CBG – A FRAMEWORK FOR INTELLIGENT DECISION SUPPORT SYSTEMS

V. Kurbalija, M. Ivanović

Department of Mathematics and Informatics, Faculty of Science,

Trg D. Obradovića 4, 21 000 Novi Sad, Serbia and Montenegro

{kurba, mira}@im.ns.ac.yu

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.

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

Keywords: Intelligent System, Database and Information Retrieval, Case-Based Reasoning

1 Introduction

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 (Aamodt 1994), (Budimac 2001), (Kurbalija 2002), (Lenz 1998). The main advantage of this technology is that it can be applied to almost any domain. This approach is extremely suitable for less examined domains – for domains where rules and connections between parameters are not known. Generally speaking, case-based reasoning is applied for solving new problems by adapting solutions that worked for similar problems in the past. 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 of the actual problem.*

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:

- the vocabulary used;
- the similarity measure;
- the case base;
- the solution transformation.

In principle, each container can carry almost all knowledge available. Manipulations on one container have little consequences on the others. As a consequence, maintenance operations (Iglezakis 2001), (Reinartz 2000) are easier to be performed then on classical knowledge based systems. 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.

During the last two years in the area of CBR at the Department of Mathematics and Informatics, several systems have been developed (Ivanović 2002), (Kurbalija 2003). Current research is connected to realization of "Casebase Generator" – a decision support system based on CBR. The main attention was to create the core system which could be used in several more specific intelligent systems. "Casebase Generator" is first step in realization of different decision support systems based on CBR engine.

The main phases of the case-based reasoning activities (Aamodt 1994) 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.

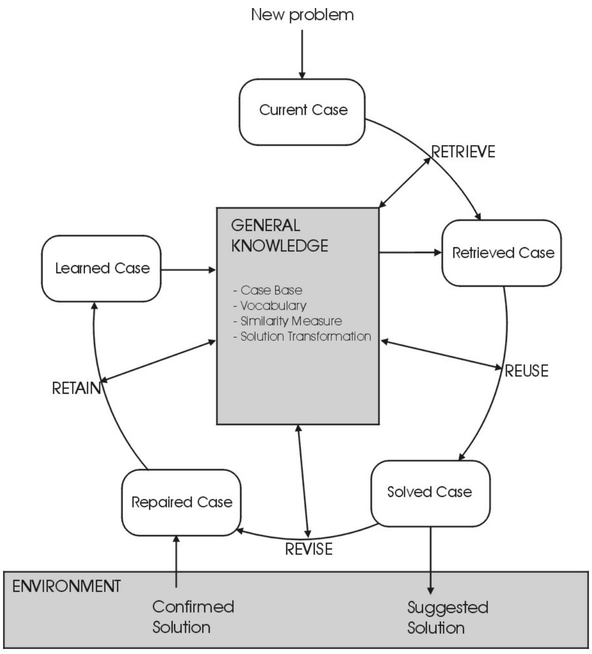


Fig. 1: The *CBR-Cycle* after Aamodt and Plaza (1994)

The rest of the paper is organized as follows. The following section elaborates necessary concepts for CBG implementation (Lenz's home-page), (Lenz 1998), (Lenz's home page). The system "CBG" is described in the third section, while the fourth section describes the application of the system in one "test" domain. Fifth section concludes the paper.

2 Foundations of CBR

In case-based reasoning very often the term *acceptance* is used instead of similarity. Acceptance includes similarity, but also other concepts like "expected usefulness", "reminds on" etc.

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. Set E divided into disjoint sets E_A, where E_A contains all attribute-value pairs from E for a certain attribute A. If cases and queries are considered as attribute-value vectors over a finite set of attributes $A_1,\dots,A_n$, then each case or query may contain at most one information entity from each E_{Ai}.

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 "a posterior criterion". The retrieval from the case memory will be based on matching of certain 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. In order to build a good acceptance (similarity) function, some additional functions must be introduced.

The *weighted query* assigns an importance value to each information entity by a function α_q, where $\alpha_q(e)$ denotes the importance of the information entity e for the query q.

A *local acceptance function* σ for the attribute A is defined over the domain dom(A), 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 σ, the acceptance of the information entity *e'* from the case for a single information entity *e'* of a query can be computed. 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 expresses the resulting acceptance value of *e* for that query. The *local accumulation function* for single information entity e of the query is a function π_e, where $\pi_e(a_1, \dots, a_n)$ denotes the accumulated acceptance in e. The values a_i denote the contributions of the comparable information entities e_i 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*, such that $a_i = f(\alpha_q(e_i),\sigma(e_i,e))$.

The strength (importance, relevance) of reminding for an information entity $e \in c$ is given by a *relevance function* ρ. The value $\rho(e,c)$ 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 **global acceptance** of the case c regarding its constituting information entities is computed by a *global accumulation function* $\pi_c(p_1, \dots, p_k)$, 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 the accumulated local acceptance value computed by $\pi_{ei}(a_1,\dots,a_n)$. The contributions p_i are computed by a function g, such that $p_i = g(\pi_{ei}(a_1,\dots,a_n),\rho(e_i,c))$.

2.1 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.

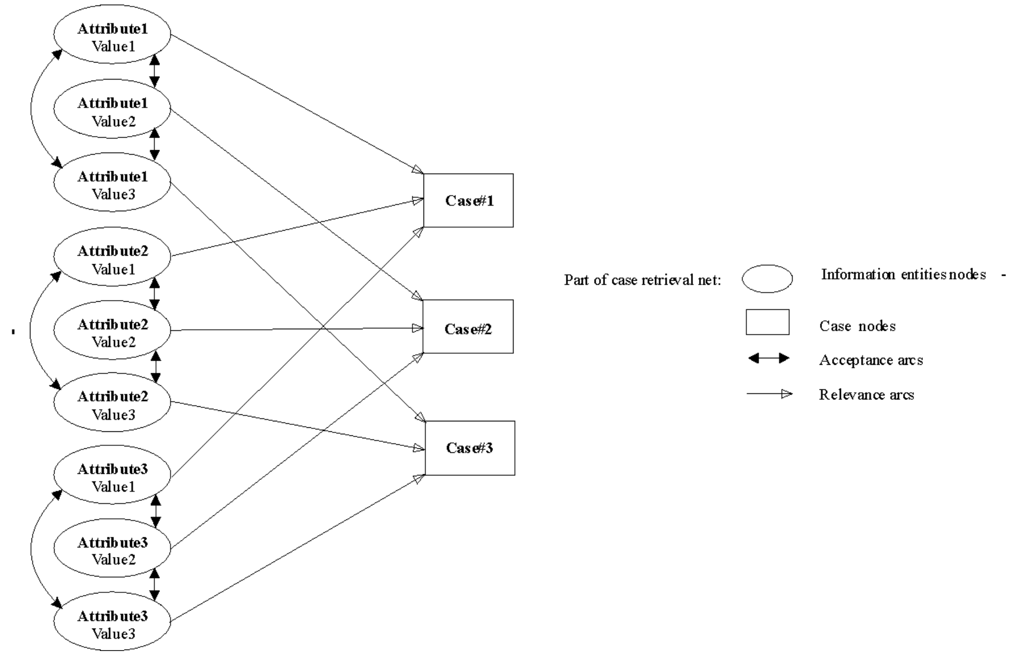


Fig. 2: Part of Case Retrieval Net

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 there 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.

In practice, it is impossible to include all information entities, since the real world domains are allmost allways infinite. 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/π_e and g/π_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

For application of CBR in a domain it would be very useful to have some basic, core framework that can be used to produce decision support systems. "CBG" (Casebase Generator) was the direct result of these intentions. It was completely implemented in Java mainly because it supports 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 two folded: the description of the case and the data for particular cases from particular chosen domain. On the basis of those data, system creates Case Retrieval Net (CRN) and it is capable to solve new problems from the domain.

In the first input file - "Case Pattern File", the description of case is stored. Case pattern file contains the list of the attributes, containing the name and the type of the attribute. 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 attributes must be listed in the correct order.

The second file - "Case Base File", contains the list of all already solved cases from the domain. Every case is described with the values of its attributes and with the final solution of that case. The final solution of the case is always listed at the end. 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. 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 information entity node e to the solution node c represents the value of the function $\rho(e,c)$. These weights are simply calculated as a number of cases (from the case base file) that contain the information entity e and whose solution is c.

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 the user has to enter the values of attributes in an appropriate 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 100% sure 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.

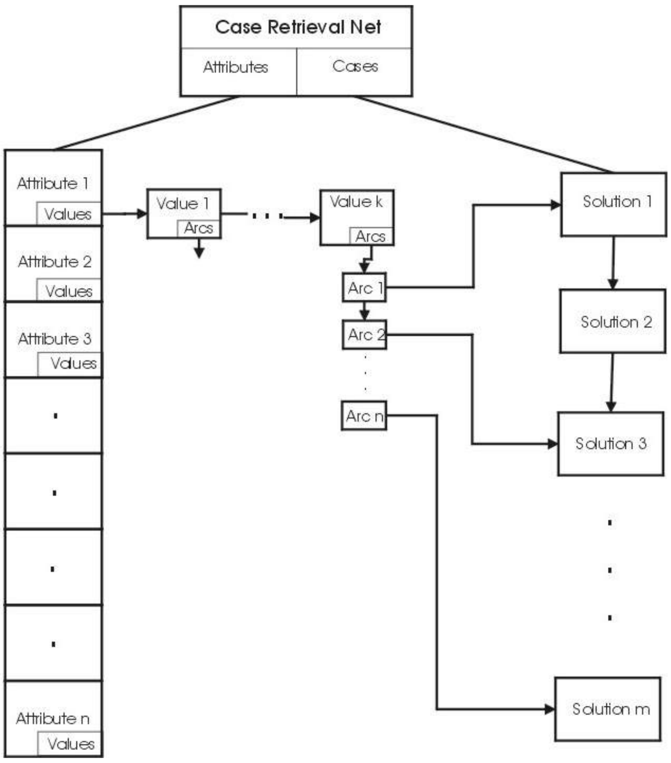


Fig. 3: Structure of CRN

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

Implemented system is tested on a one simple domain – determination species of the animals when some features of the animals are given. Casebase for this domain and also some others are available at (CBR at AIAI). The structure of a case is described in case pattern file "default.key", and it is given in the Table 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	HasABackbone	String
HasHair	String	BreathesWithLungs	String
HasFeathers	String	IsVenomous	String
LaysEggs	String	HasFins	String
GivesMilk	String	NumberofLegs	Int
CanFly	String	HasTail	String
LivesinWater	String	IsDomestic	String
Predator	String	IsCatSized	String
HasTeeth	String		

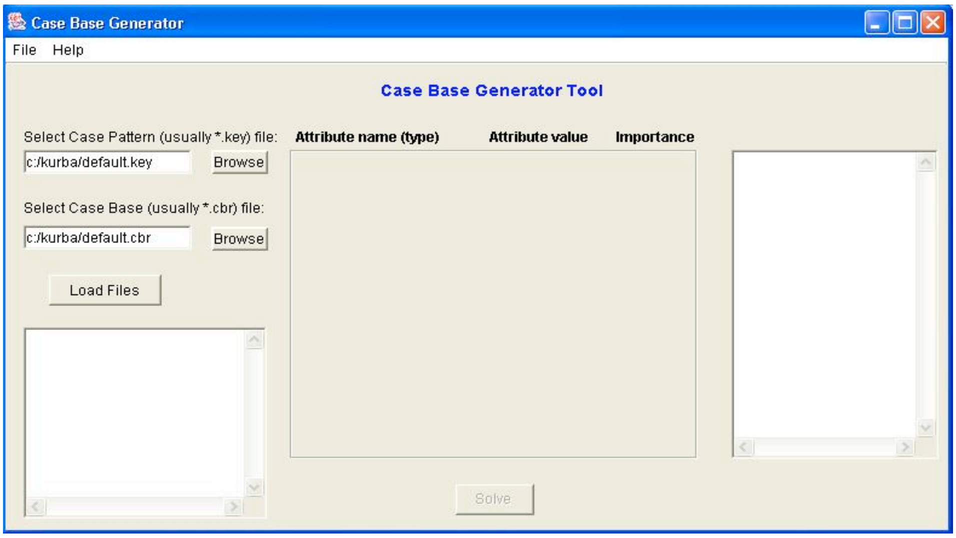
Table 1: Structure of a case

The (values of the) cases are given in the file "default.cbr". Part of this file is given in Listing 1. Every case is given in new line and a case base consists of 102 cases. 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
kiwi,No,Yes,Yes,No,No,No,Yes,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
lobster,No,No,Yes,No,No,Yes,Yes,No,No,No,No,No,6,No,No,No,Anthropod
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 1: 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.



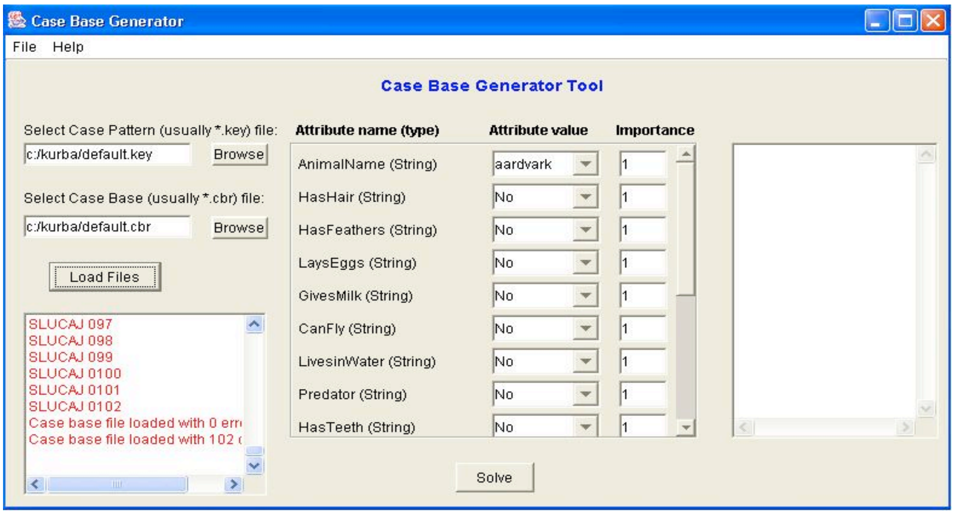
Select Case Pattern (usually *.key) file: c:/kurba/default.key

Select Case Base (usually *.cbr) file: c:/kurba/default.cbr

Attribute name (type)	Attribute value	Importance

Fig. 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.



Select Case Pattern (usually *.key) file: c:/kurba/default.key

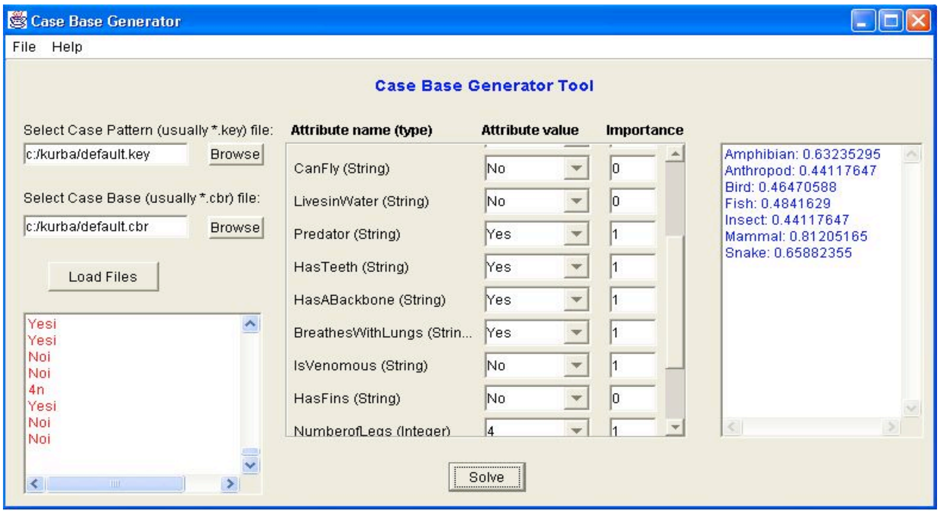
Select Case Base (usually *.cbr) file: c:/kurba/default.cbr

```
SLUCAJ 097
SLUCAJ 098
SLUCAJ 099
SLUCAJ 0100
SLUCAJ 0101
SLUCAJ 0102
Case base file loaded with 0 err
Case base file loaded with 102 
```

Attribute name (type)	Attribute value	Importance
AnimalName (String)	aardvark	1
HasHair (String)	No	1
HasFeathers (String)	No	1
LaysEggs (String)	No	1
GivesMilk (String)	No	1
CanFly (String)	No	1
LivesinWater (String)	No	1
Predator (String)	No	1
HasTeeth (String)	No	1

Fig. 5: "CBG" after loading files

After loading files and creating CRN, the system expects entering the known values of the attributes for current case. For unknown values, 0, has been entered in the corresponding importance field. When the data are entered, the process of the spreading activation is started by clicking on the "Solve" button. In the right part of the window the solution has been shown. Their all possible solutions and their activations are shown as in Figure 6.



Select Case Pattern (usually *.key) file: c:/kurba/default.key

Select Case Base (usually *.cbr) file: c:/kurba/default.cbr

```
Yesi
Yesi
Noi
Noi
4n
Yesi
Noi
Noi
```

Attribute name (type)	Attribute value	Importance
CanFly (String)	No	0
LivesinWater (String)	No	0
Predator (String)	Yes	1
HasTeeth (String)	Yes	1
HasABackbone (String)	Yes	1
BreathesWithLungs (Strin...	Yes	1
IsVenomous (String)	No	1
HasFins (String)	No	0
NumberofLegs (Integer)	4	1

```
Amphibian: 0.63235295
Anthropod: 0.44117647
Bird: 0.46470588
Fish: 0.4841629
Insect: 0.44117647
Mammal: 0.81205165
Snake: 0.65882355
```

Fig. 6: “CBG” after solving the problem

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. 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.

The CBG system focuses on the first phase of CBR cycle (Figure 1.) - retrieval of the most similar cases and proposition of the solution on the basis of retrieved cases. The second phase, revise or adaptation, is not supported by general system because the adaptation of the case strongly depends on domain of application. The phases revise and retain are not automated but can be easily accomplished by adding successfully solved cases in the Case Base File.

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 many 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 (Ivanović 2002). 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.

6 References

1. Aamodt, A., Plaza, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches, *AI Commutations*, pp. 39-58. 1994.
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, pp. 222-234.
3. Case-Based Reasoning at AIAI, http://www.aiai.ed.ac.uk/project/cbr/
4. Iglezakis, I. (2001), “The Conflict Graph for Maintaining Case-Based Reasoning Systems”, *4th International Conference on Case-Based Reasoning (ICCBR 2001),* pp. 263-276, Vancouver, Canada, July/August 2001.
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, pp. 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, pp.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., (2000), “On Quality Measures for Case Base Maintenance”, *5th European Workshop (EWCBR 2000),* pp. 247-260, Trento, Italy, September 2000.