Improving Diagnostic Efficiency in KARDIO: Abstractions, Constraint Propagation, and Model Compilation[1]

Igor Mozetič and Bernhard Pfahringer

Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
e-mail: (igor,bernhard)@ai.univie.ac.at

Abstract

The KARDIO system deals with the problem of diagnosing cardiac arrhythmias from symbolic descriptions of electrocardiograms. The system incorporates a qualitative model which simulates the electrical activity of the heart. In the paper we outline three methods for an efficient application of a simulation model to diagnosis. First, through abstractions and refinements, the model is represented at several levels of detail. Second, the model is reformulated in terms of constraints which enable efficient propagation of relational dependencies and reduce backtracking. And finally, the model is 'compiled' into surface diagnostic rules. Through simulation, a relational table is generated and subsequently compressed into efficient diagnostic rules by inductive learning. A novel contribution to KARDIO, presented here, includes a comparison of diagnostic efficiency and space complexity of five types of knowledge: a simulation model of the heart, a hierarchical four-level model, a model represented in terms of constraints, a relational table, and compressed diagnostic rules.

1 INTRODUCTION

The heart can be viewed as a device with an electrical control system consisting of interconnected components. This electrical system works autonomously within the heart and is responsible for generating the rhythmical stimulation impulses that cause the contraction of the heart muscle, and consequently changes in the electrical potentials in the body. The changes of these potentials in time can be recorded as an electrocardiogram (ECG). Disorders which can occur in the electrical control system of the heart are reflected in the

[1]This is an updated and extended version of the paper by Igor Mozetič, Diagnostic efficiency of deep and surface knowledge in KARDIO, *Artificial Intelligence in Medicine 2 (2)*, pp. 67-83, 1990.

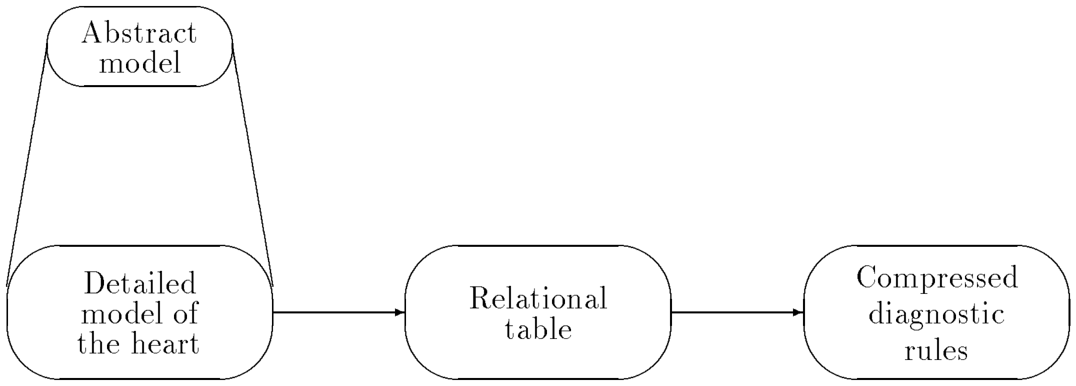

Figure 1: Deep and surface representations of the electrocardiographic knowledge.

ECG curves. For example, an impulse generator may become silent or overactive, or some electrical conductance may become partially or totally blocked. These disorders are called *cardiac arrhythmias* and cause some characteristic changes in the ECG. The diagnostic problem is to decide which cardiac arrhythmias could have caused an abnormal ECG.

In KARDIO [1], the ECG interpretation problem is formulated as follows: given a *symbolic* description of the ECG data, find *all* possible cardiac arrhythmias. There are both single and multiple disorders in the electrical system of the heart. In the medical literature (e.g., [9]), however, 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. Instead of directly constructing diagnostic rules which deal with multiple disorders we took an indirect approach. We first developed a simulation model of the electrical system 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. The model can be efficiently used for simulation, but not for diagnosis.

In the paper we present and compare three alternative approaches to efficient application of a simulation model to diagnosis. First, by representing a deep model at several levels of abstraction, second, by efficient constraint propagation, and third, by 'compiling' a model into a set of surface diagnostic rules (Figure 1). In section 2 we describe the deep model of the heart, its representation at several levels of abstraction, and the hierarchical diagnostic algorithm. The underlying idea 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. Alternatively, the model can be used directly, but with the naive, depth-first, backtracking search replaced by constraint propagation of relational dependencies.

In section 3 we show the automatic compilation of the model into a set of efficient diagnostic rules. First, by exhaustive simulation, the model is transformed into a relational table. Entries in the table are then used as examples by an inductive learning program, and the table is compressed into a set of simple if-then rules.

We compare the complexity and efficiency of different diagnostic representations in section 4. Complexity is measured by the space required to store a knowledge base, and efficiency is the average time needed to find all diagnoses. In our experiments, a nontrivial subset of the original KARDIO problem domain was used. The model described here comprises 943 cardiac arrhythmias (both single and multiple) which arise by combining 29 single heart disorders. The original KARDIO domain can be reconstructed by a set of rules specified in [1].

There are two novel contributions of the paper with respect to the KARDIO project. First, the heart model is represented by a logic program which does not require any special purpose interpreter. And second, five different representations (a one-level model, a four-level hierarchical model, a constraints-based model, a relational table, and compressed diagnostic rules) are compared on the same problem domain. Knowledge bases and diagnostic algorithms are implemented as logic programs in standard Prolog, except in the case of the constraints-based representation where an extension of the unification algorithm is needed.

2 DEEP MODEL OF THE HEART

2.1 Detailed level model

There are two fundamentally 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 (e.g., MYCIN [21]). 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., [5, 8, 6, 20]). 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.

A model of the electrical system of the heart comprises four types of components: impulse generators, conductors of impulses, impulse summators, and projectors of impulses to the ECG. In general, a component relates its qualitative state to the input and output. In the heart, the state of a component corresponds to an isolated disorder A, the input is an electrical impulse *Impulse*, and the output is either an electrical impulse or an individual ECG feature E. Specifically, the components have the following form:

- *generator*(A_{STATE}, $Impulse_{OUT}$)
- *conductor*(A_{STATE}, $Impulse_{IN}$, $Impulse_{OUT}$)

- *summator($Impulse_{IN}$, $Impulse_{IN}$, $Impulse_{OUT}$)*
- *projector($Impulse_{IN}$, E_{OUT})*

An arrhythmia *Arr* is defined as a 7-tuple of isolated disorders A_i:

$$Arr = \langle A_1, \ldots, A_7 \rangle$$

Variables $A_1, \ldots, A_7$ denote states of the heart components (impulse generators or foci, and conductors):

$$Arr = \langle \mathit{SA},\ \mathit{AF},\ \mathit{AV},\ \mathit{JF},\ \mathit{BB},\ \mathit{VF},\ \mathit{VEF} \rangle$$

SA denotes the sino-atrial node, *AF* is an atrial focus, *AV* is the atrio-ventricular conduction, *JF* is a junctional focus, *BB* denotes conduction through the bundle branches, *VF* is a regular ventricular focus, and *VEF* is an ectopic ventricular focus. Each component may be in a normal or one of several abnormal states. For example, the normal state of the heart, sinus rhythm (*sr*), is defined by

$$Arr = \langle \mathit{sr},\ \mathit{quiet},\ \mathit{normal},\ \mathit{quiet},\ \mathit{normal},\ \mathit{quiet},\ \mathit{quiet} \rangle$$

where the *SA* node is in the state *sr*, other generators (*AF, JF, VF, VEF*) are *quiet* and both conductors (*AV, BB*) are *normal*. Sometimes we use the attribute-value notation instead of pure relational notation in order to improve the readability. Each element of a relational tuple is assigned to a variable which corresponds to the element position in the tuple. For example, a multiple arrhythmia, atrial tachycardia with the LGL syndrome and junctional ectopic beats (*at, lgl, jeb*) is described by the following 7-tuple:

$$Arr = \langle \mathit{SA}{=}\mathit{quiet},\ \mathit{AF}{=}\mathit{at},\ \mathit{AV}{=}\mathit{lgl},\ \mathit{JF}{=}\mathit{jeb},\ \mathit{BB}{=}\mathit{normal},\ \mathit{VF}{=}\mathit{quiet},\ \mathit{VEF}{=}\mathit{quiet} \rangle$$

An ECG pattern *ECG* is a 10-tuple of individual ECG features E_i:

$$ECG = \langle E_1, \ldots, E_{10} \rangle$$

The above arrhythmia (*at, lgl, jeb*) has three corresponding ECG patterns:

$$
\begin{array}{lllll}
ECG = \langle & \mathit{Rhythm} = \mathit{regular}, & & \mathit{regular}, \\
 & \mathit{P\_wave} = \mathit{abnormal}, & & \mathit{abnormal}, \\
 & \mathit{Rate\_of\_P} = \mathit{between\_100\_250}, & & \mathit{between\_100\_250}, \\
 & \mathit{Relation\_P\_QRS} = \mathit{after\_P\_always\_QRS}, & & \mathit{after\_P\_always\_QRS}, \\
 & \mathit{PR\_interval} = \mathit{shortened}, & \vee & \mathit{shortened}, \\
 & \mathit{QRS\_complex} = \mathit{normal}, & & \mathit{normal}, \\
 & \mathit{Rate\_of\_QRS} = \mathit{between\_100\_250}, & & \mathit{between\_100\_250}, \\
 & \mathit{Ectopic\_P} = \mathit{abnormal}, & & \mathit{absent} \\
 & \mathit{Ectopic\_PR} = \mathit{after\_QRS\_is\_P} \vee \mathit{shortened}, & & \mathit{meaningless} \\
 & \mathit{Ectopic\_QRS} = \mathit{normal} & & \mathit{normal} \rangle
\end{array}
$$

For shortness and better readability we allow for an internal disjunction to appear in a tuple. An expression $\langle E_1 = v_1, E_2 = v_2 \vee v_3 \rangle$ is equivalent to two pairs $\langle v_1, v_2 \rangle$ and $\langle v_1, v_3 \rangle$. For example, *Ectopic_PR = after_QRS_is_P* ∨ *shortened*, together with other ECG features on the left above corresponds to two ECG patterns.

The heart model maps any arrhythmia (a single or a multiple disorder) to all corresponding ECG patterns. The mapping m from *Arr* to *ECG* is a relation (many-to-many) since each arrhythmia may have more than one corresponding ECG, and several arrhythmias may map to the same ECG pattern. For example, for the left-most ECG pattern above there are two possible arrhythmias, (*at, lgl, jeb*) and (*at, jeb*):

⟨*SA=quiet, AF=at, AV=lgl, JF=jeb, BB=normal, VF=quiet, VEF=quiet* ⟩
⟨*SA=quiet, AF=at, AV=normal, JF=jeb, BB=normal, VF=quiet, VEF=quiet* ⟩

The mapping *m(Arr,ECG)* is defined in terms of the predicate *possible(Arr)*, and the simulation model *heart(Arr,ECG)*. *Possible* eliminates physiologically impossible and medically uninteresting heart states, and *heart* simulates the heart activity for an arrhythmia *Arr*:

m(Arr, ECG) ← *possible(Arr), heart(Arr, ECG).*

Throughout the paper, we define models, rules, and algorithms by logic programs. We use the standard Edinburgh Prolog syntax (e.g., [4]), where constants start with lowercase letters, variables start with capital letters, and all variables are implicitly universally quantified. However, we divert from the standard syntax by allowing for subscripts and tuples instead of structured terms (e.g., $\langle X_1, X_2 \rangle$ instead of $f(X1, X2)$).

The simulation model is defined by its structure (a set of components and their connections) and behavior of the constituent components. The following clause defines the structure of the heart model:

```
heart( ⟨SA, AF, AV, JF, BB, VF, VEF⟩,
        ⟨Rhythm, P_wave, Rate_of_P, Relation_P_QRS, PR_interval,
        QRS_complex, Rate_of_QRS, Ectopic_P, Ectopic_PR, Ectopic_QRS⟩)  ←
    sa_node_generator( SA, ImpulseSA ),
    atrial_generator( AF, ImpulseAF ),
    summator( ImpulseSA, ImpulseAF, ImpulseATR ),
    anterograde_av_conductor( AV, ImpulseATR, ImpulseAV ),
    junctional_generator( JF, ImpulseJF ),
    regular_ventricular_generator( VF, ImpulseVF ),
    ectopic_ventricular_generator( VEF, ImpulseVF ),
    summator( ImpulseJF, ImpulseVF, ImpulseIV ),
    retrograde_av_conductor( AV, ImpulseIV, ImpulseRET ),
    summator( ImpulseAV, ImpulseIV, ImpulseINV ),
    summator( ImpulseRET, ImpulseATR, ImpulseSV ),
    summator( ImpulseAV, ImpulseJF, ImpulseHIS ),
    bundle_branches_conductor( BB, ImpulseHIS, ImpulseBB ),
    summator( ImpulseBB, ImpulseVF, ImpulseVENT ),
    atrial_projector( ImpulseSV, P_wave, Rate_of_P, Ectopic_P ),
    atrio_vent_projector( ImpulseSV, ImpulseINV, Relation_P_QRS,
                          PR_interval, Ectopic_PR ),
    ventricular_projector( ImpulseVENT, Rhythm, QRS_complex,
                          Rate_of_QRS, Ectopic_QRS ).
```

The head of the clause relates the state of the heart *Arr* to the output *ECG*. Atoms in the body represent heart components (generators, conductors, summators and projectors), and shared variables (impulses) denote connections between the components.

Due to the simulation nature of the model m, its application in the 'forward' direction

$$m(\langle A_1, \ldots, A_7 \rangle) \;\mapsto\; \langle E_1, \ldots, E_{10} \rangle$$

can be carried out efficiently. For a given disorder *Arr*, the logic program interpreter can derive all *ECG* patterns resorting only to shallow backtracking. For diagnostic purposes, however, the 'backward' application is required — for a given *ECG* find all *Arr*:

$$m^{-1}(\langle E_1, \ldots, E_{10} \rangle) \;\mapsto\; \langle A_1, \ldots, A_7 \rangle$$

Since the model m is specified by a logic program there is no inherent obstacle to the 'backward' application. 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 a 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.

Until recently, all attempts to *directly* use the model for efficient diagnosis failed. Using the naive generate-and-test method with chronological backtracking, the average diagnostic time is more than 1 second per ECG. The computational complexity is due to the large number of syntactically possible states (52,920), and high arity of predicates in the model. Impulses are structured terms with 5 arguments, and the model components have therefore between 6 and 15 primitive arguments.

2.2 Model abstractions and refinements

One approach to improve the diagnostic efficiency of the deep model is to represent it at several levels of abstraction, and to first solve the diagnostic problem at an abstract level. The abstract diagnoses are then used to restrict the search for more detailed diagnoses.

First we define three abstraction/refinement operators which can be used in a multi-level model representation. The abstraction operators are applied when one simplifies a model in a *bottom-up* fashion (from detailed to abstract). Complementary refinement operators are used in a *top-down* model development (from abstract to detailed).

- Collapse/refinement of values.
  Indistinguishable values of a variable can be abstracted into a single value. For example, the values *wide_LBBB* and *wide_RBBB* of the ECG feature *QRS_complex* are abstracted to *wide*. We represent the abstraction by a binary predicate h:

  $$h(\mathit{wide\_LBBB}, \mathit{wide}). \qquad\qquad h(\mathit{wide\_RBBB}, \mathit{wide}).$$

- Deletion/introduction of variables.
  Irrelevant variables can be deleted at the abstract level. For example, the last three ECG features *Ectopic_P, Ectopic_PR,* and *Ectopic_QRS* can be ignored. This is represented by the following clause:

  $$h(\langle E_1, \ldots, E_{10}\rangle, \langle E'_1, \ldots, E'_7\rangle) \;\leftarrow\; h(E_1, E'_1), \ldots, h(E_7, E'_7).$$

  where E'_i denote the abstract ECG features.

- Simplification/elaboration of the mapping m.
  Detailed level mapping m can be simplified to m' by ignoring and/or simplifying some model components. In general, however, the mapping abstractions are defined by a formal consistency condition which must hold between m and m' (for details see [17, 18]).

By an application of the abstraction and refinement operators, the heart model was represented at four levels of detail. All three abstraction/refinement operators were used. Apart from the introduction of new variables, values of the variables were refined at each level of detail. The hierarchical model defines different mappings $m_1, \ldots, m_4$ from *Arr* to *ECG* by introducing new components at each level. First, the three-level model was constructed in a top-down fashion, using QuMAS, a semiautomatic Qualitative Model Acquisition System [16]. The fourth, detailed level model, described in the previous subsection was then manually connected to the third level. The heart model at the first, most abstract level is very simple:

```
heart( Arr, ECG )  ←
    generator( Arr, Impulse ),
    projector( Impulse, ECG ).

    generator( brady, form(under_60) ).
    generator( rhythm, form(between_60_100) ).
    generator( tachy, form(over_100) ).

    projector( form(Rate), Rate ).
```

It consists of only two components, an impulse generator and a projector. Instead of a 7-tuple, an arrhythmia description *Arr* is a singleton and only three abstract arrhythmias are considered: a bradycardia (*brady*), a normal rhythm (*rhythm*), and a tachycardia (*tachy*). An *ECG* pattern is also a singleton, covering one ECG feature, *Rate_of_QRS*. An impulse is a structured term, but at this level it has only one argument (rate) in contrast to five arguments at the detailed level (shape, rhythm and rate of the regular part, and type and shape of the ectopic part).

Suppose that given is a list of mappings $m_1, \ldots, m_n$, ordered from abstract to detailed. Hierarchical relations between detailed and abstract ECG patterns are defined by the predicate h_E, and relations between detailed and abstract arrhythmias by the predicate h_A. The hierarchical diagnostic algorithm is then defined by the following logic program which implements a depth-first, backtracking search through the space of possible diagnoses:

```
$diag_i( ECG, Arr ) \leftarrow$
    $h_E( ECG, ECG' )$,
    $diag_{i-1}( ECG', Arr' )$,
    $h_A( Arr, Arr' )$,
    $m_i( Arr, ECG )$.
$diag_i( ECG, Arr ) \leftarrow$
    $\neg \exists Arr' \; h_A( Arr, Arr' )$,
    $m_i( Arr, ECG )$.
```

The algorithm first climbs the hierarchy of ECG patterns, and recursively finds an abstract diagnosis *Arr'* which maps to the abstract *ECG'*. The detailed model m_i then simulates refinements *Arr* of *Arr'* to verify which arrhythmias actually map to the given *ECG*. Notice that detailed arrhythmias which are not refinements of the abstract diagnosis *Arr'* are not considered at all. With appropriate abstractions, this results in a major reduction of the search space at the detailed level [17].

However, an abstract model may be incomplete with respect to the detailed level model, i.e., not all phenomena are necessarily abstracted. For example, the abstract model of the heart above does not incorporate any conduction disorders — they are introduced only at more detailed levels. Unfortunately, such incompleteness prevents the search space reduction at an abstract level. The diagnostic algorithm has to resort to the inefficient generate-and-test method for the arrhythmias *Arr* without abstractions. This is covered by the second clause of the algorithm.

Formal conditions which have to be satisfied by the hierarchical model representation, and relations to the relevant work on abstractions are in [17, 18]. If the conditions are satisfied, then the diagnostic algorithm is correct and complete. Further, with appropriate abstractions, the algorithm reduces the linear complexity of the generate-and-test method to logarithmic.

2.3 Hierarchical diagnosis

In this section we give an example of diagnostic reasoning based on the heart model represented at four levels of detail. The example is taken from [18]. Suppose the following ECG pattern at the fourth level of detail is given:

$$
\begin{aligned}
ECG_4 = \langle & \mathit{Rhythm} = \mathit{regular}, \\
& \mathit{P\_wave} = \mathit{abnormal}, \\
& \mathit{Rate\_of\_P} = \mathit{between\_100\_250}, \\
& \mathit{Relation\_P\_QRS} = \mathit{after\_P\_always\_QRS}, \\
& \mathit{PR\_interval} = \mathit{shortened}, \\
& \mathit{QRS\_complex} = \mathit{normal}, \\
& \mathit{Rate\_of\_QRS} = \mathit{between\_100\_250}, \\
& \mathit{Ectopic\_P} = \mathit{abnormal}, \\
& \mathit{Ectopic\_PR} = \mathit{after\_QRS\_is\_P}, \\
& \mathit{Ectopic\_QRS} = \mathit{normal} \rangle
\end{aligned}
$$

The hierarchical diagnostic algorithm first uses hierarchies defined by h_E to find a more abstract ECG pattern. At the third level, the last three variables *Ectopic_P, Ectopic_PR,* and *Ectopic_QRS* are deleted:

$$
\begin{aligned}
ECG_3 = \langle & \mathit{Rhythm} = \mathit{regular}, \\
& \mathit{P\_wave} = \mathit{abnormal}, \\
& \mathit{Rate\_of\_P} = \mathit{between\_100\_250}, \\
& \mathit{Relation\_P\_QRS} = \mathit{after\_P\_always\_QRS}, \\
& \mathit{PR\_interval} = \mathit{shortened}, \\
& \mathit{QRS\_complex} = \mathit{normal}, \\
& \mathit{Rate\_of\_QRS} = \mathit{between\_100\_250} \rangle
\end{aligned}
$$

At the second level of abstraction, variables *Rhythm, Rate_of_P,* and *PR_interval* are deleted. Values of *P_wave* = *abnormal* and *QRS_complex* = *normal* are both abstracted to the value *present*, and *Rate_of_QRS* = *between_100_250* is abstracted to *over_100*:

$$
\begin{aligned}
ECG_2 = \langle & \mathit{P\_wave} = \mathit{present}, \\
& \mathit{Relation\_P\_QRS} = \mathit{after\_P\_always\_QRS}, \\
& \mathit{QRS\_complex} = \mathit{present}, \\
& \mathit{Rate\_of\_QRS} = \mathit{over\_100} \rangle
\end{aligned}
$$

At the most abstract level, all variables but *Rate_of_QRS* are deleted:

$$ECG_1 = \langle \mathit{Rate\_of\_QRS}{=}\mathit{over\_100} \rangle$$

$$\begin{tikzcd}[column sep=tiny]
\mathrm{Arr}_1 = & & & & & & & \langle Arr\rangle \arrow[dlllll, dash] \arrow[dll, dash] \arrow[drrr, dash] & & & & & & \\
\mathrm{Arr}_2 = & & \langle SV, \arrow[dl, dash] \arrow[dr, dash] & & & AV, \arrow[d, dash] & & & & & IV\rangle \arrow[dlll, dash] \arrow[dl, dash] \arrow[drr, dash] & & & \\
\mathrm{Arr}_3 = & \langle SA, \arrow[d, dash] & & AF, \arrow[d, dash] & & AV, \arrow[d, dash] & & JF, \arrow[d, dash] & & BB, \arrow[d, dash] & & & VF\rangle \arrow[dl, dash] \arrow[dr, dash] & \\
\mathrm{Arr}_4 = & \langle SA, & & AF, & & AV, & & JF, & & BB, & & VF, & & VEF\rangle
\end{tikzcd}$$

Figure 2: Representation of arrhythmias at different levels of detail.

The abstract model of the heart is then used to find a possible diagnosis at this extremely simple level. The only possibility is *tachy* — a tachycardia in medical terminology. Now the algorithm resorts to hierarchies of arrhythmias h_A to refine this abstract diagnosis, and uses more detailed heart models to verify which refinements can actually produce the given ECG pattern.

Hierarchies of arrhythmias are more complicated than hierarchies of ECG patterns. At each level new variables are introduced, and typically a value of an abstract level variable depends on values of tuples of detailed level variables and not only on individual detailed level variables (as is the case with ECG patterns). Recall that individual variables correspond to the states of the heart components and that their values denote isolated disorders. Figure 2 defines hierarchies of tuples and dependencies between individual variables. At the first level *Arr* denotes the state of the heart, regarded as an impulse generator. At the second level, *SV* corresponds to a supra-ventricular focus, *AV* is the atrio-ventricular conduction, and *IV* denotes an intra-ventricular focus. The meaning of the variables at the third and the fourth level was defined in section 1.1.

Figure 3 gives some examples of hierarchical relations between values of individual variables and tuples of variables. A variable which has no value assignment in a tuple can take any value from its domain. Abbreviations for isolated arrhythmias used at the fourth level of detail correspond to the following medical terms: *st* is sinus tachycardia, *aeb* are atrial ectopic beats, *at* is atrial tachycardia, *mat* is multi-focal atrial tachycardia, *lgl* is the LGL syndrome, *wpw* is the WPW syndrome, *avb1* is the AV block, first degree, *wen* is the AV block of type Wenckebach, *mob2* is the AV block, type Mobitz 2, *avb3* is the AV block, third degree, *jt* is junctional tachycardia, *jeb* are junctional ectopic beats, *vt* is ventricular tachycardia, *lbbb* is left bundle branch block, *rbbb* is right bundle branch block, and *veb* are ventricular ectopic beats.

In our example, hierarchies in Figure 3 are used by the diagnostic algorithm to refine the abstract level diagnosis *tachy*. The following dialog with the system illustrates the depth-first search for diagnoses through abstraction spaces. Each diagnosis returned by the system is followed by the corresponding medical term.

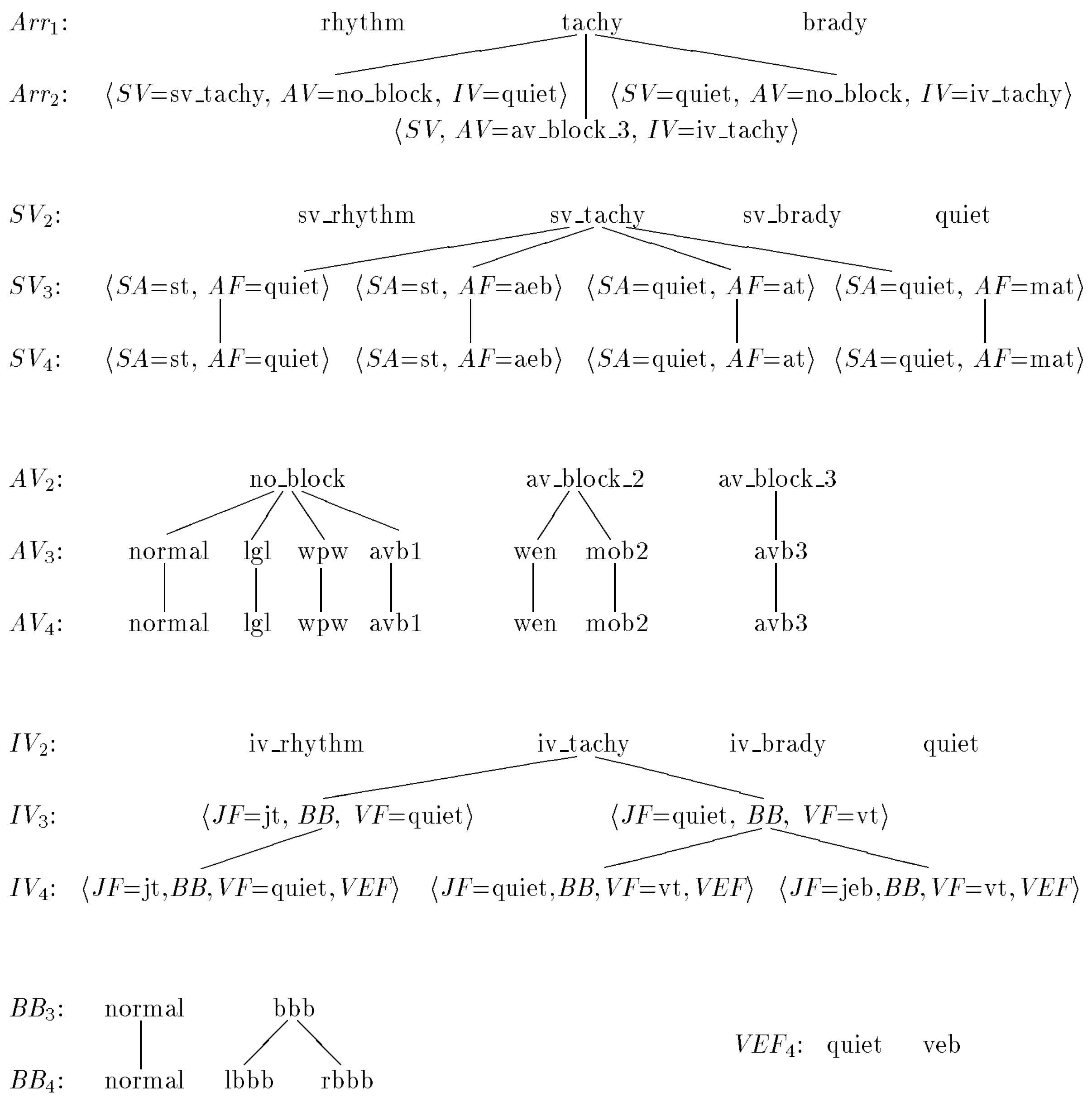
Arr_1: rhythm tachy brady
Arr_2: ⟨SV=sv_tachy, AV=no_block, IV=quiet⟩ ⟨SV=quiet, AV=no_block, IV=iv_tachy⟩
⟨SV, AV=av_block_3, IV=iv_tachy⟩
SV_2: sv_rhythm sv_tachy sv_brady quiet
SV_3: ⟨SA=st, AF=quiet⟩ ⟨SA=st, AF=aeb⟩ ⟨SA=quiet, AF=at⟩ ⟨SA=quiet, AF=mat⟩
SV_4: ⟨SA=st, AF=quiet⟩ ⟨SA=st, AF=aeb⟩ ⟨SA=quiet, AF=at⟩ ⟨SA=quiet, AF=mat⟩
AV_2: no_block av_block_2 av_block_3
AV_3: normal lgl wpw avb1 wen mob2 avb3
AV_4: normal lgl wpw avb1 wen mob2 avb3
IV_2: iv_rhythm iv_tachy iv_brady quiet
IV_3: ⟨JF=jt, BB, VF=quiet⟩ ⟨JF=quiet, BB, VF=vt⟩
IV_4: ⟨JF=jt,BB,VF=quiet,VEF⟩ ⟨JF=quiet,BB,VF=vt,VEF⟩ ⟨JF=jeb,BB,VF=vt,VEF⟩
BB_3: normal bbb
BB_4: normal lbbb rbbb
VEF_4: quiet veb

Figure 3: Some examples of the hierarchical relation h_A between the abstract and detailed level arrhythmias.

A possible diagnosis:

$\rangle$ $Arr_1 = tachy$
Tachycardia

More detailed diagnosis? *yes*

$\rangle\rangle$ $Arr_2 = \langle SV{=}sv\_tachy,\ AV{=}no\_block,\ IV{=}quiet\ \rangle$
Supra-ventricular tachycardia

More detailed diagnosis? *yes*

$\rangle\rangle\rangle$ $Arr_3 = \langle SA{=}quiet,\ AF{=}at,\ AV{=}normal,\ JF{=}quiet,\ BB{=}normal,\ VF{=}quiet\ \rangle$
Atrial tachycardia

More detailed diagnosis? *yes*

$\rangle\rangle\rangle\rangle$ $Arr_4 = \langle SA{=}quiet,\ AF{=}at,\ AV{=}normal,\ JF{=}jeb,\ BB{=}normal,\ VF{=}quiet,\ VEF{=}quiet\ \rangle$
Atrial tachycardia with junctional ectopic beats

Alternative diagnosis? *yes*

$\rangle\rangle\rangle$ $Arr_3 = \langle SA{=}quiet,\ AF{=}at,\ AV{=}lgl,\ JF{=}quiet,\ BB{=}normal,\ VF{=}quiet\ \rangle$
Atrial tachycardia with the LGL syndrome

More detailed diagnosis? *yes*

$\rangle\rangle\rangle\rangle$ $Arr_4 = \langle SA{=}quiet,\ AF{=}at,\ AV{=}lgl,\ JF{=}jeb,\ BB{=}normal,\ VF{=}quiet,\ VEF{=}quiet\ \rangle$
Atrial tachycardia with the LGL syndrome and junctional ectopic beats

Alternative diagnosis? *yes*

$\rangle\rangle$ $Arr_2 = \langle SV{=}quiet,\ AV{=}no\_block,\ IV{=}iv\_tachy\ \rangle$
Intra-ventricular tachycardia

More detailed diagnosis? *yes*

$\rangle\rangle\rangle$ $Arr_3 = \langle SA{=}quiet,\ AF{=}quiet,\ AV{=}normal,\ JF{=}jt,\ BB{=}normal,\ VF{=}quiet\ \rangle$
Junctional tachycardia

More detailed diagnosis? *yes*

$\rangle\rangle\rangle\rangle$ No consistent refinement !

$\rangle\rangle\rangle\rangle$ No more alternatives !

For the given detailed ECG pattern, there are two possible diagnoses: atrial tachycardia with junctional ectopic beats, and atrial tachycardia with the LGL syndrome and junctional ectopic beats. The first diagnosis appears to be more general than the second one, but for a physician it is important to be aware of both possibilities, since the second diagnosis is potentially more dangerous and might require a different treatment. Note that a diagnosis possible at the third level, junctional tachycardia, has several refinements at the fourth level, but none of them actually maps to the given ECG pattern.

2.4 Constraint propagation of relational dependencies

In this section we introduce CLP(gRel) and demonstrate its utility in diagnosis. CLP(gRel) is an instance of Constraint Logic Programming scheme [12] where syntactic unification is replaced by a more general constraint satisfaction. CLP(gRel) allows for an explicit manipulation of relational dependencies between variables and is a straightforward generalization of *domain variables* [23]. Domain variables range over finite sets of atomic values and as more constraints are imposed on the variables these sets become smaller. In CLP(gRel) variables range over sets of *g*round *tuples* of atomic values which represent *Rel*ations between constituent arguments. Constraints expressed by variables ranging over tuples can be used to actively prune the search space by detecting combinations of values that are bound to be unsatisfiable early (cf. *forward checking* [22]).

To motivate our discussion, we introduce the following example. These predicates are part of the detailed level heart model:

```
atrial_reg_projector( reg(_, none, zero), absent, zero ).
atrial_reg_projector( reg(SV, _, Rate), P, Prate )   ←
      non_zero( Rate ),
      supra_vent( SV, P, Rate, Prate ).

non_zero( under_60 ).
non_zero( between_60_100 ).
non_zero( over_100 ).

supra_vent( sa_node, normal, Rate, Rate ).
supra_vent( atr_focus, abnormal, Rate, Rate ).
supra_vent( wandering, changing, Rate, Rate ).
supra_vent( flutter, abnormal, Rate, Rate ).
supra_vent( fibflut, abnormal, Rate, Rate ).
supra_vent( faster, abnormal, Rate, Rate ).
supra_vent( slower, abnormal, Rate, Rate ).
supra_vent( equal, absent, _, zero ).
supra_vent( fibrill, absent, _, zero ).
```

Domain variables can be seen as a way of representing disjunction explicitly by means of a data-structure as opposed to encoding disjunction implicitly in two or more clauses for a predicate. So the above predicate *non_zero* can be rewritten to:

```
non_zero( X )   ←   X ∈ {under_60, between_60_100, over_100 }.
```

where $\in$ is a binary predicate specifying the set of legal values for a variable. The difference to the former representation is that the latter does not create a choice point during the search for a proof, and that it allows for an explicit reasoning about the disjunction. The straightforward generalization introduced in CLP(gRel) is to allow predicates of arbitrary arity to be reformulated in terms of such a data structure, instead of only unary predicates.

Assume an extended binary predicate $\in$ which relates an arbitrary number of variables — represented by a tuple — to the corresponding set of legal atomic values for the variables — represented as a set of tuples with the values as arguments. This allows the above *supra_vent* predicate to be rewritten as follows:

```
supra_vent( SV, P, Rate, Rate )   ←
      t(SV, P) ∈ { t(sa_node, normal),
                   t(atr_focus, abnormal),
                   t(wandering, changing),
                   t(flutter, abnormal),
                   t(fibflut, abnormal),
                   t(faster, abnormal),
                   t(slower, abnormal) }.
supra_vent( SV, absent, _, zero )   ←
      SV ∈ {equal, fibrill }.
```

All the original clauses, except the last two, are compressed into a single new clause. In this example the variables SV and P are constrained to a small set of tuples of possible atomic values. The extended unification algorithm (described below) ensures that when a variable, e.g. SV, gets bound, it is only bound to the first argument of one of the above tuples. Furthermore, extended unification immediately instantiates variable P accordingly, and vice versa. For example, if SV is bound to *sa_node* then P gets bound to *normal*. It is interesting to see what happens if P is bound to *abnormal*. Obviously SV cannot be bound to an atomic value as there are five different values still possible. However, extended unification assigns a new restricted domain {*atr_focus, faster, fibflut, flatter, slower*} to the variable SV. In relational database terms the set of tuples is called the table of rows which, in the above example, represents the relation *supra_vent* with columns labeled SV and P. Furthermore, we can describe a unification of constrained variables in terms of *select* and *join* relational operations.

Below we briefly sketch the procedural semantics of extended unification. The unification algorithm is extended to handle tuples of variables which are referred to as *r-variables*. Unification has to handle the following three cases:

- If a standard variable and an r-variable are unified, the standard variable is simply bound to the r-variable.
- Suppose a constant value and an r-variable have to be unified, where the r-variable is the i-th argument of the tuple of variables. The algorithm selects those *rows* of the set of legal solutions which have the constant value as the i-th argument. Depending on the number N of such rows there are three possible outcomes:
  - N = 0, unification fails,
  - N = 1, unification binds the r-variable to the constant value and all the other variables of the tuple to their corresponding values,

- N > 1, unification binds all the r-variables to newly created r-variables which are constrained to the new set of legal solutions.

- If two r-variables are unified the two sets of legal solutions are joined. Again, depending on the number N of resultant rows, there are three possibilities:
  - N = 0, unification fails,
  - N = 1, unification binds all the variables involved in the two tuples to constant values,
  - N > 1, unification binds all the r-variables of both tuples to newly created r-variables which are constrained to the join of the two sets of value-tuples.

This behavior of unification can be modeled by the following meta-interpreter. The interpreter handles unifications of atomic values, standard variables, and r-variables. An r-variable is represented by a term *attr(Var, N-dis(AllVars,AllRows))*, meaning *Var* is attributed by *N-dis(AllVars,AllRows)*. *N* specifies that *Var* is the *N*-th argument of the tuple *AllVars* which represents all the variables involved in a relation, and *AllRows* is a set of all legal solutions.

```
unify( X, Y )   ←   var(X), !, X = Y.
unify( X, Y )   ←   var(Y), !, X = Y.
unify( attr(V1, N1-dis(Vars1,Rows1)), attr(V2, N2-dis(Vars2,Rows2)) )   ←
        !, V1 = V2,
        join( N1, N2, Rows1, Rows2, NewRows ),
        join_bind( NewRows, N1, Vars1, N2, Vars2 ).
unify( attr(V, N-dis(Vars,Rows)), Atom )   ←
        !, V = Atom,
        select( Rows, N, Atom, NewRows ),
        select_bind( NewRows, N, Vars ).
unify( Atom, attr(V, N-dis(Vars,Rows)) )   ←
        !, V = Atom,
        select( Rows, N, Atom, NewRows ),
        select_bind( NewRows, N, Vars ).
unify( X, X ).
```

Since unification is such a basic operation in a logic programming environment, it cannot be delegated to a meta-interpreter if one expects reasonable efficiency. Further, a proper implementation of the CLP scheme should allow for an easy integration of specialized constraint propagation techniques into the logic programming framework. In our experiments we used a modified version of SICStus Prolog [3] which supports user-defined extensions of unification [11]. It allows for the specification and management of *attributed variables* and unification thereof. Both, the unification of two attributed variables, and of an attributed variable and an arbitrary term can be specified in Prolog.

Typically, constraint propagation is used to compute some form of local consistency, be it *node-*, *arc-*, or *path-(of some length) consistency* [10] plus backtrack search in the reduced solution space. For example, *forward-checking* as introduced in [22] essentially ensures arc-consistency. Except for [7] we are not aware of any experiments similar to ours. Our approach results in global consistency for given variables, is able to combine partial solutions dynamically, and thus interleaves backtrack search with consistency checks. This approach is of course not a panacea for all types of search problems. It is successful for the heart model since one needs all the solutions, and their number is typically small. This seems to prevent combinatorial explosion of the size of the intermediate data structures which represent the possible disjuncts. Additionally, the interaction of different constraints in the model is too complex to allow for a good pre-ordering of constraints which could be exploited by chronological backtracking.

3 SURFACE DIAGNOSTIC RULES

3.1 Derivation of a relational table

Another, *indirect* approach to use a deep model for efficient diagnosis is to 'compile' it. The 'compilation' proceeds in two steps. First, by exhaustive simulation, the model is transformed into a relational table. Entries in the table are then used as examples by an inductive learning program, and the table is compressed into a set of simple if-then rules.

The heart model m relates all arrhythmias Arr to all corresponding ECG patterns ECG, and through simulation one can generate a complete set of relations $\langle \mathrm{Arr, ECG} \rangle$:

$$m(Arr, ECG) \;\mapsto\; \langle A_1, \ldots, A_7, E_1, \ldots, E_{10} \rangle$$

Such a relational table can be used for efficient diagnosis, if not excessively large. In KARDIO, for example, a table generated from the original model of the heart consists of over 140,000 entries. When properly organized into a set of rules it still occupies over 5 Mb, 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 learning* 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). The same approach of constructing a qualitative model, exhaustive simulation, and induction of compressed diagnostic rules was taken by [19] to automatically construct a fault diagnosis system of a satellite power supply. Similarly, [2] shows the advantage of using a classical simulation model to generate a (non-exhaustive) set of learning and testing examples, which are then used to induce rules for location of errors in particle beam lines used in high energy physics.

3.2 Compression by inductive learning

In inductive learning (e.g., [13]), 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 is a pair

$$\langle object, class_i \rangle$$

where *object* is described by a tuple of attribute values $\langle v_1, \ldots, v_n \rangle$, and $class_i$ denotes an instance of the concept. The induced concept description is usually in the form of if-then rules:

$$\textbf{if}\ \ \lambda(v_1, \ldots, v_n)\ \ \textbf{then}\ \ \ class_i \qquad\qquad \text{or} \qquad\qquad \textbf{if}\ \ \ class_i\ \ \ \textbf{then}\ \ \lambda(v_1, \ldots, v_n)$$

where $\lambda(v_1, \ldots, v_n)$ is a boolean expression. The goal of learning is to find a logical expression λ for each $class_i$ which is as simple as possible, but sufficient to discriminate between the $class_i$ and all other classes $class_j,\ i \neq j$. It is worth emphasizing that in general, an if-then rule is not a logical implication, but rather a relation *if_then(*$class_i, \langle v_1, \ldots, v_n \rangle$*)*. The antecedent and the consequent of an if-then rule can be interchanged depending on the problem solving strategy since they merely indicate the direction of inference.

The inductive learning techniques were applied to the generated relational table. First, 10 sets of learning examples were prepared. For each 17-tuple relation in the table, 10 new 8-tuple relations were formed by projection:

$$\langle A_1, \ldots, A_7, E_k \rangle \ \leftarrow\ \langle A_1, \ldots, A_7, E_1, \ldots, E_{10} \rangle \qquad (k = 1, \ldots, 10)$$

Then an inductive learning program NEWGEM, an ancestor of AQ15 [14] was used. The result of learning were 10 sets of compressed diagnostic rules:

$$\begin{array}{c} \textbf{if}\ \ E_1\ \ \textbf{then}\ \ \lambda(A_1, \ldots, A_7) \\ \vdots \\ \textbf{if}\ \ E_{10}\ \ \textbf{then}\ \ \lambda(A_1, \ldots, A_7) \end{array}$$

A rule set k ($k = 1, \ldots, 10$) relates an individual ECG feature E_k to corresponding arrhythmias, described by $\lambda(A_1, \ldots, A_7)$. Each rule in a set k relates a value v_i of E_k to a minimal description of corresponding arrhythmias $\lambda(A_1, \ldots, A_7)$ which is still sufficient to discriminate between v_i and other values $v_j,\ i \neq j$ of E_k. For example, the following two rules belong to the ECG feature *QRS_complex* and discriminate between the values *wide_non_specific, normal*, and the remaining possible values.

if *QRS_complex* = *wide_non_specific*
then *VF* = *vr* ∨ *avr* ∨ *vt* ∨ *vfl*

if $QRS\_complex = normal$
then $AV \neq wpw \ \wedge$
$BB = normal \ \wedge$
$VF = quiet$

The first rule states that if a $QRS\_complex = wide\_non\_specific$ (i.e., *wide*, but not of type *LBBB* or *RBBB*) then there is either a ventricular rhythm (*vr*), an accelerated ventricular rhythm (*avr*), a ventricular tachycardia (*vt*), or a ventricular flutter (*vfl*) originating in a ventricular focus (*VF*). States of the remaining heart components are unspecified. On the other hand, if $QRS\_complex = normal$ then the ventricular focus must be *quiet*, the conduction through bundle branches (*BB*) is *normal*, and the atrio-ventricular conduction (*AV*) can be anything but the WPW syndrome ($\neq wpw$).

The following two rules are slightly more complicated:

if $P\_wave = abnormal$
then $SA = quiet \ \wedge$
$AF \neq wp \vee mat \ \wedge$
$VF \neq vfl \vee vf$

if $PR\_interval = shortened$
then $AF \neq afl \vee af \ \wedge$
$AV = wpw \vee lgl$
$\vee$
$SA = quiet \ \wedge$
$AF = at \vee aeb \vee quiet \ \wedge$
$AV = normal \ \wedge$
$VF = quiet$

Formally, an internal disjunction in the consequent of a rule is equivalent to a set membership, i.e., $A = v_1 \vee v_2 \ \Leftrightarrow \ A \in \{v_1, v_2\}$. The inequality is defined by $A \neq v_1 \vee v_2 \ \Leftrightarrow \ A \in Domain(A) - \{v_1, v_2\}$. Notice that the last rule, for the *PR_interval*, has a disjunctive consequent. In a logic program, such a rule is represented by two clauses:

$if\_then_5(\ shortened,\ \langle SA,\ AF,\ AV,\ JF,\ BB,\ VF,\ VEF\ \rangle) \ \leftarrow$
$\quad AF \in \{wp,\ at,\ mat,\ aeb,\ quiet\},$
$\quad AV \in \{wpw,\ lgl\}.$
$if\_then_5(\ shortened,\ \langle SA,\ AF,\ AV,\ JF,\ BB,\ VF,\ VEF\ \rangle) \ \leftarrow$
$\quad SA = quiet,$
$\quad AF \in \{at,\ aeb,\ quiet\},$
$\quad AV = normal,$
$\quad VF = quiet.$

Let us illustrate diagnostic reasoning with the compressed rules by an example. Suppose that given are values of three ECG features, $P\_wave = abnormal$, $PR\_interval = shortened$, and *QRS_complex* = *normal*. A combination of the three corresponding rules defined above yields:

$$\begin{array}{ll}
\textbf{if} & \mathit{P\_wave} = \mathit{abnormal} \ \wedge \ \mathit{PR\_interval} = \mathit{shortened} \ \wedge \ \mathit{QRS\_complex} = \mathit{normal} \\
\textbf{then} & SA = \mathit{quiet} \ \wedge \\
& AF = \mathit{at} \vee \mathit{aeb} \vee \mathit{quiet} \ \wedge \\
& AV = \mathit{lgl} \ \wedge \\
& BB = \mathit{normal} \ \wedge \\
& VF = \mathit{quiet} \\
& \vee \\
& SA = \mathit{quiet} \ \wedge \\
& AF = \mathit{at} \vee \mathit{aeb} \vee \mathit{quiet} \ \wedge \\
& AV = \mathit{normal} \ \wedge \\
& BB = \mathit{normal} \ \wedge \\
& VF = \mathit{quiet}
\end{array}$$

The set of possible diagnoses is now restricted and only two heart components, *JF* and *VEF* are still unconstrained. The following logic program specifies the diagnostic algorithm:

```
diag( ⟨E1, ...,E10⟩, Arr )   ←
      if_then1( E1, Arr ),
            ⋮
      if_then10( E10, Arr ),
      possible( Arr ).
```

A diagnosis is an intersection of the consequents of if-then rules for individual ECG features, filtered through the *possible* predicate which eliminates physiologically impossible and medically uninteresting arrhythmias. In an improved implementation, the algorithm could compute with domains of variables and not just their individual values [23].

The application of learning to the 10 sets of examples required 40 hours of CPU time on SUN 2 [15]. The compressed rules occupy 40 times less space than the relational table, and can be used for efficient diagnosis. The reduction is due to the generalization of arrhythmia descriptions in the process of learning. The equivalence to the relational table is regained by the application of *possible* at the end of the diagnostic algorithm. In general, however, a relational table and the corresponding set of compressed rules are not equivalent. The difference is due to the projection of relational table entries to learning examples. The conditions under which both, a relational table and compressed rules produce equivalent diagnostic results are stated in [1].

4 COMPARISON OF COMPLEXITY AND EFFICIENCY

Tables 1 and 2 outline the complexity of the hierarchical model of the heart at each level of detail. For comparison, the complexity of the KARDIO model is given when possible. Table 1 gives the number of components and the complexity of the arrhythmia descriptions. n denotes the arity of Arr tuples, $\mid A_1 \times \ldots \times A_n \mid$ is the number of syntactically possible descriptions, $\mid Arr \mid$ is the number of arrhythmias which satisfy the model constraints, and $\mid \neg Arr' \mid$ is the number of arrhythmias with no abstraction.

Level of	Model	Arrhythmias				
detail	components	Single	n	$\mid A_1 \times \ldots \times A_n \mid$	$\mid Arr \mid$	$\mid \neg Arr' \mid$
1	2	3	1	3	3	3
2	9	8	3	48	18	3
3	16	24	6	10,080	175	26
4	17	29	7	52,920	943	0
KARDIO	/	30	7	79,380	2,419	/

Table 1: The complexity of the heart model and the arrhythmia descriptions at different levels of detail, and in KARDIO.

In Table 2 the complexity of ECG descriptions and the relational table are given. m denotes the arity of ECG tuples, $\mid E_1 \times \ldots \times E_m \mid$ is the number of syntactically possible ECG patterns, $\mid ECG \mid$ is the number of distinct ECG patterns derived from the model, and $\mid \neg ECG' \mid$ is the number of ECG patterns with no abstraction. $\langle Arr, ECG \rangle$ is the number of entries in the relational table, and $\neg \langle Arr, ECG \rangle'$ is the number of entries with no abstraction.

Level of	ECG patterns				Relational table entries	
detail	m	$\mid E_1 \times \ldots \times E_m \mid$	$\mid ECG \mid$	$\mid \neg ECG' \mid$	$\langle Arr, ECG \rangle$	$\neg \langle Arr, ECG \rangle'$
1	1	3	3	3	3	3
2	4	64	12	0	23	5
3	7	41,472	263	6	333	79
4	10	3,386,880	3,096	0	5,240	0
KARDIO	7-19	/	/	/	140,966	/

Table 2: The complexity of ECG descriptions and the relational table at different levels of detail, and in KARDIO.

Tables 1 and 2 indicate that the heart model at levels 1 and 2 is incomplete with respect to levels 2 and 3, respectively. The level 3 model is complete with respect to level 4. Recall

Type of diagnostic knowledge	Space (Kb)	Time (msec)
One-level model	20	1060
Relational table	850	12
Compressed diagnostic rules	20	77
Constraints-based model	40	96
Hierarchical four-level model	50	98

Table 3: Space requirements for different representations and the average times needed to find all possible diagnoses for a given ECG pattern at the detailed level.

that in the case 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 [1] showed no considerable advantage of hierarchical diagnosis over the generate-and-test method, due exactly to the high level of incompleteness in the model. Consequently, the heart model at the level 2 was modified to decrease its incompleteness. Further, for all arrhythmias *Arr* without abstraction $(\neg Arr')$ the hierarchical diagnostic algorithm resorts to the corresponding relational table entries $\langle Arr, ECG\rangle$ in order to avoid the repetitive application of generate-and-test.

We compared the space complexity and diagnostic efficiency of the five types of diagnostic knowledge and their corresponding algorithms:

- one-level model of the heart with the naive generate-and-test,
- relational table indexed by an ECG feature,
- compressed diagnostic rules without domain variables,
- constraints-based model with inverted order of constraints and propagation of relational dependencies,
- hierarchical four-level model with hierarchical diagnosis and generate-and-test at each individual level; pairs $\langle Arr, ECG\rangle$ were pre-computed for each *Arr* without abstraction.

In all cases, knowledge bases and diagnostic algorithms are implemented and compiled by SICStus Prolog [3] and run on an Apollo DN5500 workstation. Complexity is measured by the 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 patterns at the detailed level. Results in Table 3 are the average times over 3096 ECGs.

With the one-level model of the heart, initially three constraint propagation strategies were applied: inverting the order of constraints, forward checking [22], and naive generate-and-test with chronological backtracking. Somehow surprisingly, the generate-and-test

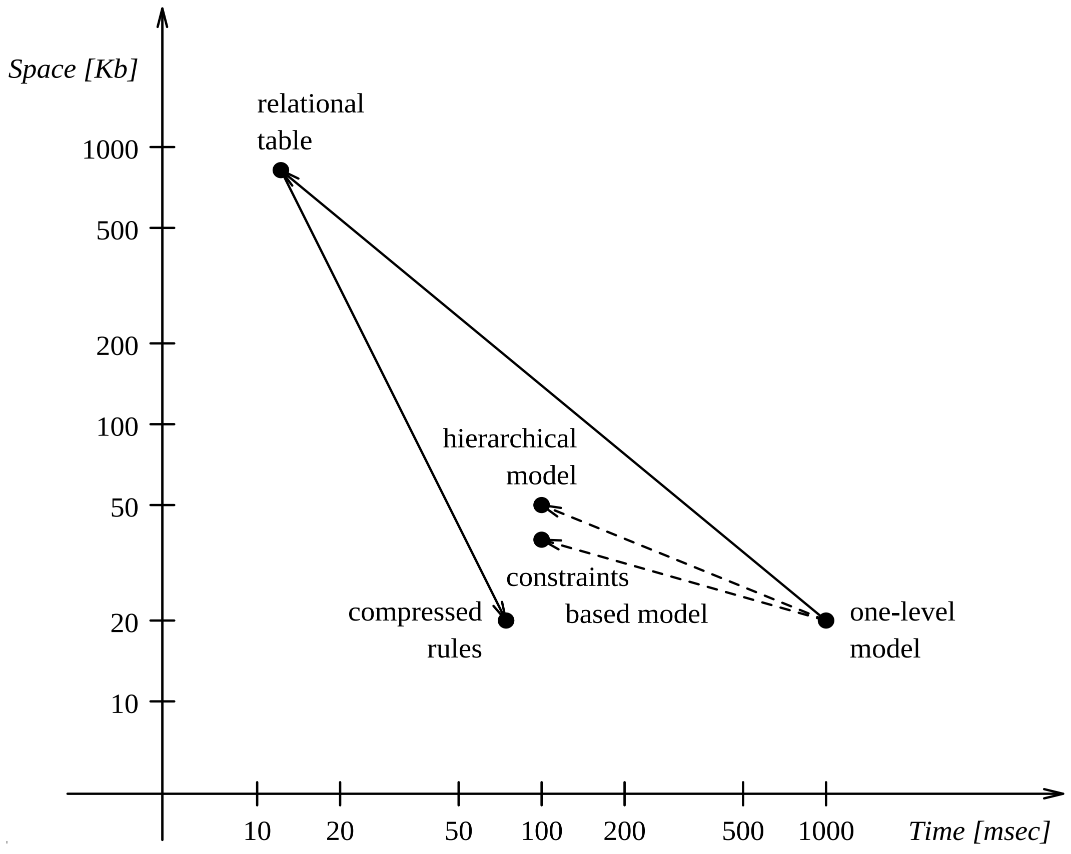

Figure 4: Space complexity and diagnostic efficiency of different model representations. Solid lines denote automatic model transformations while dashed lines denote semi-automatic transformations.

method turned out to be the most efficient. This is due to the simulation nature and high directionality bias of the model. When the model is used in the 'forward' direction, the average time to derive an ECG for a given Arr is only 3.8 milliseconds. In contrast, the model application in the 'backward' direction (from a given ECG to Arr) requires as much as 1060 milliseconds on the average. The application of forward checking was completely unsuccessful, probably due to the high arity of constraints (between 6 and 15 arguments) in the model definition. Only a more sophisticated constraint propagation of relational dependencies brought a considerable improvement in diagnostic efficiency.

The relational table representation is the most time efficient since only a simple retrieval is required, but, on the other hand, it is more space demanding. Compressed diagnostic rules are optimal in terms of space and time efficiency and currently appear to be the best representation for the ECG interpretation. Their diagnostic efficiency can be further improved by using domain variables [23] instead of the backtracking *member* predicate. Finally, hierarchical diagnosis with the four-level model is an order of magnitude faster than the one-level model, and requires only two times as much space (out of 50 Kb, 10 Kb are for the relational table entries without abstractions). An additional improvement in efficiency can be achieved by replacing the generate-and-test at each level by constraint propagation of relational dependencies.

The relation between different representations of diagnostic knowledge is better illustrated on a time/space diagram in Figure 4. Recall that the relational table and compressed diagnostic rules were automatically derived from the model. The hierarchical model was constructed semi-automatically on top of the one-level model, and the constraints-based model was in part derived by partial evaluation and in part manually from the original model.

5 CONCLUSION

We have presented three approaches to an efficient application of a deep simulation model to diagnosis. First, by abstracting the model, second, by efficient constraint propagation, and third, by compressing the model into a set of surface diagnostic rules. We have compared the deep and surface electrocardiographic knowledge representation in terms of space complexity and diagnostic efficiency. Compressed diagnostic rules are both space and time efficient, and can be derived automatically. Some of them are simple and have clear medical interpretation, but many are too complex to be easy to understand. In contrast to dedicated diagnostic rules, model-based reasoning offers better explanation facilities which can be even tuned to the desired level of detail. 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.

Acknowledgments

Thanks to Ivan Bratko and Nada Lavrač for their collaboration in the KARDIO project, to Mario Veitl, M.D., for his comments on the original paper, and to Robert Trappl for making some of this work possible. Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Research.

References

[1] Bratko, I., Mozetič, I. and Lavrač, N. (1989). *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems.* The MIT Press, Cambridge, MA.

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

[3] Carlsson, M., Widen, J. (1991). SICStus Prolog User's Manual. Swedish Institute of Computer Science, Kista, Sweden.

[4] Clocksin, W.F. and Mellish, C.S. (1984). *Programming in Prolog* (Second edition). Springer-Verlag, New York.

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

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

[7] Freuder, E. C. (1987). Synthesizing constraint expressions. *Communications of the ACM 21 (11).*

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

[9] Goldman, M.J. (1976). *Principles of Clinical Electrocardiography.* Lange Medical Publications, Los Altos, CA.

[10] Guesgen, H. W., Hertzberg, J. (1988). Some fundamental properties of local constraint propagation. *Artificial Intelligence 36*, pp. 237-247.

[11] Holzbaur, C. (1990). Specification of constraint based inference mechanisms through extended unification. Ph.D. thesis, Dept. of Medical Cybernetics and AI, University of Vienna, Austria.

[12] Jaffar, J., Lassez, J.-L., Mahler, J. (1986). A logic programming language scheme. In: D. de Groot, G. Linstrom (eds.), *Logic Programming: Functions, Relations, and Equations*, Prentice-Hall, Englewood Cliffs, NJ.

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

[14] Michalski, R.S., Mozetič, I., Hong, J. and Lavrač, N. (1986). The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proc. 6th Natl. Conference on Artificial Intelligence, AAAI-86*, Philadelphia, PA, Morgan Kaufmann, pp. 1041-1045.

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

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

[17] Mozetič, I. (1990). Reduction of diagnostic complexity through model abstractions. Report TR-90-10, Austrian Research Institute for Artificial Intelligence, Vienna, Austria. *Proc. First Intl. Workshop on Principles of Diagnosis*, Stanford University, CA, pp. 102-111.

[18] Mozetič, I. (1991). Hierarchical model-based diagnosis. *Intl. Journal of Man-Machine Studies 35 (3)*, pp. 329-362.

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

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

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

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

[23] Van Hentenryck, P., Dincbas, M. (1986). Domains in logic programming. *Proc. 5th Natl. Conference on Artificial Intelligence, AAAI-86*, Philadelphia, PA, Morgan Kaufmann, pp. 759-765.

Appendix — KARDIO availability

Major parts of the KARDIO system were recently re-implemented into three small, compatible subsystems which can be used independently of each other:

- kardio-dm.pl — deep model of the heart which simulates its electrical activity,
- kardio-h4.pl — hierarchical, four level model of the heart and hierarchical diagnostic algorithm,
- kardio-sd.pl — surface, compressed diagnostic rules.

These programs are written in standard Prolog (e.g., C-Prolog, Quintus Prolog, SICStus Prolog) and are available free of charge, on an "as is" basis. The interested users are asked to sign a standard, non-exclusive, non-transferable software license agreement. You can receive the agreement forms and copies of the relevant programs via electronic mail by contacting the first author.