LIMIT BEHAVIOUR OF DYNAMIC RULE-BASED SYSTEMS

Gennady Osipov

Abstract: The paper suggests a classification of dynamic rule-based systems. For each class of systems, limit behavior is studied. Systems with stabilizing limit states or stabilizing limit trajectories are identified, and such states and trajectories are found. The structure of the set of limit states and trajectories is investigated.

Keywords: Dynamic rule-based systems, set of attainable states, limit trajectories.

ACM Classification Keywords: I.2.8 Problem Solving, Control Methods, and Search

Introduction

Dynamic intelligent systems and dynamic knowledge bases are typically understood as a result of integrating expert systems with simulation systems and automatically updated knowledge bases [1].

Although research in this area has a rather long history, there are some issues that still remain unaddressed — those of global behavior of dynamic intelligent systems, their attainable sets, stability, and other issues that usually come up when studying dynamic systems [2].

In this paper, we focus on the kind of systems that are dynamic systems whose states, behavioral laws and other dynamic system "attributes" are described in a special way.

This special way consists in using intelligent system techniques for describing both states and behavioral laws for such systems (by intelligent system techniques we mean methods for knowledge representation, modeling of reasoning and behavior modeling that are common in artificial intelligence). The said does not mean that functions and variables defined in any other way cannot be used as components of such systems; what is more even, it is supposed that the systems in question allow for integration with various models, such as differential equation systems, finite automata, and others.

In a general case, attainability of knowledge-based systems is determined by their knowledge bases and control strategies [3, 4]. In the case when a set of rules is used as knowledge representation in a system [5], attainability is entirely determined by the structure of the set of rules, by the general principles of rule organization, and the control strategy being used [6].

If by a rule base architecture we mean 'structure of the rule set + rule structure + rule application logic' then, consequently, attainability of rule-based systems is entirely determined by the rule base architecture.

Let us remind the basic definitions following [4].

1 Rule-Based Systems

A rule [6] is said to be a triple of sets:

$\Pi = <C, A, D>$, where:

C is the applicability condition for the rule;

A is the set of facts to be added by the rule Π;

D is the set of facts to be removed by the rule Π.

C, A and D are sets of formulas of a language L, e.g. a multi-sorted first-order predicate calculus language, whose alphabet contains variables of sort t that take values from a linearly-ordered discrete set T. $A \cap D = \varnothing$ for every rule.

The word "fact" is used here as a synonym for the expression "closed atomic formula of a first-order predicate calculus language".

Formulas from C, A and D are turned into facts by some substitutions that will be described below.

Every rule will be assigned to one of two classes τ or θ and denoted as a τ-rule or θ-rule, respectively.

With each τ-rule, we associate either an action which is performed by an actuator in the environment or a procedure that computes and assigns to a variable the values of certain database attributes based on values taken by other attributes in the current state.

No actions are associated with θ-rules, as the latter do not affect the real world and merely update our knowledge of it.

It should be stressed that in conditions C of θ-rules first-order language formulas are such that the value of t ($t \in T$) in the formulas from the condition is the same as the value of t in the formulas from the sets A and D. This means that the result of a θ-rule execution changes the state in which its condition is satisfied.

As far as τ-rules are concerned, if $(\forall t)\ (n \leq t \leq m)\ (\exists x)\ F_C(t,x)$ is a formula from condition C and $(\forall t)\ (p \leq t \leq q)\ (\exists y)\ P_A(t,y)$ is a formula from the list A of formulas to be added (where n, p and m, q are the discrete start and end time points, respectively, for the "validity" term of facts $F_C(t,x)$ and $P_A(t,y)$), then the integer $v = p - n$, which is the time lag between the start point of the fact $P_A(t,y)$ being added and the start point of the period when the condition $F_C(t,x)$ is true, is a characteristic of each rule and is associated with it.

Things are the same with the sets of formulas to be removed.

Let us now look at the basic computational process in rule-based systems.

For this purpose, we need the following concepts to be introduced [3]: database and strategy of control over the system's rules.

1.1 Database

Database is a collection of finite relations, or tables (e.g. like those in relational databases), the number of which equals the number of different predicate symbols in the rules. Table columns correspond to the sorts of individual variables in atomic formulas. Interpretation of language L in the database is taken to be defined in a standard way.

One can therefore talk of satisfiability or non-satisfiability of rules' conditions.

1.2 Control Strategy

Control strategy picks up a rule from the set of rules, checks if its condition is satisfied in the current state of the working memory and, if so, applies the rule, i.e. performs the actions as prescribed by the rule; otherwise, it picks up the next rule and carries out the same manipulations on it.

For the sake of definiteness, we assume that the set of rules is ordered, e.g. in a lexicographic way.

Then the control strategy looks as follows:

1. Pick up the next rule Π_i from the set of rules.

2. Check whether condition C_i is true in the current state of the working memory.

3. If C_i is true, then substitute all free variables in formulas from C_i, A_i and D_i by the corresponding values from the database. Otherwise go to 1.

4. Apply the rule, i.e. write down to the working memory the values that make true the formulas from A_i and remove from the working memory the values that make true the formulas from D_i.

5. Go to 1.

The condition for the completion of the process is either stabilization of the working memory or exhaustion of the set of applicable rules.

Typically, the choice of rule depends on the task or domain specifics; the general principle consists in that the rule's condition should hold. If there is more than one such rule in a current state then the so-called conflict set resolution strategies are applied. With the latter not being the subject of this paper, we take the control strategy to be such that the choice of rule will only affect the computational complexity and not the result of the process. To put it differently, in what follows we are not going to be concerned with rule applicability, and we will get back to this later.

2 Dynamic Rule-Based Systems

Let X be a set of facts, $\chi \in 2^X$, $\Pi \in (\tau \cup \theta)$. Let $K(\chi,\Pi)$ denote the control strategy we described above, and we assume

$$K(\chi,\Pi^\theta) = \Phi(\chi), \text{ where } \Pi^\theta \in \theta,$$
$$K(\chi,\Pi^\tau) = \Psi(\chi), \text{ where } \Pi^\tau \in \tau.$$

$\Phi(\chi)$ will be referred to as a closure function, $\Psi(\chi)$ as a transition function.

Then

$$H = <X, T, \Phi, \Psi> \quad (1)$$

will be referred to as a dynamic rule-based system.

The fixed point of equation

$$\Phi(\chi) = \chi$$

will be referred to as a state of the system (1), and the fixed point of equation

$$\Psi(\Phi(\chi)) = \chi \text{ (if such exists) with } t \to \infty,$$

will be referred to as the limit state of the system (1).

3 Classification of Rule-Based Dynamic Systems

As the basis for classification we will take the form of system's rules and certain correlations on the sets of rule components.

First, we identify classes of systems which differ in the form of rules.

In system H1 the rules are of the form:

$$\Pi_1 = <C, \{P(t,y)\}, \varnothing>$$

(here P(t, y) is a fact to be added).

In system H2 the rules are of the form:

$$\Pi_1 = <C, \{P(t,y)\}, \{\Phi(t,z)\}>$$

(here P(t, y) is a fact to be added, Φ(t, z) is a fact to be removed).

In system H3 the rules are of the form:

$$\Pi_1 = <C, P(t,y), F(t,z)>$$

(here P(t, y) is a set of facts to be added, F(t,z) is a set of facts to be removed).

Let us now identify classes of systems, based on some correlations on the sets of rule components. Let S_0 be the initial state.

Then system H21 is a system H2, such that:

$(\cup\{P\}) \cap (\cup\{\Phi\}) = \varnothing$ (where $(\cup\{P\})$ and $(\cup\{\Phi\})$ is the union of facts being added and removed, respectively, over all of the rules of system H2);

system H22 is a system H2, such that $S0 \cap (\cup\{\Phi\}) = \varnothing$;

system H23 is a system H2, such that $(\cup\{P\}) \cap (\cup\{\Phi\}) \neq \varnothing$ and $S0 \cap (\cup\{\Phi\}) \neq \varnothing$;

system H31 is a system H3, such that $(\cup P) \cap (\cup F) = \varnothing$;

system H32 is a system H3, such that $S0 \cap (\cup F) = \varnothing$;

system H33 is a system H3, such that $(\cup P) \cap (\cup F) \neq \varnothing$ and $S0 \cap (\cup F) \neq \varnothing$ (here $(\cup P)$ and $(\cup F)$ stand for the union of the sets of facts being added and removed, respectively, over the entire set of the rules of system H3).

4 Limit States of Dynamic Rule-Based Systems

Let us give a few rather simple statements without proof:

Statement 1. The limit state of system H1 equals S0 ∪ (∪{P}).

Statement 2. The limit state of system H21 equals (S0 / (∪{Φ})) ∪ (∪{P}).

Statement 3. The limit state of system H31 equals (S0 / (∪F)) ∪ (∪ P).

Statement 4. In systems H22, H23, H32, H33 stabilization of states never occurs, as a matter of fact, but every state of every one of these systems lies in the set S0 ∪ (∪{P}) — for systems H22 and H23 or the set S0 ∪ (∪ P) — for systems H32 and H33.

Statement 5. The trajectories of systems H22, H23, H32 or H33, with t large enough, look as shown by diagram 1, where for H22 and H32 for all i: Si0 ⊆ Si1.

It is appropriate to call such trajectories *limit trajectories*.

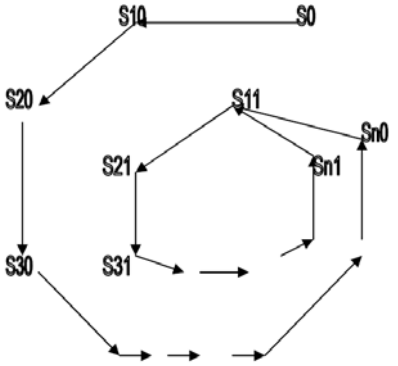

Diagram 1

5. Structure of the Set of Limit States

Let us represent the structure of the set of limit states as set inclusion diagrams. The arrow pointing from a smaller set to a larger one plays the role of the inclusion relation. We also assume that all facts to be added and removed of all the systems under consideration belong to set X. Then it is evident enough that the following diagram 2 holds for systems H1, H21 and H22:

Diagram 2

Let now systems H31, H32 and H33 be such that to each rule of system H31 a rule of system H32 is related in such a way that for each rule Π(H31) of system H31 there is such a rule Π(H32) in system H32 that C(H31) = C(H32), P(H31) = P(H32), and the inverse holds true (where C and P are the applicability conditions and the sets of facts to be added of systems H31 and H32, respectively). Then, if system H33 is such that for each rule Π(H33) of system H33 there is a rule Π(H31) in system H31 and there is a rule Π(H32) in system H32 such that C(H31) = C(H32) = C(H33), P(H31) = P(H32) = P(H33) and F(H31) ⊆ F(H33) and F(H32) ⊆ F(H33), then the following diagram 3 holds:

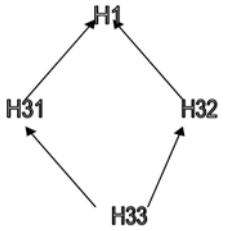

Diagram 3

In the last two diagrams, the inclusions of H22, H23, H32 and H33 in H1 have a slightly different meaning from others: they mean inclusion in H1 of every state of the limit trajectory.

Now let us come back to the postponed question of rule applicability. The situation is as follows: taking rule applicability into account may lead to some rules proving inapplicable on a certain step. It can be shown that if one sticks to the control strategy described in Section 1.2 then diagram 2 will remain the same. Diagram 3 will change its appearance to that of diagram 4:

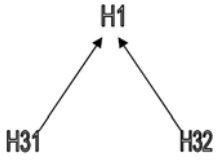

Diagram 4

Conclusion

In the paper, classes of dynamic systems have been identified from the point of view of their architecture. It has been shown that it is precisely the architectural specifics of such systems that determine their behaviour. The classes of systems with stabilizing limit states have been specified, and these states have been found. For systems with no stable limit states, the limit trajectories have been found (in case of a finite rule set).

The structures of limit states and trajectories have been established based on the criterion of attainable set inclusion.

Bibliography

[1] V.L.Stefanuk. Dynamic Expert Systems. The International Journal of Systems & Cybernetics. V. 29, Issue 5, pp. 702-709, MCB University Press, 2000.

[2] M.Mesarovich, Ya.Takakhara. General Theory of Systems: Mathematical Foundations. Moscow: Mir, 1978 (in Russian).

[3] G.S.Osipov. Dynamics in Integrated Knowledge-Based Systems. Proc. of the 1998 IEEE International Symposium on Intelligent Control (September 14-17,1998). Gaithersburg, Maryland, USA, 1998, pp.199-2003.

[4] T.G.Lebedeva, G.S.Osipov. Architecture and Controllability of Knowledge-Based Discrete Dynamical Systems. Journal of Computer and System Sciences International. NY, Vol. 39, No.5, 2000.

[5] N.Nilson. Principles of Artificial Intelligence. Moscow: Radio i svyaz', 1985.

[6] A.N.Vinogradov, L..Yu.Zhilyakova, and G.S.Osipov. Dynamic Intelligent Systems: II. Computer Simulation of Task-Oriented Behavior. Journal of Computer and System Sciences International. NY, Vol. 42, No. 1, 2003

Author's Information

Gennady Osipov – Institute of Systems Analysis, RAS, 9 Prospekt 60-letiya Oktyabrya, Moscow 117312; e-mail: gos@isa.ru

CASE-BASED REASONING METHOD FOR REAL-TIME EXPERT DIAGNOSTICS SYSTEMS

Alexander Eremeev, Pavel Varshavskiy

Abstract: The method of case-based reasoning for a solution of problems of real-time diagnostics and forecasting in intelligent decision support systems (IDSS) is considered. Special attention is drawn to case library structure for real-time IDSS (RT IDSS) and algorithm of k-nearest neighbors type. This work was supported by RFBR.

Keywords: Intelligent decision support systems, expert diagnostics systems, analogous reasoning, case-based reasoning.

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

Introduction

The problem of human reasoning simulating (so called "common sense" reasoning) in artificial intelligence (AI) systems and especially in IDSS is very actual nowadays [1,2]. That is why special attention is turned to case-based and analogous reasoning methods and models. The analogy and precedents (cases) can be used in various applications of AI and for solving various problems [3-7], e.g., for diagnostics and forecasting or for machine learning. AI experts model case-based reasoning by computers in order to develop more flexible models of search for solutions and learning.

In this paper, we consider method of case-based reasoning for a solution of problems of real-time diagnostics and forecasting in RT IDSS [5]. These systems 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 analogous reasoning in RT IDSS, it is necessary to take into account a number of the following requirements to systems of this kind [2]:

– The necessity of obtaining a solution under time constraints defined by real controlled process;