Abstractions in Model-Based Diagnosis

Igor Mozetič
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna
Austria
igor%ai-vie.uucp@uunet.uu.net

Abstract

Most of the research on abstractions with a sound formal basis was applied to theorem proving and planning, and only a few results which show some computational advantage of abstractions are known. The paper presents an application of abstractions to model-based reasoning, with the goal to improve the efficiency of diagnosis. Model-based reasoning about a system requires an explicit representation (a model) of the system's components and their connections. Diagnosing such a system consists of finding abnormal states of components, so that the faulty system behavior is accounted for. Our approach is not restricted to qualitative models, but requires an explicit fault model. We define three abstraction operators which map a detailed, complex model to an abstract, simpler one. A formal consistency condition which must be satisfied by any pair of adjacent models in a hierarchy is specified. We define a diagnostic algorithm and prove its correctness and completeness. Abstractions and the algorithm were applied to a complex medical problem: ECG interpretation based on the model of the heart's electrical activity. Results show that hierarchical diagnosis has logarithmic time complexity when compared to a one-level diagnosis—specifically, a speedup of a factor of 20 was achieved.

1 Introduction

There are two fundamentaly different approaches to diagnostic reasoning. In the first, heuristic approach, one codifies diagnostic rules of thumb and experience of human experts in a given domain. In the second, model-based approach, one starts with a model of a real-world system which explicitly represents the structure and components of the system (e.g., de Kleer 1976, Genesereth 1984, Reiter 1987). When the system's actual behavior is different from the expected behavior, the diagnostic problem arises. The model is then used to identify components and their internal states which account for the observed behavior.

In section 2 we compare our approach to model-based diagnosis, stimulated by a medical application, to Reiter's (1987) approach. In technical domains, a model typically defines just normal behavior of components. A diagnosis is a minimal set of assumptions about the normal behavior which, if removed, render the model behavior consistent with observations. There is no need for a fault model, and the minimality criterion is essential. In our approach a strong fault model exists and all diagnoses (single and multiple, including non-minimal) are required since combined disorders might be treated differently than individual ones. There is no distinction between normal and abnormal states of components, and a model defines a mapping from any internal state to external observations. The diagnostic problem is then to find the inverse mapping, i.e., all internal states for the given observations. However, due to the 'forward' directionality bias of the model, its 'backward' application might be inefficient.

In section 3 we define three abstraction operators which map a detailed, complex model to an abstract, simpler one. The operators allow for both structural and behavioral abstractions. Our approach is related to abstractions in theorem proving (Giunchiglia & Welsh 1989, Plaisted 1981) and planning (ABSTRIPS, Sacerdoti 1974).

In section 4, a hierarchical diagnostic algorithm is defined and proved to be correct and complete. The algorithm uses an abstract model to generate potential diagnoses, and a more detailed model to verify them. Since the models are always used in the 'forward' direction, for simulation, the algorithm is suitable for integrating numerical and qualitative models (e.g., Gallanti et al. 1989). With appropriate multi-level abstractions, the complexity of diagnosis may be reduced from $O(S)$ to $O(\log S)$, where the number of states to be verified at the detailed level is S. A similar complexity reduction using abstractions is reported by Genesereth (1984) and Korf (1987, in planning), but without any experimental evidence to support the claim.

Two applications of abstractions, to a constraint satisfaction problem (map coloring) and to a complex medical problem confirm the expected complexity reduction (section 5). In the medical application—originating from the KARDIO project (Bratko, Mozetic & Lavrac 1989) the problem is to diagnose heart disorders, given an ECG and a simulation model of the heart's electrical activity. Until now, all attempts to *directly* use the model for efficient diagnosis failed—in an average case more than $50 sec.$ is needed to find all diagnoses. By abstractions and refinements, the model was represented at four levels of detail, and the average diagnostic time was reduced to $2.7 sec.$

2 Model-based diagnosis

Model-based reasoning about a system requires an explicit representation (a model) of the system's components and their connections. Reasoning is typically based on theorem proving when a model is represented by first-order logic (Genesereth 1984, Reiter 1987), or on constraint propagation (Davis 1984), possibly coupled with an ATMS (de Kleer & Williams 1987). For our approach it is essential that a model explicitly relates an internal state of components to external observations. To illustrate such a representation, a well known example of a binary adder, used by Genesereth (1984) and Reiter (1987) is represented by a logic program with negation (Lloyd 1987). We found logic programs useful to represent models since they can be naturally extended to efficiently solve constraints over finite domains (e.g. by forward checking, Van Hentenryck 1989) and to solve systems of linear equations and inequalities over real arithmetic terms (e.g. by a Constraint Logic Programming language CLP(R), Jaffar & Michaylov 1987).

Example (binary adder)

```
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 ).
```

A model relates an internal state (*ok* or *ab*) to external observations (inputs and outputs, 0s and 1s), and is specified by its structure and the functions of its components. In our example a single clause is used to define the structure, where atoms in the body represent model components (*xor, and, or* gates), and shared variables denote connections between components. Normal behavior of the components is defined by the corresponding Boolean functions:

xorg(ok, In1, In2, Out) $\leftarrow$ *xor(In1, In2, Out).*
andg(ok, In1, In2, Out) $\leftarrow$ *and(In1, In2, Out).*
org(ok, In1, In2, Out) $\leftarrow$ *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).*

Abnormal behavior (a fault model) has to be defined as well. The weakest fault model allows for any behavior, and is needed when a component consists of several diagnosable subcomponents. When components are considered primitive, a stronger fault model which specifies as abnormal any behavior that is not correct, is justified:

xorg(ab, In1, In2, Out) $\leftarrow$ $\neg$*xor(In1, In2, Out).*
andg(ab, In1, In2, Out) $\leftarrow$ $\neg$*and(In1, In2, Out).*
org(ab, In1, In2, Out) $\leftarrow$ $\neg$*or(In1, In2, Out).*

Suppose that an adder is given an input *in(1,0,1)* and produces an incorrect output *out(1,0)* (the correct output is *out(0,1)*). This indicates that the adder is faulty, and we can diagnose it by submitting the following query to the model interpreter:

$\leftarrow$ *adder(State, in(1,0,1), out(1,0)).*

The interpreter returns (through backtracking) eight states which correspond to the given input-output observations:

State = state(ab,ok,ok,ok,ok);
State = state(ok,ab,ok,ab,ok);
State = state(ok,ab,ab,ab,ab); ...

Reiter (1987) defines a model as a pair (SD, COMPONENTS). SD (system description) is a set of first-order sentences defining how the system components are connected and how they *normally* behave by referring to a distinguished unary predicate AB (meaning 'abnormal'). COMPONENTS is a finite set of constants, and an observation OBS is a finite set of first-order sentences. A diagnosis Δ for (SD, COMPONENTS, OBS) is a minimal subset $\Delta \in$ COMPONENTS such that

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

is consistent. Reiter's diagnostic algorithm is 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}. The last diagnosis {X2, A2} subsumes the two states of the adder *state(ok,ab,ok,ab,ok)* and *state(ok,ab,ab,ab,ab)* found by our model interpreter. We define a diagnosis as a correct answer substitution for the state of the model which is a logical consequence of the model definition, and not only the minimal set of faulty components, such that the consistency of SD and OBS is restored. Notice, for example, that a conjecture where all gates are simultaneously abnormal {X1, X2, A1, A2, O1} always restores the consistency of SD and OBS (but is not minimal) in Reiter's approach. The corresponding *state(ab,ab,ab,ab,ab)*, however, is not a logical consequence of the model definition for the given observation.

In our approach, a model M defines a mapping m (in general m is not a function) from *any* state to external observations. We denote the domain of m (states) by τ_x, the range (observations) by τ_y, and a typed version of m by m_τ, where

$$\forall x \in \tau_x \, \forall y \in \tau_y \; m_\tau(x,y) \leftarrow m(x,y).$$

Definition (model description)
A model description M consists of a type-free definition of m_τ(1), a type theory (2 and 3, Lloyd 1987), and a formal system which defines the mapping m (4):

1. $m_\tau(x,y) \leftarrow \tau_x(x), \tau_y(y), m(x,y).$
2. $\tau(a).$ for each constant a of type τ.
3. $\tau(f(x_1,\ldots,x_n)) \leftarrow \tau_1(x_1),\ldots,\tau_n(x_n).$ for each functor f of type $\tau_1 \times \ldots \times \tau_n \rightarrow \tau$.
4. P a formal system which defines m.

Example (binary adder, continued)

$$m_\tau(x,y_1,y_2) \quad \leftarrow \quad \tau_x(x),\ \tau_y(y_1),\ \tau_y(y_2),\ \mathit{adder}(x,y_1,y_2).$$

$$\tau_x(\mathit{state}(x_1,x_2,x_3,x_4,x_5)) \quad \leftarrow \quad \tau_s(x_1),\tau_s(x_2),\tau_s(x_3),\tau_s(x_4),\tau_s(x_5).$$

$$\tau_y(\mathit{in}(y_1,y_2,y_3)) \quad \leftarrow \quad \tau_b(y_1),\tau_b(y_2),\tau_b(y_3).$$
$$\tau_y(\mathit{out}(y_1,y_2)) \quad \leftarrow \quad \tau_b(y_1),\tau_b(y_2).$$

$$\tau_s(\mathit{ok}). \qquad \tau_s(\mathit{ab}).$$

$$\tau_b(0). \qquad \tau_b(1).$$

Definition (diagnostic problem)
Given a model description M and observations $y \in \tau_y$ a diagnosis Δ is a state $\Delta \in \tau_x$ such that $M \models m_\tau(\Delta, y)$.

The diagnostic problem is to find the inverse mapping m^{-1} for given observations y. Suppose P is a simulation model and consists of a system of equations over reals. In general, it is not possible to interpret equations or run simulation 'backwards' in order to find the inverse mapping m^{-1}. Even if the domain τ_x is finite and P is a system of constraints, any constraint propagation method may be too inefficient for a system with a large number of components and large domain τ_x. One possible solution is to represent M at several levels of abstraction and 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 Abstractions and refinements

Suppose M and M' are detailed and abstract model descriptions, respectively. A relation $M \mapsto M'$ denotes an abstraction from M to M' and a refinement from M' to M. In general $\mapsto$ is a *partial* mapping (many-to-many) from M to M'. Below we define three abstraction operators which map M to M':

$$m_\tau(x,y) \leftarrow \tau_x(x), \tau_y(y), m(x,y). \quad \mapsto \quad m'_\tau(x,y) \leftarrow \tau'_x(x), \tau'_y(y), m'(x,y).$$

by abstracting constituent components (2, 3 and 4) of the model description M to M'.

Definition (abstraction/refinement operators)

1. Collapse/refinement of constants.
   Different constants can be abstracted to a single constant, for example
   $\tau(a_1). \mapsto \tau'(a').\qquad \tau(a_2). \mapsto \tau'(a').$
   We represent the abstraction $\mapsto$ by a binary predicate h:
   $h(a_1, a').\qquad h(a_2, a').$

2. Deletion/introduction of arguments.
   Irrelevant arguments can be deleted at the abstract level, for example
   $\tau(f(x_1, \ldots, x_n)) \leftarrow \tau_1(x_1), \ldots, \tau_n(x_n). \;\mapsto\; \tau'(f'(x_2, \ldots, x_n)) \leftarrow \tau'_2(x_2), \ldots, \tau'_n(x_n).$
   where $\tau \mapsto \tau', \tau_i \mapsto \tau'_i (2 \leq i \leq n)$, and f' is f with the first argument deleted. When all arguments of f are eliminated we replace the functor f' by a constant a'. Term abstractions are also represented by the predicate h:
   $h(f(x_1, \ldots, x_n), f'(x'_2, \ldots, x'_n)) \leftarrow h(x_2, x'_2), \ldots, h(x_n, x'_n).$ or
   $h(f(x_1, \ldots, x_n), a').$

3. Simplification/elaboration of the mapping m and formal system P.
   Only some useful abstractions of $m \mapsto m'$ and the corresponding formal system $P \mapsto P'$ can be defined syntactically. For example:

   (a) Partial evaluation (e.g., van Harmelen & Bundy 1988).
   m' can be defined by $m'(x', y') \leftarrow m(x, y), h_x(x, x'), h_y(y, y')$ and partially evaluated in order to eliminate predicates h_x, h_y and some $p \in P$ from its definition.

   (b) Dropping conditions (e.g., Sacerdoti 1974).
   If m is defined by $m(x, y) \leftarrow c_1, c_2, \ldots, c_n$ then m' can be simplified to
   $m'(x, y) \leftarrow c'_2, \ldots, c'_n$ where $c_i \mapsto c'_i$ $(2 \leq i \leq n)$.

   (c) Renaming predicates (e.g., Plaisted 1981).
   Different predicates $p_1, p_2 \in P$ can be abstracted to the same predicate $p' \in P'$.

   In general, abstractions of m and P are defined implicitly by conditions C1 and C2, defined below.

An abstraction was defined as a *partial* (not total) mapping since we found it useful to ignore some unimportant features at the abstract level. For example, take the *or* gate from the adder example as an abstraction of an actual realization with three transistors and resistors (for details, see Mozetic 1990a). The abstraction ignores currents and maps only relevant voltage ranges to *low* and *high* ($0 \leq V < 0.7 \mapsto 0$ and $2 \leq V \leq 5 \mapsto 1$). Irrelevant voltage ranges (e.g., $0.7 \leq V < 2$) do not have any abstraction. However, in order to exploit possible computational advantages of the multi-level over one-level model representation, two conditions must be satisfied by any pair M and M'.

Let us denote by τ^- a subset of τ which is not abstracted since it is irrelevant at the abstract level, and by τ^+ a subset of τ which is abstracted. Define $\tau^-(x) \leftarrow \neg\exists x'\, h(x, x')$ and $\tau^+(x) \leftarrow \exists x'\; h(x, x')$. The following condition restricts the relation between the mapping m_τ, and subsets of its range τ_x^- and domain τ_y^+ which are (not) abstracted.

Definition (restriction of incompleteness)
For any m_τ, if $M \models m_\tau(x,y)$ then $\tau_x^-(x)$ or $\tau_y^+(y)$.
With respect to the model M this is equivalent to:

$C1: \quad \forall x, y\; m(x,y) \Rightarrow \neg\exists x'\; h_x(x,x') \vee \exists y'\; h_y(y,y')$

If we denote a subset of the mapping m_τ which is abstracted by m_τ^+ and define $m_\tau^+(x,y) \leftarrow \tau_x^+(x), \tau_y^+(y), m(x,y)$ then the following condition defines the relation between the detailed and abstract mapping.

Definition (preservation of mapping)
For any m_τ^+, if $M \models m_\tau^+$ then there exists m'_τ such that $M' \models m'_\tau$.
With respect to M and M' this is equivalent to:

$C2: \quad \forall x, y\; (\exists x'', y''\; m(x,y) \wedge h_x(x,x'') \wedge h_y(y,y'') \Rightarrow \exists x', y'\; m'(x',y') \wedge h_x(x,x') \wedge h_y(y,y')$

Giunchiglia and Walsh (1989) define an abstraction as a total function which maps one formal system into another where a formal system consists of a language, set of axioms, and deductive machinery. They define, among others, TI (theorem increasing) and NTI (non-theorem increasing) abstractions. An abstraction is TI iff for any detailed level theorem there exists a corresponding abstract level theorem. An abstraction is NTI iff for any non-theorem (which yields an inconsistency when added to the detailed level axioms) its abstraction added to the abstract system also yields an inconsistency. If negation is preserved across the abstraction then TI and NTI abstractions are equivalent.

An abstraction in our definition is not a total, but a partial mapping, and therefore completeness is not necessarily preserved, but C1 must be satisfied. However, for the part of the space which is abstracted (where C2 applies) our abstractions are TI/NTI. The approach by Giunchiglia and Walsh is more theoretic and general than ours, since they do not restrict axioms to definite clauses and deductive machinery to resolution. Their abstractions are defined in terms of derivability and they do not make any attempt at providing syntactic abstraction operators.

Plaisted (1981) restricts abstractions to resolution systems, and uses them for theorem proving. His abstractions are inconsistency preserving, and thus NTI, but he does not capture all NTI abstractions. He gives several instances of both syntactic and semantic abstractions. Syntactic abstractions include renaming predicates, functors, and constants (our abstraction operator 1), deleting arguments of predicates and functors (our abstraction operator 2), instantiating clauses, changing signs of literals and permuting arguments. These syntactic abstractions are applied globally, to the whole set of axioms and ensure that inconsistency is preserved. Our operators are applied only locally, to terms denoting states and observations of the model. When they are applied globally, to the whole model definition, the condition C2 is always satisfied.

Hobbs (1985) presents a theory of granularity where an abstraction is defined as a mapping from a complex to a simpler 'coarse-grained' theory. He defines the *indistinguishability relation* $\sim$ by $(\forall x, y\; x \sim y) \Leftrightarrow (\forall p \in R\;\; p(x) \Leftrightarrow p(y))$ where R is a set of predicates relevant to the situation at hand. The intended meaning is that x and y are indistinguishable if no relevant predicate distinguishes between them. This is a special case of Plaisted's abstractions, where constants are renamed in a systematic (but not necessary one-to-one) way. In our approach, this corresponds to the abstraction operator (1), where the hierarchical relation h is specified by $\forall x \sim c\; h(x, c')$ where c' represents the equivalence class of all constants indistinguishable from c.

ABSTRIPS (Sacerdoti 1974) is an early application of abstraction to planning, where preconditions of operators were abstracted according to their *criticality*. A precondition p of an operator op can be defined as a mapping from a state of the world s to *true* or *false*, depending on all primitive conditions c_i being satisfied or not:

$$p(op, s) \leftarrow c_1(\kappa_1, s), ..., c_i(\kappa_i, s), ..., c_n(\kappa_n, s).$$

Each primitive condition c_i is (automatically) assigned a criticality κ_i. In the abstract space, all conditions c_i with criticality $\kappa_i < \kappa$ are deleted from the precondition definition:

$$p'(op, s) \leftarrow c_1(\kappa_1, s), ..., c_{i-1}(\kappa_{i-1}, s), c_{i+1}(\kappa_{i+1}, s), ..., c_n(\kappa_n, s).$$

This corresponds to our abstraction operator (3), where some model components are ignored. Note that in the abstract space more operators are applicable, but those that achieve details are never selected as relevant. In ABSTRIPS there is no abstraction of the world description which would correspond to our operators (1) and (2). The relation h is therefore the identity relation, and conditions C1 and C2 are obviously satisfied. Sacerdoti claims that there is no need to delete unimportant details from the world description since they can be simply ignored. In contrast, Korf (1987) proposes abstraction of both operators and state descriptions in planning, but does not provide any specific abstraction operators.

Tenenberg (1987) defines an abstraction as a predicate mapping (not necessarily one-to-one), which is a special case of Plaisted's abstractions. However, TI and NTI abstractions may map a consistent theory into an inconsistent one. This is known as the 'false proof' problem (Plaisted 1981) since there may be a proof in the abstract space that does not correspond to any proof in the detailed space. The aim of Tenenberg's work is to ensure that consistency is preserved. He places restrictions on the abstraction mappings which preserve consistency, but has to sacrifice completeness. In this respect his approach is related to ours since we also allow for the abstract model to be incomplete, but the incompleteness is restricted by the condition C1.

4 Hierarchical diagnosis

Conditions C1 and C2 which must hold for any model abstraction $M \mapsto M'$ can be conjoined into a *consistency condition.*

Definition (consistency condition)

$CC: \ \forall x, y \; m(x, y) \wedge (\exists x'' \; h_x(x, x'')) \Rightarrow \exists x', y' \; m'(x', y') \wedge h_x(x, x') \wedge h_y(y, y')$

The following logic program tests whether CC holds between models M and M'.

Algorithm (consistency test)

$$\begin{array}{rcl} \mathit{consistent} & \leftarrow & \neg \mathit{inconsistent}. \\ \mathit{inconsistent} & \leftarrow & h_x(x, x'), m(x, y), \neg \mathit{exist\_x'y'}(x, y). \\ \mathit{exist\_x'y'}(x, y) & \leftarrow & h_x(x, x'), h_y(y, y'), m'(x', y'). \end{array}$$

Theorem. The consistency test algorithm is correct and complete. The proof is a corollary to Lemmas 18.3 and 18.4 of (Lloyd 1987, p. 115).

Suppose that given is a list of models $M_1, \ldots, M_n$, ordered from abstract to detailed where corresponding state-observation mappings are defined by predicates $m_1, \ldots, m_n$. The hierarchical diagnostic algorithm is then defined by the following logic program which implements a depth-first, backtracking search through the space of possible states.

Algorithm (hierarchical diagnosis)

$$\begin{aligned} D1: \quad & \mathit{diag}_i(y,x) & \leftarrow \quad & h_y(y,y'), \mathit{diag}_{i-1}(y',x'), h_x(x,x'), m_i(x,y). \\ D2: \quad & \mathit{diag}_i(y,x) & \leftarrow \quad & \neg \mathit{exists\_x}'(x), m_i(x,y). \\ D3: \quad & \mathit{exists\_x}'(x) & \leftarrow \quad & h_x(x,x'). \end{aligned}$$

Theorem. If the consistency condition CC is satisfied by any two adjacent models then the hierarchical diagnostic algorithm is correct and complete with respect to the mapping m_i. The proof is by induction on the abstraction level i.

• Base case ($i = 1$). There is no abstraction of x above the top level and $\neg\exists x'\ h_x(x,x')$ holds. The bodies of the clauses D1 and D3 are unsatisfiable, and the clause D2 reduces to $\mathit{diag}_1(y,x) \leftarrow m_1(x,y)$. Under the usual assumption about logic programs that predicate definitions are implicitely completed (Lloyd 1987) this proves that diag_1 and m_1 are equivalent.

• General case ($i > 1$). The algorithm is obviously correct since all pairs (x,y) are explicitly verified by the mapping m_i. We prove completeness by contradiction. Assume that the algorithm is correct and complete at the level $i-1$, i.e., $\mathit{diag}_{i-1}(y,x) \leftarrow m_{i-1}(x,y)$. Suppose (A) that there exists a mapping $m_i(a,b)$ which is not found by the algorithm. Therefore $\neg \mathit{diag}_i(b,a)$ holds, and both the body B1 of D1 and B2 of D2 (resolved with D3) must be unsatisfiable. Assuming that the consistency condition CC holds, and replacing m_i by m and m_{i-1} by m' we derive contradiction by resolution. For shortness, we do not transform formulae to conjunctive normal form or skolemize variables, but instead use two additional rules: P and $\neg(P \wedge Q)$ resolve to $\neg Q$ (steps 5 and 6), and $\forall x\ \neg P(x)$ and $\exists x\ P(x)$ resolve to $\square$ (indicating contradiction, step 9).

$$\begin{aligned}
A: \quad & m(a,b) && (1)\\
B1: \quad & \neg(h_y(y,y') \wedge m'(x',y') \wedge h_x(x,x') \wedge m(x,y)) && (2)\\
B2: \quad & \neg(\neg\exists x'\ h_x(x,x') \wedge m(x,y)) && (3)\\
CC: \quad & \neg m(x,y) \vee \neg(\exists x''\ h_x(x,x'')) \vee (\exists x',y'\ m'(x',y') \wedge h_x(x,x') \wedge h_y(y,y')) && (4)\\
1,2: \quad & \neg(h_y(b,y') \wedge m'(x',y') \wedge h_x(a,x') && (5)\\
1,3: \quad & \exists x'\ h_x(a,x') && (6)\\
1,4: \quad & \neg(\exists x''\ h_x(a,x'')) \vee (\exists x',y'\ m'(x',y') \wedge h_x(a,x') \wedge h_y(b,y')) && (7)\\
6,7: \quad & \exists x',y'\ m'(x',y') \wedge h_x(a,x') \wedge h_y(b,y') && (8)\\
5,8: \quad & \square && (9)
\end{aligned}$$

Let us assume that the cost δ of finding all diagnoses for the detailed level model M_n (without using abstractions) is a function v_n of the number of states S to be verified:

$\delta(M_n) = v_n(S) = O(S)$

In the worst case, when faults of the model components are independent, all states have to be verified. With multi-level models $M_1, \ldots, M_n$, the cost of hierarchical diagnosis is the sum of costs of verifying refinements of abstract diagnoses D_{i-1} at each level i. B_{i-1} denotes a branching factor from the level $i-1$ to i. In addition, we also have to take into account the number of newly introduced states N_i (without abstraction) which must be verified. The overall cost is:

$\delta(M_1, \ldots, M_n) = \sum_{i=1}^{n} v_i(D_{i-1} \times B_{i-1} + N_i)$

where $D_0 = 0$ and $N_1 = S_1$ (the number of states at the top level). When tree-structured hierarchies are used the number of states at each level is $S_i = S_{i-1} \times B_{i-1}$. The number of detailed level states can be expressed as a product of branching factors $S = \prod_{i=1}^{n} B_{i-1}$ if we take $B_0 = S_1$. Note that the number of abstraction levels needed is $n = \log_B S$ when one chooses a constant branching factor B. If we make a simplifying assumption that v_i, D_i and B_i are constant across levels ($v(D \times B) = C$), and there is no incompleteness ($N_{i>1} = 0$), then the linear complexity of finding all diagnoses is reduced to logarithmic:

$$\delta(M_1, \ldots, M_n) = v(D \times B) \times n = C \times \log_B S = O(\log S)$$

The reduction comes from the fact that, while the total number of states grows exponentially, the number of states to be verified is kept constant across levels. In our experiments this actually turned out to be the case.

5 Experiments and results

An interesting special case of the inverse mapping problem is constraint satisfaction. In this setting m corresponds to constraints which map a tuple of variables x to $y \in \{true, false\}$, depending on constraints being satisfied or not. First we show an application of abstractions to a well known map coloring problem. Suppose constraints are represented by a conjunction of binary predicates *next/2* for each pair of bordering countries. *Next/2* holds when its two arguments are assigned two different colors from the set of $\{$*blue, green, red, yellow*$\}$. For a map of 32 European countries, a chronological backtracking algorithm requires more than 54 *days* to find a solution. An obvious abstraction (operator 1) collapses *blue, green* $\mapsto$ *dark* and *red, yellow* $\mapsto$ *light*, and the map is at first colored by the two colors. Constraints have to be abstracted as well (operator 3): any conjunction of three binary predicates *next/2* which connect three bordering countries is replaced by a ternary predicate *next/3*. For example, *next(Austria,Italy), next(Austria,Yugoslavia), next(Italy,Yugoslavia)* $\mapsto$ *next(Austria,Italy,Yugoslavia)*. As a side product, countries which do not have at least two neighbours (e.g. *Portugal*) are ignored at the abstract level (operator 2). *Next/3* holds when one argument is assigned a different abstract color than the remaining two, e.g., *next(dark,dark,light)* holds. With such a two level representation, the first solution was found in $11min$. When an intermediate level was introduced, such that at first only one abstract color is refined and the map colored by three colors, and only then all four colors are used, the first solution was found in $4sec$. All times are CPU times on Apollo DN 3000 with compiled Quintus Prolog.

The next application involves a realistic medical problem. In KARDIO (Bratko, Mozetic & Lavrac 1989), the ECG interpretation problem is formulated as follows: given a symbolic ECG description, find all possible—single and multiple—heart disorders (cardiac arrhythmias). In the medical literature there is no systematic description of ECG features which correspond to complicated multiple disorders. Instead of constructing diagnostic rules directly we developed a *simulation* model of the electrical activity of the heart. The original model of the heart in KARDIO can simulate over 2400 heart disorders to derive over 140000 corresponding ECG descriptions. In the experiments described here we used a subset of the original model which relates 943 heart failures to 5240 ECG descriptions. Due to the simulation nature of the model its application in the 'forward' direction (deriving ECGs for a given disorder) can be carried out efficiently. In contrast, diagnostic reasoning (finding disorders for a given ECG) involves deep backtracking and renders the 'backward' application inefficient. Using

Level	States	States without	Refinements of	Diagnoses	Branching
		abstraction	abstract diagnoses		factor
i	S_i	N_i	$D_{i-1} \times B_{i-1}$	D_i	B_i
1	3	3	0	1.0	5
2	18	3	5	1.9	10
3	175	26	19	1.3	9
4	943	0	12	2.1	/
Four-level, hierarchical diagnosis			36	2.7 sec.	
One-level, generate-and-test			943	50.4 sec.	

Table 1: The number of states verified and diagnostic times needed to find all diagnoses from the heart model, averaged over 3096 distinct ECG descriptions.

the naive generate-and-test method with chronological backtracking, the average diagnostic time is more than $50 sec.$, and the application of more sophisticated constraint satisfaction techniques (different goal selection strategies, forward checking) provided no improvement. The computational complexity is due to the large number of syntactically possible states (52920), and high arity of predicates in the model (components have between 6 and 15 arguments).

In order to improve diagnostic efficiency, we represented the heart model at four levels of abstraction. First, the three-level model was constructed in a top-down way, 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. All three abstraction/refinement operators were used in the hierarchical model representation (see Mozetic 1990a).

We compared diagnostic efficiency and the number of states to be verified by a hierarchical diagnostic algorithm and a one-level generate-and-test method (Table 1). Diagnostic efficiency is the time needed to find *all* possible diagnoses for a given ECG, and was averaged over all 3096 distinct ECG descriptions at the detailed level. The heart model and the diagnostic algorithm were compiled by Quintus Prolog and run on SUN 3. The experimental results are consistent with the complexity analysis. In one-level diagnosis, for each ECG, all possible states have to be verified. Thus $v_4(943) \propto 50.4 sec.$ and the average time to verify a state is $53 msec.$ In hierarchical diagnosis the cost is $\sum_{i=1}^{4} v_i(D_{i-1} \times B_{i-1} + N_i)$. We can ignore the cost of verifying states without abstraction (N_i) since corresponding pairs state-observation were cached in a table. We further simplify the matter by assuming that costs of verifying states at different levels were equal, thus yielding the overall cost $\sum_{i=1}^{4} v(D_{i-1} \times B_{i-1}) = v(36) \propto 2.7 sec.$ The approximate time to verify one state in hierarchical diagnosis is therefore $75 msec.$ This is close to one-level diagnosis and confirms that the number of states to be verified is an indicative measure of complexity.

An alternative approach to use a deep model for efficient diagnosis is to 'compile' it into shallow diagnostic rules. A general, domain independent 'compilation' procedure, and a comparison of diagnostic efficiency and space complexity between deep and shallow knowledge in

KARDIO is described in (Mozetic 1990b).

6 Conclusion

We applied abstractions to model-based diagnosis, and showed a considerable improvement of diagnostic efficiency on a non-trivial medical problem. We defined three abstraction operators, formal conditions they have to satisfy, and a provably correct and complete diagnostic algorithm. With appropriate abstractions, the linear complexity of diagnosis can be reduced to logarithmic. The complexity reduction is due to simpler models and smaller search space at the abstract levels. The search space size depends on the branching factor of hierarchical relations and on the number of newly introduced states without abstraction (due to incompleteness). Therefore, reducing incompleteness and introducing intermediate levels improves the efficiency of hierarchical diagnosis. The questions how to find appropriate abstractions and when constructing abstract models is cost-effective, remain open. Our current research indicates that partial evaluation is a powerful technique to automatically construct abstract models on top of an existing detailed model, provided that abstractions of states and observations are given. The main limitation of our approach is that it requires a strong fault model and is geared towards the problem of finding all (including non-minimal) diagnoses. In technical domains, fault models are less common than in medicine, one is usually interested in minimal diagnoses, and the question of suggesting additional measurements has to be addressed. We are currently investigating how to upgrade an existing numerical simulation model, introduce a weak fault model at the qualitative level, and then use abstraction hierarchies to find minimal diagnoses.

Acknowledgments

Thanks to Christian Holzbaur and Gerhard Widmer for valuable discussions and comments, to Robert Trappl for making some of this work possible, and to the Austrian Federal Ministry of Science and Research for financial support. Some experiments with the heart model described here were carried out during the author's stay at the Center for Artificial Intelligence, Department of Computer Science, George Mason University, USA.

References

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

Davis, R. (1984). Diagnostic reasoning based on structure and behaviour. *Artificial Intelligence 24*, pp. 347-410.

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

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

Gallanti, M., Roncato, M., Stefanini, A., Tornielli, G. (1989). A diagnostic algorithm based on models at different level of abstraction. *Proc. 11th IJCAI*, pp. 1350-1355, Detroit, MI, Morgan Kaufmann.

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

Giunchiglia, F., Walsh, T. (1989). Abstract theorem proving. *Proc. 11th IJCAI*, pp. 372-377, Detroit, MI, Morgan Kaufmann.

Hobbs, J.R. (1985). Granularity. *Proc. 9th IJCAI*, pp. 432-435, Los Angeles, CA, Morgan Kaufmann.

Jaffar, J., Michaylov, S. (1987). Methodology and implementation of a CLP system. *Proc. 4th Intl. Conference on Logic Programming*, pp. 196-218, Melbourne, Australia, The MIT Press.

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

Lloyd, J.W. (1987). *Foundations of Logic Programming* (Second edition). Springer-Verlag, Berlin.

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. (1990a). Hierarchical model-based diagnosis. Report TR-90-3, Austrian Research Institute for Artificial Intelligence, Vienna, Austria. To appear in *International Journal of Man-Machine Studies.*

Mozetic, I. (1990b). Diagnostic efficiency of deep and surface knowledge in KARDIO. Report TR-90-6, Austrian Research Institute for Artificial Intelligence, Vienna. To appear in *Artificial Intelligence in Medicine — Special Issue on Deep Models.*

Plaisted, D.A. (1981). Theorem proving with abstractions. *Artificial Intelligence 16*, pp. 47-108.

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

Sacerdoti, E.D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence 5*, pp. 115-135.

Tenenberg, J.D. (1987). Preserving consistency across abstraction mappings. *Proc. 10th IJCAI*, pp. 1011-1014, Milan, Italy, Morgan Kaufmann.

Van Harmelen, F., Bundy, A. (1988). Explanation-based generalisation = partial evaluation. *Artificial Intelligence 36*, pp. 401-412.

Van Hentenryck, P. (1989). *Constraint Satisfaction in Logic Programming.* The MIT Press, Cambridge, MA.