Вычислительные технологии, 2022, том 27, № 5, с. 89–100. © ФИЦ ИВТ, 2022 ISSN 1560-7534
Computational Technologies, 2022, vol. 27, no. 5, pp. 89–100. © FRC ICT, 2022 eISSN 2313-691X

DOI:10.25743/ICT.2022.27.5.009

On events processing in a formal logical approach to control of discrete event systems

Davydov A. V.*, Larionov A. A., Nagul N. V.
Matrosov Institute for System Dynamics and Control Theory SB RAS, 664033, Irkutsk, Russia
*Corresponding author: Davydov Artem V., e-mail: artem@icc.ru
Received June 08, 2022, accepted June 30, 2022

The paper provides an insight into the control of automata-based discrete event systems with the help of the calculus of positively constructed formulas. It is shown how the knowledge available in the system may be exploited during the inference of such formulas for events processing and supervisor implementation. The approach described may be applied at different levels of a robot or a robot group control system. As an illustration, the problem of mobile robots pushing a block to a target area is considered.

Keywords: discrete event systems, positively constructed formulas, automated theorem proving, supervisory control.

Citation: Davydov A.V., Larionov A.A., Nagul N.V. On events processing in a formal logical approach to control of discrete event systems. Computational Technologies. 2022; 27(5):89–100. DOI:10.25743/ICT.2022.27.5.009.

Introduction

In this paper we continue developing an original approach to control automata-based discrete event systems (DESs) in the framework of supervisory control theory (SCT). SCT is a powerful tool for restricting DES behaviour according to some constraints and mainly deals with DES presented in the form of finite-state automata. Since such automata are regarded as generators of formal languages, application of formal methods of language processing and analysis for solving problems of SCT looks natural. A detailed description of state-of-the-art SCT is presented e. g. in [1–3]. The developed approach suggests applying the automated theorem proving (ATP) technique to solve SCT problems stated in the calculus of positively constructed formulas (PCFs). A detailed discussion of characteristics and capabilities of the PCF calculus as a complete method for ATP can be found in [4–7]. One of the advantages of the PCF calculus relates to the prover, a computer program exploited for automation in ATP. Usage of provers for ATP is associated with the well-known difficulties: a) the proving program makes too many inference steps, most of which are redundant or irrelevant; b) the program has to store too much information in the database; c) inference rules and inference steps are not of the same size; d) inadequate focus, i. e. the program quickly stumbles on a false search path. Unlike many logical calculi that underlie the theoretical basis of modern provers, such as Vampire [8], nanoCoP [9], E [10], the features of the calculus of PCFs help to eliminate or significantly reduce the above difficulties. Its application at the upper level of a robot group control system for constructing plans of robot actions is discussed in [11].

A small overview of the current state of research on ATP in robotics is presented in [11], while a detailed review on planning in robotics, including the use of ATP, is presented in [12]. The work [13] lies at the intersection of ATP and machine learning, presenting a reinforcement learning toolkit for experiments on guiding ATP in the calculus of connections. In [14] for planning and control in swarm robotics, the PDDL language is used, which is based on the classical STRIPS-style ATP. As for the topic of this paper, an approach for testing the diagnosability of DES based on its logical representation is proposed in [15]. In contrast to the logical formalism of the PCF calculus, [15] uses the conjunctive normal form (CNF). Automata transitions are described as a set of clauses and when the well-known resolution method is applied to test whether failure events can be detected in a finite number of observable events.

Keeping in mind that CNF is a less expressive, compared to PCF, means to represent automata underlying DES, we leave the problem of diagnosability of DES for future research. For now, application of the PCF-calculus allowed solving such basic problems of SCT as controllability checking [16], supremal controllable sublanguage of a specification language construction [17] and a monolithic supervisor realization [18]. The PCF-based approach is used to create multi-level hierarchical control systems for mobile robots and robot groups [18]. The current paper continues that study and the study [19] where composite events are employed to embrace the knowledge of the environment.

Today, knowledge-based subsystems seem to be the most adequate components of control systems for the implementation of highly intelligent functions. Knowledge can be represented in some logical language, for example, the language of PCFs, and its processing using logical means allows obtaining some preferences on the set of valid controls in order to select one of them. ATP is the one of developing areas of the artificial intelligence, which is based on the methods of mathematical logic. The PCF-language as a predicate language allows one to formalize broader knowledge for subsequent processing by machine-oriented rules. Moreover, the question-answering procedure of the inference in the PCF-calculus is an intuitive way of reasoning and knowledge processing.

The paper is structured as follows. In section 1 we start with the statement of the problem of moving the block to the target area. In section 2 preliminaries on SCT are given. In section 3 the PCF-calculus is described. In section 4, the PCF-based approach to the supervisory control of DES is briefly described. Our main results, the usage of the external knowledge and supervisor realization in the control system, are presented in sections 5 and 6. In conclusion, some words on our future work are said.

1. The problem of moving a block by two robots

Let on a field, called *a scene*, there be three robots, two blocks, and the target area to which it is necessary to move the blocks (Fig. 1). Only a pair of robots can push a block, so at first, a robot should find a companion to form a pair and only then push the block to the target area. The problem under consideration may be described by a set of finite-state automata, with each of them captures one of the robot actions. In graphical representation of automata below, the incoming arrow denotes the initial state of an automaton while dotted lines denote controllable events. Marked states are denoted by double circles.

The automaton $\mathcal{G}_1$ in Fig. 2 describes the process of forming a group of two robots. Here the event rl is for "companion robot is lost", while the event rf is for "companion robot is found". At the state 0, the robot has not yet found a companion, so the event rl does not

change the state. Let $\Sigma_c^{\mathcal{G}_1} = \{rf\}$. We set rf to be controllable as the status "companion is found" is determined by external facts. For example, the notion of composite events may be exploited for environmental conditions evaluation and usage [18]. The automaton $\mathcal{G}_2$ in Fig. 3 describes the search of the direction for movement to the block. The event *dl* means "direction is lost", and *df* is for "direction is found". Again, additional information is used to decide if the right direction is found. The automaton $\mathcal{G}_3$ in Fig. 4 describes the possible directions of rotation of the robot when it tries to find the proper direction of moving. We suppose that the robot can rotate clockwise and counterclockwise (events *cw* and *ccw*), with both events controllable. Let the initial state be the north orientation. The actual information whether the direction is found or not comes from external sources, i. e. events are deducible.

The automaton $\mathcal{G}_4$ in Fig. 5 is an automaton that describes operational modes of robots. The robot can stand still, move, rotate and push. The modes restrict the robot's functions in different situations. For example, when the robot moves or pushes a block, it cannot rotate. Or, until the robot has found a companion, it should not push. This case, for example, is described by a combination of states of automata $\mathcal{G}_1$ and $\mathcal{G}_4$. The automaton $\mathcal{G}_5$ (Fig. 6) serves for checking the achievement of the goal assigned, i. e. if the block is reached the target area. The events of $\mathcal{G}_5$ are *gc* — the goal check, *ga* — the goal is achieved, *gna* — the goal is not achieved. The parallel composition of automata $\mathcal{G}_1$–$\mathcal{G}_5$ represents the current state of the robot.

Figure 7 depicts a specification automaton $\mathcal{H}_1$. This specification requires that at first the robot looks for a partner. Then it finds the right direction, then moves, pushes or rotates, depending on what action is required for the current position of the block. After that the goal achievement is checked. If the goal is not achieved the actions continue. If the goal has been achieved the task assigned has been solved. The specification automaton $\mathcal{H}_2$ (Fig. 8) limits the search for directions. If the direction has not yet been found, then the robot can only rotate clockwise or move.

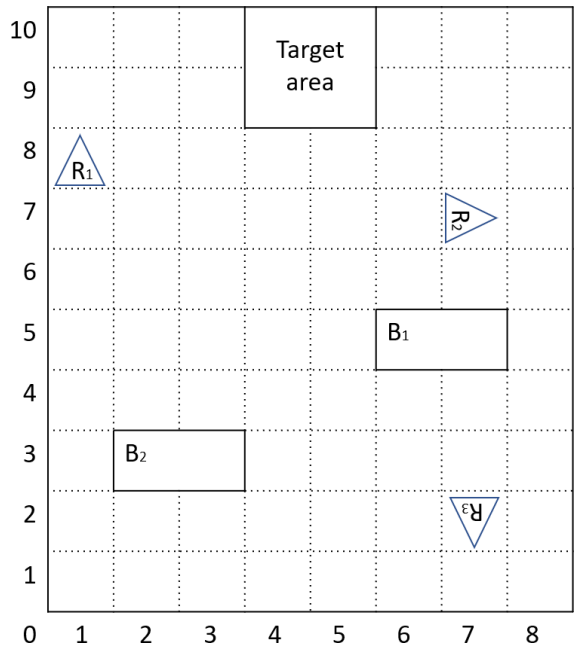

Fig. 1. The initial scene

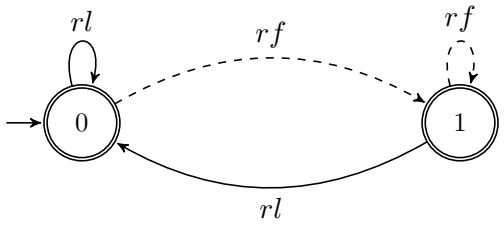

Fig. 2. $\mathcal{G}_1$. Group formation for 2 robots

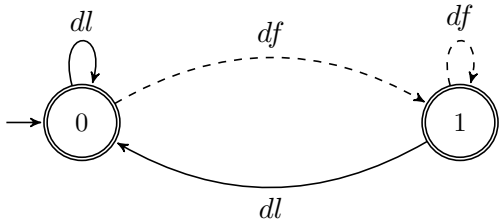

Fig. 3. $\mathcal{G}_2$. Search for direction

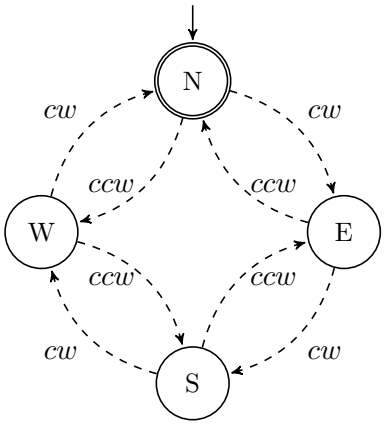

Fig. 4. $\mathcal{G}_3$. Rotation

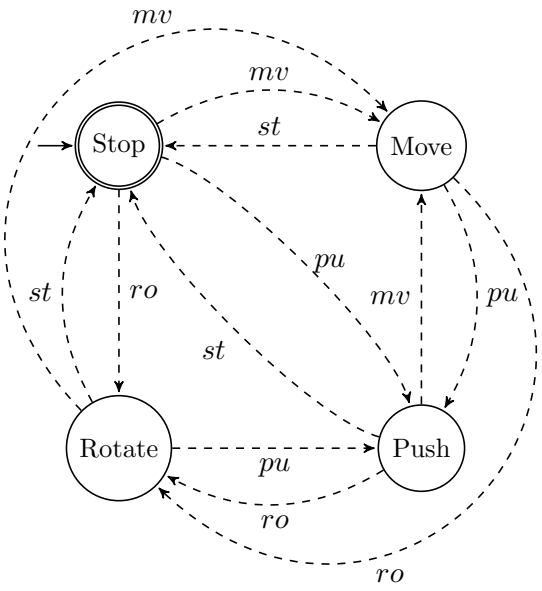

Fig. 5. $\mathcal{G}_4$. Modes switching

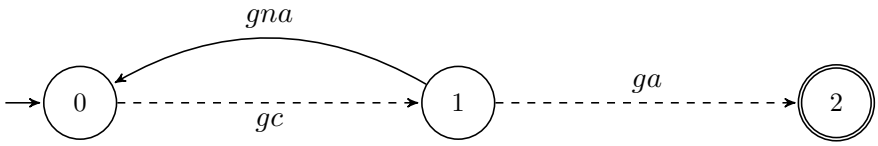

Fig. 6. $\mathcal{G}_5$. Goal checking

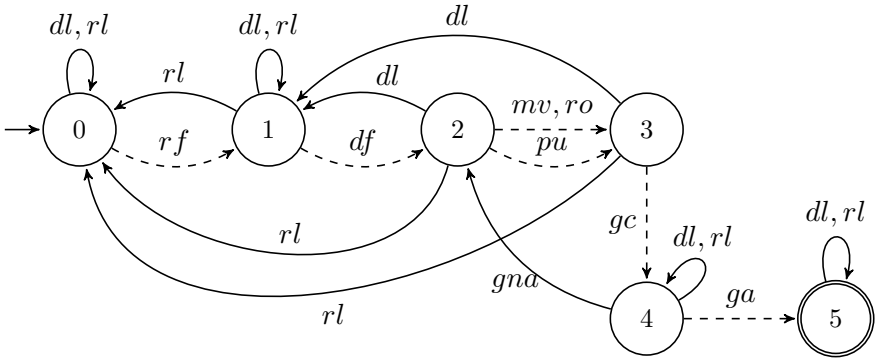

Fig. 7. Specification $\mathcal{H}_1$. The main strategy

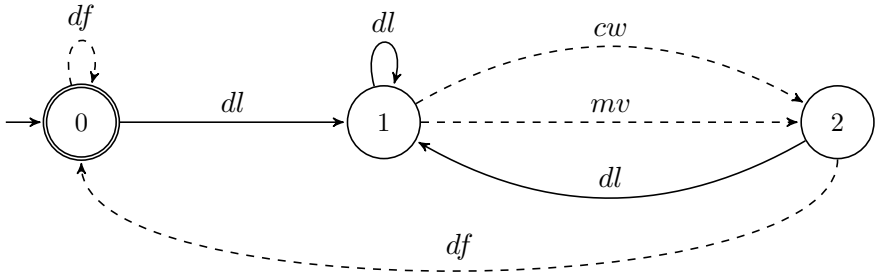

Fig. 8. Specification $\mathcal{H}_2$. Search for proper direction

2. Supervisory control of DES as generators of formal languages

Restrictions on robot's behaviour in the process of moving a block to the target area, described above, can be ensured in the framework of supervisory control theory (SCT) for discrete event systems (DES). In this case DES is represented by the set of finite-state automata of the form $\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$ where Q is the set of states q; Σ is the set of events; $\delta : \Sigma \times Q \to Q$ is the transition function; $q_0 \in Q$ is the initial state; $Q_m \subset Q$ is the set of marked states. Events label the transitions of automaton, and such a DES behaves as a generator of a formal language $L(\mathcal{G})$, which words are the sequences of the events occurred in the automaton. Note that marked states, in this case, play no termination role and serve to denote some distinguished sequences of events.

SCT supposes some events of $\mathcal{G}$ to be controllable, i. e., they may be prohibited from occurring by the mean of control called *supervisor*. Let Σ_c be a controllable event set, $\Sigma_{uc} = \Sigma \setminus \Sigma_c$, $\Sigma_c \cap \Sigma_{uc} = \emptyset$. The supervisor observes events generated by the plant and disables undesired controllable events, thus realizing a mapping $\gamma : L(\mathcal{G}) \to 2^{\Sigma_c}$, in such a way that the supervised DES generates some regular language K called *specification*. For this, formal language K must be controllable (with respect to $L(\mathcal{G})$ and Σ_{uc}). If K is a specification language, let $\mathcal{H}$ be an automaton that marks the language K. Such $\mathcal{H}$ is called a recognizer of K, and if K is controllable then $\mathcal{H}$ may be chosen as a supervisor $\mathcal{S}$ for $\mathcal{G}$, and $L(\mathcal{S}/\mathcal{G}) = L(\mathcal{H}||\mathcal{G})$ [1]. Here $L(\mathcal{S}/\mathcal{G})$ denotes a language generated by the closed-looped behaviour of the plant $\mathcal{G}$ and the supervisor $\mathcal{S}$. In the parallel composition $\mathcal{S}||\mathcal{G}$ of the automata events in the intersection of the respective event sets occur simultaneously in both automata, while events that are not shared by the automata occur independently. Thus the specification imposed is assured. In [16] the method for controllability checking using the PCF calculus is presented.

3. The calculus of PCFs

Consider a language of first-order logic that consists of first-order formulas (FOFs) built out of atomic formulas with &, $\vee$, $\neg$, $\to$, $\leftrightarrow$ operators, $\forall$ and $\exists$ quantifier symbols and constants *true* and $false$. Let $X = \{x_1, \ldots, x_k\}$ be a set of variables, $A = \{A_1, \ldots, A_m\}$ be a set of atomic formulas called *conjunct*, and $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ be a set of FOFs. The following formulas $\forall x_1 \ldots \forall x_k(A_1 \& \ldots \& A_m) \to (\mathcal{F}_1 \vee \cdots \vee \mathcal{F}_n)$ and $\exists x_1 \ldots \exists x_k(A_1 \& \ldots \& A_m) \& (\mathcal{F}_1 \& \ldots \& \mathcal{F}_n)$ are denoted as $\forall x_1, \ldots, x_k A_1, \ldots, A_m\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ and $\exists x_1, \ldots, x_k$ $A_1, \ldots, A_m\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$. They can be abbreviated as $\forall_X A\ \Phi$ and $\exists_X A\ \Phi$ respectively.

Definition 1. Let X be a set of variables, and A be a conjunct, both can be empty.

1. $\exists_X A$ is an $\exists$-PCF.
2. $\forall_X A$ is a $\forall$-PCF.
3. If $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is a set of $\forall$-PCFs, then $\exists_X A\ \Phi$ is an $\exists$-PCF.
4. If $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is a set of $\exists$-PCFs, then $\forall_X A\ \Phi$ is a $\forall$-PCF.
5. Any $\exists$-PCF or $\forall$-PCF is a PCF.
6. There are only PCFs of a form $\exists$-PCF and $\forall$-PCF.

For the sake of readability, we represent PCFs as trees whose nodes are type quantifiers, and we use corresponding notions: *node, root, leaf, branch*. Given PCFs $\mathcal{P} = \forall\{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ and $\mathcal{F}_i = \exists_{X_i} B_i\{\mathcal{Q}_{i1}, \ldots, \mathcal{Q}_{im}\}$, $i = \overline{1, n}$, then $\mathcal{F}_i$ is called *base subformula* of $\mathcal{P}$, B_i is called *base of facts* or just *base*, $\mathcal{Q}_{ij}$ are called *question subformulas*, and roots of question

subformulas are called *questions* to the base B_i, $i = \overline{1,n}$. A question of a form $\forall_X A$ (without any children) is called *goal question*.

Definition 2. [Answer] Let $\exists_X A\ \Psi$ be a base subformula of a PCF. A question of the subformula $\mathcal{Q} = \forall_Y B\ \Phi$, $Q \in \Psi$ has an answer θ if and only if θ is a substitution $Y \to H^\infty \cup X$ and $B\theta \subseteq A$, where H^∞ is Herbrand universe based on constant and function symbols that occur in the corresponding base subformula.

Definition 3. Let $\mathcal{P}_1 = \exists_X A\ \Psi$ and $\mathcal{P}_2 = \exists_Y B\ \Phi$, then $merge(\mathcal{P}_1, \mathcal{P}_2) = \exists_{X \cup Y} A \cup B\ \Psi \cup \Phi$.

Definition 4. Consider some base subformula $\mathcal{B} = \exists_X A\ \Psi$. A question subformula $\mathcal{Q} \in \Psi$ has the form $\forall_Y D\ \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, where $\mathcal{P}_i = \exists_{Z_i} C_i\ \Gamma_i, i = \overline{1,n}$, then $split(B, Q) = \{merge(B, \mathcal{P}'_1), \ldots, merge(B, \mathcal{P}'_n)\}$, where $'$ is a variable renaming operator. We say that $\mathcal{B}$ is split by $\mathcal{Q}$, and $split(\mathcal{B}, \mathcal{Q})$ is the result of the split of $\mathcal{B}$. Obviously, $split(\mathcal{B}, \forall_Y D) = \varnothing$.

Definition 5. [The inference rule ω] Consider some PCF $\mathcal{F} = \forall\ \Phi$. If there exists a base subformula $\mathcal{B} = \exists_X A\ \Psi$, $\mathcal{B} \in \Phi$ and there exists a question subformula $\mathcal{Q} \in \Psi$, and the question of $\mathcal{Q}$ has an answer θ to $\mathcal{B}$, then $\omega(\mathcal{F}) = \forall\ \Phi \setminus \{\mathcal{B}\} \cup split(\mathcal{B}, \mathcal{Q}\theta)$.

In the process of reasoning, one often proves a statement $\mathcal{F}$ by refuting its negation. In the PCF-calculus we proceed similarly. If the set Φ becomes empty after applying the ω rule, and the PCF becomes just $\forall$, then the negation of the original statement is unsatisfiable; therefore, the statement itself is true. Any finite sequence of PCFs $\mathcal{F}$, $\omega\mathcal{F}$, $\omega^2\mathcal{F}$, $\ldots$, $\omega^n\mathcal{F}$, where $\omega^s\mathcal{F} = \omega(\omega^{s-1}\mathcal{F})$, $\omega^1 = \omega$, $\omega^n\mathcal{F} = \forall$, is called *an inference* of $\mathcal{F}$ in the PCF calculus (with the axiom $\forall$). A search strategy used by default does not use repeated application of ω to a question with the same θ (question-answering method of automated inference search).

4. PCF representation of DES

Figure 9 shows a general form of a PCF representing a DES $\mathcal{G}$. It consists of the single base $B = \{L(\varepsilon, S_0),\ L_m(\varepsilon, S_0),\ \delta(S_1^i, \sigma^i, S_2^i),\ \delta_m(S_1^i, \sigma^i, S_2^i),\ \Sigma_c(\sigma^j),\ \Sigma_{uc}(\sigma^j)\}$, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$, n is the number of transitions, k is the number of events, and two questions where the following predicates are exploited. The predicate $L(s, S)$ denotes "s is a sequence of events that brings the system into the state S" and the predicate $L_m(s, S)$ denotes "s is a marked sequence of events that brings the system into the state". The first arguments of these atoms accumulate the strings of the languages $L(\mathcal{G})$ and $L_m(\mathcal{G})$. S_0 corresponds to the initial state of $\mathcal{G}$. A predicate of the form $\delta(S_1, \sigma, S_2)$ is interpreted as a transition from a state S_1 to a state S_2 due to event σ occurring. If the target state of a transition is marked, then atoms with an index m are used, i. e., $\delta_m(S_1, \sigma, S_2)$. Controlled and uncontrolled events are represented in the base by separate atoms using the predicates $\Sigma_c(\_)$ and $\Sigma_{uc}(\_)$, respectively. The function symbol "$\cdot$" denotes strings concatenation, and the symbol "ε" corresponds to the empty string. Applying the inference rules to this PCF, the words of the languages $L(\mathcal{G})$ and $L_m(\mathcal{G})$ are constructed as the first arguments of the atoms $L(s, S)$, $L_m(s, S)$ in the base.

In the PCF representation, the joint work of the system and the supervisor is carried out using the PCF $\mathcal{F}_{SC}$ in Fig. 10. Here bases B and B_S are the sets of atoms corresponding to

$$\exists\ B \begin{cases} \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta(s, \sigma', s') \text{ —— } \exists L(\sigma \cdot \sigma', s') \\ \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta_m(s, \sigma', s') \text{ — } \exists L_m(\sigma \cdot \sigma', s') \end{cases}$$

Fig. 9. A general form of PCF representation of DES

$$\exists\ B, B_S \text{ —— } \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta(s, \sigma', s'), \delta^S(s, \sigma', s') \text{ ——— } \exists L(\sigma \cdot \sigma', s')$$

Fig. 10. The general form of a PCF realizing supervisory control

the transitions of the plant and the supervisor, correspondingly. The only question of $\mathcal{F}_{SC}$ may be interpreted as follows. If the system is at the state s and an event σ occurs then according to the δ^S the system is switched to the specified state s', and σ is added to the current chain of events stored as the first argument of the predicate $L(\cdot, \cdot)$. That is, for any transition corresponding to the language $L(\mathcal{G})$ (an atom $\delta(\cdot, \cdot, \cdot)$), we trace the corresponding event in the automaton of the supervisor (an atom $\delta^S(\cdot, \cdot, \cdot)$). The rule works only on those strings that are allowed by the supervisor, i. e., atoms $\delta^S(\cdot, \cdot, \cdot)$ limit the answers that could be generated having atoms $\delta(\cdot, \cdot, \cdot)$ only. Note that problems involving marking are not considered in this paper.

5. Knowledge processing by PCF

The basic PCF $\mathcal{F}_{SC}$ in Fig. 10 should be complicated if the system under consideration has a modular structure. In this case, modular supervisory control paradigm may be applied. For example, all generator modules may be composed to a single generator, and then one supervisor is designed for each specification. The conjunction of the supervisors, captured by the parallel composition $\mathcal{S}^{mod} = \mathcal{S}_1||\mathcal{S}_2||\ldots||\mathcal{S}_m$, guarantees that any event of the global plant is enabled only if all modular supervisors that have the corresponding event in their event set enable it. The PCF $\mathcal{F}_C$ in Fig. 11 employs the local modular control [20] ensuring the satisfaction of constraints imposed by the specifications in robots and blocks example. A local supervisor is created for each control specification but only those generator modules are taken into account that are affected by the particular specification i. e. have at least one common event with this specification.

Moreover, the PCF $\mathcal{F}_C$ is specially constructed for processing the knowledge about the environment. Available information is processed with the help of the predicate T that is

$$\exists B \left\{ \begin{array}{ll}
\forall \sigma\ E(\sigma) & \text{——————————} \exists\ T^{\#}(\sigma) \\
\forall \sigma, q_i, \sigma', q_i'\ L^{G_i}(\sigma, q_i), \delta^{G_i}(q_i, \sigma', q_i') & \text{————} \exists\ N_{G_\parallel}(\sigma, \sigma', q_1' \cdot q_2' \cdot q_3' \cdot q_4' \cdot q_5') \\
\forall \sigma, q, \sigma', q', q_G'\ L^{H_1}(\sigma, q), N_{G_\parallel}(\sigma, \sigma', q_G'), \delta^{H_1}(q, \sigma', q') & \text{————} \exists\ N_{G_\parallel \times H_1}(\sigma, \sigma', q_G' \cdot q') \\
\forall \sigma, q, \sigma', q', q_G'\ L^{H_2}(\sigma, q), N_{G_\parallel}(\sigma, \sigma', q_G'), \delta^{H_2}(q, \sigma', q') & \text{————} \exists\ N_{G_\parallel \times H_2}(\sigma, \sigma', q_G' \cdot q') \\
\forall \sigma, \sigma', q, q_{H_1}', q_{H_2}'\ L_S(\sigma, q), N^*_{G_\parallel \times H_1}(\sigma, \sigma', q_{H_1}'), N^*_{G_\parallel \times H_2}(\sigma, \sigma', q_{H_2}'), E_T^*(\sigma) & \text{——} \exists L^S(\sigma \cdot \sigma', q_{H_1}' \cdot q_{H_2}')
\end{array} \right.$$

$$B = \{\{E(e)\}, I_i^G(q_0^{G_i}), I_i^H(q_0^{H_i}), \{\delta_i^G(\cdot, \cdot, \cdot)\}, \{\delta_j^H(\cdot, \cdot, \cdot)\}, L_i^G(\varepsilon, q_0^{G_i}), L_i^H(\varepsilon, q_0^{H_i}), L^S(\varepsilon, q_0)\},$$
$$i = \overline{1, 5}, j = 1, 2$$

Fig. 11. The PCF $\mathcal{F}_C$ that realizes supervisory control using the additional knowledge

also realized as a PCF. The following atoms form the base of $\mathcal{F}_C$. $E(\sigma)$ is an atom that determines all events of the DES. When implemented, each clock cycle it is replaced by the real event occurred in the system. Atoms $I_i^G(q_0^{G_i})$ and $I_i^H(q_0^{H_i})$ determine the initial states of the corresponding automata. Atoms $\{\delta_i^G(\cdot,\cdot,\cdot)\}$, $\{\delta_j^H(\cdot,\cdot,\cdot)\}$ correspond to the transitions of the considered automata. $L_i^G(\varepsilon, q_0^{G_i})$, $L_i^H(\varepsilon, q_0^{H_i})$ are atoms that will contain strings generated by the corresponding automata. During the inference, the strings of the corresponding languages are accumulated in the first arguments of these atoms. The second arguments correspond to states in which the last events of the strings of the first arguments occurred.

The first question of $\mathcal{F}_C$ serves for external processing of an event occurred in the system. It enters the computable construction $T^\#$ as an argument σ. $T^\#$ can be of any reasonable complexity, e. g. some logical formula or any black box performing computation or knowledge processing. In robots and blocks example, we use $T^\#$ in a form of a simple PCF for additional reasoning based on some knowledge of the environment. The second question of $\mathcal{F}_C$ generates the next symbol of the word of the language of the parallel composition of automata $\mathcal{G}_1$, …, $\mathcal{G}_5$, and adds an auxiliary atom $N_{\mathcal{G}_\parallel}$ to the base, which is necessary for further search for independent intersections of languages. The third and fourth questions add similar atoms for automata $\mathcal{H}_{\mathcal{G}_\parallel \times \mathcal{H}_1}$ and $\mathcal{H}_{\mathcal{G}_\parallel \times \mathcal{H}_2}$. The fifth question restricts the language event generated by the second question to the product of the automata $\mathcal{H}_1$ and $\mathcal{H}_2$. In this case, we use the $*$ operator in order not to use these atoms in the future. The $E_T^*(\sigma)$ atom in this question is a result of processing the σ event with the $T^\#$ construction.

The general view of the subformula T used in the example with robots moving a block is depicted in Fig. 12. In this PCF, σ is a parameter coming from the PCF $\mathcal{F}_C$ (see Fig. 11). The first question of this formula handles the σ parameter, and the answer to this question may be found only when the $Trigger$ condition is satisfied. In this case, based on the value of σ, some predefined procedures are executed, indicated by the predicate $Processing^\#(\sigma)$. The second question is a stub for all other events. The third question completes the inference and returns the processed E_T atom to the top-level formula $\mathcal{F}_C$. As an illustration, Fig. 13 depicts the subformula T handling the event dl. The first question is triggered if the parameter σ is dl. On the right side of this question, the argument of the atom E_T is a computable function that makes a request to the server to obtain the distances from the next hypothetical position of the robot (if it moves left, straight, or right) to the current target. The min operator

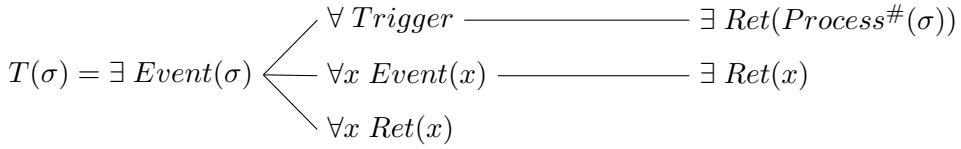

Fig. 12. The PCF computing predicate T

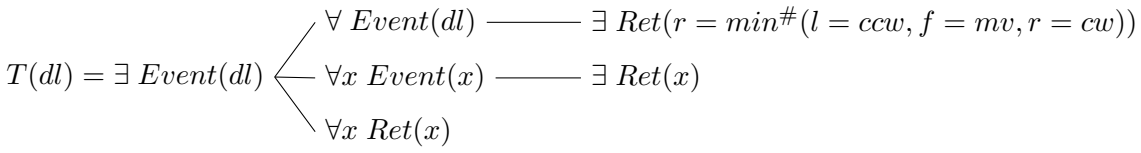

Fig. 13. The PCF computing predicate T for the event dl

selects the shortest one and substitutes the corresponding event name into the argument of the E_T atom. The resulting atom will be added to the base of the top-level formula $\mathcal{F}_C$, and the robot thus chooses the direction: left, forward or right. When the event dl occurs, as a result of the sub-inference of the formula T two atoms will be added to the base of $\mathcal{F}_C$ simultaneously: $E_T(r)$ as the result of the answering the first question and $E_T(dl)$ as the result of the answering the second question. Then, $E_T(dl)$ will be deleted by the last question of the top-level formula $\mathcal{F}_C$. The atom $E_T(r)$ remains in the base to be used in the next step of the inference.

6. PCF-based supervisory control realization in control system

Figure 14 shows a scheme of supervisory control realization as a part of a hierarchical control system for mobile robots group. The control system is being developed on the basis of a robotic stand consisting of Lego Mindstorms EV3 robots moving on the special field and a set of cameras reading the position of robots and other objects on the field. The software is distributed over a wireless network between the robots and the server. In Fig. 14, the real working environment of robots is represented by the Robots environment block. Events occurring in the working environment are recognized, labelled with predefined symbols, and fed to the symbolic data processing subsystem. In this subsystem, a DES is implemented, which is a formalization of the problem solved by robots, in the form of automata $\mathcal{G}_1, \ldots, \mathcal{G}_m$ for system and a set of automata $\mathcal{H}_1, \ldots, \mathcal{H}_l$ describing constraints on system behaviour. Then, the DES is represented in the form of a PCF, the inference of which is carried out by a specially developed software system for searching for the logical inference of PCFs, a prover. The automata underlying the DES are represented by logical atoms in the PCF base. The logical rules, represented as questions of the PCF, generate the next state of the system based on events, taking into account imposed constraints. Thus, the supervisors $\mathcal{J}_1, \ldots, \mathcal{J}_l$ responsible for achieving a predetermined goal are realized. The advantage of this approach is the possibility of additional processing and control of events based on environmental data in real-time during the inference. Special logical rules are responsible for this, presented in the PCF in the form of event processing questions. By answering them, it is possible to launch sub-inferences, in which events coming from the working environment serve as

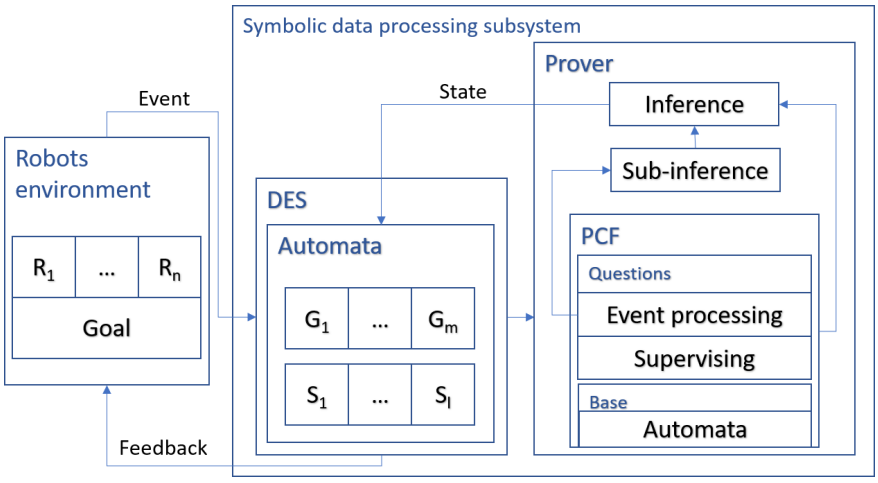

Fig. 14. Supervisory control realization as a part of control system

parameters, used in calculations or in other decisions. The work of the prover is organized in such a way that all the questions of the PCF representing the DES are bypassed in a loop while an event coming from the working environment is processed.

Conclusion

This paper continues the work on developing a new way of formalizing and solving control problems for the important class of dynamic systems known as DES. The PCF calculus provides the powerful tools for dealing with sophisticated control problems and above it was shown how the knowledge available in the modelled system may be exploited for events processing and supervisor implementation. The developed approach is currently tested as a part of a control system for the group of Lego Mindstorms EV3 robots on the robotic stand in the laboratory of Information and Control Systems of ISDCT SB RAS to be further employed in real robotic systems. PCF representation of DES allows using the modularity in an efficient way, and this issue is the subject of our further research. Another related problem is how to efficiently implement the supervisory control with partial observations of events. This problem will be also studied with the help of the PCF-calculus and ATP, as well as the problem of diagnosibility of DES.

Acknowledgments. The research was partly supported by the RFBR, project No. 20-07-00397 sections 5, 6), and by the Ministry of Science and Higher Education of the Russian Federation, project No. 121032400051-9.

References

[1] **Cassandras C.G., Lafortune S.** Introduction to discrete event systems. Boston: Springer; 2008: 772.

[2] **Seatzu C., Silva M., van Schuppen J.H. (Eds)** Control of discrete-event systems. London: Springer; 2013: 480. DOI:10.1007/978-1-4471-4276-8.

[3] **Wonham W.M., Cai K.** Supervisory control of discrete-event systems. Cham: Springer; 2019: 487.

[4] **Vassilyev S.N.** Machine synthesis of mathematical theorems. The Journal of Logic Programming. 1990; (9):235–266. DOI:10.1016/0743-1066(90)90042-4.

[5] **Zherlov A.K., Vassilyev S.N., Fedosov E.A., Fedunov B.E.** Intellektnoe upravlenie dinamicheskimi sistemami [Intelligent control of dynamic systems]. Moscow: Fizmatlit; 2000: 352. (In Russ.)

[6] **Davydov A., Larionov A., Cherkashin E.** On the calculus of positively constructed formulas for automated theorem proving. Automatic Control and Computer Sciences. 2011; (45):402–407. DOI:10.3103/s0146411611070054.

[7] **Davydov A., Larionov A., Cherkashin E.** The calculus of positively constructed formulas, its features, strategies and implementation. International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija; 2013: 1023–1028.

[8] **Kovács L., Voronkov A.** First-order theorem proving and vampire. Computer Aided Verification. Berlin, Heidelberg: Springer; 2013: 1–35.

[9] **Otten J.** Nanocop: a non-clausal connection prover. Proceedings of the 8th International Joint Conference on Automated Reasoning. Berlin, Heidelberg: Springer-Verlag; 2016: 302–312. DOI:10.1007/978-3-319-40229-1_21.

[10] **Schulz S., Cruanes S., Vukmirović P.** Faster, higher, stronger: E 2.3. Proc. of the 27th CADE. Natal, Brasil: Springer; 2019: 495–507.

[11] **Davydov A., Larionov A.** Action planning for robots using the first order logic calculus of positively constructed formulas. 2020 7th International Conference on Control, Decision and Information Technologies (CoDIT). 2020; (1):727–732. DOI:10.1109/CoDIT49905.2020.9263814.

[12] **Karpas E., Magazzeni D.** Automated planning for robotics. Annual Review of Control, Robotics, and Autonomous Systems. 2020; (3):417–439.

[13] **Zombori Z., Urban J., Brown C.E.** Prolog technology reinforcement learning prover. International Joint Conference on Automated Reasoning. Springer; 2020: 489–507.

[14] **Schader M., Luke S.** Planner-guided robot swarms. International Conference on Practical Applications of Agents and Multi-Agent Systems. Springer; 2020: 224–237.

[15] **Geng X., Ouyang D., Han C.** Verifying diagnosability of discrete event system with logical formula. Chinese Journal of Electronics. 2020; (29):304–311. DOI:10.1049/cje.2020.01.008.

[16] **Davydov A., Larionov A., Nagul N.** On checking controllability of specification languages for des. 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO). 2020: 1151–1156. DOI:10.23919/MIPRO48935.2020.9245154.

[17] **Davydov A., Larionov A., Nagul N.** The construction of controllable sublanguage of specification for DES via PCFs based inference. CEUR Workshop Proceedings. 2020; (2638):68–78.

[18] **Davydov A., Larionov A., Nagul N.** Application of the PCF calculus for solving the problem of nonblocking supervisory control of discrete event systems. Journal of Physics: Conference Series. 2021; (1864):012048. DOI:10.1088/1742-6596/1864/1/012048.

[19] **Davydov A., Larionov A., Nagul N.** Positively constructed formulas-based approach to mobile robot control using DES. 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). 2021: 1064–1069. DOI:10.23919/MIPRO52101.2021.9597024.

[20] **de Queiroz M.H., Cury J.E.R.** Modular supervisory control of large scale discrete event systems. The Springer International Series in Engineering and Computer Science. 2000; (569):103–110. DOI:10.1007/978-1-4615-4493-7_10.

Вычислительные технологии, 2022, том 27, № 5, с. 89–100. © ФИЦ ИВТ, 2022 ISSN 1560-7534
Computational Technologies, 2022, vol. 27, no. 5, pp. 89–100. © FRC ICT, 2022 eISSN 2313-691X

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

DOI:10.25743/ICT.2022.27.5.009

Об обработке событий в формальном логическом подходе к управлению дискретно-событийными системами

Давыдов А.В.*, Ларионов А.А., Нагул Н.В.

Институт динамики систем и теории управления им. В.М. Матросова, 664033, Иркутск, Россия
*Контактный автор: Давыдов Артем Васильевич, e-mail: artem@icc.ru

Поступила 08 июня 2022 г., принята в печать 30 июня 2022 г.

Аннотация

Дается представление об управлении дискретно-событийными системами, представленными в форме конечных автоматов, с помощью исчисления позитивно-образованных формул. Показано, как знания, имеющиеся в системе, в ходе построения вывода формулы могут быть использованы для обработки событий и реализации супервизора. Описанный подход может применяться на разных уровнях систем управления роботами. В качестве иллюстрации рассматривается задача о перемещении мобильными роботами блока в целевую область.

Ключевые слова: дискретно-событийные системы, позитивно-образованные формулы, автоматическое доказательство теорем, супервизорное управление.

Цитирование: Давыдов А.В., Ларионов А.А., Нагул Н.В. Об обработке событий в формальном логическом подходе к управлению дискретно-событийными системами. Вычислительные технологии. 2022; 27(5):89–100. DOI:10.25743/ICT.2022.27.5.009. (на английском)

Благодарности. Работа частично поддержана РФФИ, проект № 20-07-00397 (разделы 5, 6), и Министерством науки и высшего образования РФ, проект № 121032400051-9.