Decision Making

APPLICATION OF TEMPORAL REASONING AND CASE-BASED REASONING IN INTELLIGENT DECISION SUPPORT SYSTEMS

Alexander Eremeev, Ivan Kurilenko, Pavel Varshavskiy

Abstract: In this paper the problem of the application of temporal reasoning and case-based reasoning (CBR) in intelligent decision support systems (IDSS) is consider. To reduce static algorithm complexity of temporal reasoning some methods are investigated. The dynamic algorithm for qualitative temporal reasoning on the base of point algebra is presented. CBR method for a solution of problems of real-time diagnostics and forecasting in IDSS is described. This paper demonstrates how the temporal reasoning system and CBR system can be used in IDSS of the car access control. This work was supported by RFBR and grant of President of Russia (MK-6009.2008.9).

Keywords: Intelligent decision support systems, temporal reasoning, analogous and case-based reasoning.

ACM Classification Keywords: H.4.2 [Information systems applications]: Types of systems – Decision support; I.2.4 [Artificial intelligence]: Knowledge Representation Formalisms and Methods – Temporal logic; I.2.5 [AI]: Programming Languages and Software – Expert system tools and techniques; I.2.6 [AI]: Learning – Analogies.

Conference: The paper is selected from XVth International Conference "Knowledge-Dialogue-Solution" KDS 2009, Varna, Bulgaria, June-July 2009

Introduction

Temporal reasoning and “common sense” reasoning, in particular, CBR can be used in various applications of artificial intelligence (AI) and for solving various problems [1], e.g., for diagnostics and forecasting or for machine learning. The problem of presentation of time and temporal interconnections in AI systems (AIS) and especially in IDSS is very actual nowadays. A lot of basic notions, such as “alteration”, “cause”, “consequence/effect” and relations among them can be described by time notions [2]. However, the problem of creation of formal systems of presentation and operating by temporal information became really actual after the appearance and development of AIS, oriented towards open and dynamic problem domains. Typical representatives of these systems are real time IDSS (RT IDSS), designed for monitoring and control of complex objects and processes in rather strict time conditions and for different types of uncertainties of obtained information [1]. AI experts model CBR and temporal reasoning by computers in order to develop more flexible models of search for solutions and learning. The generalized structure of RT IDSS is given in Fig. 1.

Formally, a RT IDSS can be defined by the tuple $SS = <M, R(M), F(M), F(SS)>$, where

- $M = \{M_1, \ldots, M_n\}$ is the set of formal or logic-linguistic models, implementing defined intelligent functions;
- $R(M)$ is the function for selection of the necessary model in a current situation;
- $F(M) = \{F(M_1),\ldots, F(M_n)\}$ is the set of modification functions of models $M_1,\ldots, M_n$;
- $F(SS)$ is the function for modification of SS system, i.e. its base components $M, R(M), F(M)$.

The main problems, solved by RT IDSS, are: *diagnostics and monitoring* – revealing of problem situations; *decision searching* – searching an optimal or admissible sequence of actions allowing to achieve the desired goal or to solve the problem situations; *forecasting* – assessing the recommended actions related to the goal achievement and the sanction to solve the problem situation.

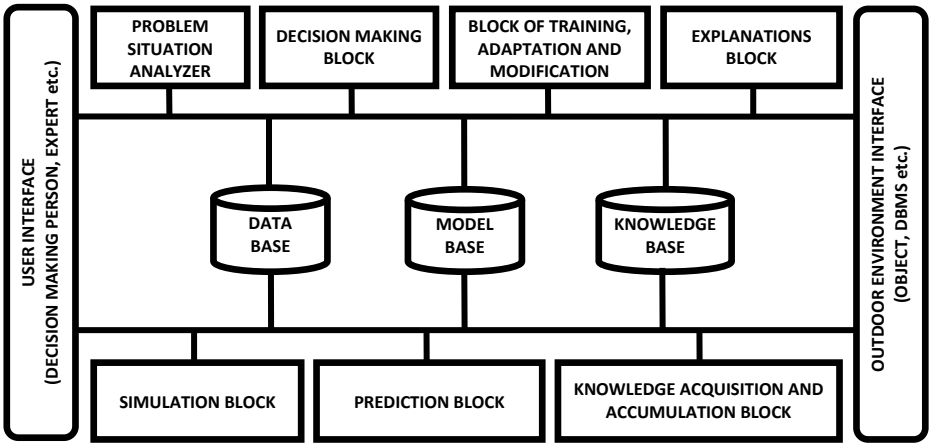

Fig. 1 Base RT IDSS structure

The methods of temporal reasoning and CBR may be applied in different blocks of RT IDSS. The necessity to present data and knowledge, changing in the course of time (indications of sensors, values of control parameters, information from decision making person (DMP), etc.) appears when solving many problems. RT IDSS must solve diagnostics, monitoring, decision searching and forecasting problems uninterruptedly in real time scale in order to help DMP to find efficient control effects in different operation modes of controlled objects, especially in abnormal modes. Using information about time while solving these problems permits to decrease search parameters greatly, which naturally positively effects reactivity of the whole system [3]. Thus, the use of the respective methods in RT IDSS broadens the possibilities of RT IDSS and increases the efficiency of making decisions in various problem (abnormal) situations.

Temporal reasoning

To implement the mechanism of temporal reasoning (MTR), it is necessary to formalize the notion of time and to provide the possibility to present and to discuss temporary aspects of knowledge. Modern approaches to presentation of time and temporal dependences in software systems can be divided into two large classes – based on *modeling of time changes* and on *explicit time modeling*. In the approaches which use changes modeling, the basic features are entities (actions), transforming one state of the system to the other. These states are regarded as momentary pictures of the world, which don't have any time duration. Time itself is regarded implicitly, via modeling of the system changes within time.

Approaches, based on changes modeling, have constraints when presenting difficult time dependences (events, which have duration, continuance of processes, competing or time overlapping events, causal relations etc). In literature different ways of constraints elimination can be found, however in the most cases they are reduced to introduction of an explicit time model. Explicit time modeling provides the possibility to make "flexible" formalized languages, which help to do reasoning on the basis of expressions, truth values of which are timed to the definite moment or time interval, and they can change in the course of time. Time is presented explicitly, taking into consideration its properties. Different *temporal logics* and *models* are included to this class. Time can be presented both syntactically (via explicit temporary structures) and semantically (modal logics are typical representatives of this approach). In the case of explicit time modeling the following specific *tasks of temporary reasoning* are raised: The support of information coordination about time – check of coordination of knowledge base when adding the new information. In the case of inconsistency of temporal constraints it is necessary to

localize plenty of expressions, responsible for this inconsistency; Answers to queries, dealing with time aspects of knowledge. These queries can be divided into finding a simple fact, true in the definite of moment and definition when a set of expressions is true at the same moment of time.

In this paper we consider the models, based on the presentation of information about time as constraints (dependences) between time primitives [4]. In temporal logics using the concept of constraint satisfaction, information about time is presented as dependences between temporal primitives (moments, intervals or their combinations). Dependences between primitives are interpreted as constraints to real time of their appearance. Main aim of the MTR is a generations of conclusions on the basis of sets of temporary constraints, i.e. new constraints for consistent input sets. Usually sets of temporal primitives and relations among them are presented as the Temporal Constraint Satisfaction Problem (TCSP), which is detailing of a more general Constraint Satisfaction Problem (CSP), what permits to use CSP methods to solve the TCSP.

The TCSP is specified by the following way $Z = (V,D,BTR,C)$ [5]:

1) $V = \{V_1,V_2,\ldots,V_m\}$ – a finite set of temporal variables;
2) D – a value domain of temporal variables;
3) $BTC = \{R_1, R_2,..R_n\}$ – a finite set of binary basic temporary constraints, and constraints entering there are mutually exclusive, but their total join is the universal constraint U;
4) $C = \{ C_{ij} \mid C_{ij} =\{r_1,\ldots,r_k\}, k>0, r_1,\ldots,r_k \in BTR, i<m, j<m \}$, a finite set of temporary constraints (STC), where C_{ij} –is the constraint for temporary variables V_i and V_j. Each constraint C_{ij} from set C is interpreted as $(V_i R_1 V_j) \vee .. \vee (V_i R_k V_j)$. In case C_{ij} consists only of one clause, it is called an *exact* restriction.

It is necessary to find such a STC $C^*=\{C_{ij}^* \mid C_{ij}^*=\{r_j\}, r_j \in C_{ij}\}$, so that exact constraints, entering it, do not conflict with each other.

Elements of V set can be interpreted as moments, time intervals or duration. The range of values of D variables, corresponding to moments of time and duration, represent a set of numbers, and for interval variables – a set of ordered value pairs.

In case in C set only exact constraints are entered, the TCSP is called *exact* TCSP, and the TCSP itself is regarded as a check of constraints consistency in C set [5].

The task of defining r constraint valid for variables V_i and V_j with the constraint $C_{ij} =\{r_1,\ldots,r_k\}$, having more than one clause, is called the task of defining the inexact constraint C_{ij}. The constraint itself is called *a single mark of the constaint* C_{ij},and the set $C^E=\{C_{ij}^E \mid C_{ij}^E=\{r_j\}, r_j \in C_{ij}\}$ *– a single mark of the TCSP.* In this case solution of the TCSP is its *consistent single mark.* The TCSP *is consistent* only when it has at least one solution.

The constraint C_{ij} *is feasible* for variables V_i and V_j if and only if at least there is one solution of the TCSP, where C_{ij} is a constraint between these variables. The minimal constraint C_{ij}^{min} is the set, consisting only of feasible constraints for V_i and V_j. The TCSP is called *minimal,* if all its constraints are minimal. It is known that for any TCSP it is always possible to find the equivalent minimal one or to show inconsistency of constraints [3].

Main operations for temporary constraints are the following:

1) complement ($\neg$): $\neg L_{ij}=U\backslash L_{ij}$;
2) inversion (~):$\sim(r_1,\ldots,r_k)=(\sim r_1,\ldots,\sim r_k)$;
3) intersection ($\cap$):$S \cap T =\{r:r \in S, r \in T\}$ – the set, consisting of equal constraints in S and T;
4) composition ($\bullet$):$T \bullet S=(t_1,\ldots,t_k) \bullet (s_1,\ldots,s_q)=((t_1 \bullet s_1),(t_1 \bullet s_2),\ldots,(t_k \bullet s_q))$ – disjunction of individual compositions of all elementary constraints in T and S.

Main subtasks of the task of TCSP are the following: *checking consistency* - checking whether there is a way of building the C^* set from C set (which is the solution of the TCSP); *finding consistent scenarios* - definition of all possible C^* *sets* ; *search of minimal presentation* - transition of initial TCSP to the minimal; *search for feasible constraints* for the given pair of temporary variables.

Allen proposed the interval algebra of temporal constraints wherein time intervals are taken as primitive. Reasoning within this algebra is NP-complete [6]. The point algebra is based on time points as primitives [4]. The major advantage of the point algebra is ability to construct the reasoning algorithms with polynomial complexity [7]. Further we will consider the point algebra as the base to construct TRS.

To solve the TCSP, a set of temporal variables and constraints are transformed into a graph, weighted by temporal information [7]. A *temporally labeled graph (TL-graph)* is a graph $G=(V,E)$ with at least one vertex ($V\neq\varnothing$) and a set of labeled edges, where each edge (v, l, w) connects a pair of distinct vertices v,w. The edges are either directed and labeled ≤ or <, or undirected and labeled ≠. Every vertex of a TL-graph has at least one name attached to it. If the vertex has more than one name, than these names are alternative for the same time point. The name sets of any two vertices are required to be disjoint. A path on a TL-graph is called ≤-path if each edge on the patch has a label < or ≤. A ≤-path is called <-path if at least one of the edges has label <. Given a TL-graph G an interpretation of G is a triple <T,I,R> where T is a totally ordered set (with ordering <), I is a function $I:P\rightarrow T$ such that for all $p_i,p_j\in P$ if $\mu(p_i)=\mu(p_j)$ than $I(p_i)=I(p_j)$; R is a function mapping each label I on the edges of G into corresponding binary constraint $R(I)$ on T. Given a TL-graph G a model of G is an interpretation such that if (v_1,I,v_2) is an edge of G, than for all $p_i,p_j\in P$, satisfying $\mu(v)=v_1$ and $\mu(p_j)=v_2$ $<I(p_i),I(p_j)>\in R(I)$. TL-graph is consistent if and only if it has at least one model. Two or more TL-graphs are logically equivalent if and only if they has same models [7].

TL-graph G contains an *implicit* < *constraint* between two vertices v_1,v_2 when the strongest constraint entailed by the set of constraints from which G has been build, is $v_1<v_2$ and there is no <-path from v_1 to v_2. A TL-graph without implicit < constraints is an explicit TL-graph. An explicit TL-graph entails $v = w$ if and only if v and w are alternative names of same vertex; $v < w$ if and only if there is a <-path from v to w; $v \leq w$ if and only if there is a ≤-path from v to w, and there is no any <-path from v to w; $v \neq w$ if and only if there is a <-path from v to w, or there is a <-path from w to v, or there is an edge $(v,\neq,w)$. A *Time-graph* is an acyclic TL-graph portioned into a set of time chains (≤-path), such that each vertex is on one and only one time chain. Search of solution of the TCSP is based on transformation of TL-graph to Time-graph, because if we pass from TL-graph to Time-graph, problems of checking consistency and definition of all feasible constraints will be solved automatically [7].

Algorithms for solving TCSP

During processing inexact information and after solving the task for set of exact constraints search algorithms with returns for processing set of inexact point constraints are used $D=\{D_i: D_i = (x\{R_1\}y) \vee (w\{R_2\}z) \vee \ldots \vee (t\{R_k\}u)\}$.

Disjunctive Time-graph (D-Time-graph) is the pair <*T,D*>, where *T* – Time-graph and *D* – a set of disjunctions in point algebra (PA). Set elements D : $D_i = (x\{R_1\}y) \vee (w\{R_2\}z) \vee \ldots \vee (t\{R_k\}u)$ ($x,y,z,w,..,t,u$ – temporal variables, $R_1,R_2,\ldots,R_k$ - constraints, i=1..n). *Realization* of *D* disjunctions set for Time-graph *T* is a STC in PA *M*, where one clause out of each *D* set disjunction enters, and Time-graph, received by adding *T* constraint from *M,* is consistent. D-Time-graph <*T,D*> is consistent only when there is the realization of D disjunctions set for Time-graph *T*. D-Time-graph <*T,D*> is *exact* in case it is consistent and doesn't contain implicit relations. In order to get explicit D-Time-graph it is necessary to define realization of sets of binary disjunctions *D* for graph *T*. In the general case, in order to solve this task for *k* disjunctions in *D* set it is necessary to check 2^k possible variants of solution in the worst case. In order to find solutions let's use modification of backtracking algorithm:

Modified backtracking algorithm

Input: *D* – set of inexact constraints; C – consistent set of exact point constraints.
Output: M – realization of inexact constraints set, solvability flag.
Operations: For constraint $D_i=(x_1\{R_1\}y_1)\vee\ldots\vee(x_k\{R_k\}y_k)$ defined operations: $|D_i = k, D_i[m]=\{(x_m\{R_m\}y_m)\}$

```
01: M ← Ø
02: Rollback ← false
03: foreach (j ∈ [0, |D|))  Active[j] ← 0
04: j ← 0
05: while (j < |D|) {
```

```
06:  Decided ← false
07:  i ← Active[j]
08:  if (Rollback)  i ← i+1
09:  Rollback ← false
10:  while ((i < |Dj|) && !Decided) {
11:    M ← M ∪ Dj[m]
12:    if (TCSP with STC C∪M is consistent)  Decided ← true else M ← M \ Dj[m]
13:    i ← i+1
14:  }
15:  if (Decided)  j ← j+1 else {
16:    Rollback ← true
17:    j ← j-1
18:  }
19:  if (j<0) return (false, M)
20:}
21: return (true, M)
```

During modification of initial TCSP the following situations are possible: set of exact constraints has changed; set of inexact constraints has changed; both set of exact and set of inexact constraints have changed. Backtracking algorithm takes significant part of time, necessary to solve the TCSP, that's why during step=by=step search of solutions it is desirable to minimize the number of its recurrent calls, what is reached by deleting corresponding constraints. In the situation when only inexact constraints change (i.e. there exists some set of inexact constraints *D*,* which is necessary to add to *D),* it is possible to initiate the backtracking algorithm not from the very beginning, but from the moment of processing the new constraints (obviously, if we start alg. 1 for the set $D \cup D^*$, we'll spend time to calculate earlier received set *M* for set *D,* and only after it will be finalized up to *M** in the result of constraints analysis from *D**). It is possible to build algorithm, which significantly reduces the number of complete repeated analysis of set of disjunctive constraints, because the introduction of the constraint α to the STC of the TCSP *C,* not requiring the solution of the TCSP with STC $C \cup \alpha$.

Case-based reasoning

CBR, like analogous reasoning, is based on analogy, however, there are certain differences in their implementation [8]. A precedent is defined as a case that took place earlier and is an example or justification for subsequent events of this kind. As the practice shows, when a new problem situation arises, it is reasonable to use CBR method without drawing an analogy. This is caused by the fact that humans operate with these reasoning schemes at the first stages, when they encounter a new unknown problem. CBR solves new problems by adapting previously successful solutions to similar problems. The processes involved in CBR can be represented by a CBR-cycle [9]. Usually, CBR-cycle includes the four main stages:

- *RETRIEVE* the most similar case(s) from the case library (CL);
- *REUSE* the retrieved case(s) to attempt to solve the current problem;
- *REVISE* the proposed solution in accordance with the current problem if necessary;
- *RETAIN* the new solution as a part of a new case (precedent).

The main advantages of CBR include the possibility to use the experience gained by the system for solution new problem situation, without the intensive involvement of experts in a particular problem domain, and the exception of the repeated erroneous decision. In addition, CBR does not require an explicit problem domain model.

The disadvantages of CBR may include the following: the description of cases is usually limited to superficial knowledge of problem domain; a large number of cases may lead to a decrease in system performance; complexities in definition of criteria for indexation and case comparison.

The successful implementation of CBR is necessary to ensure the correct case retrieval from CL. The choice of case retrieval method directly linked to the way of case representation.

Methods of case representation and case retrieval

CL could be a part of the knowledge base of IDSS, but may act as an independent component of the system. There are different ways of representation and storage of cases [10] - from the simple (linear) to the complex hierarchical. It should be noted that the simple methods, based typically on the technology of databases, require much less costly to implement, as well as maintenance and support of CL than the complex. However, the time to search a solution with a simple case representation may require substantially more than in other complex ways of representation and storage of cases.

Usually a case comprises: the *problem* that describes the object state when the case occurred, the *solution* of the problem (diagnosis of problem situation and recommendations for DMP), and/or the *outcome* which describe the object state after the case occurred.

Cases which comprise problems and their solutions can be used to derive solutions to new problems. Other cases which comprise problems and outcomes can be used to evaluate new situations. If, in addition, such cases contain solutions they can be used to evaluate the outcome of proposed solutions and prevent potential problems. Cases can be represented in a variety of forms using the full range of AI representational formalisms including frames, objects, predicates, semantic nets and rules.

In most cases, the simple parametrical representation of precedents can be used, i.e. case description in the form of a set of parameters with specific values and the solution:

$$CASE = (x_1, x_2, ..., x_n, R),$$

where $x_1 ... x_n$ – the parameters of the problem situation, describing this case ($x_1 \in X_1, ... , x_n \in X_n$), R - the solution, n – the number of case parameters, and $X_1, ..., X_n$ - the domains of permissible values for case parameters .

Well known methods for case retrieval (nearest neighbor, induction et al.) can be used alone or combined into hybrid retrieval strategies [8, 11].

1) The Nearest Neighbor (NN) method. This is the most common method of cases retrieval. The main advantages of this method are simplicity of implementation and universality in the sense of independence from the specifics of a particular problem domain. This approach involves the evaluation of similarity between stored cases and the new input case. A main constraint of this approach is the linear dependence of the retrieval time to the number of cases in the CL. Therefore this approach is more effective when the CL is relatively small. This method is also widely used to solve the problems of classification, clustering, regression and pattern recognition.

2) Induction method of case retrieval. Induction algorithms (e.g. ID3) generate a decision tree type structure to organize the cases in the CL. This method involves retrieval the required cases by resolving the decision tree tops. This approach is recommended for the grate CL in order to reduce the retrieval time.

3) Method of case retrieval on the basis of knowledge. In contrast to the methods described above, this method allows to take into account the knowledge of experts (DMP) in a specific problem domain (the importance of object parameters, identified dependencies and so on). The method can be successfully applied in combination with other methods of case retrieval, especially when the CL is great and problem domain is open and dynamic.

4) Method of case retrieval, taking into account the applicability of cases. This method of case retrieval is based not only on similarity of cases with a new problem situation, but also on their applicability in the circumstances. In some systems, this problem is solved by maintaining cases, together with comments of their application.

Use of the mechanism of cases for RT IDSS consists in output of the decision to the operator (DMP) for the current situation on the basis of cases which is contained in system. As a rule, the last stage in a CBR-cycle is excluded and performed by the expert (DMP) because the CL should contain only reliable information confirmed by the expert. Revising and adaptation of the taken decision is required seldom because the same object (subsystem) is considered. More detailed the modified CBR-cycle for RT IDSS is considered in [8, 11]. Thus,

CBR for RT IDSS consists in definition of similarity degree of the current situation with cases from CL. For definition of similarity degree in simple case (parametrical representation of precedents) the NN-algorithm and its modifications are used [11]. For more complex structure of cases like Time-graph the methods of case retrieval on the basis of structural analogy [12] (structural analogy using the context and structural analogy based on Structure Mapping Theory (SMT)) are used.

Application of temporal and CBR in RT IDSS of the car access control

One of the fields where the methods and algorithms, described above, are used is creation of distributed RT IDSS for access control. The considered apparatus is used in the payable car access control system (ACS) sPARK [3]. Modern parking solutions is the complicated complexes, which are equipped with an automatic barriers, the video cameras, fire and access alarm, etc. The major target of the car ACS is passage control of the cars, registration of the visitors and the car owners, stealing prevention and control of equipment in real time. The object of access in the car ACS is the car. The system should control that the car successfully entered to the parking territory. The necessity to control the passage process leads to take into account the temporal dependencies. Let's consider the simple rule, which system should check at the time of car passage [3]:

(1) *If operation of the entrance A_i was activated at the moment t_1 and to the time moment t_2:t_2-t_1> 2.5 min. operation A_i is still active, then display to operator the notice "delay at the entry".*

The quantity of such or more complicated rules for the basic car access point can reach several tens. The ability of analysis of the sequences of observed by the system actions permits to implement more reliable automatic parking complex control. For example, when the system recognizes the temporal situation, which represented at fig. 2, it can make decision, that visitor making attempt to leave parking in the car, which is not the car in which he had enter to the parking, and notice guards about it.

The developed CBR system (Case Libraries Constructor – CLC) was applied in RT IDSS of the car ACS for diagnostics and detection of different problem (abnormal, critical) situations on object.

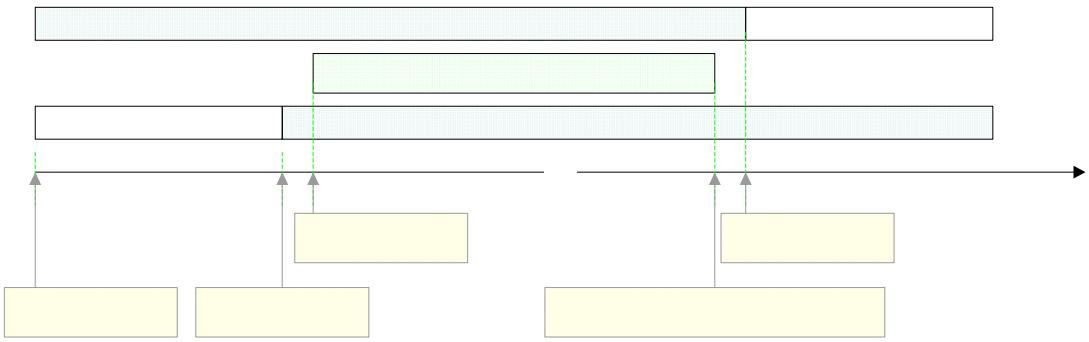

Fig. 2 The temporal diagram of the situation, which is suspicious to car stealing

Conclusion

Possibility to process information about time in the process of its coming is a very important task for plenty of field of AIS. In this work perspective algorithms of the TCSP solution are considered, which are oriented towards the use in distributive IDSS, oriented for dynamic value domains, in particular distributed RT IDSS and described algorithms are implemented in TRS PointTime [3], TRS Singularity and have been approbated in real RT IDSS – the system of access control and security provision – sPARK. The CBR method was considered from the aspect of its application in RT IDSS, in particular, for a solution of problems of real-time diagnostics, forecasting and detection of problem situations on object. The CBR-cycle is considered and different methods of case representation and retrieval are investigated. The proposed CBR method was implemented in CBR system CLC.

Bibliography

1. Vagin V.N., Yeremeyev A.P. Modeling Human Reasoning in Intelligent Decision Support Systems // Proc. of the Ninth International Conference on Enterprise Information Systems. Volume AIDSS. Funchal, Madeira, Portugal, June 12-16, INSTICC, 2007, pp.277-282.
2. Allen, J. F. 1984. Towards a General Theory of Action and Time // Artificial Intelligence Vol/.23, pp.123-154.
3. Eremeev A.P., Kurilenko I.E. Implementation of temporal reasoning in Intelligent Systems // Journal of Computer and Systems Sciences International, Vol. 2, 2007, pp. 120-136
4. Van Beek. P. Reasoning about qualitative temporal information // In Proceedings the AAAI National Conference on Artificial Intelligence, 1990, pp. 728-734.
5. Eremeev, A.P. and Troitskii, V.V. Models of Representation of Temporal Relations in Intelligent Decision Support Systems // Journal of Computer and Systems Sciences International, Vol.42, No.5, 2003, pp.732-743.
6. Drakengran T. and Jonsson P. Maximal tractable subclasses of Allen's interval algebra. Preliminary report // In Proceedings of the AAAI National Conference on Artificial Intelligence, Portland, OR, 1996, pp. 389-394.
7. Gereviny A. and Schubert L. Efficient Algorithms for Qualitative Reasoning about Time. Technical report 496, Department of Computer Science, University of Rochester, Rochester, NY, 1993.
8. Eremeev A., Varshavsky P. Analogous Reasoning and Case-Based Reasoning for Intelligent Decision Support Systems // International Journal INFORMATION Theories & Applications (ITHEA) 2006, Vol.13, No.4, pp. 316-324.
9. Aamodt A., Plaza E. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches // AI Communications, IOS Press, Vol. 7: 1, 1994, pp. 39-59.
10. Leake D.B. CBR in Context: The Present and Future // Case-Based Reasoning: Experiences, Lessons and Future Directions, AAAI Press / The MIT Press, 1996, pp. 3–31.
11. Eremeev A., Varshavsky P. Application of Case-based reasoning for Intelligent Decision Support Systems // Proceedings of the XIII-th International Conference "Knowledge-Dialogue-Solution" – Varna, vol. 1, 2007, pp. 163-169.
12. Eremeev A., Varshavsky P. Methods and Tools for Reasoning by Analogy in Intelligent Decision Support Systems // Proc. of the International Conference DepCoS - RELCOMEX 2007. Szklarska Poreba, Poland, 2007, IEEE, pp. 161-168.

Authors' Information

Alexander P. Eremeev – Applied Mathematics Department of the Moscow Power Engineering Institute (Technical University), Krasnokazarmennaya str., 14, Moscow, 111250, Russia; e-mail: eremeev@appmat.ru

Ivan E. Kurilenko – Applied Mathematics Department of the Moscow Power Engineering Institute (Technical University), Krasnokazarmennaya str., 14, Moscow, 111250, Russia; e-mail: ivan@appmat.ru

Pavel R. Varshavskiy – Applied Mathematics Department of the Moscow Power Engineering Institute (Technical University), Krasnokazarmennaya str., 14, Moscow, 111250, Russia; e-mail: varp@appmat.ru