Developing an Expert System for Diet Recommendation

Gergely Kovásznai
Department of Information Technology
Eszterházy Károly College
Eger, Hungary
Email: kovasz@aries.ektf.hu

***Abstract*—Menu construction is an important task for institutions that need to plan menus within certain constraints. There is also a personal need for professional menu construction by clients or patients who should eat according to a planned diet. For menu construction and dietary analysis, there are several approaches (e.g., linear programming, genetic algorithms, rule-based expert systems, etc.) and commercial IT systems. In this paper, we propose a case-based approach for diet recommendation. Based on this approach, we are going to construct an expert system which is intended to be employed in a health record management system. Our approach is based on ripple down rules (RDR), however, a special representation is also needed for patient attributes and rule actions.**

I. Introduction

Menu construction is an important task for schools, hospitals, nursing homes, etc., since such institutions need to plan menus within certain constraints such as available material, equipment, and financial means. There is also a personal need for professional menu construction by clients or patients who should eat according to a planned diet, due to certain reasons such as medical conditions, or desire to become or stay healthy.

Numerous IT systems and approaches are available today for dietetic applications, and, primarily, for menu construction and dietary analysis. There are several approaches based on linear programming [1]–[3], genetic algorithms [4], [5], rule-based or case-based expert systems [6], [7]. A lot of commercial menu planning systems exist, and are available [8]–[12].

In this paper, we propose a *case-based* approach for diet recommendation. Based on this approach, we are going to construct an expert system which is intended to be employed in a health record management system. The goal of the project is to develop an IT system (called the *eFilter* system) which filters the collection of available food products, considering the data stored in the health records of clients or patients. Health records include information about food intolerance, allergy, and diet. An important module of this system is going to implement menu planning. In Section II, we suggest an approach for case-based knowledge acquisition, namely the *ripple down rules (RDR)*. In Section III, we propose a representation of rules for the RDR approach within our expert system, i.e., we discuss patient attributes and rule actions. In Section IV, we focus on implementation.

II. Knowledge Acquisition

Knowledge acquisition seems to be the bottleneck in expert system development. Two experts have to be involved, a domain expert and a knowledge engineer (both highly paid specialists). They need to work together for extended periods to develop the knowledge base.

By applying *case-based* reasoning systems, we get an approach which does not require a knowledge engineer during the knowledge acquisition process.

A. Case-based Reasoning

The basic idea of case-based reasoning is to solve new problems by remembering solutions to problems which are similar to the current problem. Of course, it is domain-specific which problems are considered similar to each other. To be more precise, a domain expert is the one who can make such a decision. As it will be mentioned in Section II-B, *ripple down rules* provide a very natural interface for the domain expert to specify the appropriate similarity relation, case by case.

A major practical advantage of case-based reasoning is the fact that domain experts cannot usually specify explicitly why a case is similar to another case, but they rather refer to cases they have encountered during their careers. That is, by using case-based reasoning, domain experts are not forced to provide abstract general rules of what they do.

The major problems of case-based reasoning systems are

1) how to retrieve relevant cases, and
2) how to adapt them to fit a new problem.

B. Ripple Down Rules

By using ripple down rules (RDR) [13], the object space (i.e., the sum of all the object classes) is to be incrementally subdivided into smaller and smaller partitions. The goal is to classify all objects from the same class to the same partition.

The process of the above-mentioned subdivision is very simple. Assume an object x which has been classified incorrectly, into the partition p. In such a case, the partition p needs to be subdivided into two partitions p_1 and p_2, such that the partition to which x belongs classifies x correctly. The domain expert himself/herself can provide such a subdivision competently, with minimal effort. In RDR, the domain expert is merely required to formulate an explanation or criteria why x differs

978-1-4244-9109-4/11/$26.00 ©2011 IEEE

from the object x_p, which has previously led to the creation of the partition p.

For a domain expert, it is usually quite easy to provide such an explanation, since it is not much different to explaining their decision to colleagues. The basic idea of RDR is to allow the domain expert to directly construct the knowledge base and to *incrementally* add rules to it.

Using these explanations, the system builds an RDR tree. Such a tree can be seen in Figure 1. In each node of the tree a logical condition and a conclusion (i.e., a partition) are stored. Each node has at most two children. To one of the children an "except"/"true" link is drawn, and an "else"/"false" link to the other child node.

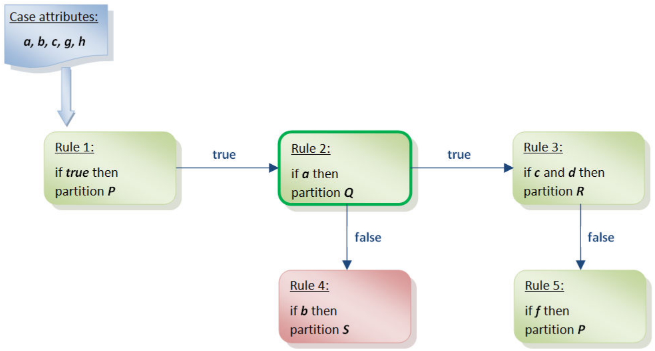


Fig. 1. An RDR tree.

The classification algorithm checks the condition of subsequent nodes. Let C denote the condition and P the partition of the current node. If C is satisfied then the algorithm classifies the input into P. However, this is not necessarily the final verdict, since the node may have an outgoing "except"/"true" link. It it has such a link then the algorithm goes along this link, and the classification process continues; otherwise, the classification ends (i.e., the last verdict gets accepted).

Something different happens when C is not satisfied: the algorithm checks if the node has an outgoing "else"/"false" link. If it does then the algorithm goes along this link, and the classification process continues; otherwise, the classification ends.

For instance, using the RDR tree and the input case that can be seen in Figure 1, the input is first classified into the partition P, by Rule 1. However, this is not the final verdict since Rule 1 has an outgoing "true" link. Since the condition of Rule 2 is also satisfied, the input is temporarily classified into the partition Q. Since the condition of Rule 3 is not satisfied, the algorithm follows the "false" link. The condition of Rule 5 is not satisfied, hence, the classification process ends, and classifies the input into the partition Q.

If the domain expert does not agree with the result of the classification, the system asks for

- the partition to the input should belong;
- a justification in terms of the input's attributes.

Then, the system adds a new node (i.e., rule) into the RDR tree. The condition of this node is based on the justification that the domain expert has provided.

III. Knowledge Representation

As it has been proposed, we represent the dietetic knowledge as an RDR tree. However, we should decide how to deal with the following questions.

- What attributes to use for the cases?
- What kind of conditions to use in the nodes?
- What kind of partitions to use?

A. Attributes

Attributes can be numerical, string, or boolean values. For example, `weight` is a numerical attribute. The `lifestyle` of a patient could be a string, with such values as "sedentary" or "critically ill". A boolean attribute `hasDiabetes` could designate whether the patient suffers from diabetes.

In the eFilter health record management system, the following attributes describe a patient:

- `height` – integer
- `weight` – integer
- `bmi` – body mass index (BMI), automatically calculated from `height` and `weight`
- `age` – floating-point number
- `sex` – string or enumerated ("male", "female")
- `lifestyle` – string or enumerated ("sedentary", "sporty", "critically ill", "handicapped")
- `hasDisease("...")` – This is not an attribute, but rather a function, which takes the name of a disease as a patameter, and returns a boolean value.

B. Conditions

Rules' conditions are to be formulated on the above-mentioned attributes. For instance, one can restrict that `weight` must be not less than 80 kilograms (`weight>=80`), `lifestyle` must be "critically ill" (`lifestyle="critically ill"`), or it is to refer only those patients who suffer from protein intolerance (`hasDisease("protein intolerance")`).

Another question is whether all logical connectives should be permitted in such conditions, or only a few of them? For the sake of the simplicity of the user interface (which is going to be used by a dietitian), we recommend only to use conjunction (i.e., logical "and") for connecting the atomic conditions.

C. Partitions

The most delicate question is how to represent partitions? As a matter of fact, what kind of object or collection of objects should be regarded as a partition, in this dietetic problem? We propose an approach that is similar to the one in [14], hence each partition is represented as a *set of actions*. Three types of actions are distinguished:

- *Add-action*. The action adds a new constraint on a certain component (e.g., carbohydrate, fat, protein) [15]. Such a constraint must specify the following data:

  - `nutrient` – The name of the certain nutrient, e.g., "protein", "fat", "carbohydrate", any vitamin, any mineral, "water", "cholesterol", "fiber", etc. In the eFilter system, it is also allowed to store any food name (e.g., "walnut", "pork meat") in this slot.
  - `direction` – The way how the above-mentioned nutrient is constrained. In the eFilter system, four values are used:
    * "prohibited"
    * "not recommended"
    * "recommended"
    * "highly recommended"
  - `count` – floating-point number
  - `period` – Number of days. Default value: 1 day. This is also a floating-point number.

  Of course, in the case of prohibition, `count` and `period` are not taken into consideration.

- *Delete-action.* It deletes all constraints on a certain component.
- *Replace-action.* When a constraint has already been added on a certain component, the user may replace it with another. A replace-action can be composed of a delete-action and an add-action.

Such an RDR tree can be seen in Figure 2. As can be seen, a "default" node is needed, which contains a constant true condition and no actions. The rules which check the sex of the patient specify default constraints on protein and carbohydrate. The case when the patient is male gets developed by checking whether the patient has a sporty lifestyle. If he has then it is highly recommended for him to consume increased amount of protein; otherwise the system checks if he is an overweighted kid. If he is then it is not recommended for him to consume fat more than 20 grams; otherwise it is checked whether he suffers from milk allergy. If he does then cow milk is prohibited for him, but he should use some other product (e.g., rice milk) to substitute cow milk [16].

D. Incremental Acquisition of Knowledge

Let us assume a patient who is an overweighted boy. For him, the system recommends to consume about 56 grams of protein and 200 grams of carbohydrate daily, and not to exceed 20 grams of fat. However, the dietitian, who is using the system, is not satisfied by this result, since he/she knows that the patient actually suffers from diabetes, thus, he should consume less carbohydrate [17]. This is why the dietitian adds a new rule to the system, by specifying a new condition (`hasDisease("diabetes type 2")`) and a new replace-action (`"carbohydrate" "not recommended" 130`). The new rule gets added as can be seen in Figure 3.

IV. Implementation

We have accomplished to implement the RDR engine of the proposed system, in C# programming language for .NET Framework. We have also developed a simple graphical user interface, that employs the Windows Forms API.

The user interface is divided into three regions, as shown in Figure 4 and described in the following sections.

A. Case Attributes

The user can feed the attribute values of the current case into the system, by using the left-hand side panel. The value of the attribute `bmi` gets calculated automatically.

When all attribute values have been keyed in, user presses the button "Classify", in order to execute the classification algorithm on the underlying RDR tree.

B. Dietetic Constraints

As the result of the classifying of the current case, dietetic constraints are listed in the middle panel. Such a constraint can formalize recommendation or prohibition on certain nutrients, as it has been proposed in Section III-C.

If the current case cannot be classified or the user does not agree with the result of the classification, a new rule (i.e., a new node) is added into the RDR tree, by using the dialog window in Figure 5. Each node contains

- a *condition*, which is a boolean expression;
- a list of *actions*;
- a *cornerstone case*, which is actually the current case.

C. RDR Tree View

On the right-hand side panel, the RDR tree is visualized, by the help of an underlying Graphviz engine.

In Figure 6, the recommended dietetic constraints and the resulting RDR tree can be seen after adding the rule in Figure 5.

V. Conclusion

Several problems have to be solved during the designing, specifying, and implementing of an IT system for menu construction and dietary analysis. We have focused on diet recommendation, and proposed a case-based approach. Using ripple down rules (RDR) and suitable rule representation, our approach is going to be applicable in the health record management system funded by and developed within the scope of the project *eFilter*[1].

References

[1] P. M. Soden and L. R. Fletcher, "Modifying diets to satisfy nutritional requirements using linear programming," *British Journal of Nutrition*, vol. 68, pp. 565–572, 1992.

[2] D. Sklan and I. Dariel, "Diet planning for humans using mixed-integer linear programming," *British Journal of Nutrition*, vol. 70, pp. 27–35, 1993.

[3] S. Henson, "Linear programming analysis of constraints upon human diets," *Journal of Agricultural Economics*, vol. 42, pp. 380–393, 1991.

[4] B. Gaál, I. Vassányi, and G. Kozmann, "A novel artificial intelligence method for weekly dietary menu planning," *Methods of Information in Medicine*, vol. 44, pp. 655–664, 2005.

[5] B. K. Seljak, "Computer-based dietary menu planning," *Journal of Food Composition and Analysis*, vol. 22, pp. 414–420, 2009.

[1] Participants: WIT-SYS Consulting Co. Ltd. (Budapest, Hungary), Eszterházy Károly College (Eger, Hungary), University of Miskolc (Miskolc, Hungary)

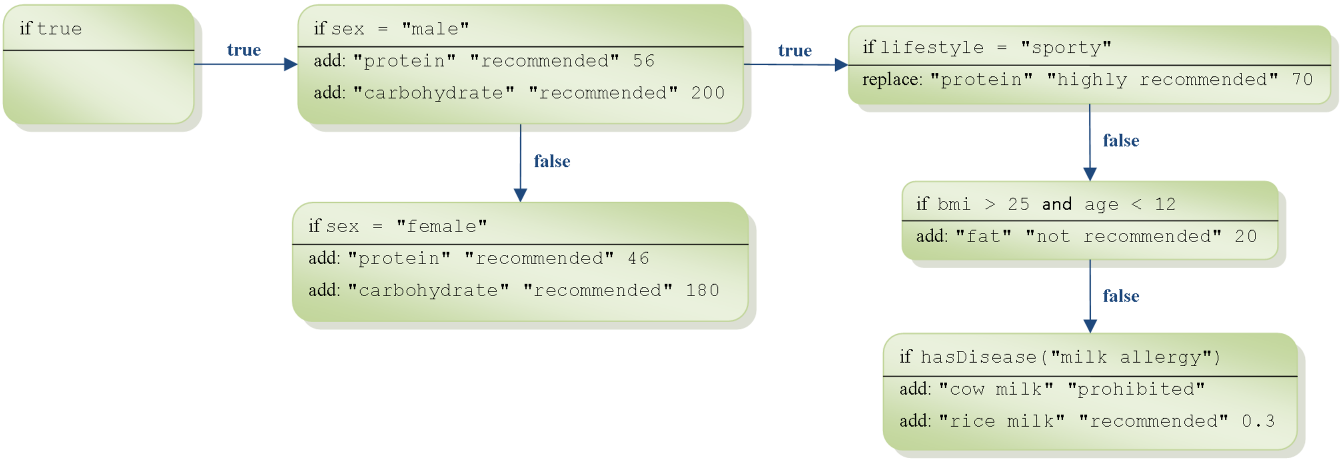

if true
true
if sex = "male"
add: "protein" "recommended" 56
add: "carbohydrate" "recommended" 200
true
if lifestyle = "sporty"
replace: "protein" "highly recommended" 70
false
false
if sex = "female"
add: "protein" "recommended" 46
add: "carbohydrate" "recommended" 180
if bmi > 25 and age < 12
add: "fat" "not recommended" 20
false
if hasDisease("milk allergy")
add: "cow milk" "prohibited"
add: "rice milk" "recommended" 0.3

Fig. 2. Dietetic knowledge as an RDR tree.

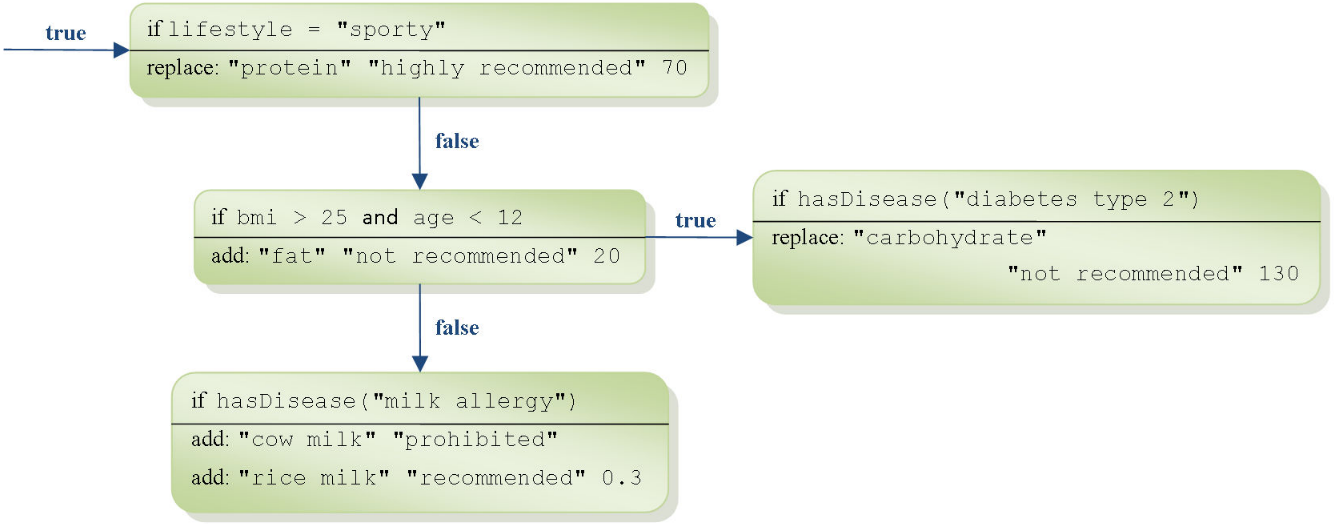

true
if lifestyle = "sporty"
replace: "protein" "highly recommended" 70
false
if bmi > 25 and age < 12
add: "fat" "not recommended" 20
true
if hasDisease("diabetes type 2")
replace: "carbohydrate"
"not recommended" 130
false
if hasDisease("milk allergy")
add: "cow milk" "prohibited"
add: "rice milk" "recommended" 0.3

Fig. 3. Dietetic knowledge as an RDR tree – New rule added.

[6] G. J. Petot, C. Marling, and L. Sterling, "An artificial intelligence system for computer-assisted menu planning," *Journal of the American Dietetic Association*, vol. 98, pp. 1009–1014, 1998.

[7] S. A. Noah, S. N. Abdullah, S. Shahar, H. Abdul-Hamid, N. Khairudin, M. Yusoff, R. Ghazali, N. Mohd-Yusoff, N. S. Shafii, and Z. Abdul-Manaf, "Dietpal: A web-based dietary menu-generating and management system," *Med Internet Res*, vol. 6, no. 1, p. e4, 2004.

[8] "Computer planned nutrition," 2010. [Online]. Available: http://www.ncconcepts.com/

[9] "Nutribase," 2010. [Online]. Available: http://www.nutribase.com/

[10] "Cnmenus." [Online]. Available: http://www.cnmenus.com/

[11] "Lunchbox nutritional analysis module." [Online]. Available: http://www.lunchbox-k12.com/

[12] "Menus & nutrition inventory management." [Online]. Available: http://www.mcssoftware.com/products/inventory

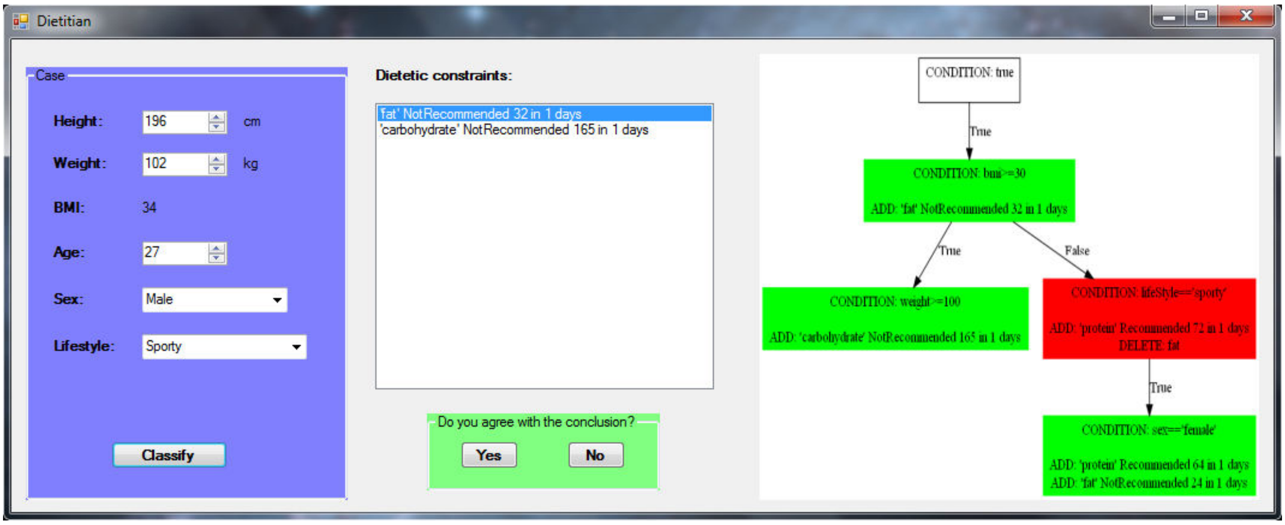
[13] P. Compton, G. Edwards, B. Kang, L. Lazarus, R. Malor, P. Preston, and A. Srinivasan, "Ripple down rules: Turning knowledge acquisition into knowledge maintenance," *Artificial Intelligence in Medicine*, vol. 4, no. 6, pp. 463–475, 1992.

[14] A. S. Khan and A. Hoffmann, "Building a case-based diet recommendation system without a knowledge engineer," *Artificial Intelligence in Medicine*, vol. 27, pp. 155–179, 2003.

[15] "List of nutrients," 2009–2011. [Online]. Available: http://nutrient.javalime.com/

[16] J. A. McMillan, R. D. Feigin, C. DeAngelis, and M. D. Jones, *Oski's Pediatrics*. Lippincott Williams & Wilkins, 2006.

[17] A. Accurso, R. K. Bernstein, A. Dahlqvist, B. Draznin, R. D. Feinman, E. J. Fine, A. Gleed, D. B. Jacobs, G. Larson, R. H. Lustig, A. H. Manninen, S. I. McFarlane, K. Morrison, J. V. Nielsen, U. Ravnskov, K. S. Roth, R. Silvestre, J. R. Sowers, R. Sundberg, J. S. Volek, E. C. Westman, R. J. Wood, J. Wortman, and M. C. Vernon, "Dietary carbohydrate restriction in type 2 diabetes mellitus and metabolic syndrome: time for a critical appraisal," *Nutrition & Metabolism*, vol. 5, no. 9, 2008.


Dietitian
Case
Classify
Dietetic constraints:
Do you agree with the conclusion?
Yes
No
CONDITION: true
True
CONDITION: bmi>=30
ADD: 'fat' NotRecommended 32 in 1 days
True
False
CONDITION: weight>=100
ADD: 'carbohydrate' NotRecommended 165 in 1 days
CONDITION: lifeStyle=='sporty'
ADD: 'protein' Recommended 72 in 1 days
DELETE: fat
True
CONDITION: sex=='female'
ADD: 'protein' Recommended 64 in 1 days
ADD: 'fat' NotRecommended 24 in 1 days

Height: 196 cm

Weight: 102 kg

BMI: 34

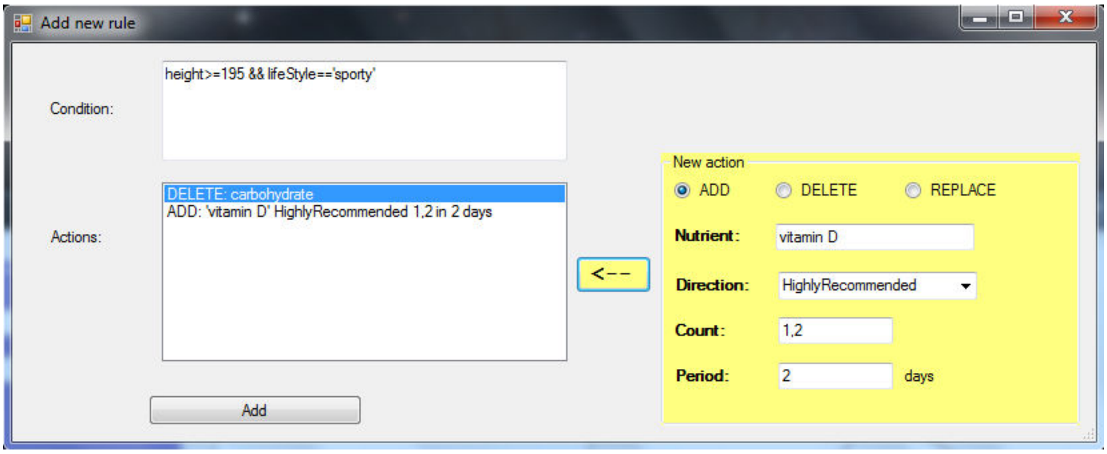
Age: 27

Sex: Male

Lifestyle: Sporty

'fat' NotRecommended 32 in 1 days
'carbohydrate' NotRecommended 165 in 1 days

Fig. 4. The main screen of the system.



Condition: height>=195 && lifeStyle=='sporty'

Actions:
DELETE: carbohydrate
ADD: 'vitamin D' HighlyRecommended 1,2 in 2 days

● ADD ○ DELETE ○ REPLACE

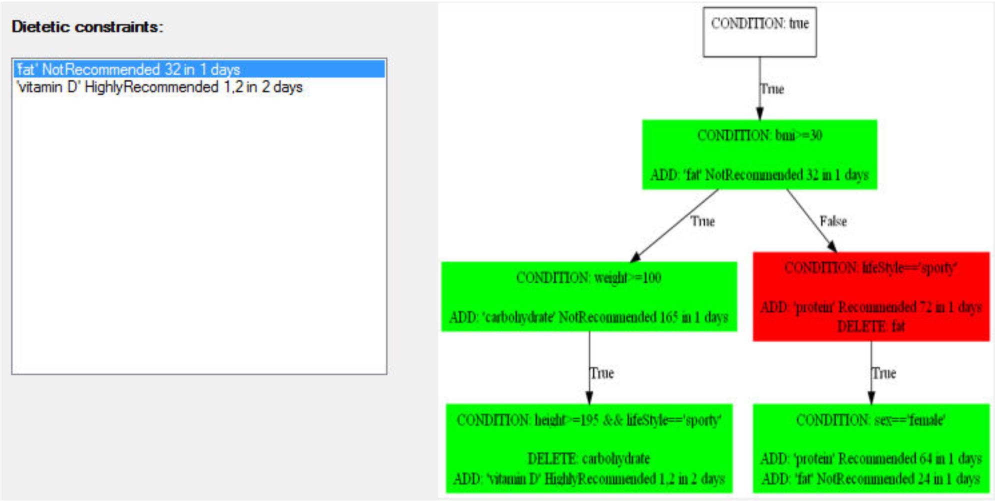
Nutrient: vitamin D

Direction: HighlyRecommended

Count: 1,2

Period: 2 days

Fig. 5. Adding a new rule into the RDR knowledgebase.


Dietetic constraints:
CONDITION: true
True
CONDITION: bmi>=30
ADD: 'fat' NotRecommended 32 in 1 days
True
False
CONDITION: weight>=100
ADD: 'carbohydrate' NotRecommended 165 in 1 days
CONDITION: lifeStyle=='sporty'
ADD: 'protein' Recommended 72 in 1 days
DELETE: fat
True
True
CONDITION: height>=195 && lifeStyle=='sporty'
DELETE: carbohydrate
ADD: 'vitamin D' HighlyRecommended 1,2 in 2 days
CONDITION: sex=='female'
ADD: 'protein' Recommended 64 in 1 days
ADD: 'fat' NotRecommended 24 in 1 days

'fat' NotRecommended 32 in 1 days
'vitamin D' HighlyRecommended 1,2 in 2 days

Fig. 6. The recommended dietetic constraints and the resulting RDR tree.