International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:5, 2015

Temporal Case-Based Reasoning System for Automatic Parking Complex

Alexander P. Eremeev, Ivan E. Kurilenko, Pavel R. Varshavskiy

***Abstract*—**In this paper the problem of the application of temporal reasoning and case-based reasoning in intelligent decision support systems is considered. The method of case-based reasoning with temporal dependences for the solution of problems of real-time diagnostics and forecasting in intelligent decision support systems is described. This paper demonstrates how the temporal case-based reasoning system can be used in intelligent decision support systems of the car access control. This work was supported by RFBR.

***Keywords*—**Analogous reasoning, case-based reasoning, intelligent decision support systems, temporal reasoning.

I. Introduction

ONE approach to solving the problem of modeling commonsense reasoning in artificial intelligence (AI) systems and especially in real-time intelligent decision support systems (RT IDSSs) is to use inductive reasoning, temporal reasoning, fuzzy logic as well as methods of reasoning based on analogies and precedents (cases) [1]–[6].

RT IDSSs are usually characterized by strict constraints on the duration of the search for the solution. One should note that, when involving models of case-based and temporal reasoning in RT IDSS, it is necessary to take into account a number of the following requirements to systems of this kind [1], [4]:

- The necessity of obtaining a solution under time constraints defined by real controlled process;
- The necessity of taking into account time in describing the problem situation and in the course of the search for a solution;
- The impossibility of obtaining all objective information related to a decision and, in accordance with this, the use of subjective expert information;
- Multiple variants of a search, the necessity to apply methods of plausible (fuzzy) search for solutions with active participation of a decision making person (DMP);
- Nondeterminism, the possibility of correction and introduction of additional information in the knowledge base of the system.

Temporal reasoning and case-based reasoning (CBR) can be used in various applications of AI and for solving various problems, e.g., for diagnostics and forecasting or for machine learning [1], [3]–[5].

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 (sensors data, values of control parameters, information from DMP, etc.) appears during the process of solution of many problems. RT IDSS must solve diagnostics, monitoring, decision searching and forecasting problems uninterruptedly in a real-time scale in order to help DMP to find efficient control actions 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 what naturally positively affects reactivity of the whole system. Thus, the use of the corresponding methods in RT IDSS broadens the possibilities of RT IDSS and increases the efficiency of making decisions in various problem (abnormal) situations.

A. P. Eremeev, I. E. Kurilenko, and P. R. Varshavskiy are with the National Research University "MPEI", Moscow, Russia (e-mail: eremeev@appmat.ru, ivan@appmat.ru, VarshavskyPR@mpei.ru).

II. Case-Based Reasoning

A. CBR Cycle

CBR, like analogous reasoning, is based on analogy; however, there are certain differences in their implementation. 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. 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 (Fig. 1).

Usually, CBR cycle includes the four main stages [7]:

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

B. Methods of Case Representation

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 a case representation. There are different ways of the representation and storage of cases – from the simple (linear) to the complex hierarchical [2], [3], [5].

Usually a case comprises [2]:

- the *problem* that describes the object state when the case occurred,
- the *solution* of the problem (diagnosis of problem

situation and recommendations for DMP),
– the *outcome* which describe the object state after the case occurred.

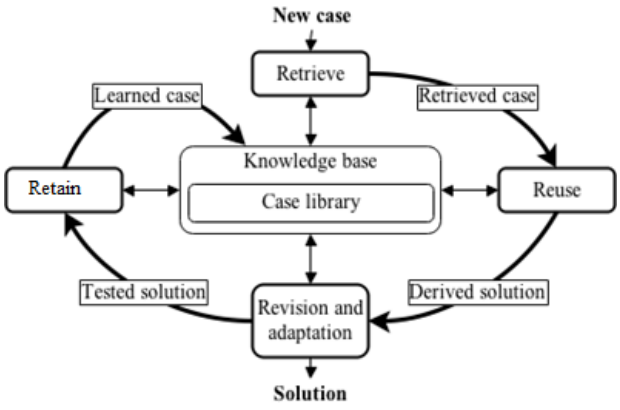

Fig. 1 CBR cycle

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. For example, the form of the case with an explicit structure of events that must occur in correspondence with the situation can be used for construct temporal cases. This structure contains two components: the time component (explicitly or implicitly) and descriptive component (not storing a time information). As a means of explicit description of temporal dependencies is proposed to use temporal logics that are based on the presenting temporal dependencies in the form of temporal constraints. The combination of the expressive possibilities of temporal logic and the descriptive component, allow constructing different classes of representations for temporal cases with different expressive power and complexity of the inference algorithms.

C. Advantages and Disadvantages of CBR Methods

The main advantages of CBR include the possibility to use the experience gained by the system for solution a 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 a 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. Another disadvantage is that in most implementations of CBR systems, a simple parametric description of the situation at certain fixed time is used. However, this approach imposes restrictions on the expressive power and range of tracked and recognizable situations. Using the "instantaneous" snapshot of the key parameters of the controlled object leads to the fact that the conclusions do not take into account the history of parameter changes, which leads to the impossibility of presenting time and causality. A lot of basic notions, such as "alteration", "cause", "consequence/effect" and relations among them can be described by time notions. The majority of real processes operate in accordance with the law and some time, given the history of changes in the state of the observed object or process allows a more accurate decisions and recommendations than on the basis of a simple analysis of the "instant" snapshot of parameters. Fig. 2 contains an example of a case description with temporal information about case parameters changes in time.

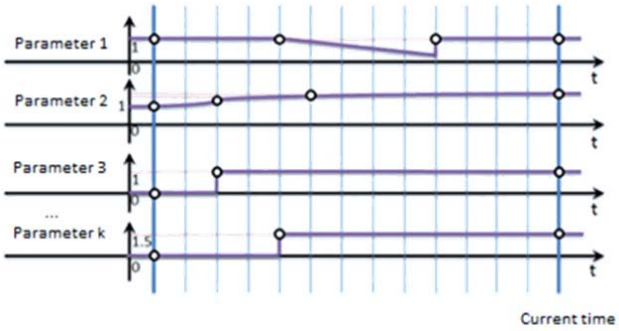

Fig. 2 Description of case parameters changes in time

The temporal CBR-method, taking into account the features of the actual process, allows obtaining more accurate solution than the methods based on "instant" snapshots.

III. Methods of Case Retrieval

Well-known methods for case retrieval (nearest neighbor, induction et al.) can be used alone or combined into hybrid retrieval strategies [5].

The Nearest Neighbor (NN) Method

This is the most common method of comparison and retrieval of cases. 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 a 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.

Induction Method of Case Retrieval

Induction algorithms (e.g. ID3, C4.5) generate a decision tree type structure to organize the cases in the CL. This method involves retrieval the required cases by resolving the nodes of decision tree. This approach is recommended for the grate CL in order to reduce the retrieval time.

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.

Method of Case Retrieval, Taking into Account the Applicability of Cases

In some systems, this problem is solved by maintaining cases, together, with comments of their application.

Using the mechanism of cases for RT IDDS consists in issuing a decision to the operator (DMP) based on the existing cases. Thus, CBR for RT IDSS consists in the definition of a similarity degree of the current situation with cases from CL. For the definition of a similarity degree in a simple case (parametrical representation) the NN algorithm and its modifications are used. For more complex structure of cases like temporal cases, the methods of case retrieval on the basis of structural analogy (structure-mapping theory (SMT) [8]–[10]) and temporal logics [6], [11]–[17] are used.

SMT allows formalizing the set of implicit constraints, which are used by a human who operates with notions such as analogy and similarity. This theory uses the fact that an analogy is a mapping of knowledge of one domain (base) in another domain (target) based on the system of relations between objects of the base domain, as well as the target domain. The main principle of SMT is a principle of systematicity, that reflects the fact that humans (DMP) prefer to deal with a system of connected relations, not just with a set of facts or relations.

According to SMT, the inference process on the basis of analogies consists of the following stages:

1. *Definition of potential analogies*. Having the target situation (target), define another situation (base) that is analogous or similar to it.
2. *Mapping and inference*. Construct mapping that consists of matchings between the base and the target. This mapping can contain additional knowledge (facts) about the base that can be transferred to the target. These pieces of knowledge are called candidates of conclusions formed by an analogy.
3. *Estimation of matching "quality"*. Estimate the obtained correspondence using structural criteria such as the number of similarities and differences, a degree of structural correspondence, and the quantity and a type of new knowledge synthesized by analogy from the conclusion candidates.

Consider the structure-mapping engine (SME) which is based on SMT. This mechanism is suited for modeling inference by an analogy providing matching an estimation independent of the problem domain.

The input data for the SME algorithm are structural representations of the base and target domains.

SME algorithm consists of the following steps:

Step1. Constructing local mappings. Determine the matches (mapping hypotheses) between separate elements in the base and target domains by means of the following rules:

1. If two relations have the same name, then create a mapping hypothesis.
2. For the mapping hypothesis between relations, test the arguments: if they are objects or functions, then create for them local mapping hypotheses. Determine plausibility estimations for these local hypotheses using the following rules:

(a) increase a plausibility degree for the correspondence if the base and the target relations have the same names;

(b) increase a plausibility degree for the correspondence if it is known that the base relation has the parent relation.

Rule (a) prefers the identity of relations, and (b) reflects the principle of the systematic character of relations.

Step2. Construction of global mappings. Form the mapping systems that use compatible pairwise of objects. Join them in systems of relations with compatible mapping of objects. With each global mapping of this kind (Gmap), relate the set of conclusion candidates.

Step3. Construct conclusion candidates. For each mapping Gmap, construct a set of the facts (possibly empty) that occur in the base domain, which does not occur originally in the target domain.

Step4. Estimation of global matches. The global matches receive a structural estimate that is formed taking into account the plausibility of local correspondence. Terminate.

Thus, as a result, the most systematic consistent mapping structure Gmap has following components:

- *matchings* is a set of pairwise mappings between base and target domains;
- *conclusion candidates* is a set of new facts that presumably are contained in the target domain;
- *structure estimation* is a numeric equivalent of the match quality based on the structural properties of Gmap.

The main advantages of SME that are especially important for RT IDSSs are the polynomiality of the considered SME algorithm and the simplicity of importing the conclusion candidates in the target domain.

This paper proposes a hybrid approach to finding solutions based on CBR. SME algorithm used to determine similarity at the first stage, and NN algorithm used at the second stage to compare values.

IV. Temporal Reasoning

To implement the mechanism of temporal reasoning, it is necessary to formalize the notion of time and to provide the possibility to present temporary aspects of knowledge.

Modern approaches to presentation of time and temporary 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 ("snapshots"), which don't have any time duration. Time itself is regarded implicitly, via modeling the system changes within time. Approaches, based on changes modeling, have constraints when presenting difficult time dependences (events, that have duration, continuance of processes, causal

relations etc.). Some different ways to eliminate this 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 some specific tasks of temporary reasoning are raised such as checking the temporal information consistency. In the case of inconsistency of temporal constraints, it is necessary to localize plenty of expressions, responsible for this inconsistency. Reasoner should be able answer 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. Explicit time modelling allows present and reason about continuous processes checking temporal constraints and dependencies.

Three main groups can be distinguished in the class of approaches, based on explicit time modeling:

1) Temporary approaches expanding on the basis of time changes modeling;
2) Introduction of a time factor to logics;
3) The models, built on the basis of paradigm of constraint agreement.

In this paper we consider the models, based on the presentation of information about time as constraints (dependences) between time primitives belong to the third group. In temporary logics using the concept of constraint satisfaction, information about time is presented as dependences between temporary primitives (moments, intervals or their combinations). Dependences between primitives are interpreted as constraints to real time of their appearance. Main aim of the temporal reasoning system (TRS) is a generations of conclusions on the basis of sets of temporary constraints, i.e. new constraints for consistent input sets. Usually sets of temporary 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 $Z = (V, D, BTR, C)$ [11]:

1) $V = \{V_1, V_2, ..., V_m\}$ – 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) $C = \{C_{ij} | C_{ij} = \{r_1, ..., r_k\}, k>0, r_1, ..., r_k \in BTR, i<m, j<m\}$, 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 $(V_i R_1 V_j) \vee \ldots \vee (V_i R_k V_j)$. The case C_{ij} consists only of one clause, it is called an *exact* restriction.

It is necessary to find such a STC $C^* = \{C_{ij}^* | C_{ij}^* = \{r_j\}, r_j \in C_{ij}\}$, 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. Constraints between time primitives are represented as qualitative binary constraints, because on their basis any constraints of a higher order can be presented [11].

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.

The task of defining r restriction valid for variables V_i and V_j with the constraint $C_{ij} = \{r_1, ..., r_k\}$, 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 restriction* C_{ij}, and the set $C^E = \{C_{ij}^E | C_{ij}^E = \{r_j\}, r_j \in C_{ij}\}$ – *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 executable* 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 executable 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 [15].

Main operations for temporary constraints are the following:

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

The set 2^{BTC} of all temporary constraints is closed relatively to the given operations and forms the algebra of temporary constraints.

Main subtasks of the task of TCSP are the following:

- *defining consistency* (or *the task of satisfiability)*, consisting of 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* - transformation of initial TCSP to the minimal one;
- *definition of sets of satisfiable constraints* for the given pair of temporary variables.

Allen proposed the interval algebra of temporal constraints wherein time intervals are taken as primitive [12]. Reasoning

within this algebra is NP-complete [13]. The point algebra is based on time points as primitives [14]. The major advantage of the point algebra is ability to construct the reasoning algorithms with polynomial complexity [15], [16]. Further, we will consider the point algebra as the base to construct TRS.

To solve the TCSP, a set of temporary variables and constraints among them are transformed into a graph, weighted by temporal information [16]. 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 $\leq$ or $<$, or undirected and labeled $\neq$. We assume that every vertex of a TL-graph has at least one name attached to it. If the vertex has more than one name, then 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 $\leq$-path if each edge on the patch has a label $<$ or $\leq$. A $\leq$-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 l on the edges of G into corresponding binary constraint $R(l)$ on T. Given a TL-graph G a model of G is an interpretation such that if (v_1,l,v_2) is an edge of G, than for all p_i, $p_j \in P$, satisfying $\mu(p_i)=v_1$ and $\mu(p_j)=v_2$, $<I(p_i),I(p_j)> \in R(l)$. 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 [16].

TL-graph G contains an *implicit $<$-relation* between two vertices v_1, v_2 when the strongest relation entailed by the set of relations 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 $<$-relations 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 $\leq$-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* (Fig. 3) is an acyclic TL-graph portioned into a set of time chains, such that each vertex is on one and only one time chain. A time chain is a $\leq$-path, plus possibly transitive edges connecting pairs of vertices on the $\leq$-path. If we pass from TL-graph to Time-graph, problems of attainability and definition of all executable constraints will be solved automatically. Search of solution of the TCSP is based on transformation of TL-graph to Time-graph [16], [17].

Now we can define new model of case (temporal case) as $S = <T, P, A, D>$, where $T=(V,C)$ is TCSP (V – set of temporal variables, C – set of temporal constraints between variables in V), P – is finite set of controlled object parameter sets, A: $V \rightarrow P$ – function for mapping sets of controlled object parameters to temporal structure, D – solution. Using this model allows to take into account temporal information in the inference mechanism.

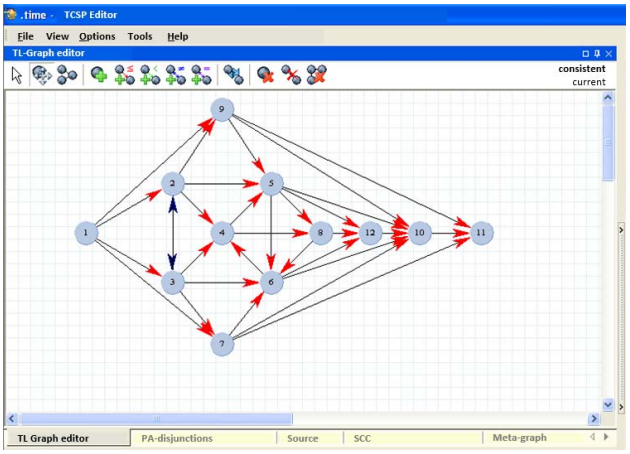

Fig. 3 Example of Time-graph editing

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: $D_i = (x\{R_1\}y)$ v $(w\{R_2\}z) \vee \ldots \vee (t\{R_k\}u)$ (x, y, z, w,..,t, u – temporary 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 sets of disjunctions D 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 obvious 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 ← {}
02: Rollback ← false
03: foreach (j ϵ [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)
```

```
13:    Decided ← true else M ← M \ Dj[m]
14:   i ← i+1
15:  }
16:  if (Decided) j ← j+1 else {
17:   Rollback ← true
18:   j ← j-1
19:  }
20:  if (j<0) return (false, M)
21:}
22: return (true, M)
```

During modification of initial TCSP the following situations are possible: set of exact constraints has changed; set of inexact constraints also 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 algorithm 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 $C \cup \alpha$ [17].

V. Application of Temporal CBR in RT IDSS of the Car Access Control

One of the fields where the methods and algorithms of temporal CBR, described above, are used is creation of distributed RT IDSS for access control. The considered apparatus and implemented temporal CBR system (Fig. 4) are used in systems of payable access control sPARK [18].

Modern parking solutions are the complicated complexes, which are equipped with automatic barriers, the video cameras, fire and access alarm, etc. The major target of the car access control system is passage control of the cars, registration of the visitors and the car owners, stealing prevention. The object of access in the system of car access control is the car. The execution units are the barriers and the gates, which system should open before the passage and close after car entrance completed (Fig. 5).

So, 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. The major task of the system is control of equipment in real time. The ability of analysis of the sequences of observed by the system actions permits to implement more reliable automatic parking complex control.

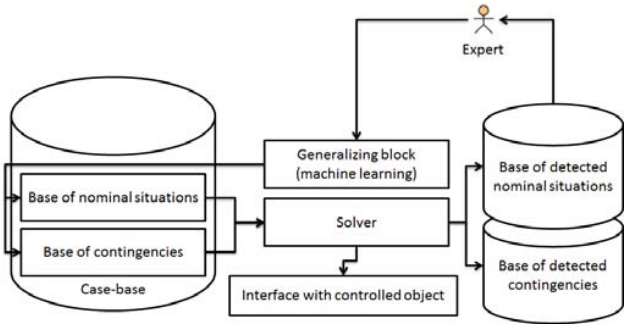

Fig. 4 Architecture of temporal CBR system

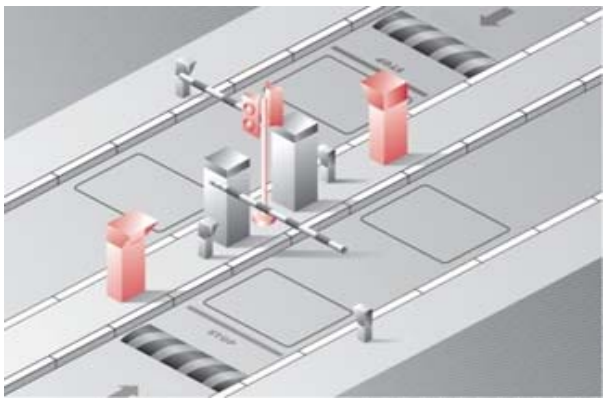

Fig. 5 Typical access point to parking

(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, that is represented at Fig. 6, 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 temporal CBR system was applied in RT IDSS of the car access control for diagnostics and detection of different problem (abnormal, critical) situations on object.

VI. Conclusion

Possibility to process information about time in the process of its coming is a very important task for plenty of field of AI systems and CBR systems. The CBR cycle is considered and different methods of case representation and retrieval are investigated. This paper proposes a hybrid approach to finding solutions based on CBR. SME algorithm used to determine similarity at the first stage, and NN algorithm – at the second stage to compare values. In this work perspective algorithms of the TCSP solution are considered, that are oriented towards the use in distributive IDSS, oriented for dynamic value

domains, in particular distributed RT IDSS and described algorithms are implemented in temporal CBR system and have been approbated in RT IDSS – the system of access control and security provision – sPARK. The temporal CBR system 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.

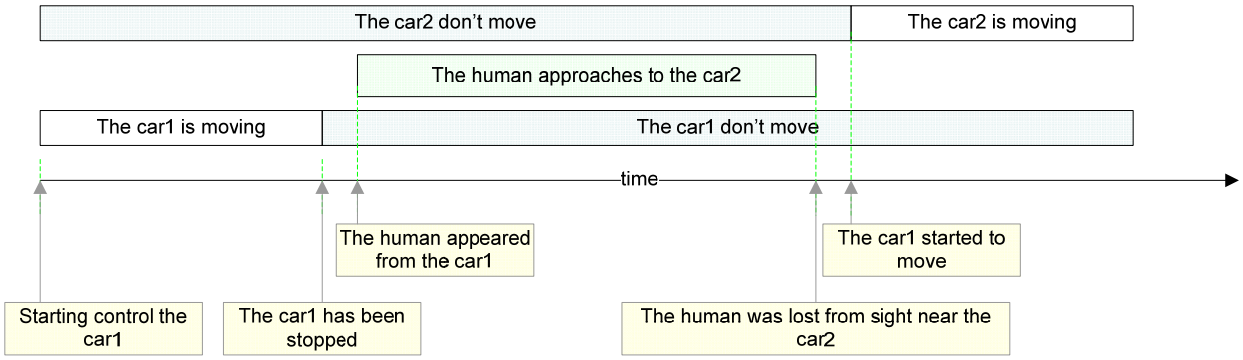

Fig. 6 The temporal diagram of the situation, which is suspicious to car stealing

References

[1] Eremeev A. P., Vagin V. N. Methods and tools for modelling reasoning in diagnostic systems // ICEIS 2009 – 11th International Conference on Enterprise Information Systems, Proceedings, AIDSS, 2009, pp. 271–276.
[2] David C. Wilson, David B. Leakes, Maintaining Case-Based Reasoners: Dimensions and Directions // Computational Intelligence, Volume 17, Number 2, 2001, pp. 196–213.
[3] 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.
[4] 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.
[5] Varshavskii P.R., Eremeev A.P. Modeling of case-based reasoning in intelligent decision support systems // Scientific and Technical Information Processing, 37 (5), 2010, pp. 336–345.
[6] Eremeev A.P., Kurilenko I.E. Temporal reasoning component for real-time intelligent decision-support systems // Scientific and Technical Information Processing, 38 (5), 2011, pp. 332–343.
[7] 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.
[8] Falkenhainer B., Forbus K., Gentner D. The Structure-Mapping Engine: Algorithm and examples // Artificial Intelligence, 41, 1989, pp. 1–63.
[9] Alexander Eremeev, Pavel Varshavsky, Analogous Reasoning and Case-Based Reasoning for Intelligent Decision Support Systems // International Journal INFORMATION Theories & Applications (ITHEA) 2006, Vol.13 № 4, pp. 316–324.
[10] Eremeev A., Varshavsky P. Methods and Tools for Reasoning by Analogy in Intelligent Decision Support Systems // Proc. of the International Conference on Dependability of Computer Systems. Szklarska Poreba, Poland, 14-16 June, 2007, IEEE, pp.161–168.
[11] Eremeev A.P. and Troitskii V.V. Representation of Temporal Constraints in Intelligent Systems, Proc. Congress "Artificial intelligence in the XXI Century" ICAI'2001, Moscow: Fizmatlit, 2001.
[12] Allen J.F. Towards a General Theory of Action and Time. Artificial Intelligence 23, 1984, pp. 123–154.
[13] T. Drakengran and P. Johnson, 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.
[14] Vila L. A survey on temporal reasoning in artificial intelligence // AI Communications, 1994. Vol.7, no. 1, pp. 4–28.
[15] Peter van Belk, Reasoning about qualitative temporal information. In Proceedings the AAAI National Conference on Artificial Intelligence, 1990, pp. 728–734.
[16] 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.
[17] Eremeev A.P., Kurilenko I.E. Implementation of temporal reasoning in Intelligent Systems // Journal of Computer and Systems Sciences International, v. 2, 2007, pp. 120–136.
[18] Borisov A.V., Kurilenko I.E. Application of temporal reasoning in distributed system of car access control // ITMU 2005, №5, pp. 786–794.