Novi Sad J. Math.
Vol. 35, No. 1, 2005, 25-40

CaseBaseGenerator for Intelligent Systems[1]

Vladimir Kurbalija [2], Mirjana Ivanović[2]

Abstract. CBG - "Casebase Generator" is a decision support system recently developed at Department of Mathematics and Informatics, University of Novi Sad. The technology used for implementing this system is Case-Based Reasoning (CBR). CBR is relatively new and promising area of artificial intelligence where every new problem is solved by adapting the solutions of the previously successfully solved problems. In the past few years CBR has become a popular technique for knowledge-based systems in different domains because the experience is included in solving every new problem. The purpose of this system is to be the main engine of some future, more specialized systems. "Case Base Generator" system is partly described in this paper and some future specialization are analyzed.

Key words and phrases: Intelligent Systems, Databases and Information Retrieval, Case Based Reasoning.

1. Introduction

During the last two years in the area of CBR at the Department of Mathematics and Informatics, several systems have been developed [5], [6]. Current direction is connected to realization of "Casebase Generator" - a decision support system based on CBR. The main attention of the authors was to create the core system which could be used in several more specific intelligent systems. "Casebase Generator" was exactly the result of these intentions, and first step in realization of different decision support systems based on CBR engine.

Case-Based Reasoning, a special area of artificial intelligence, is considered as a problem solving technology (or technique). This technology is used for solving problems in domains where experience plays an important role [1],[2],[7],[9].

Generally speaking, case-based reasoning is applied for solving new problems by adapting solutions that worked for similar problems in the past. The main supposition here is that similar problems have similar solutions. The basic scenario for mainly all CBR applications is: *In order to find a solution of an actual problem, one looks for a similar problem in an experience base, takes the solution from the past and uses it as a starting point to find a solution to the actual problem.*

[1]Research is supported by the project: "Development of (intelligent) techniques based on software agents for application in information retrieval and workflows", Ministry of science, Technologies, and Development, Republic of Serbia.(Project No. 1844).

[2]Department of Mathematics and Informatics, Faculty of Science, University of Novi Sad, Trg D. Obradovica 4, Novi Sad 21000, Serbia and Montenegro, E-mail: kurba, mira@im.ns.ac.yu

In CBR systems experience is stored in form of cases. The case is a recorded situation where problem was totally or partially solved, and it can be represented as an ordered pair (problem, solution). The whole experience is stored in case base, which is a set of cases and each case represents some previous episode where the problem was successfully solved.

The knowledge of the CBR system is stored in different knowledge containers. The knowledge container is the structural element, which contains some quantity of knowledge. The idea of the knowledge container is totally different from the traditional module concept in programming. While the module is responsible for a certain subtask, the knowledge container does not complete the subtask but contains some knowledge relevant to many tasks. Even small tasks require the participation of each container. The concept of the knowledge container is similar to concepts of the nodes and propagation rules in neural networks.

In case-based reasoning several knowledge containers can be identified:

a) the vocabulary used;

b) the similarity measure;

c) the case base; and

d) the solution transformation.

In principle, each container can carry almost all knowledge available. From a software engineering point of view there is another advantage of case-based reasoning - the content of the containers can be changed locally. This means that manipulations on one container have little consequences on the other ones. As a consequence, maintenance operations [4], [11] are easier to be performed then on classical knowledge based systems.

The task of machine learning is to improve a certain performance using some experience or instructions. In inductive learning, problems and good solutions are presented to the system. The major desire is to improve a general solution method in every inductive step. Machine learning methods can be used in order to improve the knowledge containers of a case-based reasoning system (the case base, similarity measures and the solution transformation). However, one of the greatest advantages of the case-based reasoning system is that it can learn even through the work with users modifying some knowledge containers.

The main problem in implementing almost every CBR system is to find a good similarity measure - the measure that can tell in what extent the two problems are similar. In the functional way similarity can be defined as a function $sim : U \times CB \rightarrow [0, 1]$ where U refers to the universe of all objects (from a given domain), while CB refers to the case base (objects which were examined in the past and saved in the case memory). The higher value of the similarity function means that these objects are more similar.

The case-based reasoning system has not the only goal of providing solutions to problems but also of taking care of other tasks occurring when it is used in practice. The main phases of the case-based reasoning activities [1] are described in the *CBR-cycle* (Figure 1).

In the retrieve phase the most similar case (or k most similar cases), to the problem case, is retrieved, while in the reuse phase some modifications to the

retrieved case is done in order to provide better solution to the problem (case adaptation). As the case-based reasoning only suggests solutions, there may be a need for a correctness proof or an external validation, so that system will stay consistent in regard to environment. That is the task of the phase revise. In the retain phase the knowledge, learned from this problem, is integrated in the system by modifying some knowledge containers.

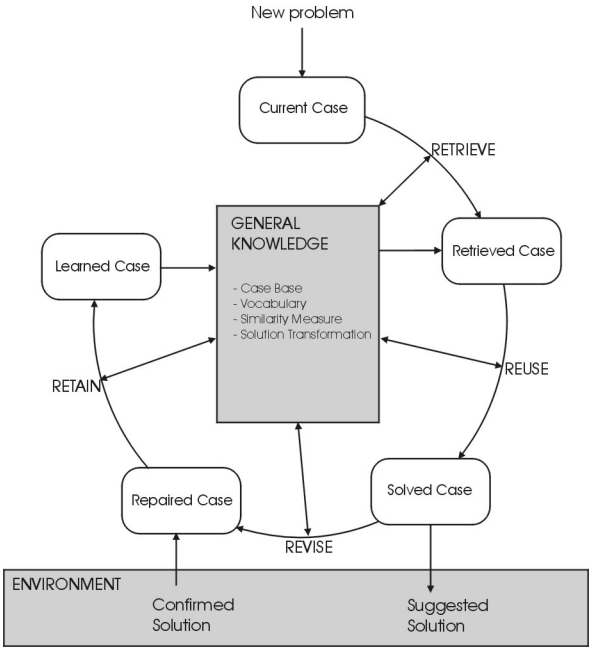

Figure 1. The CBR-Cycle after Aamodt and Plaza (1994)

The main advantage of this technology is that it can be applied to almost any domain. CBR system does not try to find rules between parameters of the problem; it just tries to find similar problems (from the past) and to use solutions of the similar problems as a solution of an actual problem. So, this approach is extremely suitable for less examined domains - for domains where rules and connections between parameters are not known. Furthermore, in the more examined domains integration of CBR in classical rule-based reasoning systems brings some efficiency. The second very important advantage is that CBR approach to learning and problem solving is very similar to human cognitive processes - people take into account and use past experience to make future decisions.

The rest of the paper is organized as follows. The following section elaborates necessary concepts for CBG implementation [8], [9], [10]. The system "CBG" is described in the third section, while the fourth section describes the application

of the system in two "test" domains. Fifth section presents the related work and concludes the paper.

2. Foundations of CBR

The case-based reasoning was developed in the context and in the neighborhood of problem solving methods, learning methods (Machine Learning, Statistics, Neural Networks) and retrieval methods (Data Bases, Information Retrieval) [9]. It has inherited the concepts of "problem" and "solution" and a notion of "similarity" based on the distance. However, very often the more general term acceptance instead of similarity is used. Acceptance includes similarity, but also other concepts like "expected usefulness", "reminds on" etc.

2.1 Information entities

Information entities are the atomic constituents of cases and queries. We consider a case as the result of the case completion process. Each step of that process adds some information entities. The current situation during the elaboration of a task is described by the information entities known at the time point. The final case, as it later may appear in the case memory, is a completed set of information entities.

The collected information entities result from the real world (an outcome of the test, a decision in an intermediate design step, etc). They are not a direct result of the case-based reasoning process - case-based reasoning is used to propose the next step (some test, the next design decision etc).

The number of information entities in a case may be variable. It is up to a human decision at which time point the task is finished.

The information entities, which are later used for retrieval, (which appear in the case memory) may be only a subset of the information entities collected during the case completion. These information entities (in the case memory) serve as the indexes for retrieval. The case memory consists of cases, which are sets of such information entities. These cases may then point to related complete descriptions in a collection of "full cases".

The information entity is an atomic part of a case or query. E denotes the set of all information entities in a given domain.

- A case is a set of information entities: $c \subseteq E$.
- The set of cases (in the case memory) is denoted by C, $C \subseteq P(E)$.
- A query is a set of information entities: $q \subseteq E$.

In many applications, the information entities are simply attribute-value pairs. Some examples of information entities are: (price, 1000), (price, 324), (color, blue), (mass, 54 kg). We say that the first two information entities are comparable (because they have the same attribute) while the other information entities are not comparable. This causes a structuring of the set E into disjoint sets E_A, where E_A contains all attribute-value pairs from E for a certain attribute A.

2.2 Acceptance

We want to use the association of information entities for reminding cases with the expectation that these cases are useful for a given query. Usefulness of a case in the case completion process depends on real world circumstances that are not completely known at the retrieval time. This means that usefulness is only an aposterior criterion. The retrieval from the case memory will be based on matching of certain information entities. Usefulness of former cases is not restricted to those cases that are similar to a given query for all information entities. Cases may contain information entities that have no counterpart in the query. It is also possible that some information entities of the query are not present in the useful case.

Some special desirable properties of acceptance are following:

P1: A case might be acceptable for a query even if there exist some information entities that are not comparable.

P2: A case might be unacceptable for a query if there exist an unacceptable information entity (a fix budget may forbid expensive offers).

P3: The same information entity may have different importance for different cases (information entity (sex, male) has different importance in pregnancy testing and in testing for influenza).

P4: The same information entity may have different importance for different queries according to the user's intentions (material have different priorities in design queries).

P5: Information entities may not be independent of each other.

2.3 Acceptance Functions

In this section, the proposal of the acceptance function which satisfies properties P1-P5 will be given. Queries have been defined as sets of information entities. A weighted query is the generalization of this concept.

Weighted query. The weighted query assigns an importance value to each information entity by a function:

$$\alpha_q : E \to R,$$

where $\alpha_q(e)$ denotes the importance of the information entity e for the query q.

High values indicate a high importance; negative values indicate the rejection of related cases. The value 0 is used as a neutral element ($\alpha_q(e) = 0$, means that information entity e is unimportant to the query q). Of course, values for $\alpha_q(e)$ can be taken from the set $\{0, 1\}$, meaning that "e is (un)important for the q".

Local Acceptance Functions for Attributes. A local acceptance function σ for the attribute A is defined over the domain $dom(A)$:

$$\sigma : dom(A) \times dom(A) \to R,$$

such that higher value $\sigma(e, e')$ denotes a higher acceptance of the value e (of a case c) for the value e' (of a query q).

By using we can compute the acceptance of the information entity e' from the case for a single information entity e of a query. However, a query may contain several information entities e such that $\sigma(e, e')$ is defined for the single information entity e'. The question is: how these values can be combined to a single value for e' which express the resulting acceptance value of e' for that query.

Local Accumulation Function. Let $E_e = \{e_1, \cdots, e_n\}$ denote the set of all information entities to which the information entity e is comparable concerning acceptance ($E_e = \{e' | \sigma(e', e)\ is\ defined\}$). The local accumulation function π_e for e is a function:

$$\pi_e : \underbrace{R \times \cdots \times R}_{n} \rightarrow R,$$

such that $\pi_e(a_1, \cdots, a_n)$ denotes the accumulated acceptance in e. The values a_i denote the contributions of the information entities $e_i \in E_e$ according to their occurrence in the query q and their local acceptance computed by $\sigma(e_i, e)$. The contributions are computed by a function:

$$f : R \times R \rightarrow R,$$

such that $a_i = f(\alpha_q(e_i), \sigma(e_i, e))$.

The retrieval of the cases can be considered as a process of reminding. Reminding may be of different strength; cases are in competition for retrieval according to the query. The cases receiving more reminders of more strength are the winners. The strength (importance, relevance) of reminding for an information entity $e \in c$ is given by a relevance function.

Relevance Function. The relevance between information entities and cases is described by a relevance function:

$$\rho : E \times C \rightarrow R.$$

The relevance $\rho(e, c)$ is considered as a measure for the relevance of information entity e for the retrieval of a case c. $\rho(e, c)$ is defined if and only if $e \in c$.

The acceptance of a case c for the query q is accumulated from the contributions of the information entities $e \in c$ according to their relevancies $\rho(e, c)$. The contributions of the information entities are computed by their local accumulation functions π_e. The accumulation in the cases is evaluated by *Global accumulation function.*

Global Accumulation Function. The global accumulation function π_c has the form:

$$\pi_c : \underbrace{R \times \cdots \times R}_{k} \rightarrow R,$$

for $c = \{e_1, \cdots, e_k\}$. The accumulated acceptance of the case c regarding its constituting information entities is then computed by $\pi_c(p_1, \cdots, p_k)$, where p_i is the contribution of the information entity $e_i \in c$. This contribution p_i depends on $\rho(e_i, c)$ and another real value x_i assigned to e_i (x_i is the accumulated

local acceptance value computed by $\pi_{e_i}(a_1, \cdots, a_n)$). The contributions p_i are computed by a function:

$$g : R \times R \to R,$$

such that $p_i = g(x_i, \rho(e_i, c))$.

Global Acceptance function. Acceptance between weighted queries and cases is expressed by an Global acceptance function:

$$acc : R^E \times P(E) \to R.$$

The acceptance $acc(\alpha_q, c)$ of a case c for a weighted query q can now be accumulated by using the introduced functions:

$$acc(\alpha_q, c)) = \pi_c(p_1, \cdots, p_k),$$

where $c = \{e_1, \cdots, e_k\}$ and the values p_i are contributions of the information entities e_i from the case c.

If, for example, f and g are considered as products and π_c and π_e as sums then Global acceptance function looks like this:

$$acc(\alpha_q, c) = \sum_{e' \in c} \rho(e', c) \sum_{e \in E_{e'}} \sigma(e, e') \cdot \alpha_q(e)$$

Here, the properties P1,...,P4 are satisfied, but for satisfaction of the property P5, the appropriate selection of the functions f,g,π_c and$\pi_e$is needed.

2.4 A possible Implementation - Case Retrieval Net

Case Retrieval Net (CRN) is a special memory structure that has been developed especially for being employed in large case bases. CRNs are able to deal with vague and ambiguous terms, they support the concept of information completion and can handle case bases of reasonable size efficiently.

CRN is a net structure with information entity node for each information entity and case node for each case. An example of CRN is shown on Figure 2. There exists an "acceptance" arc from the information entity nodes e to the information entity node e' if $\sigma(e, e')$ is defined, and it exists an "relevance" arc from the information entity node e to the case node c if $\rho(e, c)$ is defined. The

arcs in the net are weighted by the values $\sigma(e, e')$ and $\rho(e, c)$ respectively.

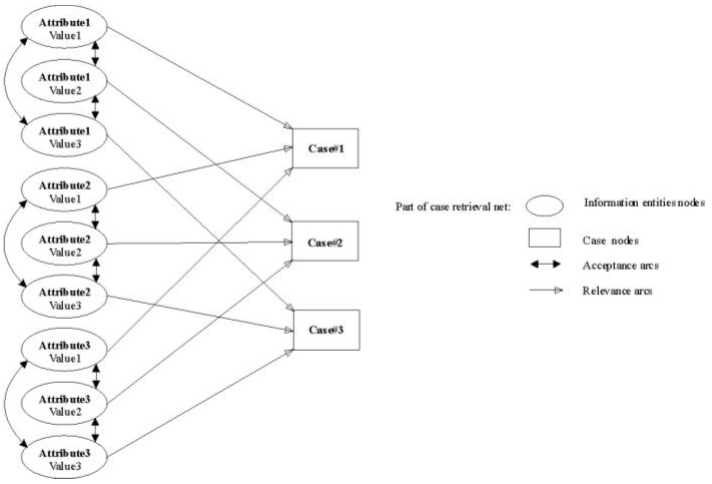

Figure 2. Part of Case Retrieval Net

In practice, it may be impossible to include all information entities. Usually, it is sufficient to build a net from the information entities which occur in the cases from the case base only.

Acceptance values are computed by a spreading activation process in the net as follows: Information entity nodes are initially activated by $\alpha_q(e)$. The computation is performed by propagating along the acceptance arcs to further information entity nodes, and from these nodes over relevance arcs to case nodes. The functions $f/\pi_e$and $g/\pi_c$are responsible for the accumulation of activities in the information entity nodes and in the case nodes respectively. The final activation at the case nodes denote the acceptance value of the case for the weighted query.

3. "CBG" - Implementation Issues

Since summer, 2000 a small group at the Department of Mathematics and Informatics in Novi Sad is doing a research in the area of CBR. During that research one natural demand came out: it would be very useful to have some basic, core system (framework) that can produce decision support systems in different domains. "CBG" (Casebase Generator) was the direct result of these intentions.

"CBG" system was completely implemented in Java - JDK 1.3. Java programming language was chosen mainly because it support all concepts of object-oriented technology, but also because it's main characteristic - platform independence. The system was realized as an application, but the small modifications

are necessary in order to make an applet or servlet. Of course, *javax.swing* components are used for creating a graphical user interface (GUI).

The main advantage of this system is that it is domain independent. The input for the system is the description of the database and the database from any domain. On the basis of those data, system creates Case Retrieval Net (CRN) and it is capable to solve new problems (as a matter of fact to propose solutions) from a domain of the input database.

As already mentioned the system read the data from two input files. In the first input file (which is called the "Case Pattern File") the description of the database or better to say the description of the case is stored. Case pattern file contains the information about the structure of every case. There, the list of the attributes, containing the name and the type of the attribute is listed. The type of the attribute can be: *int, float* or *string*. Boolean type can be simulated, for example, with the string type where only two values ("Yes"/"No") are allowed. The number of the attributes is arbitrary, but all the attributes in the correct order must be listed.

The second file (which is called the "Case Base File") contains the list of all already solved cases. Every case is described with the values of its attributes and with the final solution of that case. The values of the attributes must be sorted in the correct way according to the order of the attributes listed in the case pattern file. The final solution of the case is always listed at the end of the case (after values of all attributes). Type of the solution can also be : *int, float* or *string* and it is determined dynamically, when all cases are parsed. At this moment it is assumed that the case base file is textual file where every case is listed in one line, and the values of the attributes and solution are separated with commas.

Together with the reading of the case base file, system creates case retrieval net. The basic structure of CRN is given in the figure 3. Two main parts of the CRN are array of attributes and list of solutions. The array of attributes is created using case pattern file, while the list of the solutions is created from case base file - solution is the last value from every case. Every attribute consists of its name and the list of values. Every value for every attribute represents one information entity because the information entity is an ordered pair (*attribute, value*). Every value (or information entity) contains the list of arcs to the solution nodes. The arc is given with its weight and a pointer to the solution node; just the arcs whose weights are different from zero are saved. Weights of the arcs between the the information entity node e to the solution node c represents the value of the function $\rho(e, c)$. These weights are calculated as a number of cases (from the case base file) that contain the information entity e

and whose solution is c.

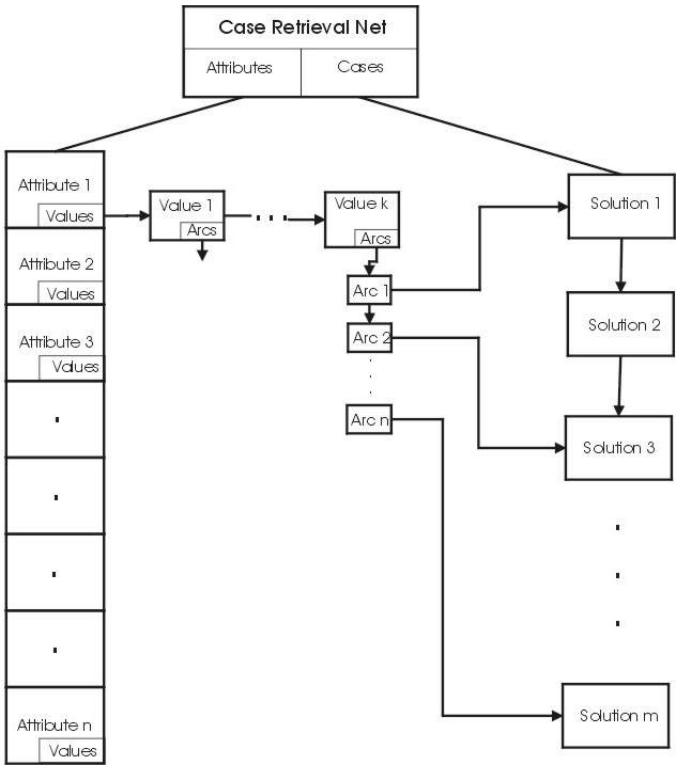

Figure 3. Structure of CRN

After creating the CRN the system expects from the user to enter the current problem (query). Since the query and the case have the same structure (a set of the information entities) the user has to enter the values of some (or all) attributes in a form. In order to better describe the problem, the user should enter all known values of the attributes although it is not necessary. The form contains one more field for every attribute - importance. The importance is the value from the interval (0,1), and describes how much is the user sure in the validity of the value of the attribute he is entering. Value 1 means that he is 100sure that the data are valid, while the value 0 means that he doesn't know the value of that attribute at all. The value of importance corresponds to the previously described value of the weighted query.

After entering the query, the system searches for the possible solution in the following way: *The information entities (attribute, value) that occur in the query are initially activated with the value of importance (weighted query). The activation is propagating trough the arcs to the solution nodes, by multiplying the value of the activation of the information entity node and the weight of*

the corresponding arc. Final activation of the solution nodes is calculated by summing all gained activations.

4. Application of the System

The application of the system will be shown on two simple test domains: a domain for determination species of the animals when some features of the animals are given, and a domain for diagnosis of breast cancer. Databases for these domains and also some others are freely available at [3].

4.1 Classification of the Animals

The structure of a case is described in case pattern file "default.key", and it is given in the Listing 1. All attributes except `NumberofLegs` are of string type, but all of these attributes except `AnimalName` are in fact of Boolean type which is simulated with string type with just two values: "Yes" or "No".

```
AnimalName,string
HasHair,string
HasFeathers,string
LaysEggs,string
GivesMilk,string
CanFly,string
LivesinWater,string
Predator,string
HasTeeth,string
HasABackbone,string
BreathesWithLungs,string
IsVenomous,string
HasFins,string
NumberofLegs,int
HasTail,string
IsDomestic,string
IsCatSized,string
```

Listing 1. Structure of a case

The cases are given in the case base file "default.cbr". Part of this file is given in Listing 2. Every case is given in new line and a case base consists of 102 cases. The values of the attributes must be given in the correct order according to the case pattern file. Last value in the description of the case is always the solution of the case. These values are determined dynamically and in this domain can be: *Amphibian, Anthropoid, Bird, Fish, Insect, Mammal* or *Snake*.

```
.....
hamster,Yes,No,No,Yes,No,No,No,Yes,Yes,Yes,No,No,4,Yes,Yes,No,Mammal
hawk,No,Yes,Yes,No,Yes,No,Yes,No,Yes,Yes,No,No,2,Yes,No,No,Bird
herring,No,No,Yes,No,No,Yes,Yes,Yes,Yes,No,No,Yes,0,Yes,No,No,Fish
```

```
honeybee,Yes,No,Yes,No,Yes,No,No,No,No,Yes,Yes,No,6,No,Yes,No,Insect
housefly,Yes,No,Yes,No,Yes,No,No,No,No,Yes,No,No,6,No,No,No,Insect
kiwi,No,Yes,Yes,No,No,No,Yes,No,Yes,Yes,No,No,2,Yes,No,No,Bird
lark,No,Yes,Yes,No,Yes,No,No,No,Yes,Yes,No,No,2,Yes,No,No,Bird
leopard,Yes,No,No,Yes,No,No,Yes,Yes,Yes,Yes,No,No,4,Yes,No,Yes,Mammal
lion,Yes,No,No,Yes,No,No,Yes,Yes,Yes,Yes,No,No,4,Yes,No,Yes,Mammal
lobster,No,No,Yes,No,No,Yes,Yes,No,No,No,No,No,6,No,No,No,Anthropod
mongoose,Yes,No,No,Yes,No,No,Yes,Yes,Yes,Yes,No,No,4,Yes,No,Yes,Mammal
moth,Yes,No,Yes,No,Yes,No,No,No,No,Yes,No,No,6,No,No,No,Insect
newt,No,No,Yes,No,No,Yes,Yes,Yes,Yes,Yes,No,No,4,Yes,No,No,Amphibian
octopus,No,No,Yes,No,No,Yes,Yes,No,No,No,No,No,8,No,No,Yes,Anthropod
pitviper,No,No,Yes,No,No,No,Yes,Yes,Yes,Yes,Yes,No,0,Yes,No,No,Snake
.....
```

Listing 2. A part of a case base file.

In the Figure 4, the main window of the system is shown. The system expects that user enters the paths of the case pattern and case base file.

Figure 4. The main window of the "CBG"

If the paths are good, after clicking on the "Load Files" button, these two files are loaded and the case retrieval net is created. Also, the dynamically created form will appear in the middle of the window. This situation is shown

in the Figure 5.

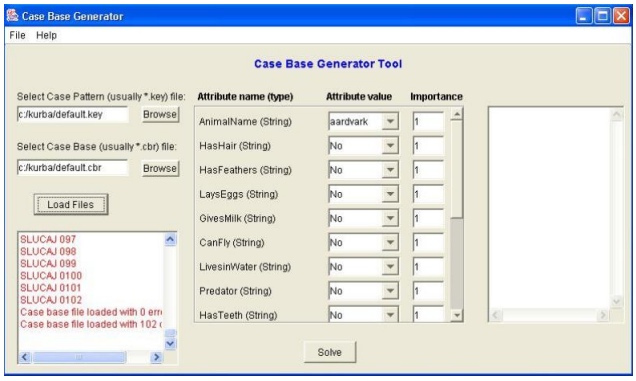

Figure 5. "CBG" after loading files

After loading files and creating CRN, the system expects from the user to enter current problem - to enter the known values of the attributes. If some values are not known then the user should enter the zero value in the corresponding importance field. When the entering the data into the form is finished, the process of the spreading activation is started by clicking on the "Solve" button. In the right part of the window the solution is shown after finishing the process. There all possible solutions and their activations are shown as in Figure 6. The solution with the highest number is the "suggested solution". For this example it is most possible that the "problem animal" is Mammal, since it has the highest activation.

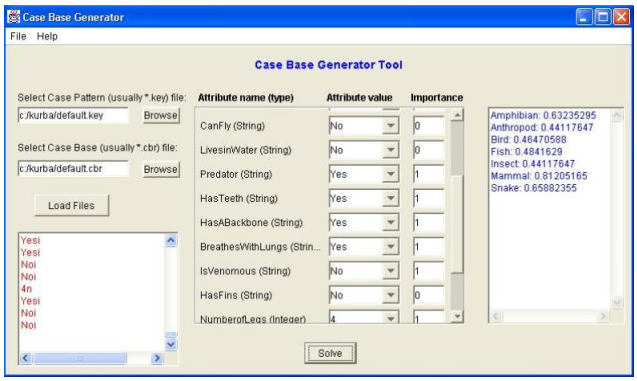

Figure 6. "CBG" after solving the problem.

Of course, some new "problem animal" can be entered in the form. The system always suggests some solution, but the quality of the solution depends on the quality of the input data.

4.2 Diagnosis of Breast Cancer

The structure of a case is described in case pattern file "cancer.key", and it is given in the Listing 3. All attributes are of integer type and they represent some medical checkups.

```
IDNumber, int
ClumpThickness,int
UniformityofCellSize,int
UniformityofCellShape, int
MarginalAdhesion,int
SingleEpithelialCellSize,int
BareNucleoli,int
BlandChromatin,int
NormalNucleoli,int
Mitoses,int
```

Listing 3. Structure of a case

The cases are given in the case base file "cancer.cbr". Part of this file is given in Listing 4. Every case is given in new line and a case base consists of 700 cases. As already mentioned, the values of the attributes must be given in the correct order according to the case pattern file. Last value in the description of the case represents the solution of the case. These values are determined dynamically and in this domain can be: 2 or 4, with the meaning 2-for benign cancer and 4-for malignant cancer.

```
1160476,2,1,1,1,2,1,3,1,1,2
1164066,1,1,1,1,2,1,3,1,1,2
1165297,2,1,1,2,2,1,1,1,1,2
1165790,5,1,1,1,2,1,3,1,1,2
1165926,9,6,9,2,10,6,2,9,10,4
1166630,7,5,6,10,5,10,7,9,4,4
1166654,10,3,5,1,10,5,3,10,2,4
1167439,2,3,4,4,2,5,2,5,1,4
1167471,4,1,2,1,2,1,3,1,1,2
1168359,8,2,3,1,6,3,7,1,1,4
1168736,10,10,10,10,10,1,8,8,8,4
1169049,7,3,4,4,3,3,3,2,7,4
1170419,10,10,10,8,2,10,4,1,1,4
```

Listing 4. A part of a case base file.

The final solution (after loading the files, creating the CRN and setting up the query) is shown on the figure 7. In this example the solution 2 gained more

activation so the suggested solution is benign cancer.

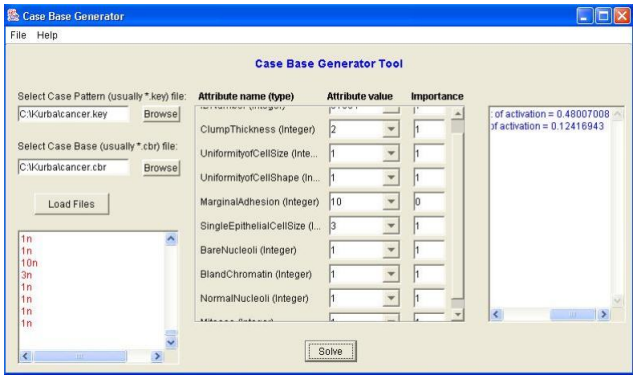

Figure 7. "CBG" after solving the problem.

5. Conclusions and Related Work

Case-based reasoning is a reasoning method that facilitates knowledge management in which knowledge is a case base acquired by a learning process. Case-based reasoning can be used for solving problems in many practical domains such as: mechanical engineering, medicine, business administration etc. Furthermore, for each domain, various task types can be implemented. Some of them are: classification, diagnosis, configuration, planning, decision support etc.

Also, for every task type all the domains are possible. That is the exactly the case with "CBG". "CBG" is the framework suitable for building different decision support systems from any domain. Currently we are trying to define concrete application of CBG together with the *Institute of Neurology*, Novi Sad. The final goal of this cooperation will be to build the decision support system that will help medical experts in determination of the Multiple sclerosis disease [5]. The quality of the system will depend on the quality of the case base that will provide colleagues from the Institute of neurology from their patient files of this disease during last 10 years.

References

[1] Aamodt, A., Plaza, E., Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. AI Commutations (1994), 39-58.

[2] Budimac Z., Kurbalija V.. Case-Based Reasoning - A Short Overview, Proceedings of Second International conference on Informatics and Information Technology "Molika 2001", Molika, Macedonia, December 20-23, 2001, 222-234.

[3] Case-Based Reasoning at AIAI, `http://www.aiai.ed.ac.uk/project/cbr/`

[4] Iglezakis, I., The Conflict Graph for Maintaining Case-Based Reasoning Systems. 4th International Conference on Case-Based Reasoning (ICCBR 2001), Vancouver, Canada, July/August 2001, 263-276.

[5] Ivanović M., Kurbalija V., Budimac Z., Semnic M., Role of Case-Based Reasoning in Neurology Decision Support, Proceedings of the Fifth Joint Conference on Knowledge-Based Software Engineering "JCKBSE 2002", Maribor, Slovenia, September 11-13, 2002, 255-264.

[6] Kurbalija, V., On Similarity of Curves - project report. Humboldt University, AI Lab, Berlin, 2003.

[7] Kurbalija V., Ivanović M., Case-Based Reasoning - A Powerfull Artificial Intelligence Approach. Journal of Electrical Engineering 12/s, Slovak Centre of IEE, Vol. 53 (2002) 20-24. (Proceedings of Conference of Applied Mathematics for undergraduate and graduate students "SCAM 2002", Bratislava, Slovak Republic, April 19-20, 2002).

[8] Lenz M., Case Retrieval Nets Applied to Large Case Bases. Draft, Lenz's homepage.

[9] Lenz M., Bartsh-Sporl B., Burkhard HD., Wess S., Case-Based Reasoning Technology: From Foundations to Aplications. Springer Verlag, October 1998.

[10] Lenz M., Burkhard H.D., Bruckner S., Applying Case Retrieval Nets to Diagnostic Tasks in Technical Domains. Lenz's home-page.

[11] Reinartz, T., Iglezakis, I., Roth-Bergofer, T., On Quality Measures for Case Base Maintenance. 5th European Workshop (EWCBR 2000), Trento, Italy, September 2000, 247-260,

Received by the editors December 29, 2003