10

Prototypes and Case-Based Reasoning for Medical Applications

R. Schmidt, T. Waligora, and O. Vorobieva

Institut für Medizinische Informatik und Biometrie, Universität Rostock

Summary. Already in the early stages of case-based reasoning (CBR) prototypes were considered as an interesting technique to structure the case base and to fill the knowledge gap between single cases and general knowledge. Unfortunately, later on prototypes never became a hot topic within the CBR community. However, for medical applications they have been used rather regularly, because they correspond to the reasoning of doctors in a natural way. In this chapter, we illustrate the role of prototypes by application programs, which cover all typical medical tasks: diagnosis, therapy, and course analysis.

10.1 Introduction

Cases are the most specialised form of knowledge representation. The knowledge of physicians consists of general knowledge they have read in medical books and of their experiences in form of cases they have treated themselves or colleagues have told them about. Not all cases are of the same importance. Some are typical while others are rather exceptional, e.g. a paediatrician does not remember all his patients with measles, but maybe those with serious complications or those where his measles diagnosis was surprisingly wrong. Doctors consider differences between their current patient and typical or known exceptional cases.

We believe that medical case-based reasoning (CBR) systems should take the reasoning of doctors into account [1]. Such systems should not only consist of general medical domain knowledge plus a flat case base, but the case base should be structured by typical case generalisations called prototypes [2].

Though the use of prototypes had been early introduced in the CBR community [3, 4], their use is still rather seldom. Later on it fell into oblivion and was brought up again by Bergman in form of generalised cases [5], which are similar but not identical with the idea of prototypes. While generalised cases are general or abstract in contrast to concrete cases, prototypes contain the typical features of a set of cases.

However, since doctors reason with typical cases anyway, in medical CBR systems prototypes are a rather common knowledge form, they are used in a variety of applications, e.g. for diabetes [6], for eating disorders [7], and for pulmonology [8]. Prototypical images that can be transformed after certain image processing steps in prototypes are used for the diagnosis of medical images [9].

A prototype is generalised from a set of single cases. The cases in this set are very similar to each other or they belong in some other specific way together and form a sort of class. For example, in a diagnostic system all patients that are diagnosed as measles patient might be grouped together. Usually, prototypes have the same structure as cases but have less and more general features, namely just the typical ones. Sometimes prototypes are defined by medical experts, sometimes they can be found in literature (e.g. the typical symptoms for measles), and sometimes they are computed.

The use of case-oriented generalised knowledge presents the opportunity to structure case bases. Cases can be clustered into groups, prototypical diseases, or schema. Clancey [10] distinguishes between prototypes that represent specific expressions of diseases or therapies and schema that contain essential features of diseases or therapies. As Selz [11] characterises a schema as a description of an entity where at least one part remains vague, the distinction between prototypes and schema seems to be fluid. We only use the term prototype and refer to a hierarchy of prototypes where the most general prototypes that contain the most common features are situated on top and the most specific ones are placed at the bottom.

This notion of prototypes differs from the usual notion of classes and clusters [12] in many ways. Prototypes are not the result of a classification process. Whether a case belongs to a prototype is determined by its features or defined by an expert. There may be a hierarchy of prototypes but there are not relations (similarity, is–a and so on), and the set of cases belonging to a prototype is not represented by its most representative case but by the prototype.

The main purpose of such generalised knowledge is to guide the retrieval and sometimes to decrease the amount of storage by erasing redundant cases. In domains with rather weak domain theories another advantage of case-oriented techniques is their ability to learn from cases. Only gathering new cases may improve the systems ability to find suitable similar cases for current problems, but it does not elicit the intrinsic knowledge of the stored cases. To learn the knowledge contained in cases a generalisation process is necessary. Generally speaking, prototypes fill the knowledge gap between the specificity of single cases and abstract knowledge usually expressed as rules.

In this chapter we present systems we developed during the last 10 years and focus on the role of prototypes within them. We start with a prototype-based system for diagnosis of dysmorphic syndromes. Subsequently we present a system for course analysis and prognosis of the kidney function and finally we present two therapeutic systems, namely one for antibiotic therapy advice and ISOR, a system that deals with therapeutic problems in the endocrine domain.

10.2 Prototype-Based Diagnosis of Dysmorphic Syndromes

In this application, retrieval does not search for former single cases but only for prototypes. Each prototype represents and characterises one specific diagnosis. We assume that this idea is rather typical for diagnostic tasks, because it seems to be reasonable to search for a general description of a disease instead of searching for single patients.

When a child is born with dysmorphic features or with multiple congenital malformations or if mental retardation is observed at a later stage, finding the correct diagnosis is extremely important. Knowledge of the nature and the etiology of the disease enables the pediatrician to predict the patient's future course. So, an initial goal for medical specialists is to diagnose a patient to a recognised syndrome. Genetic counselling and a course of treatments may then be established.

A dysmorphic syndrome describes a morphological disorder and it is characterised by a combination of various symptoms, which form a pattern of morphologic defects. An example is Down syndrome which can be described in terms of characteristic clinical and radiographic manifestations such as mental retardation, sloping forehead, a flat nose, short broad hands, and generally dwarfed physique [13]. The main problems of diagnosing dysmorphic syndromes are as follows [14]:

- More than 200 syndromes are known
- Many cases remain undiagnosed with respect to known syndromes
- Usually many symptoms are used to describe a case (between 40 and 130)
- Every dysmorphic syndrome is characterised by nearly as many symptoms

Furthermore, knowledge about dysmorphic disorders is continuously modified, new cases are observed that cannot be diagnosed (it exists even a journal that only publishes reports of newly observed interesting cases [15]), and sometimes even new syndromes are discovered. Usually, even experts of paediatric genetics only see a small count of dysmorphic syndromes during their lifetime.

So, we have developed a diagnostic system that uses a large case base. Starting point to build the case base was a large case collection of the paediatric genetics of the University of Munich, which consists of nearly 2,000 cases and 229 prototypes. A prototype (prototypical case) represents a dysmorphic syndrome by its typical symptoms. Most of the dysmorphic syndromes are already known and have been defined in literature. And nearly one-third of the prototypes were determined by semiautomatic knowledge acquisition, where an expert selected cases that should belong to same syndrome and subsequently a prototype, characterised by the most frequent symptoms of his cases, was generated. To this database we have added rare dysmorphic syndromes, namely from "clinical dysmorphology" [15] and from the London dysmorphic database [16].

10.2.1 Diagnostic Systems for Dysmorphic Syndromes

Systems to support diagnosis of dysmorphic syndromes have already been developed in the early 1980s. The simple ones perform just information retrieval for rare syndromes, namely the London dysmorphic database [16], where syndromes are described by symptoms, and the Australian POSSUM, where syndromes are visualised [17]. Diagnosis by classification is done in a system developed by Wiener and Anneren [18]. They use more than 200 syndromes as database and apply Bayesian probability to determine the most probable syndromes. Another diagnostic system, which uses data from the London dysmorphic database was developed by Evans [19]. Though he claims to apply CBR, in fact it is again just a classification, this time performed by Tversky's measure of dissimilarity [20]. The most interesting aspect of his approach is the use of weights for the symptoms. That means the symptoms are categorised in three groups – independent of the specific syndromes, instead only according to their intensity of expressing retardation or malformation. However, Evans admits that even features, that are usually unimportant or occur in very many syndromes sometimes play a vital role for discrimination between specific syndromes.

10.2.2 Prototypicality Measures

In CBR usually cases are represented as attribute-value pairs. In medicine, especially in diagnostic applications, this is not always the case, instead often a list of symptoms describes a patient's disease. Sometimes these lists can be very long, and often their lengths are not fixed but vary with the patient. For dysmorphic syndromes usually between 40 and 130 symptoms are used to characterise a patient.

Furthermore, for dysmorphic syndromes it is unreasonable to search for single similar patients (and of course none of the systems mentioned above does so) but for more general prototypes that contain the typical features of a syndrome. To determine the most similar prototype for a given query patient instead of a similarity measure a prototypicality measure is required. One speciality is that for prototypes the list of symptoms is usually much shorter than for single cases.

The result should not be just the one and only most similar prototype, but a list of them – sorted according to their similarity. So, the usual CBR retrieval methods like indexing or nearest neighbour search are inappropriate. Instead, rather old measures for dissimilarities between concepts [20, 21] are applied.

Since our system is still in the evaluation phase, the user has three choices for a prototypicality measure. As humans look upon cases as more typical for a query case as more features they have in common [21], distances between prototypes and cases usually mainly consider the shared features. The first, rather simple measure (10.1) just counts the number of matching symptoms of the query patient (X) and a prototype (Y) and normalises the result by dividing it by the number of symptoms characterising the syndrome. This normalisation is done, because the lengths of the lists of symptoms of the various prototypes vary very much. It is performed by the two other measures too.

The following equations are general (as they were originally proposed) at the point that a general function "f" is used, which usually means a sum that can be weighted. In general these functions "f" can be weighted differently. However, since we do not use any weights at all, in our application "f" means simply a sum.

$$\mathbf{D}(\mathbf{X},\mathbf{Y}) = \frac{\mathbf{f}(\mathbf{X}+\mathbf{Y})}{\mathbf{f}(\mathbf{Y})} \tag{10.1}$$

The second measure (10.2) was developed by Tversky [20]. It is a measure of dissimilarity for concepts. In contrast to the first measure, additionally two numbers are subtracted from the number of matching symptoms. Firstly, the number of symptoms that are observed for the patient but are not used to characterise the prototype (X−Y), and secondly the number of symptoms used for the prototype but are not observed for the patient (X − Y) is subtracted.

$$\mathbf{D}(\mathbf{X},\mathbf{Y}) = \frac{\mathbf{f}(\mathbf{X}+\mathbf{Y}) - \mathbf{f}(\mathbf{X}-\mathbf{Y}) - \mathbf{f}(\mathbf{Y}-\mathbf{X})}{\mathbf{f}(\mathbf{Y})} \tag{10.2}$$

The third prototypicality measure (10.3) was proposed by Rosch and Mervis [21]. It differs from Tversky's measure only in one point: the factor X–Y is not considered:

$$\mathbf{D}(\mathbf{X},\mathbf{Y}) = \frac{\mathbf{f}(\mathbf{X}+\mathbf{Y}) - \mathbf{f}(\mathbf{Y}-\mathbf{X})}{\mathbf{f}(\mathbf{Y})} \tag{10.3}$$

10.2.3 Our System

Our system consists of four steps (Fig. 10.1). At first the user has to select the symptoms that characterise a new patient. This selection is a long and very time consuming process, because we consider more than 800 symptoms. However, diagnosis of dysmorphic syndromes is not a task where the result is very urgent, but it usually requires thorough reasoning and subsequently a long-term therapy has to be started. Secondly, the user can select one of the prototypicality measures explained in Sect. 10.6. In routine use, this step shall be dropped and the measure with best evaluation results shall be used automatically.

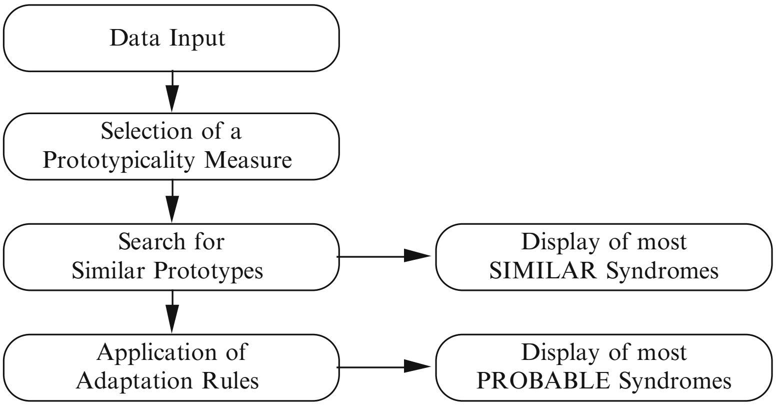
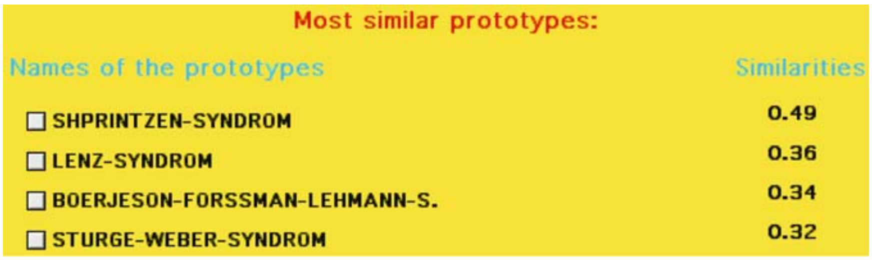

Data Input
Selection of a Prototypicality Measure
Search for Similar Prototypes
Display of most SIMILAR Syndromes
Application of Adaptation Rules
Display of most PROBABLE Syndromes

Fig. 10.1. Steps to diagnose dysmorphic syndromes



Most similar prototypes:

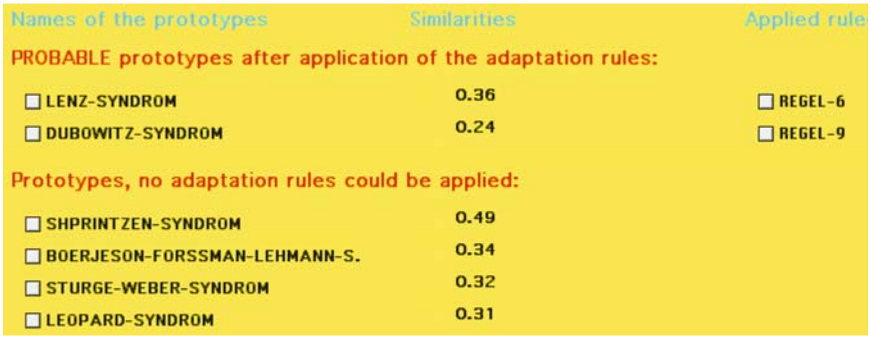
Names of the prototypes	Similarities
☐ SHPRINTZEN-SYNDROM	0.49
☐ LENZ-SYNDROM	0.36
☐ BOERJESON-FORSSMAN-LEHMANN-S.	0.34
☐ STURGE-WEBER-SYNDROM	0.32

Fig. 10.2. Top part of the listed prototypes after applying a prototypicality measure

In the third step to diagnose dysmorphoic syndromes, the chosen measure is sequentially applied on all prototypes (syndromes). Since the syndrome with maximal similarity is not always the right diagnosis, the 20 syndromes with best similarities are listed in a menu (Fig. 10.2).

In the fourth and final step, the user can optionally choose to apply adaptation rules on the syndromes. These rules state that specific combinations of symptoms favour or disfavour specific dysmorphic syndromes. Unfortunately, the acquisition of these adaptation rules is very difficult, because they cannot be found in textbooks but have to be defined by experts of paediatric genetics. So far, we have got only 18 of them and so far, it is not possible that a syndrome can be favoured by one adaptation rule and disfavoured by another one at the same time. When we, hopefully, acquire more rules such a situation should in principle be possible but would indicate some sort of inconsistency of the rule set.

How shall the adaptation rules alter the results? Our first idea was that the adaptation rules should increase or decrease the similarity scores for favoured and disfavoured syndromes. But the question is how. Of course no medical expert can determine values to manipulate the similarities by adaptation rules and any general value for favoured or disfavoured syndromes would be



Names of the prototypes	Similarities	Applied rule
PROBABLE prototypes after application of the adaptation rules:		
☐ LENZ-SYNDROM	0.36	☐ REGEL-6
☐ DUBOWITZ-SYNDROM	0.24	☐ REGEL-9
Prototypes, no adaptation rules could be applied:		
☐ SHPRINTZEN-SYNDROM	0.49	
☐ BOERJESON-FORSSMAN-LEHMANN-S.	0.34	
☐ STURGE-WEBER-SYNDROM	0.32	
☐ LEOPARD-SYNDROM	0.31	

Fig. 10.3. Top part of the listed prototypes after additionally applying adaptation rules

arbitrary. So, instead the result after applying adaptation rules is a menu that contains up to three lists (Fig. 10.3).

On top the favoured syndromes are depicted, then those neither favoured nor disfavoured, and at the bottom the disfavoured ones. Additionally, the user can get information about the specific rules that have been applied on a particular syndrome.

In the example presented in Figs. 10.2 and 10.3, the correct diagnosis is Lenz syndrome. The computation of the prototypicality measure of Rosch and Mervis provided Lenz syndrome as the most similar but one syndrome (here Tversky's measure provides a similar result, only the differences between the similarities are smaller). After application of adaptation rules, the ranking is not obvious. Two syndromes have been favoured, the more similar one is the right one. However, Dubowitz syndrome is favoured too (by a completely different rule), because a specific combination of symptoms makes it probable, while other observed symptoms indicate a rather low similarity.

10.2.4 Results

Cases are difficult to diagnose when patients suffer from a very rare dymorphic syndrome for which neither detailed information can be found in literature nor many cases are stored in our case base. This makes evaluation difficult. If test cases are randomly chosen, frequently observed cases resp. syndromes are frequently selected and the results will probably be fine, because these syndromes are well known. However, the main idea of the system is to support diagnosis of rare syndromes. So, we have chosen our test cases randomly but under the condition that every syndrome can be chosen only once. For 100 cases we have compared the results obtained by both prototypicality measures (Table 10.1).

The results may seem to be rather poor. However, diagnosis of dysmorphic syndromes is very difficult and usually needs further investigation, because often a couple of syndromes are very similar. The first step is to provide the

Table 10.1. Comparison of prototypicality measures

Right Syndrome	Rosch and Mervis	Tversky
on Top	29	40
among top 3	57	57
among top 10	76	69

Table 10.2. Results after applying adaptation rules

Right syndrome	Rosch and Mervis	Tversky
on Top	32	42
among top 3	59	59
among top 10	77	71

Table 10.3. Results after applying some more adaptation rules

Right Syndrome	Rosch and Mervis	Tversky
on Top	36	44
among top 3	65	64
among top 10	77	73

doctor with information about probable syndromes, so that he gets an idea which further investigations are appropriate. That means, the right diagnose among the three most probable syndromes is already a good result.

Obviously, the measure of Tversky provides better results, especially when the right syndrome should be on top of the list of probable syndromes. When it should be only among the first three of this list, both measures provide equal results.

Adaptation Rules

Since the acquisition of adaptation rules is a very difficult and time consuming process, the number of acquired rules is rather limited, namely at first just ten rules. Furthermore, again holds: The better a syndrome is known, the easier adaptation rules can be generated. So, the improvement mainly depends on the question how many syndromes involved by adaptation rules are among the test set. In our experiment this was the case only for five syndromes. Since some had been already diagnosed correctly without adaptation, there was just a small improvement (Table 10.2).

Some More Adaptation Rules

Later on we acquired eight further adaptation rules and repeated the tests with the same test cases. The new adaptation rules again improved the results (Table 10.3). It is obvious that with the number of acquired adaptation rules the quality of the program increases too. Unfortunately, the acquisition of these rules is very difficult and especially for very rare syndromes probably nearly impossible.

10.3 Time Course Prognosis

In this section we present a method for prognosis of temporal courses based on multiparametric numeric values for organ functions.

Since traditional time series techniques [22] work well with known periodicity, but do not fit in domains characterised by possibilities of abrupt changes, much research has been performed in the field of medical temporal course analysis. However, the methods developed so far either require a complete domain theory or well-known standards (e.g. course pattern or periodicity).

An ability of RÉSUMÉ [23] is the abstraction of many parameters into one single parameter and to analyse courses of this abstracted parameter. However, interpretation of these courses requires complete domain knowledge. Haimowitz and Kohane [24] compare many parameters of current courses with well-known standards (trend templates). In VIE-VENT [25] both ideas are combined: Courses of single quantitative measured parameters are abstracted into qualitative course descriptions that are matched with well-known standards.

When we started building a system for course analysis and prediction of the kidney function, we were confronted with a domain where the domain theory is extremely incomplete and no standards were known. So we had to design our own method. For temporal courses, our general idea is to search with CBR retrieval methods [8, 9] for former patients with similar courses and to consider their course continuations as possible prognosis for a query patient.

To make CBR applicable an appropriate case representation has to be found. Usually, a list of attribute-value pairs that contains all case attributes is sufficient. However, for multiparametric time courses the choice of suitable attributes is not obvious. Firstly, not complete courses (they may differ in their length, they may go much further back than it is relevant for the current situation), but only patients' current developments of a certain length should be compared with parts of former patients' courses, which should have about the same length. Secondly, each course consists of a sequence of measured or calculated parameter sets. It cannot be assumed that all parameters are of the same importance, especially the more recent parameter sets are usually more important than older ones.

And even the importance of parameters within the same set may extremely differ. One idea is to look for appropriate weightings for the parameters. However, hundreds of parameters might be involved, much domain knowledge may be required, and weights can be very subjective. Furthermore, it seems to be impossible to visualise a sequence of parameter sets in such a way that a user can rapidly discern the important characteristics.

In the kidney function domain, we chose a different alternative. With the help of medical experts we defined kidney function states based on the most important parameters and subsequently we abstracted each daily parameter set into such a function state. So, courses are represented as a sequence of function states.

10.3.1 Prognostic Model

We propose a prognostic model for multiparametric time courses that combines two abstraction steps with CBR (Fig. 10.4).

The first step is a state abstraction from a set of parameter values to a single function state. Therefore few requirements have to be met. Meaningful states to describe the parameter sets and a hierarchy of these states must exist. Furthermore, knowledge to define the states must be available. These definitions may consist of obligatory or optional conditions on the parameter values. Of course, all obligatory conditions should be met, while for the optional ones some alternatives exist how to determine the appropriate state. One simple idea is to count the met conditions. Additionally, the quality of meeting graduated conditions may be considered (e.g. fuzzy methods may be applied).

The second abstraction means to describe a course of states. An often-realised idea is to use different trend descriptions for different periods of time, e.g. short-term or long-term trend descriptions etc. (e.g. [25]).

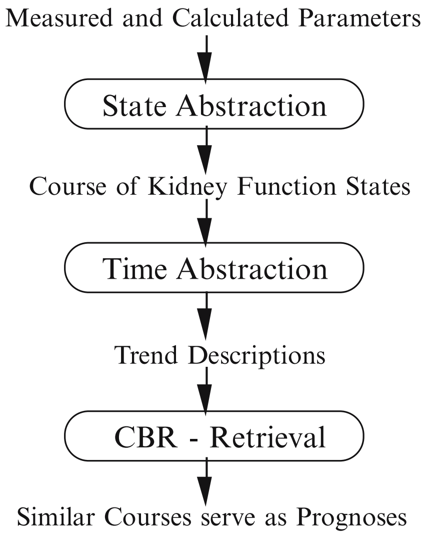

Measured and Calculated Parameters
State Abstraction
Course of Kidney Function States
Time Abstraction
Trend Descriptions
CBR - Retrieval
Similar Courses serve as Prognoses

Fig. 10.4. Prognostic model for time course

The lengths of each trend description can be fixed or they may depend on concrete values (e.g. successive equivalent states may be concatenated).

However the trend descriptions may be defined, they can be expressed by four parameters: the length, the first and last state, and an assessment. The lengths and the assessments of the descriptions can vary with domain-dependent demands, while the state definitions and their hierarchy are domain dependent anyway.

The third step means CBR retrieval. Since especially for large case bases a sequential search for similar cases is too time consuming, a few nonsequential retrieval algorithms have been developed in the CBR community. Most of the retrieval algorithms can handle various sorts of attributes, but usually they only work well for those sorts of attributes or problems they have been developed for. So, the choice of the retrieval algorithm should mainly depend on the sort of values of case attributes and sometimes additionally on application characteristics.

The question arises: Of which sort are the four parameters that describe a trend? The states are obviously nominal valued ordered according to their hierarchy. The assessments should have ordered nominal values too, e.g. steady, decreasing etc. Only the lengths should have numeric values. If the time points of the parameter measurements are few integers, they can be treated as ordered nominal values. The proposed retrieval algorithms for ordered nominal valued attributes are CBR-Retrieval-Nets [26], which are based on Spreading Activation [28]. So, if all four parameters have ordered nominal values, the choice of the retrieval algorithm should obviously be CBR-Retrieval-Nets.

However, we made some assumptions that may not necessarily be met in every domain. For example, the lengths may not be transformable into nominal values, the trend assessments may not be just simple nominal values, but more sophisticated descriptions, and there are of course alternatives to describe trends, e.g. even a computed real value might somehow express a trend.

10.3.2 Kidney Function Courses

Up to 60% of the body mass of an adult person consists of water. The electrolytes dissolved in body water are of great importance for an adequate cell function. The human body tends to balance the fluid and electrolyte situation. But intensive care patients are often no longer able to maintain adequate fluid and electrolyte balances themselves due to impaired organ functions, e.g. renal failure, or medical treatment, e.g. parenteral nutrition of mechanically ventilated patients. Therefore physicians need objective criteria for the monitoring of fluid and electrolyte balances and for choosing therapeutic interventions as necessary.

At our ICU, physicians daily get a printed renal report from the monitoring system NIMON [29] which consists of 13 measured and 33 calculated parameters of those patients where renal function monitoring is applied. For example, the urine osmolality and the plasma osmolality are measured parameters that are used to calculate the osmolar clearance and the osmolar excretion. The interpretation of all reported parameters is quite complex and needs special knowledge of the renal physiology.

The aim of our knowledge-based system ICONS is to give an automatic interpretation of the renal state to elicit impairments of the kidney function on time and to give early warnings against forthcoming kidney failures. That means, we need a time course analysis of many parameters without any well-defined standards.

However, in the domain of fluid and electrolyte balance, neither a prototypical approach in ICU settings is known nor exists complete knowledge about the kidney function. Especially, knowledge about the behaviour of the various parameters on time is yet incomplete. So, we combined the idea of RÉSUMÉ [23] to abstract many parameters into one single parameter with the idea of Haimowitz and Kohane [24] to compare many parameters of current courses with well-known standards. Since well-known standards were not available, we used former similar cases instead.

The method in ICONS corresponds to the general method proposed above and shown in Fig. 10.4.

State Abstraction

For the data abstraction we use states of the renal function, which determine states of increasing severity beginning with a normal renal function and ending with a renal failure. Based on the kidney function states (e.g. in Fig. 10.5 a reduced kidney function), characterised by obligatory and optional conditions for selected renal parameters, we first check the obligatory conditions. For each state that satisfies the obligatory conditions we calculate a similarity

Reduced Kidney Function

Obligatory Condtion:	c_kreat40 - 80	
Optional Conditions:		
Retention Rates:	p_kreat_se	<2
	p_urea_se	< 150
Tubular Function:	u_osmol320 – 600	
	u_p_osmol	1.1–1.8
	u_kreat 10 – 40	
	u_p_kreat	20–50
Urine Volume:	urine volume	0.7–3.0
	osmol_ex	800–3000

Fig. 10.5. Definition of the reduced kidney function state. Abbreviations: c, clearance; p, plasma; u, urine; kreat, kreatinin; osmol, osmolality; se, serum; ex, excretion

value concerning the optional conditions. We use a variation of Tversky's [20] measure of dissimilarity between concepts. Only if two or more states are under consideration, ICONS presents them to the user sorted according to their similarity values together with information about the satisfied and not satisfied optional conditions.

The user can accept or reject a presented state. When a suggested state has been rejected, ICONS selects another one. Finally, we determine the central state of occasionally more than one states the user has accepted. This central state is the closest one towards a kidney failure. Our intention is to find the state indicating the most profound impairment of the kidney function.

Temporal Abstraction

First, we have fixed five assessment definitions for the transition of the kidney function state of one day to the state of the, respectively, next day. These assessment definitions are related to the grade of renal impairment:

steady. both states have the same severity value.
increasing. exactly one severity step in the direction towards a normal function.
sharply increasing. at least two severity steps in the direction towards a normal function.
decreasing. exactly one severity step in the direction towards a kidney failure.
sharply decreasing. at least two severity steps in the direction towards a kidney failure.

These assessment definitions are used to determine the state transitions from one qualitative value to another. Based on these state transitions, we generate three trend descriptions. Two trend descriptions especially consider the current state transitions.

short-term trend:=	current state transition; Abbreviation: T1
medium-term trend:=	looks recursively back from the current state transition to the one before and unites them if they are both of the same direction or one of them has a "steady" assessment; Abbreviation: T2
long-term trend:=	characterises the considered course of at most seven days; Abbreviation: T3

For the long-term trend description we additionally introduced four new assessment definitions. If none of the five former assessments fits the complete considered course, we attempt to fit one of these four definitions in the following order:

alternating. at least two up and two down transitions and all local minima are equal.
oscillating. at least two up and two down transitions.

fluctuating. the distance of the highest to the lowest severity state value is greater than 1.

nearly steady. the distance of the highest to the lowest severity state value equals one.

Only if there are several courses with the same trend descriptions, we use a minor fourth trend description T4 to find the most similar among them. We assess the considered course by adding up the state transition values inversely weighted by the distances to the current day. Together with the current kidney function state these four trend descriptions form a course depiction, that abstracts the sequence of the kidney function states.

Looking back from a time point t, these four trend descriptions form a pattern of the immediate course history of the kidney function considering qualitative and quantitative assessments.

Why These Four Trend Descriptions?

There are domain specific reasons for defining the short-, medium-, and long-term trend descriptions T1, T2, and T3. If physicians evaluate courses of the kidney function, they consider at most one week prior to the current date. Earlier renal function states are irrelevant for the current situation of a patient. Most relevant information is derived from the current function state, the current development and sometimes a current development within a slightly longer time period. That means, very long-term trends are of no interest in this domain. In fact, very often only the current state transition or short continuous developments are crucial.

The short-term trend description T1 expresses the current development. For longer time periods, we have defined the medium- and long-term trend descriptions T2 and T3, because there are two different phenomena to discover and for each, a special technique is needed. T2 can be used for detecting a continuous trend independent of its length, because equal or steady state transitions are united recursively beginning with the current one. As the long-term trend description T3 describes a well-defined time period, it is especially useful for detecting fluctuating trends.

Since every abstraction loses some specific information, information about the daily kidney function states is lost in the second abstraction step. The course description contains only information about the current and the start states of the three trend descriptions. The intermediate states are abstracted into trend description assessments.

Example. The following kidney function states may be observed in this temporal sequence (Fig. 10.6):

selective tubular damage, reduced kidney function, reduced kidney function, selective tubular damage, reduced kidney function, reduced kidney function, sharply reduced kidney function

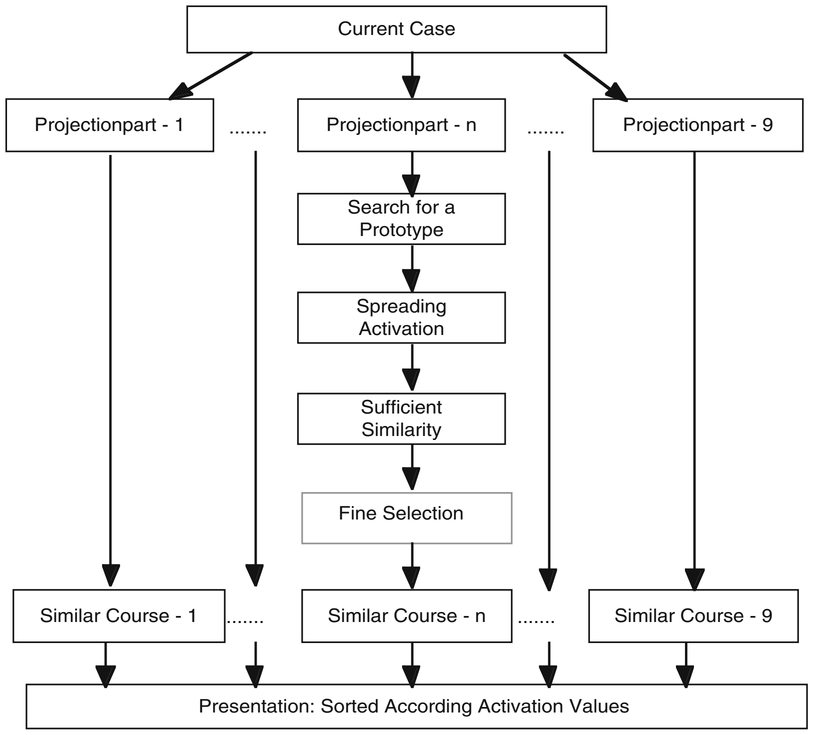


Fig. 10.6. Comparative presentation of a current and a similar course

So we get these six state transitions:
decreasing, steady, increasing, decreasing, steady, decreasing
with these trend descriptions:

current state: sharply reduced kidney function
T1: decreasing, reduced kidney function, one transition
T2: decreasing, selective tubular damage, three transitions
T3: fluctuating, selective tubular damage, six transitions
T4: 1.23

In this example, the short-term trend description T1 assesses the current state transition as "decreasing" from a "reduced kidney function" to a "sharply reduced kidney function." Since the medium-term trend description T2 accumulates steady state transitions, T2 determines a "decrease" in the last four days from a "selective tubular damage" to a "sharply reduced kidney function." The long-term trend description T3 assesses the entire course of seven days as "fluctuating," because there is only one increasing state transition and the difference between the severity values of a "selective tubular damage" and a "sharply reduced kidney function" equals two.

Retrieval

We use the parameters of the four trend descriptions and the current kidney function state to search for similar courses. As the aim is to develop an early warning system, we need a prognosis. For this reason and to avoid a sequential runtime search along the entire cases, we store a course of the previous seven days and a maximal projection of three days for each day a patient spent on the intensive care unit.

Since there are many different possible continuations for the same previous course, it is necessary to search for similar courses and for different projections. Therefore, we divided the search space into nine parts corresponding to the possible continuation directions. Each direction forms an own part of the search space. During the retrieval these parts are searched separately and each part may provide at most one similar case. The similar cases of these parts together are presented in the order of their computed similarity values.

Before the main retrieval, we search for a prototypical case (see Sect. 10.3.3) that matches most of the trend descriptions. Below this prototype the main retrieval starts (Fig. 10.7). It consists of two steps for each part. First we search with an activation algorithm concerning qualitative features.

Subsequently, we check the retrieved cases with a similarity criterion [27] that looks for sufficient similarity, because even the most similar course may differ from the current one significantly. This may happen at the beginning of the use of ICONS, when there are only a few cases known to ICONS, or when the query course is rather exceptional.

If two or more courses are selected in the same projection part, we use the sequential similarity measure of TSCALE [30], which goes back to Tversky [20], concerning the quantitative features in a second step.

Continuation of the example. For the example above, the following similar course (Fig. 10.6) with these transitions is retrieved:
decreasing, increasing, decreasing, steady, steady, decreasing
with these trend descriptions:

current state: sharply reduced kidney function
T1: decreasing, reduced kidney function, one transition
T2: decreasing, selective tubular damage, four transitions
T3: fluctuating, selective tubular damage, six transitions
T4: 1.17

In the lower part of each course the (abbreviated) kidney function states are depicted. The upper part of each course shows the deduced trend descriptions.

T1 describes a "decrease" from a "reduced kidney function" and T2 describes a "decrease" from a "selective tubular damage" to a "sharply reduced kidney function" in the last five days. T3 assesses the considered course as "fluctuating." For T4, a slightly lower value in comparison to the current

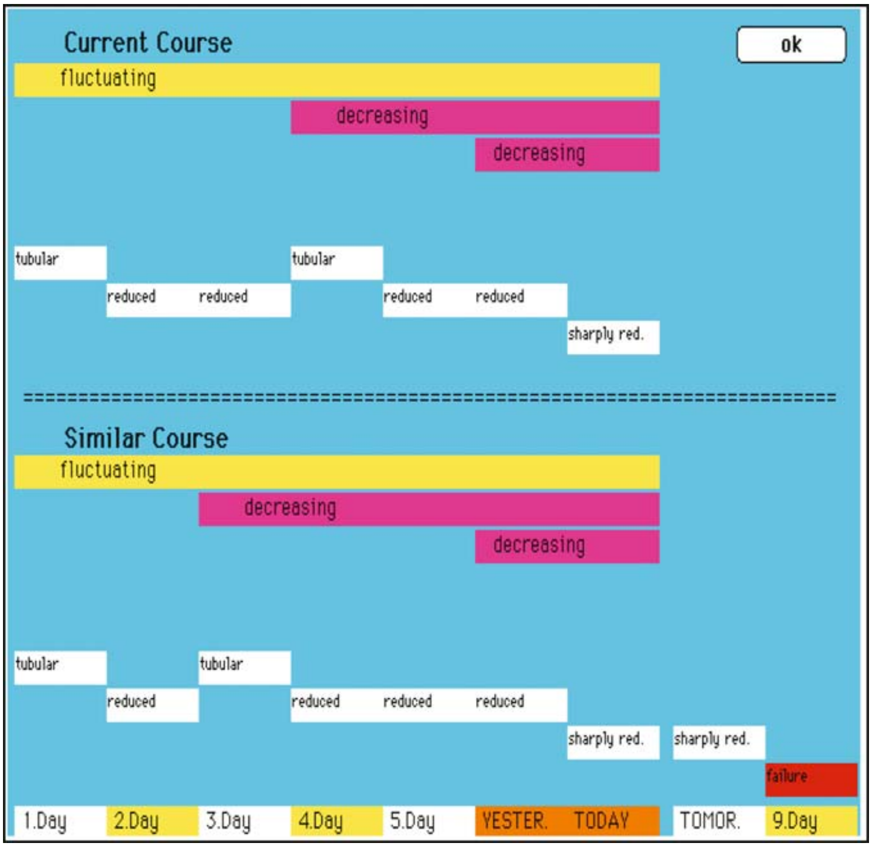


Fig. 10.7. The retrieval procedure

course has been calculated, because the change from a "selective tubular damage" to a "reduced kidney function" state occurs earlier.

After another day with a "sharply reduced kidney function" the patient belonging to the similar course had a kidney failure. The physician may notice this as a warning and it is up to him to interpret it.

This former course was retrieved, because especially the features with the highest weights (the current state and all assessments) equal the features of the query course. As there is no significant difference between both courses, there is no reason for the sufficient similarity criterion to reject this similar course.

10.3.3 Learning a Tree of Prototypes

Prognosis of multiparametric courses of the kidney function for ICU patients is a domain without a medical theory. Moreover, we cannot expect such a theory

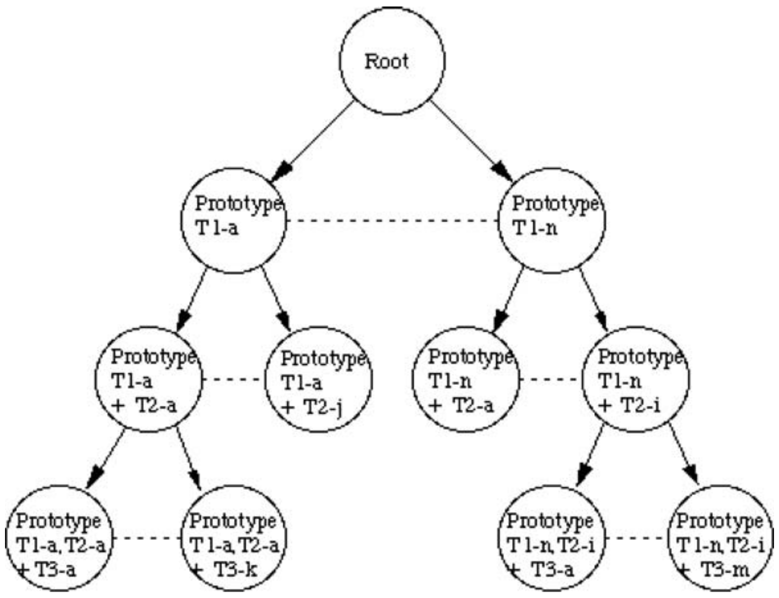


Fig. 10.8. Prototype architecture for the trend descriptions T1, T2, and T3

to be formulated in the near future. So we attempt to learn prototypical course pattern. Therefore, knowledge on this domain is stored as a tree of generalised cases (prototypes) with three levels and a root node (Fig. 10.8).

Except for the root, where all not yet clustered courses are stored, each level corresponds to one of the trend descriptions T1, T2, or T3. As soon as enough courses that share another trend description are stored at a prototype, a new prototype with this trend is created. At a prototype at level 1, we cluster courses that share T1, at level 2, courses that share T1 and T2 and at level 3, courses that share all three trend descriptions T3.

We can do this, because regarding their importance, the short-, medium-, and long-term trend descriptions T1, T2, and T3 refer to hierarchically related time periods. T1 is more important than T2 and T3, and so forth.

We start the retrieval with a search for a prototype that has most of the trend descriptions in common with the query course. The search begins at the root node with a check for a prototype with the same short-term trend description T1. If such a prototype can be found, the search goes on below this prototype for a prototype that has the same trend descriptions T1 and T2, and so forth. If no prototype with a further trend in common can be found, we search for a course at the last accepted prototype.

If no prototype exists that has the same T1 as the query course, we search at the root node, where links to all courses in the case base exist.

Continuation of the example. In the example above, we can create just one prototype at level 1, because at the second level the query course and the similar one, called “similar-1” differ in their length. Although the long-term trend description T3 is equal for both courses, we cannot create a prototype at level 3 because of the strictly hierarchical organisation of the prototype tree. However, learning a prototypical description “fluctuating in seven days from a selective tubular damage to sharply reduced kidney function” which does not consider any more similarities or deviations within this time period would be too general and therefore too impracticable.

Assuming we find another similar course, called “similar-2”, for the current case of the example above with the following kidney function states:

reduced kidney function, reduced kidney function, selective tubular damage, selective tubular damage, reduced kidney function, reduced kidney function, sharply reduced kidney function with these trend descriptions:

current state: sharply reduced kidney function
T1: decreasing, reduced kidney function, one transition
T2: decreasing, selective tubular damage, four transitions
T3: oscillating, reduced kidney function, six transitions
T4: 1.33

The current query course, “similar-1”, and “similar-2” will be clustered at level 1 to prototype T1-a, defined by T1 as “decreasing, reduced kidney function, one transition”. Afterwards at level 2 the current course and “similar-2” will be clustered to a prototype T1-a + T2-a, defined by T1 as “decreasing, reduced kidney function, one transition” plus by T2 as “decreasing, selective tubular damage, four transitions.” The attempt to create another prototype at level 3 fails, because the trend descriptions T3 have different assessments and different start states. The result, a tree of prototypes learned from the three courses is shown in Fig. 10.9.

10.3.4 Retrieval Experiments

Since we wished to be convinced that CBR-Retrieval-Nets really are the appropriate retrieval algorithm for our prognostic model, we compared them with an indexing algorithm, which had been developed for nonordered nominal values [31]. The results of this comparison are as follows: The indexing algorithm works faster (Table 10.4), but provides worse results, because stored cases get only activation values for attribute values that exactly match the query case values. The CBR-Retrieval-Nets additionally send smaller activation values to cases with attribute values similar to query case values. Hence, courses can be determined to be similar which have attribute values that slightly deviate from the query case values.

Since one idea of using prototypes is to speed up the retrieval by structuring the case base, we additionally compared both algorithms with and without using prototypes. To decide when a prototype should be generated, a threshold parameter is required. We set this parameter to the value of 2, which

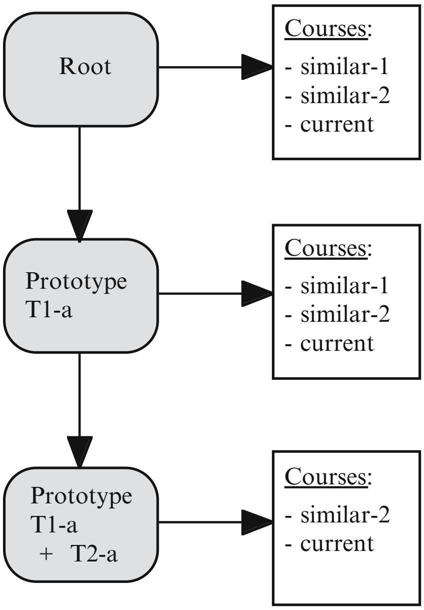


Fig. 10.9. Generated prototype tree from three example courses

Table 10.4. Retrieval times (in seconds) for the retrieval algorithms "CBR-Retrieval-Nets" and "indexing" with and without using prototypes

Courses	Retrieval nets	Retrieval nets, Use of prototypes	Indexing	Indexing, use of prototypes
No. 1	0.163	0.163	0.155	0.155
No. 2	0.284	0.281	0.214	0.218
No. 3	0.316	0.366	0.165	0.213
No. 4	0.455	0.513	0.404	0.452
No. 5	0.514	0.544	0.428	0.506
No. 6	1.328	0.759	0.600	0.717
No. 7	0.649	0.401	0.246	0.347
No. 8	0.685	0.642	0.376	0.469
No. 9	0.550	0.617	0.444	0.551
No. 10	0.386	0.476	0.257	0.394
No. 11	0.537	0.553	0.234	0.367
No. 12	1.396	0.870	0.743	0.890
No. 13	0.577	0.607	0.244	0.332
No. 14	0.518	0.425	0.340	0.494

means, that already two cases with the same trend description are sufficient to generate a prototype. Hence many prototypes were generated.

At first glance the results (Table 10.4) are not very encouraging for using prototypes. However, for the CBR-Retrieval-Nets the time differences between with and without prototypes are very small except for those two courses where the retrieval worked noticeably slower (No.6 and No.12): Here, using prototypes reduces the retrieval by at least a third.

However, so far the determination of the appropriate prototype occurred by sequentially matching the trend description parameters. So, most of the time gained by reducing the number of cases worth to consider is used up to determine the appropriate prototype. This indicates that not only the retrieval algorithm for cases, but also the determination of appropriate prototypes should be organised in a nonsequential way.

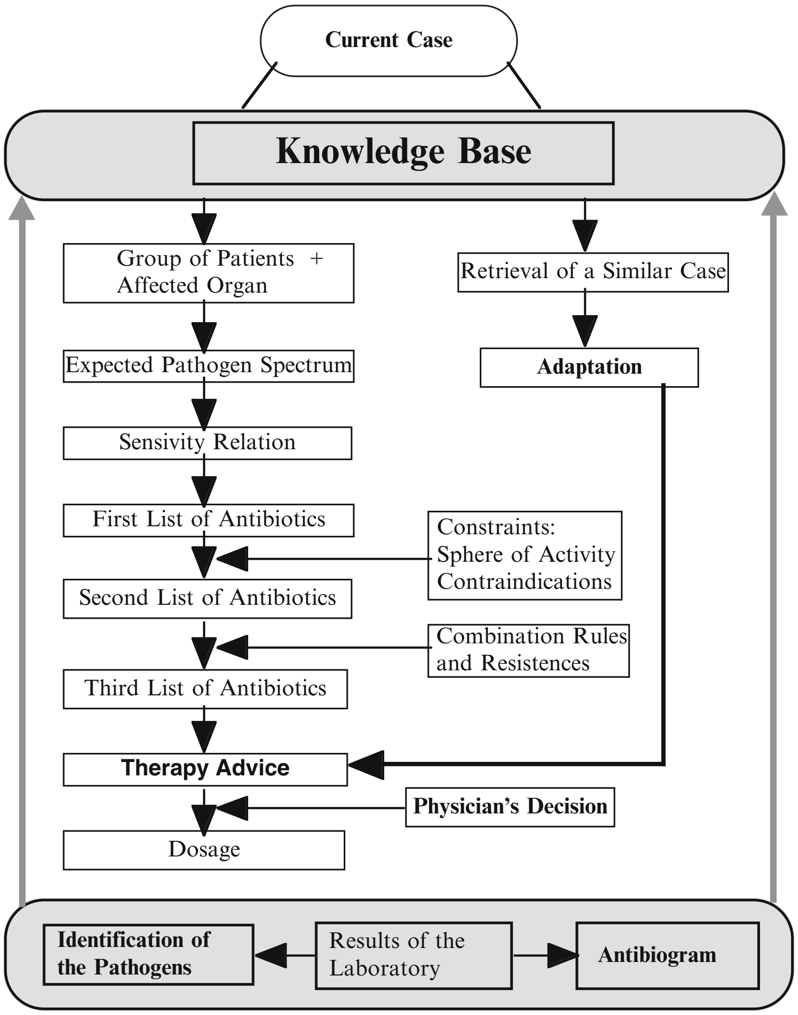
10.4 Antibiotics Therapy Advice

We developed an antibiotics therapy advice system called ICONS for patients in an intensive care unit who have caught an infection as additional complication. To speed up the process of finding suitable therapy recommendations, we applied CBR techniques. As information about antibiotics therapy changes in time, new cases are incrementally incorporated into the case base and outdated ones are updated or erased.

10.4.1 Antibiotics Therapy

Severe bacterial infections are still a life-threatening complication in intensive care medicine, correlating with a high mortality [32]. Identification of bacterial pathogens is often difficult. It usually requires at least 24 hours to identify the pathogen that is responsible for an infection and at least another 24 hours to find out which antibiotics have therapeutic effects against the identified pathogen. In order not to endanger the patient, physicians sometimes have to start an antimicrobial therapy before the responsible pathogen and its sensitivities are determined. This sort of antibiotic therapy is called "calculated," in contrast to a "selective" therapy, which is used when microbiological results are already available. For an adequate calculated antibiotic therapy, it is essential to access information about the expected pathogen spectrum and its expected susceptibility, existing contraindications, and the side effects of antibiotics.

The main task of our adviser is to present suitable calculated antibiotics therapy advice for intensive care patients who have caught a bacterial infection as an additional complication. Since, for such critical patients, physicians cannot wait for the laboratory results, we use an expected pathogen spectrum based on medical background knowledge. Each recommended antibiotics therapy should completely cover this spectrum. Furthermore, as advice is needed very quickly we speed up the process of computing recommended antibiotic therapies by using CBR methods (the right path in Fig. 10.10). This means that we search for a previous similar patient and transfer the therapies



Current Case
Knowledge Base
Group of Patients + Affected Organ
Expected Pathogen Spectrum
Sensivity Relation
First List of Antibiotics
Second List of Antibiotics
Third List of Antibiotics
Therapy Advice
Dosage
Retrieval of a Similar Case
Adaptation
Constraints: Sphere of Activity Contraindications
Combination Rules and Resistences
Physician's Decision
Identification of the Pathogens
Results of the Laboratory
Antibiogram

Fig. 10.10. Program flow in ICONS

suggested for his situation to the current patient. These previous therapies are then adapted to take account of any differences between the situations of the previous and current patients.

10.4.2 Strategy for Selecting Recommended Antibiotic Therapies

As ICONS is not a diagnostic system, we do not attempt to deduce evidence for diagnoses based on symptoms, frequencies, or probabilities, but instead pursue a strategy that can be characterised as follows: find all possible solutions, and subsequently reduce them using the patient's contraindications and the requirement to completely cover the calculated pathogen spectrum (establish-refine strategy).

Firstly, we distinguish between different groups of patients (infection acquired in or outside the ward, respectively, the hospital; immunocompromised patients). An initial list of antibiotics is generated by a susceptibility relation, which for each group of pathogens provides all antibiotics that usually have therapeutic effects. This list contains all antibiotics that cover at least a part of the potential pathogen spectrum. We obtain a second list of antibiotics by reducing the first one through applying two constraints: the patient's contraindications and the desired sphere of activity. Using the antibiotics on this second list, we try to find antibiotics that cover the whole pathogen spectrum individually.

Except for some community-acquired infections, monotherapies have to be combined with antibiotics that have synergistic or additive effects. If no adequate single therapy can be found, we use combination rules to generate combinations of antibiotics. Each possible combination must be tested for the ability to cover the expected spectrum completely.

10.4.3 Case-Based Reasoning

In this application, the main argument for using CBR methods is to speed up the process of finding adequate therapies. We shorten the strategy described above for selecting recommended antibiotic therapies by searching for a similar case, retrieving its suggested therapies, and by adapting them according to the contraindications of the current patient.

The retrieval consists of three steps. Firstly we select the part of the case base in which all cases share the following two attributes with the current patient: the group of patients, and the infected organ system. This means a selection of the appropriate prototype tree. Subsequently, we apply the tree-hash retrieval algorithm of Stottler, Henke, and King [33] for nominal valued contraindications and the similarity measure of Tversky [20] for the few integer valued contraindications. Furthermore, we use an adaptability criterion, because not every case is adaptable [34]. The attributes used for the retrieval are the contraindications, which work as constraints on the set of possible antibiotics. It is therefore obvious that we should use only former cases whose contraindications are shared by the current patient. To guarantee this condition the adaptability criterion has to be checked during retrieval.

In ICONS three different sorts of adaptations occur: A CBR adaptation to obtain sets of calculated advisable therapies for current patients, an adaptation of chosen therapies according to laboratory findings, and a periodical update of laboratory information (resistance situation, frequently observed pathogens).

Each contraindication restricts the set of advisable therapies. During the retrieval we require that the retrieved case does not have any additional contraindications besides those of the current case. Otherwise the solution set for the current case would be inadmissibly reduced by the additional contraindications of a previous case.

Because of this criterion, the adaptation of a previous similar case is rather simple. It is simply a matter of transferring the set of advisable therapies and if necessary of reducing this set according to the additional contraindications of the current case.

10.4.4 Prototypes

Since in an incrementally working system the number of cases increases continuously, storing each case would slow down the retrieval time and exceed any space limitations. We therefore decided to structure the case base by prototypes and to store only those cases that differ significantly from their prototype. Like for diagnosis (see Sect. 10.2), we create prototypes that include the most frequent features of the corresponding cases. In diagnostic applications, prototypes correspond to typical diseases or diagnoses. So, for antibiotic therapies, prototypes are expected to correspond to typical antibiotic treatments associated with typical clinical features of patients. However, as the attributes are contraindications that are responsible not for the generation, but for the restriction of the solution set, this is only partly true.

We investigated the growth of a hierarchical prototype structure built up from a randomly ordered stream of cases. The results are presented and discussed in Sect. 10.4.5.

Selection of a Prototype Tree

In ICONS there is not just one prototype tree, but a forest of trees, which are all independent from each other. A specific tree can be generated for each affected organ system combined with each group of patients. So, for nearly 20 organ systems and five patient groups there are nearly 100 possible prototype trees. We generate them dynamically only when required. For example a tree for “community-acquired kidney infections” will be generated as soon as the first data input occurs from a patient who has a kidney infection which he has acquired outside the hospital.

Since all cases within the same prototype tree belong to the same group of patients, and the same organ system is affected, it follows that the same expected pathogen spectrum deduced from background knowledge has to be covered. Cases within the same prototype tree are only discriminated from each other by their contraindications. These are allergies against specific antibiotics, reduced organ functions (kidney and liver), specific diagnoses (e.g. CNS disease), special blood diseases, pregnancy, and the patient’s age.

Generating Prototypes

The aim of our concept of prototypical cases is to structure the case base, to keep the prototypes always up to date, and to erase redundant cases. As the prototypes are generated incrementally and as they should always contain the typical features of their cases, we use two threshold parameters:

(1) The parameter "minimum frequency" determines how (relatively) often a contraindication has to occur in the set of cases to be incorporated into the prototype.
(2) The parameter "number of cases" determines the required number of cases that are necessary to fill a prototype or to create an alternative prototype. The lower this threshold the more prototypes are created and the fewer cases are stored.

First, all cases are stored below the prototype they belong to. If the threshold "number of cases" is reached after storing a new case below a prototype, the prototype will be "filled". At this point, every contraindication that occurs in the prototype's cases at least as often as the "minimum frequency" threshold will be included into the prototype. Subsequently, the "filled" prototype can be treated like a case. The same holds for prototypes as for cases: Each contraindication restricts the set of advisable therapies. The contraindications of a prototype are those that occur most often within its cases. So from the viewpoint of frequency they are the typical ones. Those cases that have no additional contraindications in comparison with their prototypes are erased.

When new cases are added later on to an already filled prototype, the observed frequencies may change and consequently the contraindications of the prototype may have to be recomputed. If the contraindications of a prototype change, the suggested antibiotic therapies have to be recomputed, too. In addition, all cases must be inspected again to determine whether they need to be stored.

We create an "alternative" prototype below an already existing prototype if for the latter enough cases exist (which means the threshold "number of cases" is reached) that have at least one contraindication in common, which the already existing prototype does not include. We generate the alternative prototype using those cases that share at least one contraindication not included in the existing prototype. We place this new prototype in the hierarchy directly below the already existing prototype. Alternative prototypes differ from their superior prototypes by their contraindications and therefore also by their sets of advisable antibiotic therapies.

10.4.5 Experimental Results with Prototype Generation Strategies

The general idea of our concept is to keep the prototypes always up to date. They should contain the typical features of their cases. We have tested two contrasting policies for deleting redundant cases and a strategy of keeping all cases. Our evaluation had two aims. First, we wished to find a strategy that best fits the two contrasting aims of finding many adaptable cases or prototypes and requiring little memory. Secondly, we wished to find good settings for the threshold parameters.

Normally, cases without additional features in comparison to their prototype are redundant, because they do not contain any additional information [31]. However, in our application the attributes are contraindications, which are not used to generate a solution, but to restrict a solution set. This means they are applied as constraints. A case with fewer contraindications than its prototype has a greater chance of being adaptable to a query case, because only a case without additional contraindications in comparison to the query case is adaptable.

We have therefore tested two opposing strategies: firstly, deleting cases without additional attributes, and secondly, deleting only cases with additional attributes. Additionally, we have tested a strategy without deleting any cases at all.

The memory size without any stored cases is about 2.248 MB for all three strategies. The argument about the memory might seem to be unreasonable, because the differences between the strategies are only about 40 KB for 75 test cases. However, we performed our tests in just one of about 100 possible parallel sets. 75 cases in each set might lead to differences of up to 4 MB. This leads to the question of whether our system should require about 12 or about 16 MB memory. Certainly, problems should not occur until the number of cases per set exceeds 75.

Without generating any prototypes at all, for 51 of the 75 test cases a similar adaptable case can be found. As prototypes are treated like cases, this number can be exceeded.

Strategy A: Deleting Cases Without Additional Attributes

We have tested the strategies with 75 cases, which were incrementally incorporated into the system. For strategy A, we varied the threshold parameter "number of cases," which indicates how many cases are necessary to generate a prototype. The second threshold parameter "relative frequency" was set to 33%, which means that a contraindication is incorporated into a prototype if at least a third of its cases have this contraindication.

The results (Table 10.5) can be summarised as follows: The more cases necessary to generate a prototype (this is achieved by increasing the value "number of cases") the higher the number of stored cases and the higher the number of retrieved adaptable cases. After a while there is only little to be gained by increasing this threshold parameter any further (fourth setting). A surprise is the big increase in the number of retrieved adaptable cases in the second setting compared with the first one. This cannot be simply explained by the four additionally stored cases, but by the following two phenomena. Firstly, those cases that have no additional information (contraindications) in comparison to their prototype are deleted. This means that the deleted cases would be more likely to be adaptable to future queries.

Secondly, under the second setting the prototypes are generated later and consequently cases are deleted later as well.

Table 10.5. Test results for strategy A

	1. Setting number of cases = 2	2. Setting number of cases = 3	3. Setting number of cases = 4	4. Setting number of cases = 5
memory size	in Mbytes	in Mbytes	in Mbytes	in Mbytes
after 75 cases	2.392	2.390	2.401	2.402
number of prototypes	9	7	8	8
number of stored cases	53	57	62	63
number of deleted cases	22	18	13	12
number of adaptations	12	26	31	31

Table 10.6. Test results for strategy B

	1. Parameter setting: relative frequency = 33 %	2. Parameter setting relative frequency = 25 %
memory size after 75 cases	2.373 MB	2.368 MB
number of generated prototypes	9	6
number of stored cases	20	25
number of deleted cases	55	50
number of adaptations	14	14

One aim of using prototypes is the hope of reducing the memory size. For strategy A this benefit does not occur, because the storage requirement for prototypes is bigger than for cases. This is because prototypes contain some additional information: The intersection of advisable therapies for their cases (cases only contain additional specific therapy suggestions), observed frequencies of contraindications of their cases, etc.

Strategy B: Deleting Cases with Additional Attributes

Our aim with strategy B was to keep in the case base those cases that have a higher chance to be adaptable. These are cases with few contraindications. We therefore adapted a strategy opposite to strategy A, namely deleting cases with additional information (contraindications) to their prototype. As many cases are deleted, we set the threshold parameter “number of cases” to the value two. Here, we varied the second parameter “relative frequency,” which determines the frequency with which contraindications have to be observed among the cases to be incorporated into a prototype.

The difference between the results for the two settings for strategy B is rather small (Table 10.6). With a smaller relative frequency (second setting) more contraindications are incorporated into the prototypes. So, fewer stored cases have additional contraindications in comparison to their prototypes and consequently fewer cases are deleted. Furthermore, fewer prototypes are generated, because the prototypes cover more cases. The memory size is nearly the same and the number of retrieved adaptable cases is exactly the same for both settings.

In comparison to strategy A, it is noticeable that about the same number of prototypes have been generated, but much more cases have been deleted. Though those cases which have a bigger chance to be adaptable remain in the case base, the number of retrieved adaptable cases slightly increases in comparison to the first setting of strategy A, but the number is not as high as in the other settings of strategy A.

So, the strategy of keeping those cases that are easily adaptable results in such a small case base that only few adaptable cases can be retrieved.

Strategy C: All Cases Remain in the Case Base

For strategy C no cases are deleted at all. We have evaluated the same threshold parameter settings as for strategy A. It can be seen that many more adaptable cases can be retrieved in comparison to the corresponding settings of strategy C, while the memory requirement increases only slightly (Table 10.7).

Since two cases are sufficient to generate a prototype in the first setting, many prototypes are created and the memory requirement increases correspondingly. It is a little surprising that fewer adaptable cases are retrieved, but this is because a hierarchy with three levels of prototypes has been generated, and since the prototypes are treated as cases, the right prototype on each level has to be determined to be the most similar case.

Really surprising is the big increase of retrieved adaptable cases in the third setting. There are two possible explanations. Firstly, as the number of generated prototypes decreases, the prototype hierarchy is simpler and it is easier to find the appropriate case. Secondly, and probably the main reason, the number of cases which are necessary to generate a prototype is higher (= 4), so that more cases are considered when a generated prototype is filled, and consequently fewer contraindications are incorporated into the prototype. This means the prototypes themselves become more adaptable.

Table 10.7. Test results for strategy C

	1. Setting number of cases = 2	2. Setting number of cases = 3	3. Setting number of cases = 4	4. Setting number of cases = 4
memory size	in Mbytes	in Mbytes	in Mbytes	in Mbytes
after 75 cases	2.439	2.426	2.421	2.419
number of prototypes	19	10	8	7
number of stored cases	75	75	75	75
number of deleted cases	0	0	0	0
number of adaptations	29	32	52	51

However, when the number of generated prototypes decreases, there are fewer cases available to be used for adaptation (fourth setting).

Summary of the Evaluation Results for the Prototype Strategies

Keeping all cases in the case base increases the memory requirement, but increases the number of retrieved adaptable cases dramatically. Considering the number of retrieved adaptable cases, strategy A provides results that are nearly as good as for strategy C, but the achieved reduction is rather small. Keeping more adaptable cases (strategy B) results in a small case base, but only few adaptable cases can be found.

Too many prototypes should be avoided, because a complex hierarchy results in difficulties in finding the desired case. This means the threshold parameter "number of cases" should not be set too low.

The most preferable setting is the third one of strategy C. If the memory limitations become a real problem, strategies that delete redundant cases should be considered. Every stored case increases the memory requirement of our system by approximately 1.7 KB. This might lead to performance problems for much bigger case bases, keeping in mind that our test set of 75 cases covers just one out of a set of more than 80 medical areas.

The best settings, whether all cases are retained (strategy C) or cases without additional information (strategy A) are deleted, are those where the threshold parameter "number of cases" is set sufficiently high. It leads to more retrieved adaptable cases.

10.5 ISOR

ISOR is a CBR system for long-term therapy support in the endocrine domain [35]. It performs typical therapeutic tasks, such as computing initial therapies, initial dose recommendations, and dose updates. Apart from these tasks ISOR deals especially with situations where therapies become ineffective. Causes for inefficacy have to be found and better therapy recommendations should be computed. In addition to the typical CBR knowledge, namely former already solved cases, ISOR uses further knowledge forms, especially medical histories of query patients themselves and prototypical cases (prototypes).

ISOR uses prototypes in two ways, namely in form of guidelines for dose calculations and as generalised solutions for therapy inefficacy.

10.5.1 Computing Initial Doses: Guidelines as Prototypes

For hypothyroidism only one drug exists, namely Levothyroxine. The problem is to calculate effective initial doses (Fig. 10.11). Firstly, a couple of prototypes exist. These are recommendations that have been defined by expert commissions [36]. Though we are not sure whether they are officially accepted, we call

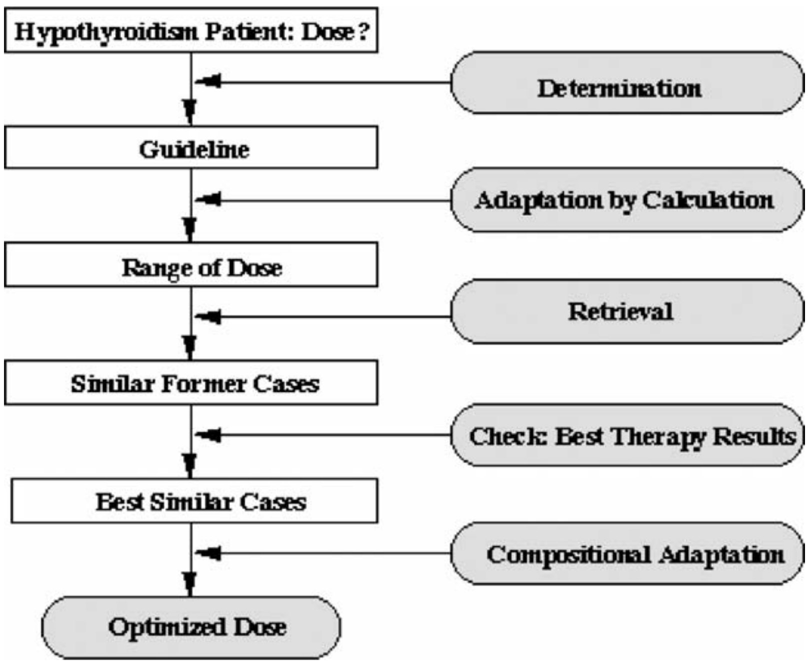


Fig. 10.11. Determination of an initial Levothyroxine dose

them guidelines. The assignment of a patient to a fitting guideline is obvious because of the way the guidelines have been defined. With the help of these guidelines a range for good doses can be calculated.

To compute a dose with best expected impact, we retrieve similar cases whose initial doses are within the calculated ranges. Since cases are described by few attributes and since our case base is rather small, we use Tversky's sequential measure of dissimilarity [20]. On the basis of those retrieved cases that had best therapy results an average initial therapy is calculated. Best therapy results can be determined by values of a blood test after two weeks of treatment with the initial dose. The opposite idea to consider cases with bad therapy results does not work here, because bad results can also be caused by various other reasons.

10.5.2 Generalised Solutions for Therapy Inefficacy

When long-term therapies become inefficient, ISOR searches for reasons and attempts to find better therapies. Solutions are reasons for inefficacy. General solutions like "irregular intake" or "changes of hormonal situation" are used as prototypes on a first level. On a second level these prototypes are more specific. All prototypes are described by three main attributes (problem code, diagnosis, and therapy) and some additional attributes like age, sex, prescribed drug etc. All prototypes have been defined by medical experts.

At first the retrieval searches among the prototypes on the top level and checks which solution might be probable for the query patient. Subsequently prototypes on lower level are considered and finally the single cases, which belong to the retrieved prototype.

10.6 Summary: The Role of Prototypes

The presented systems have one thing in common that distinguishes them from most CBR systems: They use prototypes as a form of knowledge representation that fills the gap between specific cases and general rules. The main purpose of such generalised knowledge is to structure the case base, to guide the retrieval process, and sometimes to decrease the amount of storage by erasing redundant cases.

In domains with rather weak domain theories another advantage of case-oriented techniques is their ability to learn from cases. Only gathering new cases may improve the system's ability to find suitable similar cases for current problems, but it does not elicit the intrinsic knowledge of the stored cases. To learn the knowledge contained in cases a generalisation process is necessary. In our early warning system concerning the kidney function, apart from guiding the retrieval and structuring the case base prototypes mainly serve to learn typical course pattern, because just the relevant kidney parameters are known but no knowledge about their temporal course behaviour exists.

For diagnosis of dysmophic syndromes prototypes correspond directly to the physician's sense of prototypes. As comparisons with single cases are unable to identify typical features, in this application the use of prototypes is not only sensible, but even necessary.

In ISOR, the prototypes for dose calculation are guidelines and the prototypes for therapy inefficacy are similar to those for diagnosis of dysmorphic syndromes. The main difference is that in ISOR all prototypes are defined by medical experts.

Summarising our experiences we would like to make quite clear that the role of prototypes depends on the application and the task. For medical diagnoses they even seem to be necessary because of their correspondence to medical prototypes which guide the physicians diagnoses. In domains with very poor domain theories they may help to learn general knowledge.

References

1. Strube G, Janetzko D (1990) Episodisches Wissen und fallbasiertes Schließen: Aufgaben für die Wissensdiagnostik und die Wissenspsychologie. Schweizerische Zeitschrift für Psychologie 49 (4): 211–221

2. Swanson DB, Feltovich PJ, Johnson PE (1977) Psychological Analysis of Physician Expertise: Implications for Design of Decision Support Systems. In: Shires DB, Wolf H (eds.): Proceedings of MEDINFO 77, North-Holland, Amsterdam, pp 161–164
3. Schank RC (1982) Dynamic Memory: a theory of learning in computer and people. Cambridge University Press, New York
4. Bareiss R (1989) Exemplar-based knowledge acquisition. Academic Press, San Diego
5. Maximini K., Maximini R., Bergmann R.(2003) An Investigation of Generalized Cases. In: Asley, K.D., Bridge, D.G. (eds.): Proc ICCBR 2003, Springer, Berlin Heidelberg New York, pp 261–275
6. Bellazzi R, Montani S, Portinale l (1998) Retrieval in a prototype-based case library: a case study in diabetes therapy revision. In: Smyth B, Cunningham P (eds) Proc European Workshop on Case-Based Reasoning. Springer-Verlag, Berlin Heidelberg New York, pp 64–75
7. Bichindaritz I (1995) Case-based reasoning adaptive to several cognitive tasks. In: Aamodt A, Veloso M (eds) Case-Based Reasoning Research and Development, Proceedings International Conference on Case-Based Reasoning, ICCBR-95, Springer-Verlag, Berlin Heidelberg New York, pp 391–400
8. Turner R (1988) Organizing and Using Schematic Knowledge for Medical Diagnosis. In: Kolodner J (ed) Proceedings of a Workshop on Case-Based Reasoning, Florida, pp 435–446
9. Perner P (2006): A Comparative Study of Catalogue-Based Classification. In: Roth-Berghofer TR et al (eds.): Proc ECCBR, Springer Berlin 301–308
10. Clancey WJ (1985) Heuristic Classification. Artificial Intelligence 27: 289–350
11. Selz O (1913) Über die Gesetze des geordneten Denkverlaufs. Stuttgart
12. Perner P (2004) Are case-based reasoning and dissimilarity-based classification two sides of the same coin?, Journal Engineering Applications of Artificial Intelligence, 15/2: 205–216
13. Taybi H, Lachman RS (1990) Radiology of Syndromes, Metabolic Disorders, and Skeletal Dysplasia. Year Book Medical Publishers, Chicago
14. Gierl L, Stengel-Rutkowski S (1994) Integrating Consultation and Semi-automatic Knowledge Acquisition in a Prototype-based Architecture: Experiences with Dysmorphic Syndromes. Artificial Intelligence in Medicine 6: 29–49
15. Clinical Dysmorphology. www.clindysmorphol.com
16. Winter RM, Baraitser M, Douglas JM (1984) A computerised data base for the diagnosis of rare dysmorphic syndromes. Journal of medical genetics 21 (2): 121–123
17. Stromme P (1991) The diagnosis of syndromes by use of a dysmorphology database. Acta Paeditr Scand 80 (1): 106–109
18. Weiner F, Anneren G (1989) PC-based system for classifying dysmorphic syndromes in children. Computer Methods and Programs in Biomedicine 28 111–117
19. Evans CD (1995) A case-based assistant for diagnosis and analysis of dysmorphic syndromes. International Journal of Medical Informatics 20: 121–131
20. Tversky A (1977) Features of Similarity. Psychological Review 84 (4): 327–352
21. Rosch E, Mervis CB (1975) Family Resemblance: Studies in the Internal Structures of Categories. Cognitive Psychology 7: 573–605

22. Robeson SM, Steyn, DG (1990) Evaluation and comparison of statistical forecast models for daily maximum ozone concentrations. Atmospheric Environment 24 B (2): 303–312
23. Shahar Y (1999) Timing is Everything: Temporal Reasoning and Temporal Data Maintenance in Medicine. In: Horn W, Shahar Y, Lindberg G, Andreassen S, Wyatt J (eds) Proceedings of AIMDM'99, Springer-Verlag, Berlin Heidelberg New York, 30–46
24. Haimowitz IJ, Kohane IS (1993) Automated trend detection with alternate temporal hypotheses. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo, pp 146–151
25. Miksch S, Horn W, Popow C, Paky F (1995) Therapy planning using qualitative trend descriptions. In: Barahona P, Stefanelli M, Wyatt J (eds) Proc AIME'95, Springer-Verlag, Berlin Heidelberg New York, pp 197–208
26. Lenz M, Auriol E, Manago M (1998) Diagnosis and decision support. In: Lenz M, Bartsch-Spörl B, Burkhard H-D, Wess S (eds) Case-Based Reasoning Technology, From Foundations to Applications., Springer-Verlag, Berlin Heidelberg New York, pp 51–90
27. Smyth B, Keane MT (1998) Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. Artificial Intelligence 102: 249–293
28. Anderson JR (1989) A theory of the origins of human knowledge. Artificial Intelligence 40, Special Volume on Machine Learning, pp 313–351
29. Wenkebach U, Pollwein B, Finsterer U (1992) Visualization of large datasets in intensive care. Proc Annu Symp Comput Appl Med Care, pp 18–22
30. DeSarbo WS et al. (1992) TSCALE: A new multidemensional scaling procedure based on Tversky's contrast model. Psychometrika 57: 43–69
31. Kolodner J (1993) Case-based reasoning. Morgan Kaufmann Publishers, San Mateo
32. Bueno-Cavanillas A et al. (1994) Influence of nosocomial infection on mortality rate in an intensive care unit. Crit Care Med 22: 55–60
33. Stottler RH, Henke AL, King JA (1989) Rapid retrieval algorithms for Case-Based Reasoning. In: Proceedings of International Joint Conference on Artificial Intelligence, 233–237
34. Smyth B, Keane MT (1993) Retrieving adaptable cases: the role of adaptation knowledge in case retrieval. In: Richter MM et al. (eds) Proceedings of 1st European Workshop on Case-Based Reasoning, pp 76–81
35. Schmidt R, Vorobieva O (2006) Case-Based Reasoning Investigation of Therapy Inefficacy. Knowledge-Based Systems 19 (5): 333–340
36. Working group for paediatric endocrinology of the German society for endocrinology and of the German society for children and youth medicine (1998) 1–15