Logic inference based construction of a supervisor for a discrete event system

Artem Davydov, Aleksandr Larionov and Nadezhda Nagul

Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the Russian Academy of Sciences, 134 Lermontov str., 664033 Irkutsk, Russia

E-mail: artem@icc.ru, bootfrost@zoho.com, sapling@icc.ru

Abstract. The paper provides a general view on the original logical inference based approach to dealing with discrete event systems as subject to supervisory control theory. The approach proposes a representation of automata-based discrete event system as a positively constructed formula and implementation of the calculus of positively constructed formulas. The stages of a supervisor construction are illustrated with a simplified model of an autonomous underwater vehicle operational modes switching. The supremal controllable sublanguage of the specification and the supervisor are constructed.

1. Introduction

The aim of this paper is to give a general view of the logical inference based approach to dealing with discrete event systems (DES) as subject to supervisory control [1]. The approach proposes a representation of automata-based DES as a positively constructed formula (PCF), and implementation the PCF calculus for proving it. During the inference, languages generated and marked by DES as a generator of formal languages are obtained. Thus, automata are simulated. The same technique is used to construct parallel composition and product of automata. These structures are then applied for supervisory control theory (SCT) basic problems solving. The approach was developed in authors' several works [2, 3, 4] that concerned DES formalization, controllability, observability and coobservability of specifications on DES functioning checking, and supervisor implementation. To provide the main ideas of the method and algorithms involved, many technical details were omitted in those papers due to the space limitations. This paper fills in the blanks which might remain.

The PCF calculus was presented in [5, 6] and further developed in [7]. Due to its features, the PCF calculus allows one to combine the automatic search for logic inferences with special domain-based heuristics which are customizable for a task being solved. Moreover, modifiability of semantics, which may be constructive, non-monotonic or temporal, and an ability to construct intuitionistic inferences of some non-Horn formulas, are helpful at solving problems of dynamic systems control. Problems of automated theorem proving software in the PCF calculus design and implementation are briefly discussed in [8].

To illustrate the stages of a supervisor construction procedure, a simplified model of an autonomous underwater vehicle (AUV) operational modes switching is chosen. As shown in [9]

and [10], both the PCF calculus and DES may be employed at the various levels of a hierarchical control system for AUV and AUVs groups. Indeed, SCT is actively utilized in robotics nowadays. Recent publications in this area concern single robot control [11], [12], robot groups control [13], [14], [15], robots formation control [16], and swarm robotics [17], [18]. On the other hand, the inference search machines, implemented at a robot control system, are applied to represent and process the knowledge obtained during the robot functioning. Constructive semantics of the PCF calculus allows one to extract the knowledge (for example, AUV action plans) from inferences, while non-monotonicity and time tracking help to construct plans in dynamically changing subject areas. Interactive properties of the PCF calculus, its application for dynamic systems control and action planning are described in [6] with the examples of elevator group control, mobile robot action planning and telescope guidance. Application of PCFs based ATP and logical inference, embedded at the upper level of the control system, to DES at the middle level allows efficiently solving control problems emerging during the performance of the robot's mission.

The paper is organized as follows. The basic notions of DES and SCT are provided in section 2. The third section contains a brief description of the PCF calculus. In section 4, a way of automata-based DES analysis using PCFs is described. In section 5, a supervised DES is constructed. In conclusion, we discuss some crucial features of the PCF calculus.

2. Supervisory control of DES

Let a discrete event system is specified in the form of a finite-state automata $\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$ as a generator of a formal language [19]. Here Q is the set of states q; Σ is the set of events; δ: $\Sigma \times Q \to Q$ is the transition function; $q_0 \in Q$ is the initial state; $Q_m \subset Q$ is the set of marked states. $\mathcal{G}$ is also called a *plant* in the automatic control theory. Let $\Sigma = \Sigma_c \cup \Sigma_{uc}$, $\Sigma_c \cap \Sigma_{uc} = \emptyset$ where Σ_c is a controllable event set. $\Sigma_{uc} = \Sigma \setminus \Sigma_c$. Let Σ^* denote a Kleene closure, ε is an empty string. δ is easily extended on strings from Σ^*. Language generated by $\mathcal{G}$ is $L(\mathcal{G}) = \{w : w \in \Sigma^* \text{ and } \delta(w, q_0) \text{ is defined}\}$, while language marked by $\mathcal{G}$ is $L_m(\mathcal{G}) = \{w : w \in L(\mathcal{G}) \text{ and } \delta(w, q_0) \in Q_m\}$. For any $L \subset \Sigma^*$ a *closure* of L is the set of all strings that are prefixes of words of L, i.e., $\overline{L} = \{s | s \in \Sigma^* \text{ and } \exists t \in \Sigma^* : s \cdot t \in L\}$. Symbol $\cdot$ denotes string concatenation and is often omitted. K is called *prefix-closed* if $\overline{K} = K$.

The Ramadge–Wonham supervisory control framework assumes the existence of a means of control $\mathcal{G}$ represented by a *supervisor* [19]. The supervisor switches control patterns in such a way that the supervised discrete event system achieves a control objective described by some regular language K. A language generated by the closed-looped behavior of the plant and the supervisor is denoted as $L(\mathcal{J}/\mathcal{G})$. Let $L_m(\mathcal{J}/\mathcal{G})$ denotes the language marked by the supervisor: $L_m(\mathcal{J}/\mathcal{G}) = L(\mathcal{J}/\mathcal{G}) \cap L_m(\mathcal{G})$. The main goal of supervisory control is to construct such supervisor that $L(\mathcal{J}/\mathcal{G}) = \overline{K}$ and $L_m(\mathcal{J}/\mathcal{G}) = K$. The definition of controllability characterizes the languages which may be achieved by the closed-loop structure of the plant and the supervisor.

Definition 1 [19] K is controllable (with respect to $L(\mathcal{G})$ and Σ_{uc}) if $\overline{K}\Sigma_{uc} \cap L(\mathcal{G}) \subseteq \overline{K}$.

To verify controllability condition, a product of automata for the system and the specification is built to check if the same uncontrollable transitions present in both the specification and the plant. If the specification under consideration happen to be not controllable, a controllable part of it may be used for designing a supervisor. A set $\mathfrak{C}_{in}(K) = \{L \subseteq K : \overline{L}\Sigma_{uc} \cap L(\mathcal{G}) \subseteq \overline{L}\}$ is a set of all controllable sublanguages of a given language K [20]. It is well known that since the set of controllable sublanguages of a given regular language L is closed under the union, the supremal controllable sublanguage of L exists, and it is also regular. According [20], we denote this language $K^{\uparrow C}$. Note that in the worst case $K^{\uparrow C} = \emptyset$, while $K^{\uparrow C} = K$ if K is controllable.

As a rule, a supervisor is designed as another automaton, and control is implemented by constructing the parallel composition of the automata of the plant and the supervisor. Though the parallel composition of automata can be built using the PCF representation of the automata

involved, the supervised DES will be designed in the other way in section 5.

Example 1. Consider a simplified DES model, depicted in figure 1, of autonomous underwater vehicle (i.e., AUV) main operational modes switching while implementing a mission in some underwater area. Here $\Sigma = \{m, l, g, r, s, c\}$ with m having the meaning 'the mission starts', l - 'switch to the loiter mode', g - 'activate the gathering mode', r - 'activate the reconfiguration mode', s - 'surface', c - 'activate the communication mode'. States of the set $Q = \{S, M, L, R, G\}$ correspond to the functioning modes of AUV. Here M ('mission implementation') is the main mode. G corresponds to the gathering mode which is on when the mission is completed or a corresponding command is received. L is the standby mode: AUV is loitering and waiting for the next orders. R is the reconfiguration mode. S is the mode which corresponds to the start of the mission or surfacing in case of emergency or a command. C is communication mode which supposes a wide-range communication equipment activation for finding other AUVs, a mother ship or command receiving. Let $\Sigma_{uc} = \{c, s, r\}$. All events are observable.

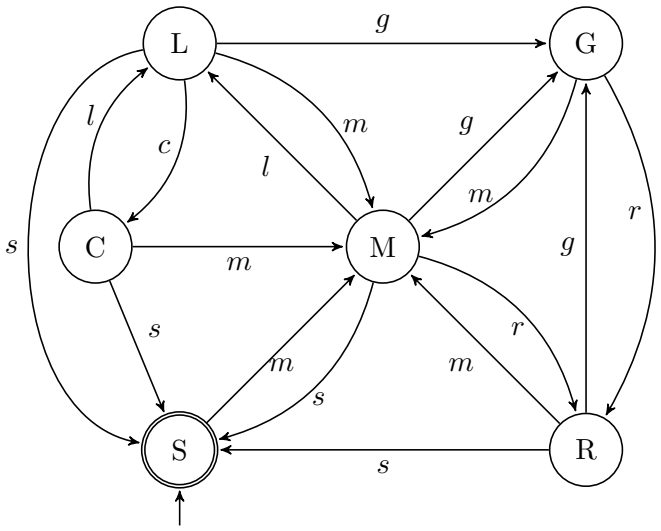


Figure 1. Generator of the language for the plant $\mathcal{G}$.

Let the specification language K corresponds to the automaton $\mathcal{H}$ in figure 2. K is supposed to be prefix-closed, i.e., $K = \overline{K}$. This specification means that communication mode for the AUV is not allowed under any circumstances. This condition may be necessary, for example, if the mission is implemented at a hostile area and the AUV should not be detected by an enemy. Although the model does not pretend to be realistic, it will help us to illustrate the results that follow. It may be noted that such K is not controllable.

3. Representation of DES in the calculus of PCFs

Let us consider a language of first-order logic that consists of first-order formulas (FOFs) built out of atomic formulas with $\&, \vee, \neg, \rightarrow, \leftrightarrow$ operators, $\forall$ and $\exists$ quantifier symbols and constants *true* and $false$. The concepts of term, atom, literal we define in the usual way. Hereafter, non-atomic formulas and subformulas will be denoted by capital calligraphic letters ($\mathcal{F}, \mathcal{P}, \mathcal{Q}$, etc.), in the general case with indices. Sets of formulas will be denoted by Greek capital letters (Φ, Ψ, etc.), possibly with indices.

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 $\forall x_1 \ldots \forall x_k (A_1 \& \ldots \& A_m) \rightarrow (\mathcal{F}_1 \vee \ldots \vee \mathcal{F}_n)$ and $\exists x_1 \ldots \exists x_k (A_1 \& \ldots \& A_m) \& (\mathcal{F}_1 \& \ldots \& \mathcal{F}_n)$ are

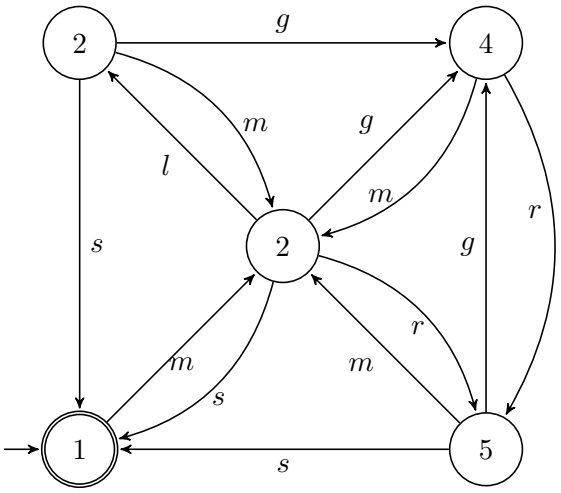


Figure 2. Recognizer $\mathcal{H}$ for the specification K.

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, keeping in mind that the $\forall$-quantifier corresponds to $\rightarrow \Phi^{\vee}$, where $\Phi^{\vee}$ means disjunction of all the formulas from Φ, and $\exists$-quantifier corresponds to $\&\Phi^{\&}$, where $\Phi^{\&}$ means conjunction of all the formulas from Φ. Any of sets X, A, Φ may be empty, and in this case they could be omitted in formulas. Thus, if $Q \in \{\forall, \exists\}$ then $Q_X A\ \varnothing \equiv Q_X A$, $Q_X \varnothing\ \Phi \equiv Q_X\ \Phi$ and $Q_{\varnothing} A\ \Phi \equiv QA\ \Phi$. Since empty disjunction is identical to $false$, whereas empty conjunction is identical to $true$, the following equivalences are correct: $\forall_X A\ \varnothing \equiv \forall_X A \rightarrow false \equiv \forall_X A$ and $\exists_X A\ \varnothing \equiv \exists_X A \& true \equiv \exists_X A$ and $\forall \varnothing\ \Phi \equiv true \rightarrow \Phi \equiv \forall\ \Phi$ and $\exists \varnothing\ \Phi \equiv true \& \Phi \equiv \exists\ \Phi$.

Definition 2. Let X be a set of variables, and A be a conjunct, both can be empty.

(i) $\exists_X A$ is an $\exists$-PCF,

(ii) $\forall_X A$ is a $\forall$-PCF,

(iii) If $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is a set of $\forall$-PCFs, then $\exists_X A\ \Phi$ is an $\exists$-PCF,

(iv) If $\Phi = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ is a set of $\exists$-PCFs, then $\forall_X A\ \Phi$ is a $\forall$-PCF,

(v) Any $\exists$-PCF or $\forall$-PCF is a PCF,

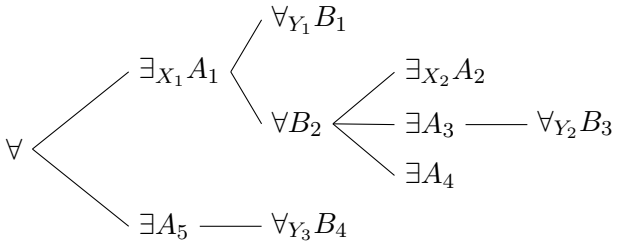
(vi) There are only PCFs of a form $\exists$-PCF and $\forall$-PCF.

The term "positively" comes from the fact that according to definition 3 PCFs contain no negation operator $\neg$.

For the sake of readability, we represent PCFs as trees whose nodes are type quantifiers, and we use corresponding notions: *node, root, leaf, branch.* For example, a PCF

$$\forall\ \{\exists_{X_1} A_1\ \{\forall_{Y_1} B_1,\ \forall B_2\ \{\exists_{X_2} A_2,\ \exists A_3\ \{\forall_{Y_2} B_3\},\ \exists A_4\}\},\ \exists A_5\ \{\forall_{Y_3} B_4\}\}$$

is represented as a tree as follows:



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*.

Inside each of the base subformulas, any variable cannot be free and bound simultaneously. Furthermore, it cannot be bound by different quantifiers simultaneously.

Example 2. The general form of PCF representing some automaton consists of the single base $\mathcal{B} = \{I(S), L(\varepsilon, S),\ L^m(\varepsilon, S), \delta(S_1^i, \sigma^i, S_2^i), \delta^m(S_1^i, \sigma^i, S_2^i), \Sigma_c(\sigma^j), \Sigma_{uc}(\sigma^j)\}$, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, k\}$, n is the number of transitions, k is the number of events, and two questions shown in figure 3. Here $L(s, S)$ denotes "s is a current sequence of events in the state S" and $L^m(s, S)$

$$\exists\, \mathcal{B} \left\langle \begin{array}{l} \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta(s, \sigma', s') \text{ —— } \exists L(\sigma \cdot \sigma', s') \\ \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta^m(s, \sigma', s') \text{ — } \exists L^m(\sigma \cdot \sigma', s') \end{array} \right.$$

Figure 3. General form of PCF representation of an automaton.

denote "s is a current sequence of events in the state S, and s is a marked string". The first arguments of these atoms will accumulate the strings of languages generated and marked by the automaton. Predicate of the form $\delta(S_1, \sigma, S_2)$ will be interpreted as the automaton transition from a state S_1 to a state S_2 with an event σ. If the target state of a transition is marked, then delta atoms with an index m are used, i.e., $\delta^m(S_1, \sigma, S_2)$ if S_2 is a marked state. The predicate $I(\_)$ denotes the initial state of the automaton. Controlled and uncontrolled events will be represented in the base by separate atoms using the predicates $\Sigma_c(\_)$ and $\Sigma_{uc}(\_)$, respectively. As usual, the function symbol "$\cdot$" denotes strings concatenation, and the "ε" symbol corresponds to the empty string.

Definition 3. [Answer] Consider some base subformula $\exists_X A\ \Psi$ 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 4. 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 5. 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 6. [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 $\omega(\mathcal{F}) = \forall\ \Phi \setminus \{\mathcal{B}\} \cup split(\mathcal{B}, \mathcal{Q}\theta)$.

Note, that 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 $\forall$).

Suppose that 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).

Example 3. The generator $\mathcal{G}$ depicted in figure 1 may be represented by the PCF shown in

figure 3 with B being the conjunct

$\{I_1(L), L(\varepsilon, L), \Sigma_{uc}(c), \delta_1(S, r, M),$
$\delta_1(M, s, S), \delta_1(M, l, L), \delta_1(M, g, G), \delta_1(M, r, R), \delta_1(L, s, S), \delta_1(L, c, C), \delta_1(L, m, M), \delta_1(L, g, G),$
$\delta_1(G, r, R), \delta_1(G, m, M), \delta_1(R, s, S), \delta_1(R, g, G), \delta_1(R, m, M), \delta_1(C, l, L), \delta_1(C, m, M), \delta_1(C, s, S),$
$\delta_1^m(L, s, S), \delta_1^m(C, s, S), \delta_1^m(M, s, S), \delta_1^m(R, s, S)\}.$

Applying the inference rules to this PCF, the words of the languages generated and marked by the automaton are constructed as the first arguments of the atoms $L(s, S)$, $L^m(s, S)$ in the base.

One of the essential features of the calculus of PCFs which will be used later is that we can build a *non-monotonic* inference by only slightly adjusting the definition of the inference rule. For this, we introduce the operator *, which will mark the atoms in the questions. Now, if a question with atoms marked with the * operator has an answer, then after applying the inference rule, the atoms in the base that participated in the matching search with the marked atoms should be removed from the base. In general, the operator * affects the property of completeness of the PCF calculus, but for the problem considered in this paper, thanks to a proper formalization, the inference using * always stops.

4. Supremal controllable sublanguage construction using a PCF inference

4.1. Product of automata construction

Suppose that behaviour of $\mathcal{G}$ should be constrained within the specification language K, and let automaton $\mathcal{H}$ generates, or recognizes, K. Before deigning a supervisor, the controllability of K must be verified by comparing uncontrollable transitions in automata $\mathcal{G}$ and $\mathcal{H}$. To merge the corresponding states of the automata, a product of $\mathcal{G}$ and $\mathcal{H}$ is built. The PCF $\mathcal{F}_{\mathcal{G}\times\mathcal{H}}$ (figure 4) constructing the product of automata consists of one base subformula, which base conjunct is $B_{\mathcal{G}\times\mathcal{H}} = \{I_1(S_0), I_2(P_0), \delta_1(S_1^i, \sigma^i, S_2^i), \delta_2(S_1^k, \sigma^k, S_2^k)\}$, containing atoms for transitions δ_1, $i = \overline{1, n_1}$, of the automaton $\mathcal{G}$ and transitions δ_2, $k = \overline{1, n_2}$, of the automaton $\mathcal{H}$.

$$\exists\, B_{\mathcal{G}\times\mathcal{H}} \left\langle \begin{array}{ll} \forall s, p\ I_1(s), I_2(p) & \text{——————} \exists I_3(s \cdot p) \\ \forall s, p\ I_1(s), I_2(p) & \text{——————} \exists \delta_3(\xi \cdot \rho, \varepsilon, s \cdot p) \\ \forall \sigma, s_1, p_1, s_2, p_2, \sigma', s'\ \delta_1(s_1, \sigma, s_2), & \\ \quad \delta_2(p_1, \sigma, p_2), & \text{——} \exists \delta_3(s_1 \cdot p_1, \sigma, s_2 \cdot p_2) \\ \quad \delta_3(s', \sigma', s_1 \cdot p_1) & \end{array} \right.$$

Figure 4. PCF constructing the product of automata.

Index one in the subscripts of atoms of this formula corresponds to the atoms of the automaton $\mathcal{G}$, two corresponds to the $\mathcal{H}$, and three corresponds to the atoms of $\mathcal{G} \times \mathcal{H}$. Here, the functional symbol $\cdot$ is used to trace the pairs of states in the product automaton. The first question adds the initial state of the product automaton to the base. The second question adds an undetermined transition atom corresponding to the third automaton, thanks to which the connectivity of the product automaton is controlled in the third question. The latter adds the transition atoms of the product automaton to the base.

Example 4. figure 5 shows the product of the automaton $\mathcal{G}$ shown in figure 1 with the automaton $\mathcal{H}$ shown in figure 2.

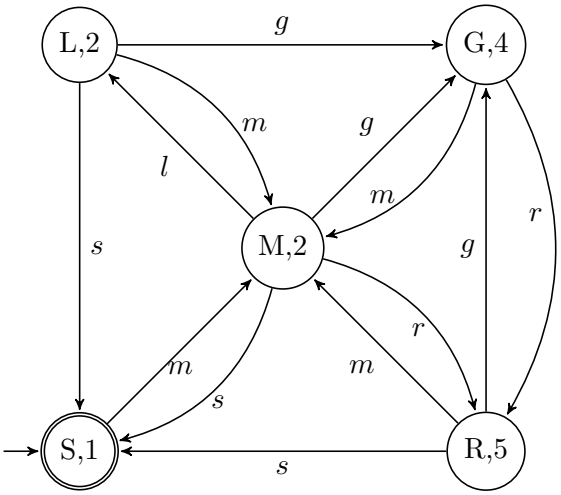


Figure 5. The product of $\mathcal{G}$ and $\mathcal{H}$.

4.2. Controllability checking

To check controllability, we use PCF $\mathcal{F}_C$ with the base $B_C = B_{\mathcal{G}\times\mathcal{H}} \cup \{\delta_3(S_{11}^k \cdot S_{21}^k, \sigma^k, S_{12}^k \cdot S_{22}^k)\}$, $k = \overline{1, n_3}$, that contains predicate descriptions of states and transitions of $\mathcal{G}$, $\mathcal{H}$, and recently added transitions of the function δ_3 of $\mathcal{G} \times \mathcal{H}$. Let the questions to the base of $\mathcal{F}_C$ be as depicted in figure 6. The questions have the following meaning.

$$\exists\, B_C \Leftarrow \begin{cases} Q_1 \\ Q_2 \\ \dots \\ Q_6 \end{cases}$$

$Q_1:\ \forall s_1, s_2, p_1, p_2, \sigma\ I_1(s_1), I_3(p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) - \exists\, N(s_1, p_1)$

$Q_2:\ \forall s_1, s_2, p_1, \sigma\ N(s_1, p_1), \delta_1(s_1, \sigma, s_2), E_{uc}(\sigma) - \exists\, Chk(p_1, \sigma, 0)$

$Q_3:\ \forall p_1, p_2, \sigma\ Chk^*(p_1, \sigma, 0), \delta_3(p_1, \sigma, p_2) - \exists\, Chk(p_1, \sigma, 1)$

$Q_4:\ \forall p_1, \sigma\ Chk(p_1, \sigma, 0) - \exists\, UC(p_1, \sigma)$

$Q_5:\ \forall p_1, \sigma\ UC(p_1, \sigma)$

$Q_6:\ \forall s_1, s_2, p_1, p_2, \sigma\ N(s_1, p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) - \exists\, N(s_2, p_2).$

Figure 6. PCF $\mathcal{F}_C$ for controllability checking.

Q_1 adds to the base the initial states of $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ as the first pair of states to be checked for violating controllability. We call states which are simultaneously achieved from the states of $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ by the same event the *neighbouring* states. The predicate $N(\_, \_)$ is used to store pairs of neighbouring states. Initial states of $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ are not neighbouring in our sense, but $N(s_1, p_1)$ with them is necessary for further inference.

Q_2 adds to the base the atom $Chk(p_1, \sigma, 0)$ which means that there is an uncontrollable transition σ from the state p_1 of the automaton $\mathcal{G}$. If there is no such transition from the state

p_1 of $\mathcal{H}$, K is not controllable. This is checked by the next rule.

Q_3 checks if both neighbouring states share the same uncontrollable event. A successful answer to this question means that the uncontrollable event is legal, i.e., allowed by the specification, since a transition labelled by it exists in both $\mathcal{G}$ and $\mathcal{H}$. In this case, the atom $Chk(p_1, \sigma, 0)$ is deleted from the base with the help of * operator.

Q_4 checks if the atom $Chk(p_1, \sigma, 0)$ is present in the base. Such atom contains information about the state p_1 and the event σ that violate the controllability condition.

Q_5 is the goal question that ends the inference.

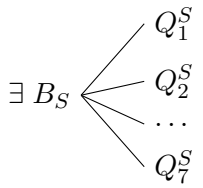
Q_6 adds the next pair of states to be checked to continue the inference search.

Since the formalized automata are finite with the finite sets of events, then the search space for the inference in this formalization is finite. If the inference has finished with the exhaustion of the search options for substitutions and the answer to the goal question has not been found, then the specification language is controllable.

Example 5. It may be noted that the specification in figure 2 is not controllable due to the event e at the state L. We do not provide here the inference proving it but leave this issue till the next subsection.

4.3. Controllable sublanguage construction

Slightly changing questions of the PCF $\mathcal{F}_C$, we can construct $K^{\uparrow C}$ during the checking controllability of K. Consider the PCF $\mathcal{F}_S$ depicted in figure 7. Let questions Q_1^S – Q_3^S are the same as the questions Q_1 – Q_3 of the PCF $\mathcal{F}_C$ above and have the same interpretation, Q_5^S coincides with Q_6. The rest of the questions are interpreted as follows.



$Q_4^S:\ \forall p_1, \sigma\ Chk(p_1, \sigma, 0) - \exists\ Del(p_1)$

$Q_5^S:\ \forall s_1, s_2, p_1, p_2, \sigma\ N(s_1, p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) - \exists\ N(s_2, p_2)$

$Q_6^S:\ \forall p_{11}, p_{12}, p_{21}, p_{22}, \sigma\ Del(p_{11} \cdot p_{12}), \delta_2^*(p_{12}, \sigma, p_{22}), \delta_3^*(p_{11} \cdot p_{12}, \sigma, p_{21} \cdot p_{22}) -$
$- \exists\ Deleted(p_{12}, \sigma, p_{22})$

$Q_7^S:\ \forall p_{11}, p_{12}, p_{21}, p_{22}, \sigma\ Del(p_{21} \cdot p_{22}), \delta_2^*(p_{12}, \sigma, p_{22}), \delta_3^*(p_{11} \cdot p_{12}, \sigma, p_{21} \cdot p_{22}) -$
$- \exists\ Deleted(p_{12}, \sigma, p_{22})$

Figure 7. PCF $\mathcal{F}_S$ for controllable sublanguage construction.

As Q^4 does, Q_4^S checks if the atom $Chk(p_1, \sigma, 0)$ is present in the base and, additionally, marks the state p_1 for deletion with the help of the predicate $Del(\_)$. The atom $Del(p_1)$ is further used in questions Q_6^S, Q_7^S to delete transitions associated with the state p_1. The rules Q_6^S, Q_7^S are simple and straightforward. If the $Del(p_1)$ atom was added to the base, then state p_1 should be deleted from the automaton $\mathcal{H}$. Thus, Q_6^S removes all transitions leaving this state and Q_7^S removes all transitions leading to this state.

To construct a supremal controllable sublanguage of K, we use the following strategy. The inference is built by checking the applicability of the inference rule to questions in order from

Q_1^S to Q_7^S. The rules Q_1^S – Q_5^S check whether the specification is controllable or not. The rules Q_6^S, Q_7^S remove transitions associated with the state in which an event occurs that violates the controllability condition. If during the application of the Q_1^S – Q_5^S rules the answer to question Q_4^S was found, an atom $Del(\_)$ denoting uncontrollability of K is added to the base. Then we apply the rules Q_6^S, Q_7^S, ignoring the rules from the Q_1^S – Q_5^S, until possible substitutions for applying the inference rule to Q_6^S and Q_7^S are not exhausted. Next, the process is repeated until the inference stops, having exhausted all possible substitutions. Thus, if for the given specification a controllable sublanguage is empty, then during the inference all transitions of the automata corresponding to the specification will be deleted. Consider this strategy with an example.

Example 6. Table 1 shows the inference of the PCF, which constructs the supremal controllable sublanguage of the specification K represented by the automaton shown in figure 2. It is assumed that the product automaton (figure 5) is already built, and the base from which the inference starts includes the next subset of atoms:

$\{I_1(S), I_3(S \cdot 1), \Sigma_{uc}(c), \Sigma_{uc}(r), \Sigma_{uc}(s), \delta_1(S, r, M),$
$\delta_1(M, s, S), \delta_1(M, l, L), \delta_1(M, g, G), \delta_1(M, r, R), \delta_1(L, s, S), \delta_1(L, c, C), \delta_1(L, m, M), \delta_1(L, g, G),$
$\delta_1(G, r, R), \delta_1(G, m, M), \delta_1(R, s, S), \delta_1(R, g, G), \delta_1(R, m, M), \delta_1(C, l, L), \delta_1(C, m, M), \delta_1(C, s, S),$
$\delta_2(S, r, M), \delta_2(M, s, S), \delta_2(M, l, L), \delta_2(M, g, G), \delta_2(M, r, R), \delta_2(L, m, M), \delta_2(L, g, G), \delta_2(G, r, R),$
$\delta_2(L, s, S), \delta_2(G, m, M), \delta_2(R, s, S), \delta_2(R, g, G), \delta_2(R, m, M),$
$\delta_3(S \cdot 1), m, M \cdot 2), \delta_3(M \cdot 2), s, S \cdot 1), \delta_3(M \cdot 2), l, L \cdot 3), \delta_3(M \cdot 2), g, G \cdot 4),$
$\delta_3(M \cdot 2), r, R \cdot 5), \delta_3(L \cdot 3), s, S \cdot 1), \delta_3(L \cdot 3), m, M \cdot 2), \delta_3(L \cdot 3), g, G \cdot 4),$
$\delta_3(G \cdot 4), r, R \cdot 5), \delta_3(G \cdot 4), m, M \cdot 2), \delta_3(R \cdot 5), s, S \cdot 1), \delta_3(R \cdot 5), g, G \cdot 4), \delta_3(R \cdot 5), m, M \cdot 2)\}$

For convenience, in the table, we omit the superscript S in questions names. The automaton recognizing $K^{\uparrow C}$ is shown in figure 8.

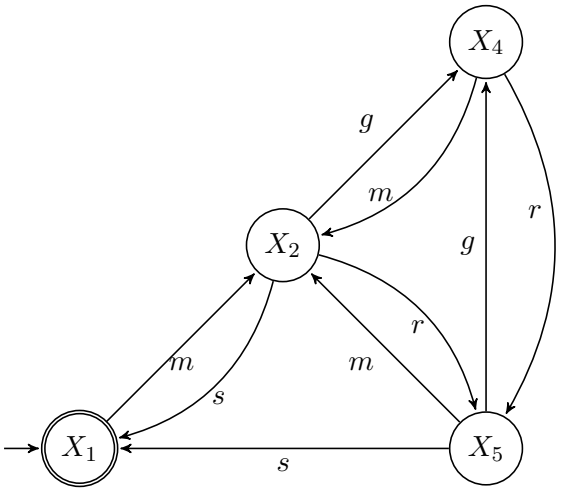


Figure 8. Recognizer for the supremal controllable sublanguage of K.

Table 1. Constructive inference building the supremal controllable sublanguage of the specification.

Q	Base atoms used	Substitution	Atoms added	Step Result
1 Q_1	$I_1(S), I_3(S\cdot 1)$, $\delta_1(S,m,M)$, $\delta_3(S\cdot 1, m, M\cdot 2)$	$\{s_1 \to S, s_2 \to M$, $p_1 \to S\cdot 1, p_2 \to M\cdot 2$, $\sigma \to m\}$	$N(S, S\cdot 1)$	The pair of states S and $(S,1)$ in the automata $\mathcal{G}$ and $\mathcal{G}\times\mathcal{H}$ is to be checked for violation of controllability
2 Q_5	$N(S, S\cdot 1)$, $\delta_1(S,m,M)$, $\delta_3(S\cdot 1, m, M\cdot 2)$	$\{s_1 \to S, s_2 \to M$, $p_1 \to S\cdot 1, p_2 \to M\cdot 2, \sigma \to m\}$	$N(M, M\cdot 2)$	The same as at the first step of this inference
3 Q_2	$N(M, M\cdot 2)$, $\delta_1(M,s,S)$ $\Sigma_{uc}(s)$	$\{s_1 \to A, s_2 \to C$, $p_1 \to A\cdot 1, \sigma \to a\}$	$Chk(M\cdot 2, s, 0)$	In automaton $\mathcal{G}$ an uncontrolled transition labeled by the event s was found, i.e., the check of the corresponding transition in the automaton $\mathcal{G}\times\mathcal{H}$ is necessary
4 Q_2	$N(M, M\cdot 2)$, $\delta_1(M,r,R)$ $\Sigma_{uc}(r)$	$\{s_1 \to M, s_2 \to R$, $p_1 \to M\cdot 2, \sigma \to a\}$	$Chk(M\cdot 2, r, 0)$	The same as in previous step, the check assigned for the event r
5 Q_5	$N(M, M\cdot 2)$, $\delta_1(M,l,L)$, $\delta_3(M\cdot 2, l, L\cdot 3)$	$\{s_1 \to M, s_2 \to L$, $p_1 \to M\cdot 2, p_2 \to L\cdot 3, \sigma \to l\}$	$N(L, L\cdot 3)$	The same as at the first step of this inference
6 Q_5	$N(M, M\cdot 2)$, $\delta_1(M,g,G)$, $\delta_3(M\cdot 2, g, G\cdot 4)$	$\{s_1 \to M, s_2 \to G$, $p_1 \to M\cdot 2, p_2 \to G\cdot 4, \sigma \to g\}$	$N(G, G\cdot 4)$	The same as at the first step of this inference
7 Q_5	$N(M, M\cdot 2)$, $\delta_1(M,r,R)$, $\delta_3(M\cdot 2, r, R\cdot 5)$	$\{s_1 \to M, s_2 \to R$, $p_1 \to M\cdot 2, p_2 \to R\cdot 5, \sigma \to r\}$	$N(R, R\cdot 5)$	The same as at the first step of this inference
8 Q_3	$Chk(M\cdot 2, s, 0)$ $(deleted)$, $\delta_3(M\cdot 2, s, S\cdot 1)$	$\{p_1 \to M\cdot 2$, $p_2 \to S\cdot 1$, $\sigma \to s\}$	$Chk(M\cdot 2, s, 1)$	The check assigned in the step 3 is passed, i.e., violation of controllability is not found
9 Q_3	$Chk(M\cdot 2, r, 0)$ $(deleted)$, $\delta_3(M\cdot 2, r, R\cdot 5)$	$\{p_1 \to M\cdot 2$, $p_2 \to R\cdot 5$, $\sigma \to r\}$	$Chk(M\cdot 2, r, 1)$	The check assigned in the step 4 is passed, i.e., violation of controllability is not found
10 Q_2	$N(L, L\cdot 3)$, $\delta_1(L,s,S)$ $\Sigma_{uc}(s)$	$\{s_1 \to L, s_2 \to S$, $p_1 \to L\cdot 3, \sigma \to s\}$	$Chk(L\cdot 3, s, 0)$	The same as in step 4, the check assigned for the event s, i.e., if there is the transition from L labeled with s there must be the same transition labeled with s in the product automaton from the state $L, 2$
11 Q_2	$N(G, G\cdot 4)$, $\delta_1(G,r,R)$ $\Sigma_{uc}(r)$	$\{s_1 \to G, s_2 \to R$, $p_1 \to G\cdot 4, \sigma \to r\}$	$Chk(G\cdot 4, r, 0)$	Another check is assigned for the event r which goes from the state G
				Continued on the next page

Table 1 – continued from the previous page

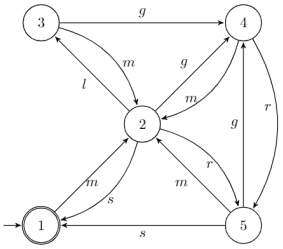
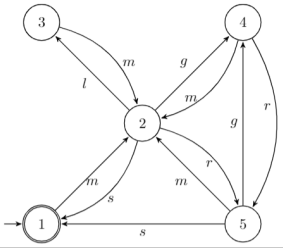
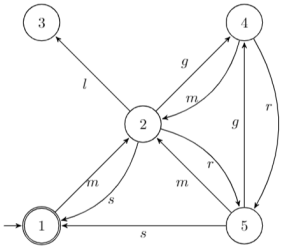
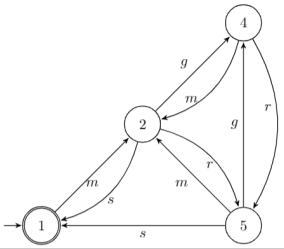
Q	Base atoms used	Substitution	Atoms added	Step Result
12 Q_2	$N(R, R\cdot 5)$, $\delta_1(R, s, S)$ $\Sigma_{uc}(s)$	$\{s_1 \to R, s_2 \to S,$ $p_1 \to R\cdot 5, \sigma \to s\}$	$Chk(R\cdot 5, s, 0)$	Another check is assigned for the event s which may occur at the state R
13 Q_3	$Chk(L\cdot 3, s, 0)$ $(deleted)$, $\delta_3(L\cdot 3, s, S\cdot 1)$	$\{p_1 \to L\cdot 3,$ $p_2 \to S\cdot 1,$ $\sigma \to s\}$	$Chk(L\cdot 3, s, 1)$	The check assigned at the step 10 is passed, i.e., violation of controllability is not found
14 Q_3	$Chk(G\cdot 4, r, 0)$ $(deleted)$, $\delta_3(G\cdot 4, r, R\cdot 5)$	$\{p_1 \to G\cdot 4,$ $p_2 \to R\cdot 5,$ $\sigma \to r\}$	$Chk(G\cdot 4, r, 1)$	The check assigned in the step 11 is passed, i.e., violation of controllability is not found
15 Q_3	$Chk(R\cdot 5, s, 0)$ $(deleted)$, $\delta_3(R\cdot 5, s, S\cdot 1)$	$\{p_1 \to R\cdot 5,$ $p_2 \to S\cdot 1,$ $\sigma \to s\}$	$Chk(R\cdot 5, s, 1)$	The check assigned at the step 12 is passed, i.e., violation of controllability is not found
16 Q_2	$N(L, L\cdot 3)$, $\delta_1(L, c, C)$ $\Sigma_{uc}(c)$	$\{s_1 \to L, s_2 \to C,$ $p_1 \to L\cdot 3, \sigma \to c\}$	$Chk(L\cdot 3, c, 0)$	The same as at the previous step, the check assigned for another uncontrollable transition from L labeled with c
17 Q_4	$Chk(L\cdot 3, c, 0)$	$\{p_1 \to L\cdot 3,$ $\sigma \to c\}$	$Del(L\cdot 3, c)$	The check assigned at the previous step was failed; therefore, the specification corresponding to the automaton $\mathcal{H}$ is uncontrollable.
Since the tested specification is uncontrollable, the rules for deleting the transitions associated with the state found in the previous steps are triggered.				
18 Q_6	$Del(L\cdot 3)$, $\delta_2(3, s, 1)$, $\delta_3(L\cdot 3, s, S\cdot 1)$	$\{p_{11} \to L, p_{12} \to 3,$ $p_{21} \to S, p_{22} \to$ $1, \sigma \to s\}$	$Deleted(3, s, 1)$	
19 Q_6	$Del(L\cdot 3)$, $\delta_2(3, g, 4)$, $\delta_3(L\cdot 3, g, G\cdot 4)$	$\{p_{11} \to L, p_{12} \to 3,$ $p_{21} \to G, p_{22} \to$ $4, \sigma \to g\}$	$Deleted(3, g, 4)$	
				Continued on the next page

Table 1 – continued from the previous page

Q	Base atoms used	Substitution	Atoms added	Step Result
20 Q_6	$Del(L \cdot 3)$, $\delta_2(3, m, 2)$, $\delta_3(L{\cdot}3, m, M{\cdot}2)$	$\{p_{11} \to L, p_{12} \to 3$, $p_{21} \to M, p_{22} \to 2, \sigma \to m\}$	$Deleted(3, m, 2)$	
21 Q_7	$Del(L \cdot 3)$, $\delta_2(2, l, 3)$, $\delta_3(M \cdot 2, l, L \cdot 3)$	$\{p_{11} \to M, p_{12} \to 2$, $p_{21} \to L, p_{22} \to 3, \sigma \to l\}$	$Deleted(2, l, 3)$	

5. Supervisor realization

In the previous section, we have considered the way of building the supremal controllable sublanguage for the given specification. Taking this language as a new specification, we can find a solution of the basic problem of supervisory control, i.e., a supervisor $\mathcal{J}$ and the closed-looped system such that $L(\mathcal{J}/\mathcal{G}) = K^{\uparrow C}$. A closed-looped behaviour of the plant and the supervisor is usually realized by the parallel composition of the corresponding automata of the plant and the supervisor, i.e., $L(\mathcal{J}/\mathcal{G}) = L(\mathcal{J}||\mathcal{G})$ [20]. In our formalization, the joint work of the system and the supervisor is carried out using the PCF $\mathcal{F}_{SC}$ below.

$$\exists\, B, B_S \text{ —— } \forall \sigma, s, \sigma', s'\ L(\sigma, s), \delta_1(s, \sigma', s'), \delta_2(s, \sigma', s')\text{-}\exists L(\sigma \cdot \sigma', s')$$

Here bases B and B_S are the sets of atoms corresponding to 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_2, 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 $L(\_, \_)$. That is, for any transition corresponding to the language $L(\mathcal{G})$ (an atom $\delta_1()$), we simultaneously trace the corresponding event in the automaton of the supervisor (an atom $\delta_2()$). The rule works only on those strings that are allowed by the supervisor, i.e., atoms $\delta_2()$ limit the answers that could be generated with atoms $\delta_1()$ only.

Note that the inference is organized in such a way as to assure a sequential accumulation of events. This means that, first of all, all possible continuations from the initial state will be added to the empty string. After then, all events from the neighbouring states will be added to all strings of the length one in the base, and so on. The search strategy can also be configured to analyze strings. For example, when all strings of a given length are generated, each next transition can be controlled in addition to the supervisor, by applying additional rules on the appeared strings. This feature of the PCF calculus may be utilized for the optimal supervisory control. The mentioned rules may be defined by an operator. Moreover, all the inference can be made interactive, for example, by pausing after each event, or after an event leading the system to the marked state.

Example 7. Table 2 demonstrates the fist steps of the inference generating $L(\mathcal{J}/\mathcal{G})$ for the AUV model from the Example 1. Taking the language recognized by the automaton $\mathcal{H}_{K^{\uparrow C}}$ in figure 8 as a new specification, we build the closed-looped system such that $L(\mathcal{J}/\mathcal{G}) = K^{\uparrow C}$

using the PCF $\mathcal{F}_{SC}$ with B being the set of atoms corresponding to the transitions of $\mathcal{G}$ and B_S being the set of atoms corresponding to the transitions of $\mathcal{H}_{K^{\uparrow C}}$:

$$\{L(\varepsilon, X_1), \Sigma_{uc}(c), \Sigma_{uc}(r), \Sigma_{uc}(s), \delta_2(X_1, m, X_2), \delta_2(X_2, s, X_1), \delta_2(X_2, g, X_4),$$
$$\delta_2(X_2, r, X_5), \delta_2(X_4, r, X_5), \delta_2(X_4, m, X_2), \delta_2(X_5, s, X_1), \delta_2(X_5, g, X_4), \delta_2(X_5, m, X_2),$$
$$\delta_2(C, m, X_2), \delta_2(C, s, X_1), \delta_2^m(C, s, X_1), \delta_2^m(X_2, s, X_1), \delta_2^m(X_5, s, X_1)\}$$

Table 2. Generating $L(\mathcal{J}/\mathcal{G})$ for the AUV model.

Q	Base atoms	Substitution	Atoms added	Step Result
1 Q_1	$L(\varepsilon, X_1)$, $\delta_1(X_1, m, X_2)$ $\delta_2(X_1, m, X_2)$	$\{\sigma \to \varepsilon, \sigma' \to m,$ $s \to X_1, s' \to X_2\}$	$L(\varepsilon \cdot m, X_2)$	Adding the first event m to the chain of events Current chain of events: m
2 Q_1	$L(\varepsilon \cdot m, X_2)$, $\delta_1(X_2, s, X_1)$ $\delta_2(X_2, s, X_1)$	$\{\sigma \to \varepsilon \cdot m, \sigma' \to s,$ $s \to X_2, s' \to X_1\}$	$L(\varepsilon \cdot m \cdot s, X_1)$	Adding the next possible event s, which corresponds to the transition from X_2 to X_1 Current chain of events: ms
3 Q_1	$L(\varepsilon \cdot m, X_2)$, $\delta_1(X_2, g, X_4)$ $\delta_2(X_2, g, X_4)$	$\{\sigma \to \varepsilon \cdot m, \sigma' \to g,$ $s \to X_2, s' \to X_4\}$	$L(\varepsilon \cdot m \cdot g, X_4)$	Adding the next possible event g, which corresponds to the transition from X_2 to X_4 Current chain of events: mg
4 Q_1	$L(\varepsilon \cdot m, X_2)$, $\delta_1(X_2, r, X_5)$ $\delta_2(X_2, r, X_5)$	$\{\sigma \to \varepsilon \cdot m, \sigma' \to r,$ $s \to X_2, s' \to X_5\}$	$L(\varepsilon \cdot m \cdot r, X_5)$	Current chain of events: mr
5 Q_1	$L(\varepsilon \cdot m \cdot s, X_1)$, $\delta_1(X_1, m, X_2)$ $\delta_2(X_1, m, X_2)$	$\{\sigma \to \varepsilon \cdot m \cdot s, \sigma' \to$ $m,$ $s \to X_1, s' \to X_2\}$	$L(\varepsilon{\cdot}m{\cdot}s{\cdot}m, X_2)$	Current chain of events: msm
6 Q_1	$L(\varepsilon \cdot m \cdot g, X_4)$, $\delta_1(X_4, m, X_2)$ $\delta_2(X_4, m, X_2)$	$\{\sigma \to \varepsilon \cdot m \cdot g, \sigma' \to$ $m,$ $s \to X_4, s' \to X_2\}$	$L(\varepsilon{\cdot}m{\cdot}g{\cdot}m, X_2)$	Current chain of events: mgm
7 Q_1	$L(\varepsilon \cdot m \cdot g, X_4)$, $\delta_1(X_4, r, X_5)$ $\delta_2(X_4, r, X_5)$	$\{\sigma \to \varepsilon{\cdot}m{\cdot}g, \sigma' \to r,$ $s \to X_4, s' \to X_5\}$	$L(\varepsilon{\cdot}m{\cdot}g{\cdot}r, X_5)$	Current chain of events: mgr
8 Q_1	$L(\varepsilon \cdot m \cdot r, X_5)$, $\delta_1(X_5, s, X_1)$ $\delta_2(X_5, s, X_1)$	$\{\sigma \to \varepsilon{\cdot}m{\cdot}r, \sigma' \to s,$ $s \to X_5, s' \to X_1\}$	$L(\varepsilon{\cdot}m{\cdot}r{\cdot}s, X_1)$	Current chain of events: mrs
9 Q_1	$L(\varepsilon \cdot m \cdot r, X_5)$, $\delta_1(X_5, g, X_4)$ $\delta_2(X_5, g, X_4)$	$\{\sigma \to \varepsilon{\cdot}m{\cdot}r, \sigma' \to g,$ $s \to X_5, s' \to X_4\}$	$L(\varepsilon{\cdot}m{\cdot}r{\cdot}g, X_4)$	Current chain of events: mrg
10 Q_1	$L(\varepsilon \cdot m \cdot r, X_5)$, $\delta_1(X_5, m, X_2)$ $\delta_2(X_5, m, X_2)$	$\{\sigma \to \varepsilon \cdot m \cdot r, \sigma' \to$ $m,$ $s \to X_5, s' \to X_2\}$	$L(\varepsilon{\cdot}m{\cdot}r{\cdot}m, X_2)$	Current chain of events: mrm
11 Q_1	$L(\varepsilon{\cdot}m{\cdot}s{\cdot}m, X_2)$, $\delta_1(X_2, s, X_1)$ $\delta_2(X_2, s, X_1)$	$\{\sigma \to \varepsilon{\cdot}m{\cdot}s{\cdot}m, \sigma' \to$ $s,$ $s \to X_2, s' \to X_1\}$	$L(\varepsilon \cdot m \cdot s \cdot m \cdot$ $s, X_1)$	Current chain of events: $msms$
And so on... The inference will never terminate...				

Conclusion

In this paper, we intended to provide a full picture on the PCF calculus based approach to solving some basic problems of SCT for DES in the automata form. The formalization of DES as PCF is presented and algorithms for controllability checking and the supremal controllable sublanguage construction are illustrated.

A few words must be said about the end of the work of the inference search machine for the PCFs exploited above. In the classical variant of the PCF calculus, there are three possible results of the search for an inference of some formula $\mathcal{F}$:

1) the inference of $\mathcal{F}$ is constructed, i.e., according to the definition, all base subformulas have been refuted, so $\mathcal{F}$ is proved to be unsatisfiable;

2) the inference of $\mathcal{F}$ is not constructed, but the inference search machine has stopped, having exhausted all variants of substitutions for the application of the inference rule; in this case, we may speak about the completed proof of satisfiability of $\mathcal{F}$;

3) the inference of $\mathcal{F}$ is not constructed and the inference search machine has stopped without exhausting all the substitution variants for applying the inference rule, i.e., stopped by the forced shutdown; in this case, nothing can be said about the satisfiability of $\mathcal{F}$.

When considering a particular PCF in this article, we stipulate which of the options for completing the inference we mean when applying the inference search machine to this formula. The exception is the last formula. This formula simulates the generation of a possibly infinite formal language, and therefore it is constructed in such a way that its inference does not end.

The further work supposes modular and partially observed DES study, design of decentralized supervisors, and temporal logic implementation for SCT via the PCF calculus. Results obtained will be embedded at the different levels of the hierarchical control system for mobile robots.

Acknowledgments

The research was partly supported by the RFBR, project no. 20-07-00397 (sections 2 and 5), and by the Basic Research Program of SB RAS, project no. IV.38.1.1.

References

[1] Wonham W M and Cai K 2019 *Supervisory Control of Discrete-Event Systems* (Springer International Publishing)

[2] Davydov A, Larionov A and Nagul N 2017 *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (IEEE) pp 1161–1165

[3] Davydov A, Larionov A and Nagul N 2018 *AIP Conference Proceedings* **2046** 020021 (*Preprint* `https://aip.scitation.org/doi/pdf/10.1063/1.5081541`) URL `https://aip.scitation.org/doi/abs/10.1063/1.5081541`

[4] Davydov A, Larionov A and Nagul N V 2019 *Proceedings of the 1st International Workshop on Information, Computation, and Control Systems for Distributed Environments, ICCS-DE 2019, Irkutsk, Russia, July 8-9, 2019* (*CEUR Workshop Proceedings* vol 2430) ed Bychkov I and Tchernykh A (CEUR-WS.org) pp 29–41 URL `http://ceur-ws.org/Vol-2430/paper3.pdf`

[5] Vassilyev S N 1990 *The Journal of Logic Programming* **9** 235–266

[6] Zherlov A K, Vassilyev S N, Fedosov E A and Fedunov B E 2000 *Intelligent control of dynamic systems* (Moscow: Fizmatlit) in Russian

[7] Davydov A, Larionov A and Cherkashin E 2011 *Automatic Control and Computer Sciences* **45** 402–407

[8] Larionov A, Davydov A and Cherkashin E 2013 *International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija* **2013** 1023–1028

[9] Bychkov I, Davydov A, Nagul N and Ul'yanov S 2018 *Information Technology in Industry* **6** 20–26

[10] Bychkov I, Davydov A, Kenzin M, Maksimkin N, Nagul N and Ul'yanov S 2019 *2019 International Siberian Conference on Control and Communications (SIBCON)* pp 1–6

[11] Jayasiri A, Mann G and Gosine R 2011 *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **41** 1224 – 1238 ISSN 1083-4419

[12] Torrico C R, Leal A B and Watanabe A T 2016 *IFAC-Papers OnLine* **49** 240 – 245 ISSN 2405-8963

[13] Dai X, Jiang L and Zhao Y 2016 *Applied Intelligence* **45** 18–29 ISSN 1573-7497

[14] Tsalatsanis A, Yalcin A and Valavanis K P 2012 *Robotica* **30** 721—730

[15] Hill R C and Lafortune S 2017 *2017 American Control Conference (ACC)* pp 3840–3847 ISSN 2378-5861
[16] Gamage G W, Mann G K I and Gosine R G 2009 *Proceedings of the 2009 IEEE RSJ International Conference on Intelligent Robots and Systems* IROS 09 (Piscataway, NJ, USA: IEEE Press) pp 4831–4836 ISBN 978-1-4244-3803-7
[17] Lopes Y K, Trenkwalder S M, Leal A B, Dodd T J and Groß R 2016 *Swarm Intelligence* **10** 65–97 ISSN 1935-3820
[18] Mendiburu F J, Morais M R A and Lima A 2016 *IEEE International Conference on Automatica (ICA-ACCA)* **Oct. 2016** 1 – 7 ISSN 2405-8963
[19] Ramadge P J and Wonham W M 1987 *SIAM Journal on Control and Optimization* **25** 206–230
[20] Cassandras C G and Lafortune S 2008 *Introduction to Discrete Event Systems* (Springer US)