ELSEVIER Artificial Intelligence in Medicine 27 (2003) 155–179

Artificial Intelligence in Medicine

www.elsevier.com/locate/artmed

Building a case-based diet recommendation system without a knowledge engineer

Abdus Salam Khan*, Achim Hoffmann

School of Computer Science and Engineering, The University of New South Wales, Sydney 2052, Australia

Received 14 February 2002; received in revised form 12 September 2002; accepted 9 October 2002

Abstract

We present a new approach to the effective development of menu construction systems that allow to automatically construct a menu that is strongly tailored to the individual requirements and food preferences of a client. In hospitals and other health care institutions dietitians develop diets for clients which need to change their eating habits. Many clients have special needs in regards to their medical conditions, cultural backgrounds, or special levels of nutrient requirements for better recovery from diseases or surgery, etc. Existing computer support for this task is insufficient—many diets are not specifically tailored for the client's needs or require substantial time of a dietitian to be manually developed. Our approach is based on case-based reasoning, an artificial intelligence technique that finds increasing entry into industrial practice. Our approach goes beyond the traditional case-based reasoning (CBR) approach by allowing an incremental improvement of the system's competency during routine use of the system. The improvement of the system takes place through a direct expert user–system interaction while the expert is accomplishing their tasks of constructing a diet for a given client. Whenever the system performs unsatisfactorily, the expert will need to modify the system-produced diet 'manually', i.e. by entering the desired modifications into the system. Our implemented system, menu construction using an incremental knowledge acquisition system (MIKAS), asks the expert for simple explanations for each of the manual actions he/she takes and incorporates the explanations automatically into its knowledge base (KB) so that the system will perform these manually conducted actions automatically at the next occasion. We present MIKAS and discuss the results of our case study. While still being a prototype, the senior clinical dietitian involved in our evaluation studies judges the approach to have considerable potential to improve the daily routine of hospital dietitians as well as to improve the average quality of the dietary advice given to patients within the limited available time for dietary consultations. Our approach opens up a new avenue towards building highly specialised CBR systems in a more cost-effective way. Hence, our approach promises to allow a significantly

* Corresponding author. Tel.: +61-2-9385-6908; fax: +61-2-9385-4936.
E-mail addresses: askhan@cse.unsw.edu.au (A.S. Khan), achim@cse.unsw.edu.au (A. Hoffmann).

more widespread development and practical deployment of CBR systems in a large variety of application domains including many medical applications.

Keywords: Dietary menu construction; Case-based reasoning; Knowledge acquisition; Ripple-down rules

1. Introduction

The task of menu construction is important in a number of contexts. Firstly, for institutions, such as hospitals, nursing homes, schools, etc. there is the need to plan menus that can be prepared and administered within the constraints given by the available space and equipment for preparation and delivery as well as the available personnel and financial means. Secondly, there is a need of providing suitable menus to clients who should eat according to a planned diet, due to medical conditions or the desire to obtain or maintain bodily fitness, etc.

A number of systems and approaches are available today, such as [41–45,48] for menu construction and dietary analysis, Vienna expert system for parenteral nutrition of neonates (VIE-PNN) [22] is designed to perform specific task of calculating the daily changing composition of parenteral nutrition for small newborn infants. Other software, such as [40,46,47] provide assistance for the management of the logistics for hospital kitchens as well as the design of menus under the aspects of cost and ease of preparation. These latter systems are good for institutional menu planning where the actual meal preparation casts substantial constraints on what is an acceptable menu, i.e. the various meals for the different patients in a hospital need to be prepared in the same kitchen by using a limited set of ingredients. Also the actual preparation cannot be too time-consuming. As a consequence, existing systems are capable of taking the mentioned constraints into account and they produce a menu plan for each patient for a day or a number of days. Most patients will receive the same menu and only a few patients with special conditions will receive some modification of the standard menu. For example, at the Prince of Wales Hospital in Sydney, the CBord system [40] is used, which allows to retrieve initially designed menus from its database according to some important medical conditions and a rough energy content, such as 1500 or 2000 kcal, etc. The system will produce a menu but does not allow the dietitian to modify the diet nor does it tell the totals of the nutrient contents. This information is important to allow the dietitian an efficient modification and further adaptation of the menu to the patient's needs.

While the existing systems are reasonably useful for the purposes of institutional menu planning given the constraints on meal preparation, etc. for developing menus for the hospital, they are not suited to develop customised menu plans for patients who leave the hospital. At the moment, the hospital dietitians at the Prince of Wales Hospital discuss general dietary recommendations with patients for which diet is critical. However, it would be much more desirable to be able to hand out to each patient a set of automatically generated menus which are suitable to the patient and from which a patient can then choose what they want to eat when they return home from hospital.

In our research we targeted this latter need for more sophisticated menu recommendation systems that are capable of producing highly customised menus. This paper reports on our approach to build a sophisticated menu construction system. Our system architecture is that of a CBR system, a common type of system in artificial intelligence and also increasingly used in medical applications. However, we also developed a new way of developing the necessary components of a CBR system which need to be tailored towards the application domain and whose development has always required the cooperation and substantial time-commitment of a domain expert—in our case a dietitian and a knowledge engineer.

Usually, a dietitian evaluates a client's dietary conditions and enters those into a computer-based diet construction system. Often the diet constructed by the system requires modification by the dietitian or nutritionist in order to meet certain integrity constraints, such as ensuring a meat portion in each lunch, juice for breakfast, etc. or simply milk when a cereal is planned for breakfast. Since menu integrity requirements are very difficult to comprehensively formalise, the currently available diet construction systems violate at times such restrictions which are rather obvious for the human user.

Over the years various mathematical models, such as linear programming, have been proposed and applied to diet construction with little success to completely automate the diet construction process (see, e.g. [9,12,14,15,18,39]).

Those models focus basically on meeting the nutrient requirements as best as possible. While these models do a reasonable job of finding cost-effective diets based on a limited number of foods, they are not very good at designing flexible diets that accommodate a client's food preferences and have well-integrated meals.

The paper is organised as follows: in Section 2, we explain about a menu planning system that is used in our local hospital, and in Section 3, we present the motivation and technical details of our approach. Section 4 illustrates, how the approach is implemented in our menu design system MIKAS. Section 5 presents the results of our evaluation studies. In Section 6, we discuss the obtained results. The conclusions of the paper are found in Section 7.

2. Currently used menu planning system

Hospital dietitians perform a number of tasks. One concerns the development and choice of menus to be provided to the hospitalised patients. Another task concerns the dietary advice of those patients who are leaving the hospital. When deciding on a diet for a patient the dietitian will take the diagnosed medical conditions, potential food allergies as well as general aspects of the patient, such as age, fitness, usual activities, etc. into account. Where feasible, also the food preferences of a patient are considered. The CBord [40] system is widely used in hospitals in Australia and New Zealand. For the patients staying in the Prince of Wales Hospital in Sydney, Australia, CBord is consulted, i.e. a query is formed on the basis of the desired energy content of the menu as well as a number of critical medical conditions which require special diets, such as diabetes. CBord usually will retrieve one or more menus from its database from which the dietitian will select. Some of the diets in CBord's database have a number of alternatives for some of the menu slots, such as red and white meat for the lunch main part. The selected menu is then forwarded to the kitchen, where the chef will decide which option is taken, subject to availability, cost of other required ingredients as well as the required resources for preparation. Finally, the prepared meals are delivered to the patients—sometimes the patients may still have a choice, e.g. between different types of meat or desserts.

CBord does not allow the dietitian to alter a retrieved menu within the system. Neither does CBord support any modification by, e.g. calculating the nutrient content of foods or an entire menu for a day. The only way of customising a retrieved menu is to print the menu and change parts of it on the printout which then can be given to the hospital kitchen. Similarly, CBord does not support the development of customised menus for patients leaving the hospital.

3. Building intelligent menu construction systems

For developing more satisfactory diet construction systems as well as more generally for developing intelligent systems, the main difficulty to overcome is to put the relevant expert knowledge into the machine. The field of knowledge acquisition deals with this problem.

3.1. Knowledge acquisition

The main research directions in knowledge acquisition have been approaches such as KADS [7,38] or Generic Tasks [8] where certain generic program structures, such as problem-solving methods, are to be re-used and complemented by problem-specific knowledge. This shifts the problem of providing all the domain-specific knowledge to partly providing domain-specific knowledge by selecting an appropriate problem-solving method. However, a substantial task of knowledge acquisition still remains and is usually addressed by a software-engineering style approach including a problem analysis phase and the synthesis of the required knowledge structures. More recent research in the field also tries to develop re-usable domain ontologies representing generally useful information about domains and their objects.

The knowledge acquisition bottleneck is so persistent because it is very difficult for an expert to anticipate all possible problems the system may face once it is put into routine use. A major cost factor and, hence, a practical hurdle is the requirement of a knowledge engineer who communicates between the domain expert and the technicalities of the used expert system shell, problem solving methods, or the CBR system. Hence, the system development is rather expensive since a *knowledge engineer* and a domain expert (both highly paid specialists) need to get together for extended periods to develop the system.

A somewhat unconventional approach to knowledge acquisition has been ripple-down rules (RDRs) and related techniques, which skip the usual phase of problem analysis and also do not require a knowledge engineer during the knowledge acquisition process. RDRs proved to be very successful for building classification systems.

Our work is the first which explores the potential of using the idea of RDRs for building CBR systems. In this paper, we present a new approach to the acquisition of the required knowledge, based on the idea of RDR, which only requires the knowledge engineer for the initial set-up phase of the system. After a short training period, the expert can add knowledge to the system on their own and during routine use, as adding a single rule takes only a few minutes of their time.

More importantly, our approach allows the incremental refinement of a KB which the system already uses. In case the system encounters situations for which it does not have appropriate knowledge yet the KB is refined on the spot (see also [24,25]). We have implemented the case-based diet construction system MIKAS, which allows the expert user to refine the system while already being used.

3.2. Ripple-down rules

The basic idea of RDR [10,23] is to develop a KB by allowing the expert to directly interact with the system and to incrementally add rules to a KB. RDR has been applied successfully in building what appears to be one of the largest expert system in routine use [13]. It has also been applied to some construction tasks, such as the Sisyphus I problem in [34] and to search in computer chess in [6].

In RDR, the object space to be classified is incrementally subdivided into smaller and smaller partitions, until all objects in each single partition belong to the same class. The rules for subdividing the object space are specified by the expert, whenever an object is encountered, which the system classifies in disagreement with the expert. That is, the current object x falls into a partition p of the object space, which classifies the object incorrectly. Hence, this partition needs to be further subdivided into two partitions p_1, p_2, such that the partition to which x belongs, classifies x correctly. Such a subdivision can be provided by the expert competently and with minimal effort, as the expert is merely required to provide an explanation of why x is different from the previously presented object x_p, which led to the creation of the partition p in the first place. That is, the expert needs only to provide a criterion by which x differs from x_p and which justifies the different classification. To provide such an explanation is usually easy for an expert as it is not much different to explaining their decision to colleagues.

Using these explanations a RDR tree is developed (see Fig. 1 for a RDR tree). An object is classified by this tree as follows: initially the ‘default rule’ in node 1 is evaluated, i.e. class ‘$-$’ is obtained. Before this becomes the final ‘verdict’, it is checked whether any ‘except’ link from the current node exists. If there is such a link, the connected node’s condition is checked.

Here, we check the condition of node 2: if “C” is true, then the corresponding class ‘$+$’ overwrites the previous ‘verdict’, unless this in turn is overwritten by another except link. If an except link exists, but the connected rule condition is not satisfied, say in node 3, then the nodes along a possibly existing chain of ‘if not’ links are checked. For instance, if the condition of node 2 is satisfied, node 3 is checked. If the condition of node 3 is not satisfied, then node 5 is checked and, if node 5 is not satisfied, node 6 is checked. If any of these nodes 3, 5, or 6 is satisfied, the classification of the first satisfied node is the final verdict, unless this node has again an overwriting except link.

If the expert is not satisfied, say with the classification due to node 5, the system asks for a justification in terms of the current object’s attributes. This explanation is then used as the condition to a new exception rule to node 5.

3.3. Case-based reasoning

While the CBR is another approach to alleviate the problem of knowledge acquisition to some degree (by providing knowledge in form of cases), it turned out that still a substantial

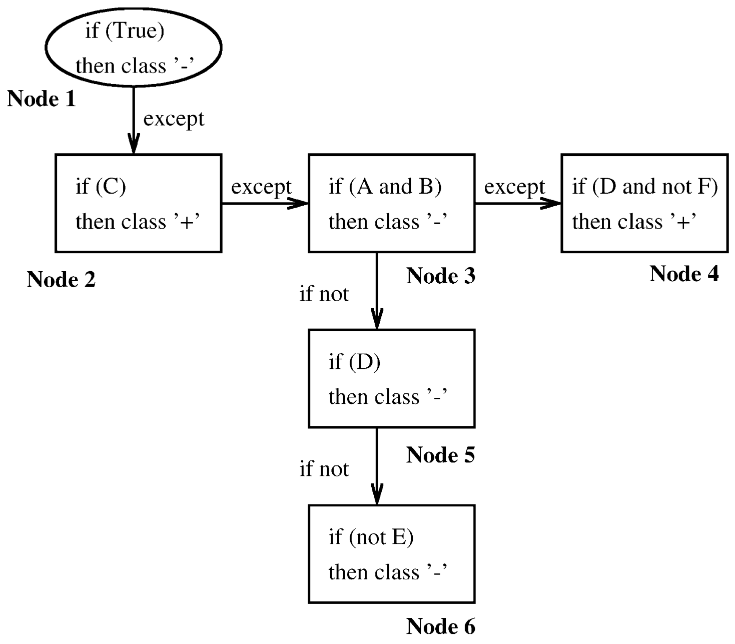


Fig. 1. An example of RDR tree.

difficulty remains to define what case should be retrieved for which problem and how should that case be adapted to produce a suitable solution to the current problem.

CBR is an approach to build intelligent system which increasingly finds entry into the industrial practice. Recently, CBR approaches have been used in the medical domain, e.g. [1,2,11,16,29,32,35–37].

The basic idea is to solve new problems by remembering solutions to problems which are similar to the current problem. Usually, one or multiple cases are remembered, which are similar to the current problem case and allow the derivation of a solution for the current problem case from the solutions of the remembered cases. If necessary, a remembered case is modified in a way that at least parts of the case is re-used. This process is known as the case adaptation [33]. CBR techniques have also been used in the field of menu construction. CAMPER [33] is the only CBR system for *diet prescription*. Well-known CBR systems addressing a somewhat related problem are CHEF [17], which is a recipe planner system, and JULIA [19–21], which is a meal design system. Both of the latter systems focus on the actual preparation of the meals as the primary objective rather than the nutrient composition and the overall suitability for certain medical conditions. The adaptation strategies of all these systems are hard coded. As a result, these systems cannot satisfy the requirement of an unusual client because of the absence of the appropriate adaptation knowledge. To handle this problem, a dietitian adapts the menu manually in the paper or a kitchen chef adapts the recipe of the meal.

A major practical advantage in CBR is the fact that experts are often eager to tell their "war stories"—the cases they encountered. This is opposed to the situation where experts are asked to provide abstract general rules of what they do, which is a much harder task for them.

However, the effectiveness of a CBR system depends not only on having and retrieving relevant cases but also on selecting a retrieved case for adaptation and actually adapting it to fit the new problem [27]. Both, suitable retrieval as well as adaptation of a case will usually require domain-dependent knowledge. CBR systems generally do not refine the methods they use to retrieve or adapt prior cases. Instead they rely on static pre-defined rules or procedures [31]. It is generally impossible to anticipate all the difficulties and problems one may encounter in a domain. Hence, the problem of defining proper adaptation rules represents a major bottleneck for successfully developing CBR systems. The problems are so acute that many CBR applications simply omit case adaptation [26]. This bottleneck demands the need of acquiring case-specific and general domain knowledge as an ongoing process for effective CBR performance.

In MIKAS we developed RDRs further and made them applicable for incrementally building a CBR system. RDRs [10] have been developed as an extremely effective approach for the acquisition of classification knowledge, as they require the expert only to provide explanations of the decision taken in a given situation.

In this paper, we present our new approach in MIKAS for developing a suitable adaptation function. We encode our adaptation knowledge in rules, which resemble the rules used in the INRECA system [5]. That is, the representation of conditions regarding attributes in the problem description is a subset of what has been developed in the language CASUEL [30], intended to be a general-purpose description language for CBR systems. However, we structure a set of rules in a unique way that supports very strongly the maintenance of the KB. Furthermore, we introduced the concept of abstract actions for our system MIKAS. In any case, the experiences in the INRECA project also showed the difficulty of actually encoding the suitable adaptation knowledge [4,5]. We address this experienced difficulty with our new way, how an expert interacts with the system in order to provide suitable adaptation rules. Our approach for the *acquisition of adaptation knowledge* is based on ideas of RDRs [10].

We lend ideas from RDRs that ensure an incremental coverage of the expert's judgements and decision making. That is, for all problem cases for which the expert provided advice to the system on how to adapt a retrieved case, the system will have to reproduce the expert's performance. We assume a domain expert will have a sufficient understanding of how to adapt a retrieved case in order to find a solution to the current problem.

3.4. *Preliminaries of our case-based reasoning approach to menu construction*

In general we consider attributes to be either numerical or discrete valued. We describe a case $C = \{\mathscr{P}, \mathscr{S}\}$ by two attribute vectors with the domains $P_1 \times P_2 \times \cdots \times P_n$ and $S_1 \times S_2 \times \cdots \times S_m$, respectively.

- The attribute vector $P = \langle p_1, \ldots, p_n \rangle$ represents the problem specification which contains a number of nutrient requirements along with a client description including the relevant medical conditions.
- The attribute vector $S = \langle s_1, \ldots, s_m \rangle$ represents a solution to the menu construction problem P and is a list of menu slots which are grouped according to the various meals during the day. A group of slots are available for the main meals, breakfast, lunch and

dinner to allow for a number of foods to be represented. Each slot contains a single food along with its quantity or is empty.

Each food is selected from our food database which contains the nutrients contained in the food as well as a key to its food group allowing to identify related foods. The case base contains a number of client descriptions along with menus that satisfy their needs.

3.5. Incremental acquisition of adaptation knowledge

After cases have been retrieved, the system tries to adapt the few highest ranked cases in order to fit the current problem.

If the system cannot produce a satisfactory menu, additional knowledge must be provided to the system. Generally speaking, this can either be a refined retrieval function, or a refinement of the adaptation function. The expert has to decide whether the retrieved cases are feasible to be adapted or whether other cases need to be retrieved.

If the expert decides to adapt the menu of a retrieved case, he/she tries to manually adapt the menu so that a menu for the current client is found. This adaptation is done by removing and adding individual foods. Subsequently, the system requests explanations for each adaptation action the expert chose. To do so, the expert has to provide a condition under which the chosen adaptation action (the removal or addition of a component) should be executed. This condition is expressed, generally speaking, in terms of the current client description as well as the retrieved menu being adapted.

Furthermore, the action needs to be specified beyond being either a deletion or an addition of a component. The particular food which is added or deleted from the particular slot of the menu needs to be specified. This is done again by providing conditions which a food has to meet to be eligible for deletion or addition from or to a particular menu slot or class of slots. The new rule r' is integrated into the existing ripple-down rules structure as an exception to the rule r, which produced the undesired adaptation action, which the expert intends to supersede by the new rule. The system ensures that the expert entered conditions do not apply to any of those cases, to which the previous rule r applied successfully, i.e. for which rule r led to a desired adaptation action.

3.5.1. Abstract actions

The purpose of *abstract actions* is to allow a way of generalisation. An adaptation rule is provided by an expert, who encounters an individual case which needs to be adapted, while observing the system's performance. The expert will be able to decide which component (food) should be replaced by which new component (new food).

However, just to provide the identity of those components (foods) will render the CBR system incapable to adapt a case which contains not exactly the same but perhaps a very similar component (food). Similarly, the new component (food) may need some variation for a new case which need to be adapted. As a consequence, our system lets the expert specify the action, he/she suggests for adaptation of the given case, and then asks the expert to abstract from the individual action and to give a more abstract description of that action (usually a replacement of a component (food), an increase or decrease in the amount or size of a component (food)).

An abstract action can be considered as a set of rules in itself. That is, depending on the features of the solution to be adapted, the abstract action may result in different changes to a different case such as scaling the portion sizes of a food or using similar foods.

Formally, we allow to define abstract actions by a set of rules. Each rule has a condition part as described in the previous subsection and an action part. The action to be specified is either to delete or to add a certain component (food) or to increase or decrease the amount/size of a component (food), which can be identified using certain attributes of the component (food or meal slot) and whose numerical attributes can be described using the attribute values of parts of the current problem and/or solution case. A *replacement-action* is then composed of a *delete-action* and an *add-action*.

A *delete-action* just needs to identify the component (food) to be deleted. This is done by specifying attribute value ranges for the component (food), which have to be matched by the component which is deleted from the current case.

4. Building a diet construction system with MIKAS

MIKAS implements the above sketched strategy for developing a CBR system. In the following, we describe the problem domain first. This is followed by a description of our MIKAS system.

4.1. The problem domain of diet construction

To illustrate the requirements for our CBR system MIKAS, we give a brief explanation of what kind of problems the system has to deal with in this domain.

A menu is recommended by a dietitian for a patient with specific health conditions. These health conditions represent stringent requirements on the menus to be designed. This does not only include certain amounts of various nutrients which should be contained in the diet. It may also refer to certain ways of preparing foods, e.g. use of spices, certain types of foods (vegetarian, kosher meat, etc.), or certain other aspects of the involved foods, such as their texture or flavour. The significance of this is that for 'normal' menus it seems appropriate to retrieve cases according to how well they match the required nutrients. Adaptation may be necessary in order to scale the amount of foods to match nutrient requirements or to remove or replace certain types of foods which are unsuitable for the given patient.

We give a few examples for illustrating possible requirements:

1. *Patient*: anaemic, diabetic and has high blood pressure. Furthermore, the patient is allergic to garlic.

   Diet requirements: the iron content should be fairly high due to anaemia, sugar content should be very low due to diabetes, fat, cholesterol and cooking salt (sodium) should be minimised due to high blood pressure and the menu must not contain any food that has garlic in the preparation due to the patient's allergy.
2. *Patient*: anaemic, diabetic, has high blood pressure and is allergic to garlic. In addition, the patient has duodenal ulcer.

   Diet requirements: same requirements as for the diet in patient 1. In addition, due to duodenal ulcer, additional snacks in regular intervals are recommended to minimise damage of gastric lining. Bedtime snacks should be avoided as they will cause nocturnal acid secretion. Black pepper and chili powder should be avoided because they may cause distress. Caffeine should be avoided as it increases gastric secretion.

3. *Patient*: kidney disease with glomerular filtration rate (GFR) less than 20–25 ml/min and plasma urea level greater than 20 mmol/l.

   Diet requirements: low protein content in order to lessen the workload for kidneys (protein rich food can increase levels of waste products leading to symptoms such as nausea, loss of appetite, vomiting and tiredness). Sodium content should be kept low to prevent fluid retention. Foods high in sodium are avoided because salty foods can cause fluid retention and can result in high blood pressure, oedema and shortness of breath. Potassium should be kept low to prevent abnormal cardiac rhythms and contractions. Potassium restriction limits foods like milk, meat, whole-grain products, most fruit and vegetables. No salt should be used in cooking or added at the table.

Patients 1 and 2 have similar conditions. So it appears appropriate to retrieve a menu for patient 2 which was already used for patient 1. However, the additional requirements would need to be accommodated by adaptation. If a bedtime snack is removed, some other foods may need to be increased in order to match the nutrient requirements. If black pepper should be avoided, a main meal may need to be replaced entirely, as the peppered meat portion in the retrieved menu may not be acceptable.

If a menu is retrieved for patient 3, which was used for patient 1 or 2, again adaptation steps need to be taken, which would reduce the salt content. That means, all foods which are rich in salt would need to be replaced or deleted from the menu.

If there are many such foods, a different menu should be retrieved, otherwise adaptation can possibly handle the problem.

4.2. The architecture of MIKAS

Fig. 2 shows the general architecture of our diet construction system MIKAS. The user interacts through the user interface with the system. The system has two basic modes of operation which interplay with each other. The consultation mode: MIKAS accepts client requirement specifications and will then construct a menu and present it to the user. The second mode is the knowledge acquisition mode which can be entered into after the user judges the constructed menu as unacceptable. In that case MIKAS will ask the user to correct MIKAS' process of constructing the unacceptable menu. Each step of the construction process can be shown to the user: first, the retrieval of a menu which was stored previously as being suitable for a similar client. If the user does not agree that the retrieved menu is suitable—either as is or for being adapted—the user can add a rule to the retrieval KB.

It follows the adaptation of that retrieved menu, usually taking multiple adaptation steps. For each adaptation step, the user can decide whether this step should be taken or another one or none instead. If the step taken by the system is rejected by the user, MIKAS will ask the user to provide an explanation for that judgement which is then automatically converted into a new rule which is added to the adaptation KB.

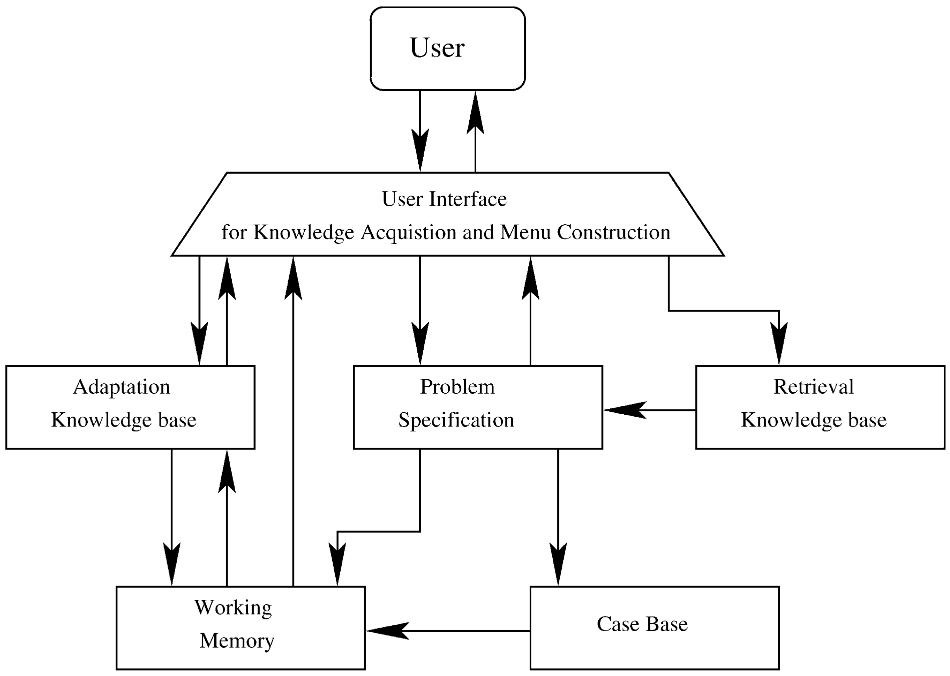


Fig. 2. The architecture of MIKAS.

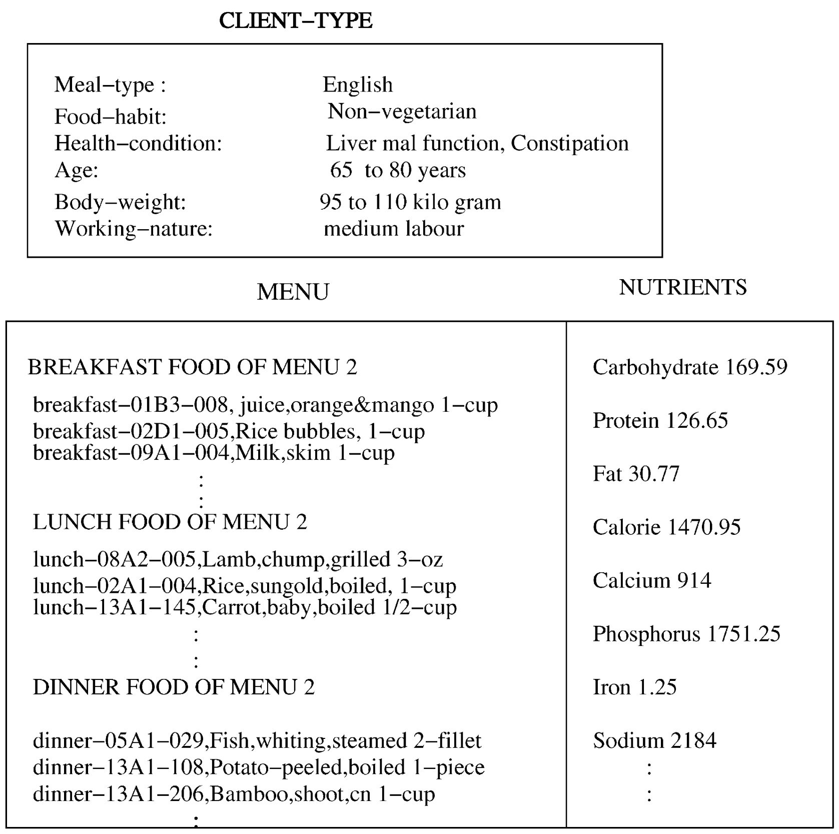
MIKAS accepts a description of the patient along with nutritional requirements. Applicable restrictions are not necessarily explicitly given, but acquired, largely in the form of adaptation rules, provided by the expert.

Once a problem description has been entered, the system retrieves and proposes a most appropriate menu from the case base using a trapezoid-shaped fuzzy scoring scheme against nutrient requirement. The corners of the trapezoid are provided by the expert. This scoring scheme ranks all the cases of the case base. A typical case in a case base is shown in Fig. 3. The case is comprised of a problem specification (which is a description of a client), the nutrient requirement of the client and a menu.

The expert then considers the retrieved menus. If he/she is not satisfied with the retrieval result, the retrieval function will be enhanced. Otherwise, he/she tries to adapt one of the retrieved menus, using various adaptation steps including the addition or deletion of food or the increase/decrease of portions. The rationales of taking particular adaptation actions are captured as conditions of rules and are stored in a RDR tree. Fig. 4 shows the window through which the expert performs adaptation actions to modify a menu. Fig. 5 shows an example of growth of RDR tree with addition of the exception rules.

4.2.1. Auto-adaptation

When a problem specification and nutrient requirements for a client are given to the system, the difference is calculated between the required nutrients and the



CLIENT–TYPE

Meal–type :	English
Food–habit:	Non–vegetarian
Health–condition:	Liver mal function, Constipation
Age:	65 to 80 years
Body–weight:	95 to 110 kilo gram
Working–nature:	medium labour

MENU	NUTRIENTS
BREAKFAST FOOD OF MENU 2	Carbohydrate 169.59
breakfast–01B3–008, juice,orange&mango 1–cup	Protein 126.65
breakfast–02D1–005,Rice bubbles, 1–cup	
breakfast–09A1–004,Milk,skim 1–cup	Fat 30.77
:	
:	
LUNCH FOOD OF MENU 2	Calorie 1470.95
lunch–08A2–005,Lamb,chump,grilled 3–oz	Calcium 914
lunch–02A1–004,Rice,sungold,boiled, 1–cup	
lunch–13A1–145,Carrot,baby,boiled 1/2–cup	Phosphorus 1751.25
:	
:	
DINNER FOOD OF MENU 2	Iron 1.25
dinner–05A1–029,Fish,whiting,steamed 2–fillet	Sodium 2184
dinner–13A1–108,Potato–peeled,boiled 1–piece	:
dinner–13A1–206,Bamboo,shoot,cn 1–cup	:
:	

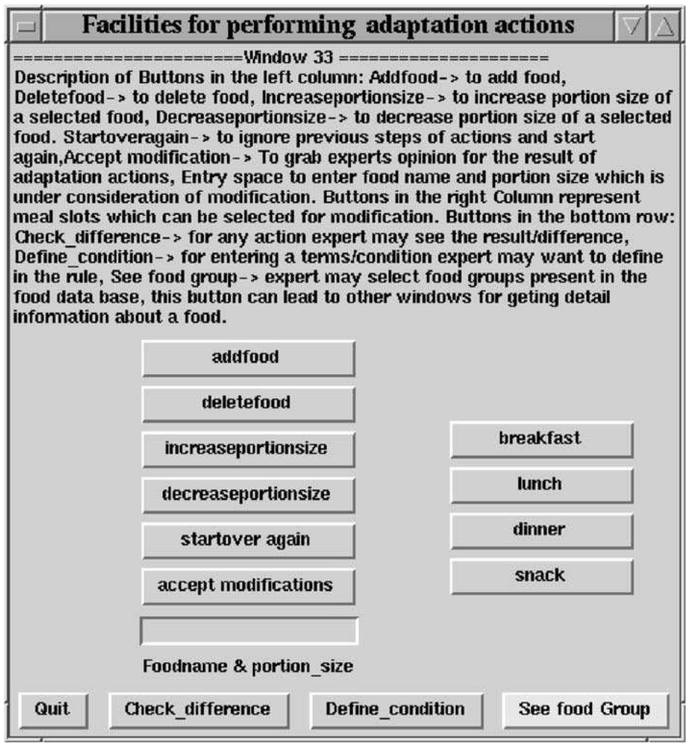
Fig. 3. An example case.

nutrients of the menu. If the given client description and the generated nutrient differences satisfy the conditions of a rule then this rule will trigger the associated adaptation action.

The overall process of constructing a suitable diet involves the following steps:

1. A problem description including nutrient requirements is submitted to the system.
2. The system retrieves a menu and adapts it if necessary.
3. Rules in the adaptation RDR tree are evaluated and determine the next applicable adaptation action. If the system does not find an applicable adaptation action, the system stops and reports that the adapted menu is still not satisfactory.
4. After the action from step 3 has been applied to the current menu the new menu is assessed whether it is acceptable for the patient at hand. If the new menu is not acceptable, another adaptation action is sought by going back to step 3.

If the system cannot produce a satisfactory menu it is up to the expert user to choose manually further adaptation steps in order to produce an acceptable menu. Examples of auto-adaptation actions and their reasons are provided in Section 4.3.


Facilities for performing adaptation actions

======================Window 33 ====================
Description of Buttons in the left column: Addfood-> to add food, Deletefood-> to delete food, Increaseportionsize-> to increase portion size of a selected food, Decreaseportionsize-> to decrease portion size of a selected food. Startoveragain-> to ignore previous steps of actions and start again,Accept modification-> To grab experts opinion for the result of adaptation actions, Entry space to enter food name and portion size which is under consideration of modification. Buttons in the right Column represent meal slots which can be selected for modification. Buttons in the bottom row: Check_difference-> for any action expert may see the result/difference, Define_condition-> for entering a terms/condition expert may want to define in the rule, See food group-> expert may select food groups present in the food data base, this button can lead to other windows for geting detail information about a food.

Fig. 4. Window for choosing an adaptation action in MIKAS.

4.3. A sketch of a knowledge acquisition session with MIKAS

In the following we describe a session with MIKAS that shows how diets are constructed. The examples will focus on the acquisition of new rules for situations in which MIKAS does not produce satisfactory diets. Consider the following clients:

- Client 1: a client who usually eats English and non-vegetarian food, 55 years of age, body weight 85 kg, has constipation with complications of his liver function, and of hard working nature.

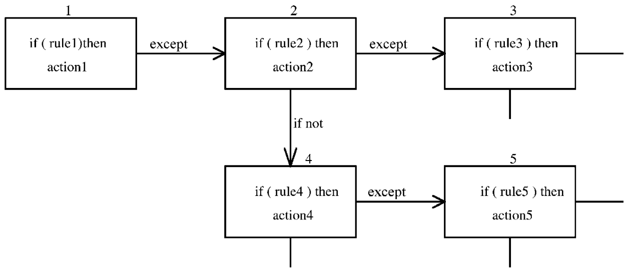

1
if (rule1)then action1
except
2
if (rule2) then action2
except
3
if (rule3) then action3
if not
4
if (rule4) then action4
except
5
if (rule5) then action5

Fig. 5. Adaptation rule in RDR tree.

- Client 2: a client who usually eats English and non-vegetarian food, 65 years of age, body weight of 65 kg, has constipation with complications of his kidney function, and of moderate working nature.
- Client 3: a client who usually eats English and non-vegetarian food, 55 years of age, body weight of 65 kg, having abnormal function of liver and severe renal complication, and of sedentary working nature, whose diet is constrained by restricted quantity of protein, sodium and potassium.

The following diets and their modifications demonstrate how the knowledge acquisition process helps to incrementally build a competent diet construction system. The MIKAS system performed all modification steps in order to meet the required nutrient levels to the highest possible degree. Table 1 shows how and when rules were acquired.

A rule is always acquired when the expert user is not satisfied with the diet constructed by MIKAS. Then the user modifies the constructed diet until a satisfactory diet is found. For each modification action (either delete or add a food to a particular meal slot) a rule is acquired which is based on the user's explanations for why the action is necessary. Also, the expert may check all adaptation actions performed on a menu. As a result, the expert may revise some of the actions taken by adding exception rules to the adaptation RDR tree.

Table 1 also shows the more general applicability of the acquired adaption rules for future similar clients. For example, rule 106 was acquired by the system from the expert while modifying menu 19 to increase its carbohydrate content. Rule 79 was acquired by refining rule 4 while modifying menu 33. Rule 4 was refined as the action of rule 4 introduced excess meat in this particular diet. Rule 237 was acquired while modifying menu 14 by refining rule 52. Rule 275 was acquired while modifying menu 18 by refining rule 273 which introduced excess iron and repetition of the beef liver in the same meal slot.

We describe below the adaptation steps emphasising some interesting actions that were taken by the system while constructing a diet for each client. This demonstrates why a rule

Table 1
Illustration of rule acquisition and application in menu construction

Refinements	Refined rule[a]	Added rule	Past-modified menu[b]	Client[c]	Menu being modified[d]
1	1	106	19	1	11
2	4	79	33	1	11
3	1	52	22	2	11
4	4	115	25	2	11
5	52	237	14	3	14
6	32	238	14	3	14
7	248	284	20	3	14
8	48	248	15	3	14
9	1	60	25	3	14
10	273	275	18	3	14

[a] Expert refines a rule when expert does not agree with the action suggested by that rule.
[b] As a refinement, a new rule is acquired and saved as an exception rule of the refined rule.
[c] The client for whom past-modified menu was modified.
[d] Menu that is currently being modified using rules that were acquired before.

Table 2
Illustration of rules

Rule 1: if client-type = X and 49 $\le$ age[a] $\le$ 65 and 75 $\le$ bwt[b] $\le$ 105 and 9 $\le$ protein-difference[c] $\le$ 20, then
Action 1: add food, lunch-02E6-003, sausage roll, party one-piece 35 g altmsl[d] = dinner
Rule 2: if client-type = X and 49 $\le$ age $\le$ 65 and 49 $\le$ bwt $\le$ 75 and 1 $\le$ iron-difference $\le$ 5, then
Action 2: increase portion size, dinner-08A1-008, beef blade, gl one-piece 50 g altmsl = lunch
Rule 3: if client-type = X and 49 $\le$ age $\le$ 65 and 49 $\le$ bwt $\le$ 75 and -12 $\le$ fat-difference $\le$ -5 and 1 $\le$, then
Action 3: increase portion size, dinner-08A1-008, beef blade, gl one-piece 50 g altmsl = lunch
Action 3: delete food, lunch-09A1-002, milk, whole, fluid half-glass 122 g altml=dinner
Rule 4: if client-type = X and 49 $\le$ age $\le$ 65 and 49 $\le$ bwt $\le$ 75 and 10 $\le$ phosphorus-difference $\le$ 20, then
Action 2: add food, snack-09D1-004, ice cream, natural, van one-cup 70 g altmsl = dinner
Rule 5: if client-type = X and 49 $\le$ age $\le$ 65 and 49 $\le$ bwt $\le$ 75 and 15 $\le$ thiamine-difference $\le$ 25, then
Action 5: add food, dinner-02E6-008, meat pie, individual one-piece 140 g altmsl = snack

The client types (above as X) is a composition of multiple conditions including medical condition, meal-type (such as English, Indian, Chinese, etc.) and food habit (different types of vegetarian, etc.).

[a] A client's age in years.

[b] A client's body weight in kg.

[c] Total nutrients in menu—required nutrient amount.

[d] Alternate meal slot.

was refined by acquiring a new rule. The definitions of a rule and similar food are shown in Tables 2 and 3, respectively.

- Client 1: menu 11 was modified by the following actions suggested by rules 106 and 79, respectively.
    - Added: one piece of sausage roll to the breakfast slot to increase the carbohydrate level.
    - Deleted: scrambled egg from the breakfast to decrease iron level.

  Rule 79 was acquired as a refined rule of rule 4 while modifying menu 33. The suggested food was egg in the form of an omelette which was present in menu 33 but

Table 3
Definitions of similar food

smfr[a] 106: if food group[b] = 06D1 and 17 $\le$ carbohydrate-content[c] $\le$ 23, then similar food = yes
smfr 79: if food group = 03B1 and 163 $\le$ calorie-content $\le$ 173, then similar food = yes
smfr 52: if food group = 10D1 and 29 $\le$ fat-content $\le$ 39, then similar food = yes
smfr 115: if food group = l08C1 and 25 $\le$ protein-content $\le$ 33, then similar food = yes
smfr 237: if food group = 02A1 and 19 $\le$ carbohydrate-content $\le$ 29, then similar food = yes
smfr 238: if food group = 08F1 and 3 $\le$ iron-content $\le$ 9, then similar food = yes
smfr 284: if food group = 09A1 and 1 $\le$ carbohydrate-content $\le$ 10, then similar food = yes
smfr 60: if food group = 08C1 and 25 $\le$ carbohydrate-content $\le$ 35, then similar food = yes
smfr 275: if food group = 08C1 and 4 $\le$ carbohydrate-content $\le$ 8, then similar food = yes

[a] Similar food rule: a similar food is identified by its content of specific nutrients per 100 g of that food and by the food group.

[b] Food group consists of food subgroups, each subgroup contains many other foods. The food groups are taken from the Australian food group system [28].

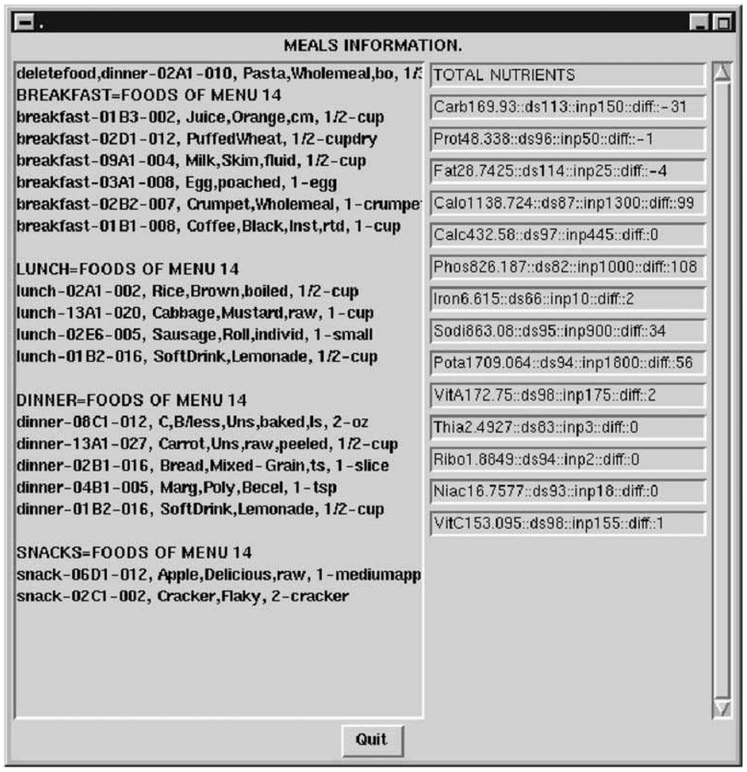
[c] Condition of nutrient and calorie content of similar food in a rule per 100 g of a food.

was absent in menu 11. The system deleted scrambled egg as a similar food from menu 11. The definition of this similar food was acquired from the expert when rule 79 was acquired while modifying menu 33.

- Client 2: menu number 11 was modified by the system as a result of applying the actions of rules 52, 115 and 79, respectively:
  - Added: half-ounce of cheese, blue-vein to the dinner slot to increase fat content of the diet.
  - Decreased: one serving portion of chicken, drum-stick from the dinner slot to decrease phosphorus content.
  - Added: half-ounce of cheese, blue-vein to the lunch slot to increase fat content.
  - Added: one pack of potato, straw to the lunch slot to increase fat content.
  - Deleted: scrambled egg from breakfast slot to decrease iron content.

  The protein requirement was less and fat requirement was more for client 2 than client 1. The suggested action of rule 115 was to decrease a portion size of food *turkey*, *breast*, *bk* from dinner. MIKAS did not find the suggested food or a similar food in the dinner meal slot. Hence, it checked for the suggested food in the alternate meal slot which is the lunch slot for rule 115. MIKAS found a similar food in the lunch slot, which was *chicken*, *drum-stick*, and decreased the portion size. Rule 52 tried to add *cheese*, *blue-vein* to the dinner. The food was already present in that meal slot, so the food was added to the lunch as an alternate meal slot. The next application of rule 52 tried to add again *cheese*, *blue-vein* to the dinner. By then the food was already present in the suggested meal slot and in the alternate meal slot, so MIKAS added *potato*, *straw* as an alternate food to the alternate meal slot. Finally, rule 79 tried to delete *omelette* from the breakfast slot. As the food was not present, a similar food, *scrambled egg*, was deleted from the breakfast.
- Client 3: menu number 14 was modified by the system as a result of the following actions suggested by rules 237, 238, 284, 248, 60 and 275, respectively.
  - Deleted: portion of pasta, whole-meal, boiled from the dinner slot to decrease carbohydrate content.
  - Added: one ounce of beef liver, simmered from the dinner slot to increase iron content.
  - Deleted: half-cup of milk, skim, fluid from the breakfast slot to decrease carbohydrate content.
  - Added: one ounce of beef liver, simmered to the lunch slot to increase iron content.
  - Decreased: one serving portion of chicken drum-stick, baked from dinner slot to decrease fat content.
  - Added: half-cup of pea, split, boiled to the lunch slot to increase protein content.

The suggested action of rule 284 was to delete *cup of milk*, *whole*, *fluid* from the lunch slot of the meal, but the suggested food or a similar food was absent in the meal slot. Hence, MIKAS did not find the suggested food in the alternate meal slot (breakfast slot) either, so the system deleted *half-cup of milk*, *skim*, *fluid* as a similar food from the alternate meal slot leaving cereal to be taken without milk. This action of the system was unacceptable. At this step the expert can refine rule 284 to delete both milk and cereal in the breakfast slot and take subsequently some other appropriate steps of adaptation. The



```
deletefood,dinner-02A1-010, Pasta,Wholemeal,bo, 1/
BREAKFAST=FOODS OF MENU 14
breakfast-01B3-002, Juice,Orange,cm, 1/2-cup
breakfast-02D1-012, PuffedWheat, 1/2-cupdry
breakfast-09A1-004, Milk,Skim,fluid, 1/2-cup
breakfast-03A1-008, Egg,poached, 1-egg
breakfast-02B2-007, Crumpet,Wholemeal, 1-crumpe
breakfast-01B1-008, Coffee,Black,Inst,rtd, 1-cup

LUNCH=FOODS OF MENU 14
lunch-02A1-002, Rice,Brown,boiled, 1/2-cup
lunch-13A1-020, Cabbage,Mustard,raw, 1-cup
lunch-02E6-005, Sausage,Roll,individ, 1-small
lunch-01B2-016, SoftDrink,Lemonade, 1/2-cup

DINNER=FOODS OF MENU 14
dinner-08C1-012, C,B/less,Uns,baked,ls, 2-oz
dinner-13A1-027, Carrot,Uns,raw,peeled, 1/2-cup
dinner-02B1-016, Bread,Mixed-Grain,ts, 1-slice
dinner-04B1-005, Marg,Poly,Becel, 1-tsp
dinner-01B2-016, SoftDrink,Lemonade, 1/2-cup

SNACKS=FOODS OF MENU 14
snack-06D1-012, Apple,Delicious,raw, 1-mediumapp
snack-02C1-002, Cracker,Flaky, 2-cracker
```

```
TOTAL NUTRIENTS
Carb169.93::ds113::inp150::diff::-31
Prot48.338::ds96::inp50::diff::-1
Fat28.7425::ds114::inp25::diff::-4
Calo1138.724::ds87::inp1300::diff::99
Calc432.58::ds97::inp445::diff::0
Phos826.187::ds82::inp1000::diff::108
Iron6.615::ds66::inp10::diff::2
Sodi863.08::ds95::inp900::diff::34
Pota1709.064::ds94::inp1800::diff::56
VitA172.75::ds98::inp175::diff::2
Thia2.4927::ds83::inp3::diff::0
Ribo1.8849::ds94::inp2::diff::0
Niac16.7577::ds93::inp18::diff::0
VitC153.095::ds98::inp155::diff::1
```

Fig. 6. A window displaying menu 14 from the case base.

acquired rule is added to the RDR tree. Fig. 6 shows menu 14 before any adaptation actions are taken.

5. Experimental studies

We developed a CBR system for diet construction using the presented incremental knowledge acquisition approach as implemented in MIKAS. In our experiments, we focussed on the development of the adaptation process of our CBR system. The retrieval function was also developed incrementally, but for most cases the retrieval function was not changed, even if the retrieved case appeared rather inappropriate. This was done to better evaluate the potential of building adaptation knowledge bases with our approach. We implemented our approach for diet construction in MIKAS. The main difference between our system and other diet construction tools is the in-built incremental knowledge acquisition capability. From our experiments so far we learned that the approach of incremental knowledge acquisition is a feasible approach towards building complex diet construction systems. We built a KB of more than 330 rules and we were able to recognise a clear trend: the larger the KB, the less automatically constructed diets needed to be manually modified. While our KB still cannot handle *all* possible client requirements, it continuously improves its competence. With every client requirement the system does not handle properly, the expert user will enter one or more new rules to increase the competence of the system. Our use of ripple-down rules ensures that MIKAS maintains its competence of handling all client requirements it has seen in the past and additionally is able to handle all newly encountered client requirements.

Fig. 7a shows how the proportion of actions being acceptable increases markedly with the KB size, and the growth rate of the KB reduces with increasing size. The menus counted were either directly retrieved from the case base or they were already partially adapted by the system and needed further adaptation. Both are shown, the number of rules which were added to the KB and the number of acceptable adaptation actions proposed by the system. Fig. 7a shows how the discrepancy between the two curves grows with increasing 'experience'. The increasing 'experience' here means a growing number of presented menus, for which the expert judged whether the system's KB proposed an appropriate adaptation action or not. After the KB grew to some 330 rules, it was already capable of suggesting adaptation steps which were acceptable in the majority of the presented cases. Given the large variety of patient requirements and retrieved menus, this result is very satisfactory.

While the current KB cannot handle the complete adaptation of every retrieved case, for virtually every patient of the considered patient types it can at least make some useful adaptation steps automatically. A system that assists an expert in semi-automatically constructing a suitable menu for a new patient is already a substantial help.

Fig. 7b shows a steady growth of the number of clients properly handled with an increased number of actions approved. The figure also indicates that an increased number of adaptation actions were performed with the increased size of KB.

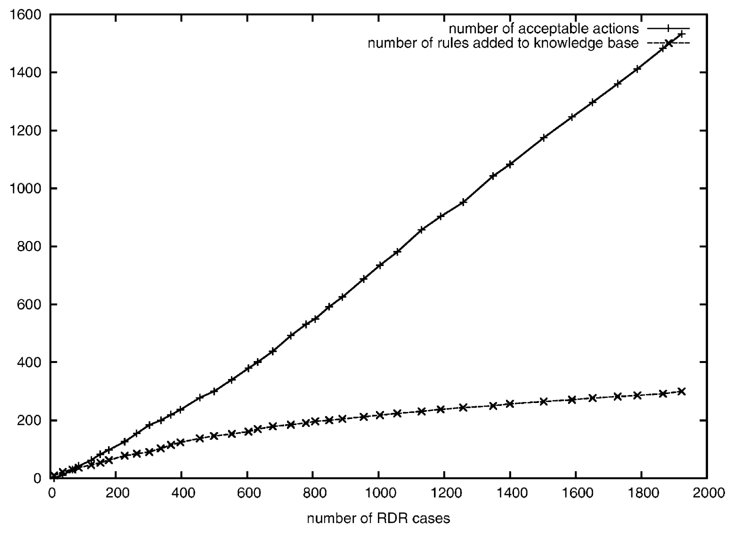
For a second patient type (ulcer patients requiring a liquid diet), we obtained the results shown in Fig. 8. The results show that a similar level of acceptable actions can at least as quickly be achieved as for the first type of patients (liver patients). Reason being that some of the rules acquired while correcting the system in constructing menus for liver patients are also applicable to other patient types. Fig. 8a and b show the competence level of our system.

For most retrieved cases it took 3–10 adaptation steps to turn a retrieved menu into a satisfactory menu for the patient at hand. We presented the constructed menus to a professional hospital dietitian who judged more than 95% of the adaptation actions taken by the system to construct menus are appropriate for the respective clients.

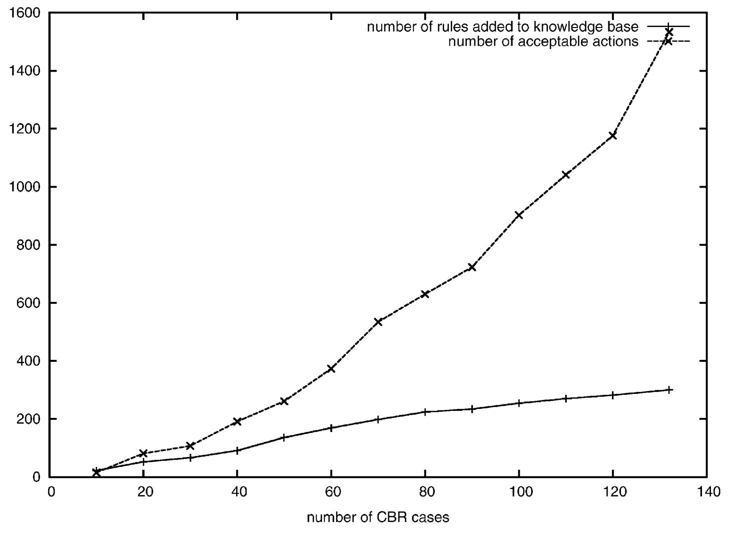
We conducted our experiments with a constant case base to find how well our approach allows to build really flexible adaptation functions. However, to build a fully fledged menu construction system, we would also add a number of the satisfactorily adapted menus to the case base to allow retrieval of a more suitable case for a similar dietary requirements.

5.1. The domain coverage of the acquired knowledge base

The menu construction and recommendation task is a complex design process as discussed above. The acquired knowledge in our prototype system mainly cover groups of patients who have liver or ulcer complications with some associated health problems. It

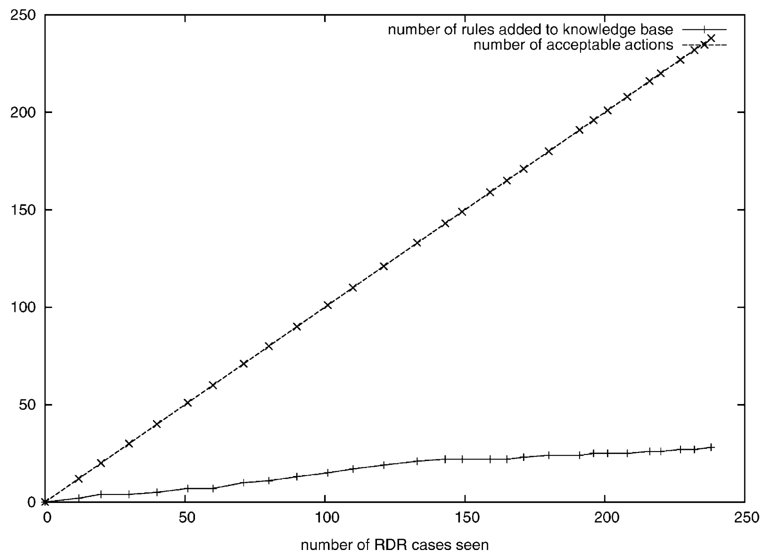


(a) RDR cases vs acceptable actions and number of rules added.

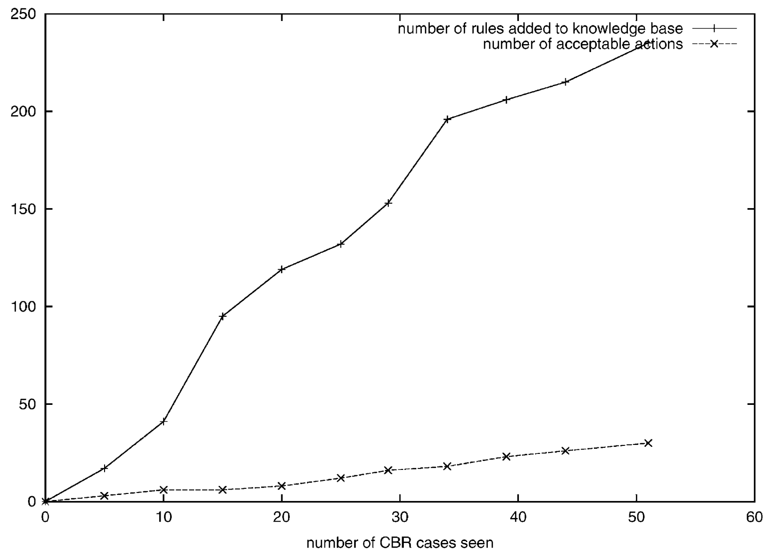


(b) CBR cases vs acceptable actions and the number of rules added.

Fig. 7. (a) The growth of the KB and its competence to handle adaptation with the number of menus presented. The population of cases was drawn from liver patients. A RDR case here is a menu that requires adaptation and is presented to the system and the expert. A retrieved menu (a CBR case) may require multiple adaptation steps, and thus would result in multiple presentations of modified menus to the expert. Accordingly, the various menu versions will be counted as multiple RDR cases. (b) How the competence of the adaptation KB in handling cases increases with experience. A single CBR case usually involves a sequence of multiple adaptation actions to be determined by the KB.



(c) RDR cases vs acceptable actions and the number of rules added.



(d) CBR cases vs acceptable actions and number of rules added.

Fig. 8. The further development of the KB for ulcer patients. The corresponding charts are shown as given in the previous Fig. 7. See there for further explanations. Much less new rules are required as quite a number of rules for the liver patients are also effective for ulcer patients.

means that during our case study two patient types were considered. The coverage of more patient types by the acquired adaptation knowledge might be possible, depending on the degree to which menus for other patient types require special treatment in all adaptation steps.

For example, if vegetarian patients need to be catered for, only some of the rules acquired for other patients would need refinement; those which introduce meat into the menu.

However, the worst case for scaling up the current KB to a more general coverage of patient types would be to add for each patient type their own set of rules. That is, replicating the effort that we undertook to cater for our groups of patients. Due to the structure of RDRs, no increased complexity and adverse interaction among a large number of rules is expected—a common problem with other approaches to organising knowledge bases. Hence, we consider the results of our case study as very encouraging indeed to allow the development of a more complete menu construction system.

6. Implications of the results for the dietary practice

CBord has a set of pre-designed menus from which it retrieves one on demand. These menus had to be manually constructed with the objective of being economical to prepare in the hospital kitchen as well as suiting most patients reasonably well. In the hospital environment, the feasibility of preparing the meals within the constraints given by the available time, space, personnel and other logistic issues takes priority over optimally tailoring a menu towards the needs and preferences of an individual patient. After all, a less than perfect meal is far better than an inappropriate meal.

CBord does not support the manual change and adaptation of a menu. To set up CBord to fit all the existing constraints, however, took about 18 months including training when CBord was introduced at the Prince of Wales Hospital in Sydney. A full-time dietitian was specifically employed for the task of developing 104 menu-types which are all stored in CBord's database and are retrieved on the basis of the rough calorie requirements for a patient and a limited number of major diseases, such as diabetes, which requires a special diet. Multiple menus may be retrieved for a single requirement, hence giving some choice to the kitchen and/or to the patient. However, no detailed tailoring of a menu towards the full set of medical conditions of a patient is supported—let alone the accommodation of a patient's specific food preferences.

In comparison to the effort that was necessary to make CBord operational in its hospital environment, the effort required by our approach MIKAS to build a menu construction system appears rather little.

Nevertheless, we consider the primary use and advantage of MIKAS over existing systems in the area of dietary advice to patients who leave the hospital as they do not have the constraints of the hospital environment with respect to cost and feasibility of meal preparation.

The strength of the menu construction system we built using MIKAS lies primarily in its flexibility being able to construct menus which are highly tailored to the needs of the individual clients. The menu construction system we built using MIKAS is not a fully fledged menu recommendation system to be used for constructing menus to be administered within the hospital environment. However, MIKAS has clearly the potential to be integrated into the existing IT environment of the Prince of Wales Hospital and other hospitals.

- The feasibility of entering new rules into MIKAS by the expert directly, appears generally feasible, after a short training period (training appears generally feasible within 1 week).
- We developed a KB of more than 330 rules, which proved to cover two different diseases where each of the diseases requires a special diet. The developed KB constructed menus for more than 80% of the patient descriptions we provided. The patient descriptions varied in the energy content of the required diet as well as in the requirements of a few special nutrients, such as iron (minimum requirement) and fat (maximum requirement). MIKAS was capable of satisfying those requirements rather well. The total time to enter rules into MIKAS' KB is around 5 min per rule, i.e. the total time taken from the professional dietitian would have been around 25–30 h.

  This compares very favourably to CBord, which required about 18 months of a full-time dietitian to set up a set of menus which are suitable for the local requirements in the Prince of Wales Hospital together with providing training for using the system.
- While the menu suggestions for the hospital kitchen generated by CBord are not very flexible, there is a limited use for a more flexible system, as the constraints given by the hospital environment would not allow to be too versatile in the various meals being served to patients. Hence, we see the primary use of MIKAS for situations where clients seek dietary advice in a face-to-face consultation with the dietitian where they would also discuss their food preferences. This includes also dietary advice given to patients leaving the hospital.
- After the development of our KB for MIKAS, more than 95% of the menus constructed and approved by MIKAS were judged to be acceptable by our hospital dietitian Dianne Muniz.
- After the dietitian modified a produced menu, MIKAS will be able to construct more suitable menus automatically by using the new rules which were added to the KB based on th provided explanations.
- Our experiments with MIKAS have shown that this is a very promising approach towards a powerful support tool for the dietary practice.

7. Conclusion

Our experiments showed that the acquisition of effective adaptation knowledge for a limited range of problem cases is effective. As our RDR-based approach avoids any adverse interaction of multiple rules in the knowledge base we believe that the approach scales up well. Hence, we expect that the current KB can be extended to cover other patient types without encountering any significant problem. The required knowledge can be provided by an expert and can be organised into a RDRs like structure with relative ease. Our approach offers an alternative to the traditional labour and time-intensive approaches for building CBR systems. We believe that the presented approach can also be adapted for allowing the development of CBR systems for many other design tasks, help desk systems and other applications, where CBR systems have been successfully employed recently (see e.g. [3,36]). The strengths of our approach include the following:

- It allows a domain expert to directly interact with the system without the need for a knowledge engineer or a system engineer. Adaptation rules are entered by the expert. The expert is guided by the system in providing suitable conditions for the rule's application.
- The system will continuously improve its performance during routine use, as the expert user will occasionally add more knowledge to the system's KB.
- The expert is only required to explain, why certain solutions should be adapted in the demonstrated way for a given problem. This is much easier than to provide general rules for adaptation. In our approach, the expert needs only to justify the particular adaptation steps he chose.
- If an explanation results in overly general adaptation rules, this will be repaired as soon as the expert encounters a situation, in which the system would perform unsuitable adaptations.

Finally, we consider our approach to build a CBR system to have significant potential to be productively applied for developing other case-based applications in a variety of areas, including many medical applications.

Acknowledgements

We are deeply indebted to Dianne Muniz, senior clinical dietitian at the Prince of Wales Hospital in Sydney for her kind assistance in evaluating MIKAS and the diet construction system we built with it.

References

[1] Althoff K-D, Bergmann R, Wess S, Manago M, Auriol E, Larichev O, et al. Case-based reasoning for medical decision support tasks: the INRECA approach. Artif Intell Med 1998;12:25–41.

[2] Armenogo E, Palaudaries A, Plaza E. Individual prognosis of diabetes long-term risks: a CBR approach. Methods Inf Med 2001;40:46–51.

[3] Bandini S, Manzoni S. A support system based on CBR for the design of rubber compounds in motor racing. In: Blanzieri E, Portinale L, editors. Proceedings of the European Workshop on Case-Based Reasoning, EWCBR00. Berlin (Germany): Springer; 2000. p. 348–57.

[4] Bergmann R, Breen S, Göker M, Manago M, Wess S. Developing industrial case-based reasoning applications: the INRECA methodology. Lecture notes in artificial intelligence, state-of-the-art-survey, LNAI 1612. Berlin (Germany): Springer; 1999.

[5] Bergmann R, Breen S, Fayol E, Göker M, Manago M, Schmitt S, et al. Collecting experience on the systematic development of CBR applications using the INRECA-II methodology. In: Smith B, Cunningham P, editors. Proceedings of the European Workshop on Case-Based Reasoning, EWCBR98. Berlin (Germany): Springer; 1998. p. 460–70.

[6] Beydoun G, Hoffmann A. Incremental acquisition of search knowledge. Int J Hum Comp Stud 2000;52(3):493–530.

[7] Breuker JA, Van de Velde W, editors. The common KADS library for expertise modelling. Amsterdam (The Netherlands): IOS Press; 1994.

[8] Chandrasekaran B. Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. IEEE Expert 1986;3(1):23–30.

[9] Colavita C, D'Orsi R. Linear programming and pediatric dietetics. Br J Nutr 1990;64:307–17.

[10] Compton P, Edwards G, Kang B, Lazarus L, Malor R, Preston P, et al. Ripple down rules: turning knowledge acquisition into knowledge maintenance. Artif Intell Med 1992;4:463–75.

[11] Craw S, Wiratunga N, Rowe R. Case-based design for tablet formulation. In: Smith B, Cunningham P, editors. Proceedings of the 4th European Workshop on Case-Based Reasoning EWCBR98. Berlin (Germany): Springer; 1998. p. 358–69.

[12] David S, Dariel I. Diet planning for humans using mixed-integer linear programming. Br J Nutr 1993;70:27–35.

[13] Edwards G, Compton P, Malor R, Srinivasan A, Lazarus L. PEIRS: a pathologist maintained expert system for the interpretation of chemical pathology reports. Pathology 1993;25:27–34.

[14] Fletcher L, Soden P. Diet construction using linear programming. Diabetes Nutr Metab 1991;4(Suppl):169–74.

[15] Fletcher L, Soden P. Linear programming techniques for the construction of palatable human diets. J Operat Res Soc 1994;45:489–96.

[16] Haddad M, Adlassnig K-P, Porenta G. Feasibility analysis of a case-based reasoning system for automated detection of coronary heart disease from myocardial scintigrams. Artif Intell Med 1997;9:61–78.

[17] Hammond KJ. CHEF: a model of case-based planning. In: Weld DS, de Kleer J, editors. Proceedings of the 5th National Conference on Artificial Intelligence, AAAI-86. Cambridge (MA): AAAI Press; 1986. p. 267–71.

[18] Henson S. Linear programming analysis of constraints upon human diets. J Agric Econ 1991;42:380–93.

[19] Hinrichs TR. Strategies for adaptation and recovery in a design problem solver. In: Buchanan BG, Shortliffe EH, editors. Proceedings of the Workshop on Case-Based Reasoning (DARPA). San Mateo (CA): Morgan Kaufmann; 1989. p. 115–8.

[20] Hinrichs TR. Problem solving in open worlds: a case study in design. Northvale (NJ): Lawrence Erlbaum; 1992.

[21] Hinrichs TR, Kolodner JL. The role of adaptation in case-based design. In: Proceedings of the AAAI-91. Cambridge (MA): AAAI Press/MIT Press; 1991. p. 34–9.

[22] Horn W, Popow C, Miksch S, Kirchner L, Seyfang A. Development and evaluation of VIE-PNN, a knowledge-based system for calculating the parenteral nutrition of newborn infants. Artif Intell Med 2002;24:217–28.

[23] Kang B, Compton P. A maintenance approach to case-based reasoning. In: Haton J, Keane K, Manago M, editors. Advances in case-based reasoning. Berlin (Germany): Springer; 1995. p. 226–39.

[24] Khan AS, Hoffmann A. A new approach for the incremental development of adaptation functions for CBR. In: Blanzieri E, Portinale L, editors. Proceedings of the Case-Based Reasoning of 5th European Workshop, EWCBR00. Berlin (Germany): Springer; 2000. p. 260–72.

[25] Khan AS, Hoffmann A. Acquiring adaptation knowledge for CBR with MIKAS. In: Stumptner M, Corbett D, Brooks M, editors. Advances in artificial intelligence, AI 2001. Berlin (Germany): Springer; 2001. p. 201–12.

[26] Leake D. Combining rules and cases to learn case adaptation. In: Proceedings of the 17th Annual Conference of the Cognitive Science Society. Hillsdale (NJ): Lawrence Erlbaum; 1995. p. 84–9.

[27] Leake D, Kinley A, Wilson D. Multistrategy learning to apply cases for case-based reasoning. In: Proceedings of the 3rd International Workshop on Multistrategy Learning. Menlo Park (CA): AAAI Press; 1996. p. 155–64.

[28] Lewis J, Milligan G, Hunt A. Nuttab95: nutrient data table for use in Australia. Canberra (Australia): Australian Government Publishing Service.

[29] Lopez B, Plaza E. Case-based learning of plans and goal states in medical diagnosis. Artif Intell Med 1997;9:29–60.

[30] Manago M, Bergmann R, Wess S, Traphoener R. CASUEL: a common case representation language. Technical report. INRECA Consortium, University of Kaiserslautern, 1994.

[31] Munoz-Avilla H, Hendler J, Aha DW. Conversational case-based planning. Rev Appl Expert Syst 1999;5:163–74.

[32] Ochi-Okorie AS. Disease diagnosis validation in TROPIX using CBR. Artif Intell Med 1998;12:43–60.
[33] Petot G, Marling C, Sterling L. An artificial intelligence system for computer-assisted menu planning. J Am Diet Assoc 1996;98:1009–14.
[34] Richards D, Compton P. Revisiting sisyphus I—an incremental approach to resource allocation using ripple down rules. In: Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management. Banff (Canada): SRDG Publications; 1999. p. 7.1–7.21.
[35] Rowe R, Craw S, Wiratunga W. Case-based reasoning—a new approach to tablet formulation. Pharm Technol Eur 1999;11(2):36–40.
[36] Schmidt R, Gierl L. Evaluation strategies for generalised cases within a case-based reasoning antibiotics therapy advice system. In: Blanzieri E, Portinale L, editors. Proceedings of the European Workshop on Case-Based Reasoning, EWCBR00. Berlin (Germany): Springer; 2000. p. 491–503.
[37] Schmidt R, Gierl L. Case-based reasoning for antibiotics therapy advice: an investigation of retrieval algorithms and prototypes. Artif Intell Med 2001;23:171–86.
[38] Schreiber AT, Wielinga BJ, Breuker JA, editors. KADS: a principled approach to knowledge-based system development. Knowledge-based systems book series, vol. 11. London (UK): Academic Press; 1993.
[39] Soden P, Fletcher L. Modifying diets to satisfy nutritional requirements using linear programming. Br J Nutr 1992;68:565–72.
[40] CBord: web page: http://www.cbord.com.
[41] Compu-cal: web page: http://www.compu-cal.com.
[42] Deluxe: web page: http://www.ncconcepts.com.
[43] Food Works: web page: http://www.NutritionCo.com.
[44] Food Works (Tm): web page: http://www.xyris.com.au.
[45] Nutri-Calc: web page: http://www.nutri-calc.com.
[46] Nutribase: web page: http://www.nutribase.com.
[47] Nutritionist Pro: web page: http://www.firstdatababk.com.
[48] Serve: web page: http://www.serve.com.au.