which is consistent and complete with respect to the examples. A learning example e is usually represented as a tuple of variable values, where one designated variable denotes a class c, and the remaining values $v_1, ..., v_n$ are features of the object belonging to the class c:

$$e(v_1, ..., v_n, c)$$

The induced concept description is usually in the form of if-then rules:

if c **then** $v_1, ..., v_n$ or **if** $v_1, ..., v_n$ **then** c

where c denotes an instance of the concept, and $v_1, ..., v_n$ is a logical expression, as simple as

possible, but sufficient to discriminate between the class c and all other classes. Note that in general, an if-then rule is not a logical implication, but rather a relationship, merely indicating the direction of inference. Consequently, depending on the problem domain, the left and right-hand sides can be interchanged.

The inductive learning techniques were applied to the exhaustive set of pairs $\langle Arr, ECG \rangle$. First, ten sets of learning examples were prepared, in each a different ECG feature E_i representing the class variable:

$$e_1(A_1, ..., A_7, E_1)$$
$$\vdots$$
$$e_{10}(A_1, ..., A_7, E_{10})$$

An algorithm for learning from examples was then used, and ten sets of compressed diagnostic rules were induced:

if E_1 **then** $A_1, ..., A_7$
$$\vdots$$
if E_{10} **then** $A_1, ..., A_7$

Each rule relates an individual ECG feature E_i to a minimal description of corresponding arrhythmias $A_1, ..., A_7$ which is still sufficient to discriminate between the E_i and other ECG features. Since the set of learning examples was exhaustive and some additional conditions were satisfied, no generalization occurred in the process, and consequently the compressed diagnostic rules are logically equivalent to the original exhaustive set of if-then rules. The compressed rules are compact and can be efficiently used for diagnosis. However, their induction required 40 hours of (user) CPU time on SUN 2 (Mozetic 1986).

The same approach of constructing a qualitative model, exhaustive simulation, and induction of compressed diagnostic rules was taken by Pearce (1988) to automatically construct a fault diagnosis system of a satellite power supply. Similarly, Buchanan *et al.* (1988) show the advantage of using a classical simulation model to generate a (non-exhaustive) set of learning and testing examples, which is then used to induce rules for location of errors in particle beam lines used in high energy physics.

5.2 Time/space tradeoff

The four-level hierarchical model of the heart was developed in two stages. First, the three-level model was constructed in a top-down fashion, using QuMAS, a semiautomatic Qualitative Model Acquisition System (Mozetic 1987). The fourth, most detailed level was then added manually, by rewriting the original KARDIO heart model (which required a special interpreter) into a logic program which can be interpreted directly.

Table 2 outlines the complexity of the hierarchical model of the heart at each level of detail. The right-hand side of the table indicates the incompleteness of abstract levels, where the number of entities without abstraction for each adjacent detailed level is given. Notice that level 0 is totally incomplete with respect to level 1, and level 3 is complete with respect to level 4.

Level of detail	Hierarchical heart model			Without abstraction		
	Arr	*ECG*	⟨*Arr, ECG*⟩	*Arr*	*ECG*	⟨*Arr, ECG*⟩
1	3	3	3	3	3	3
2	18	12	23	3	3	5
3	175	263	333	26	69	79
4	943	3096	5240	0	0	0

Table 2. Number of distinct entities in the hierarchical heart model at different levels of detail, and corresponding model incompleteness.

Recall that in the cases of incompleteness, the hierarchical diagnostic algorithm has to resort to the naive generate-and-test method, thus potentially decreasing the efficiency of diagnosis. First experiments with the three-level model of the heart (Mozetic, Bratko & Urbancic 1989) showed no considerable advantage of hierarchical diagnosis over the generate-and-test method, due precisely to the high level of incompleteness in the model. In the experiments described here, we slightly modified the heart model at level 2, thus decreasing its incompleteness. Further, a set of surface if-then rules for all pairs ⟨*Arr, ECG*⟩ without abstractions was generated in advance in order to avoid the repetitive application of generate-and-test.

We compared space requirements and diagnostic efficiency of the three types of diagnostic knowledge (described in the previous subsection) to the hierarchical model of the heart. In all cases, knowledge bases and diagnostic algorithms are implemented as logic programs and compiled by Quintus Prolog. We measured space required by each representation together

with the corresponding algorithm, when both stored as text files. Diagnostic efficiency is the time needed to find *all* possible diagnoses for a given ECG, and was measured on all 3096 distinct ECG descriptions at the detailed level. Results in Table 3 are the average times over 3096 ECGs.

Type of diagnostic knowledge			Space [Kb]	Time [sec]
(1)	One-level model	generate-and-test	15	50.35
		used 'backwards'		66.30
(2)	Surface if-then rules		750	0.22
(3)	Compressed diagnostic rules		25	0.55
(4)	Hierarchical four-level model		45	2.67

Table 3. Space requirements for different representations and times spent to find *all* possible diagnoses for a given ECG description, averaged over all 3096 distinct ECGs.

Notice the very high directionality bias of the one-level heart model in Table 3. When the model is used in the 'forward' direction, the average time to derive an *ECG* for a given *Arr* is only 0.063 seconds (this is consistent with the 50.35 seconds for the generate-and-test, where the model is applied 943 times in the 'forward' direction, once for each distinct *Arr*). In contrast, the average 'backwards' application (for diagnosis) requires as much as 66.30 seconds. As a consequence, even the naive generate-and-test method turns out to be more efficient than the model used in the 'backwards' direction. Surface if-then rules are the most time efficient since only simple memory retrieval is required, but, on the other hand, they are very space demanding. Compressed diagnostic rules are optimal in terms of space and time efficiency and appear to be the best representation for the ECG interpretation. Finally, the four-level model is obviously outperformed by the compressed diagnostic rules, but achieves satisfactory performance from the practical point of view. More importantly, it is 20 times more efficient than the one-level model, and requires only three times as much space (out of 45 Kb, 11 Kb are for surface if-then rules without abstractions).

The relation between different representations of diagnostic knowledge is better illustared on a time/space tradeoff scale in Figure 5. Recall that representations (2) and (3) were automatically derived from (1), while (4) was constructed semiautomatically on top of (1).

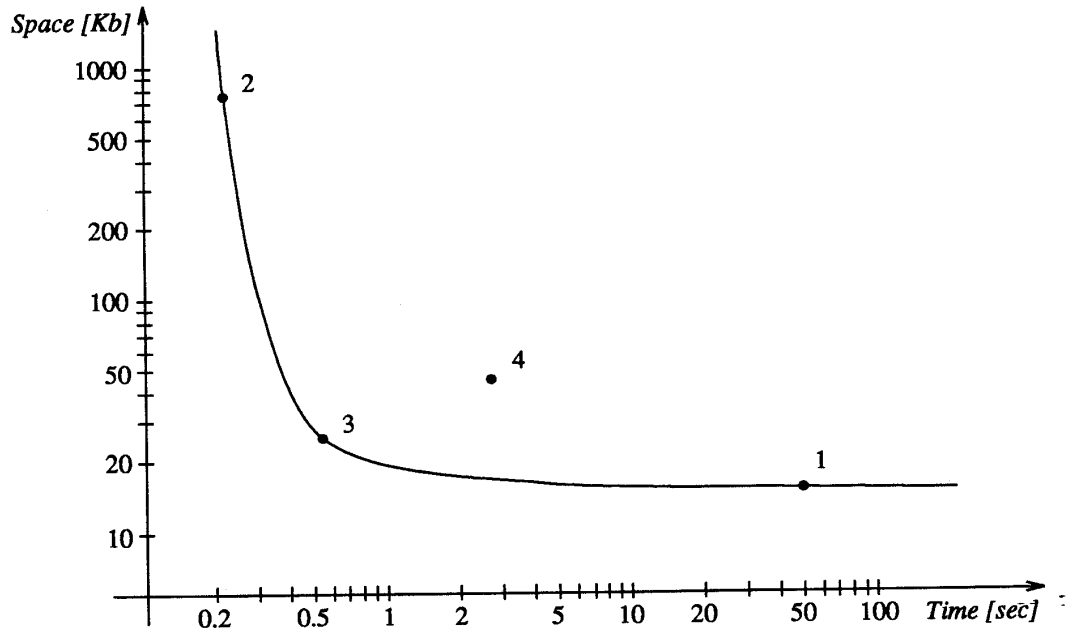

Figure 5. A tradeoff between the average diagnostic time and space requirements for different representation: (1) one-level model, (2) surface if-then rules, (3) compressed diagnostic rules, and (4) hierarchical four-level model.

In contrast to dedicated diagnostic rules, model-based reasoning offers better explanation facilities which can be even tuned to the desired level of detail (Mozetic, Bratko & Urbancic 1989). Further, the hierarchical diagnostic algorithm can be easily modified to accommodate diagnostic reasoning under time constraints, and to offer a tradeoff between diagnostic specificity and certainty. The current algorithm implements a depth-first search, favoring specificity (more detailed diagnoses) over certainty. In a breadth-first search implementation, certainty (a proportion of possible diagnoses at a given level of detail) would be favored over specificity.

6 Conclusion

In the paper, we proposed a model representation at several levels of detail with the goal to increase the efficiency of model-based diagnosis. We defined the consistency condition which has to be satisfied by the hierarchical representation, and we specifed the diagnostic algorithm. The algorithm turns out to be general, and is independent of the choice of the model representation at any single level. Further, the model is always used only in the 'forward' direction which is preferred and often the only feasible option. In particular, we envision the possibility of taking an existing simulation model, adding a few more abstract levels to it, and then using it for efficient diagnosis.

The efficiency improvement is due to the smaller search spaces at more abstract levels and the reduced search at the detailed level. The improvement depends on the branching factor of hierarchical relations and on the degree of incompleteness. In particular, it is known that in numerical equation solving, the bisection method has lower time complexity than the k-section, $k > 2$. A hierarchy in the form of a binary tree is therefore preferred over a k-ary tree or a non-tree structured hierarchy. As a consequence, to improve the efficiency, one should introduce new, intermediate levels in the hierarchical representation. For example, in the eight queens problem, it seems to be advantageous to introduce an intermediate 4×8 board. It is domain dependent, however, when such intermediate levels are meaningful, and if corresponding mappings can be easily formulated.

There is another possibility of improving diagnostic efficiency, when a component-oriented model representation is used. Instead of specifying only hierarchical relations between different level models, one could specify hierarchical relations between their constituent components as well. In this case, the verification if a detailed level model behaves consistently with the abstact level can be terminated as soon as an inconsistent behavior of a component (or a set of components) is encountered. The idea of using hierarchical relations between components was already successfully applied in QuMAS, where a model is constructed semiautomatically, in a top-down fashion, through cycles of learning, interpretation, and debugging (Mozetic 1987).

Another interesting direction of further research concerns automatic construction of abstract level models on top of an existing detailed level model. Given a class of problems to be solved by a model, it may well turn out that the existing model is unnecessarily detailed, and that a more abstract model is sufficient and even more efficient at problem solving. Such

goal-oriented reasoning may help in identifying useful abstractions and simplifications to be carried out automatically.

References

Bibel, W. (1988). Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence 35*, pp. 401-413.

Bratko, I. (1986). *Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading, MA.

Bratko, I., Mozetic, I., Lavrac, N. (1988). Automatic synthesis and compression of cardiological knowledge. In *Machine Intelligence 11* (J.E.Hayes, D.Michie, J.Richards, Eds.), pp. 435-454, Clarendon Press, Oxford, UK.

Bratko, I., Mozetic, I., Lavrac, N. (1989). *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems*. The MIT Press, Cambridge, MA (in press).

Bruynooghe, M., Pereira, L.M. (1984). Deduction revision by intelligent backtracking. In *Implementations of PROLOG* (J.A.Campbell, Ed.), pp. 194-215, Ellis Horwood, Chichester, UK.

Buchanan, B.G., Sullivan, J., Cheng, T., Clearwater, S.H. (1988). Simulation-assisted inductive learning. *Proc. 7th Natl. Conference on Artificial Intelligence, AAAI-88*, pp. 552-557, Saint Paul, MN, Morgan Kaufmann.

Cox, P.T., Pietrzykowski, T. (1987). General diagnosis by abductive inference. *Proc. 1987 Symposium on Logic Programming*, pp. 183-189, San Francisco, CA, IEEE.

de Kleer, J. (1976). Local methods for localizing faults in electronic circuits. MIT AI Memo 394, Cambridge, MA.

Geffner, H., Pearl, J. (1987). An improved constraint-propagation algorithm for diagnosis. *Proc. 10th Intl. Joint Conference on Artificial Intelligence, IJCAI-87*, pp. 1105-1111, Milan, Italy, Morgan Kaufmann.

Genesereth, M.R. (1984). The use of design descriptions in automated diagnosis. *Artificial Intelligence 24*, pp. 411-436.

Korf, R.E. (1987). Planning as search: a quantitative approach. *Artificial Intelligence 33*, pp. 65-88.

Michalski, R.S. (1983). A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach* (R.S.Michalski, J.G.Carbonell, T.M.Mitchell, Eds.), pp. 83-134, Tioga, Palo Alto, CA.

Mozetic, I. (1986). Knowledge extraction through learning from examples. In *Machine Learning: A Guide to Current Research* (T.M.Mitchell, J.G.Carbonell, R.S.Michalski, Eds.), pp. 227-231, Kluwer Academic Publishers, Boston, MA.

Mozetic, I. (1987). The role of abstractions in learning qualitative models. *Proc. 4th Intl. Workshop on Machine Learning*, pp. 242-255, Irvine, CA, Morgan Kaufmann.

Mozetic, I., Bratko, I., Urbancic, T. (1989). Varying level of abstraction in qualitative modelling. In *Machine Intelligence 12* (J.E.Hayes, D.Michie, E.Tyugu, Eds.), Oxford University Press, Oxford, UK (in press).

Pearce, D.A. (1988). The induction of fault diagnosis systems from qualitative models. *Proc. 7th Natl. Conference on Artificial Intelligence, AAAI-88*, pp. 353-357, Saint Paul, MN, Morgan Kaufmann.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence 32*, pp. 57-95.

Shortliffe, E.H. (1976). *Computer-Based Medical Consultation: MYCIN*. American Elsevier, New York.