Application of the PCF calculus for solving the problem of nonblocking supervisory control of discrete event systems

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 suggests a representation of discrete event system as a positively constructed formula to imply automated logical inference in the calculus of positively constructed formulas. During the inference, languages of the discrete event system are generated and analyzed. The minimally restricting supervisor for uncontrollable specification may be also designed. A nonblocking supervisor design is illustrated with a simplified model of autonomous underwater vehicle operational modes switching.

1. Introduction
The paper discusses the solving of the problem of nonblocking supervisory control of discrete event systems (DES) [1] with the help of the logical inference and automatic theorem proving (ATP) in the calculus of positively constructed formulas (PCFs). The blocking issue in DES arises due to the using of marked states to point completion of some tasks in the modeled system. Despite the existence of algorithms for checking the system for blocking, when considering large systems, modelled as several components interacting with each other, such a check becomes a computationally complex task that requires new methods and approaches [2, 3]. Also blocking property becomes crucial when building modular or decentralized supervisors, as combination of nonblocking supervisors may block.

Leaving modular and decentralized problems till future research, in this paper we give a general view of the logical inference based approach to dealing with supervisory control of DES. The approach suggests a representation of automata-based DES as a PCF and implementation inference search machine in the PCF calculus to DES analysis and designing control. The PCF calculus proved to be effective at solving problems of dynamic systems control because its constructive semantics may be employed to extract the knowledge from proofs, while non-monotonicity and a treat of time help to construct plans in dynamically changing subject areas [4, 5, 6]. The PCF calculus allows one to apply to the automatic search for logic inference special domain-based heuristics which are customizable for a task being solved. Moreover, semantics of the PCF language may be modified to be constructive, non-monotonic or temporal, and provides tools to construct intuitionistic inferences of some non-Horn formulas which arise in problems
of dynamic systems control. Examples of elevator group control, mobile robot action planning and telescope guidance may be found in [5].

During the inference of the PCF, languages, generated and marked by the DES, are accumulated in the PCF base. Thus, automaton as a generator of a formal language is simulated. Using PCFs, parallel composition and product of automata may be also built. These structures are then applied for supervisory control theory problems solving. The approach was developed in authors’ several works [7, 8, 9] that concerned DES formalization, controllability, observability and coobservability of specifications on DES functioning checking, and supervisor implementation.

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 [10] and [11], 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 [12], [13], robot groups control [14], [15], [16], robots formation control [17], and swarm robotics [18], [19]. 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 structure is the following. The problem of nonblocking supervisory control of DES is stated in section 2. The third section contains a description of DES representation in the PCF calculus. In section 4, a way of supremal controllable sublanguage of a given language is described. In section 5, a supervised DES is constructed. We conclude by discussion of crucial features of the prover utilized for inference in the PCF calculus search and give the future lines of research.

2. Nonblocking supervisory control of DES

Let a discrete event system is specified in the form of a finite-state automaton \( G = (Q, \Sigma, \delta, q_0, Q_m) \) as a generator of a formal language [20]. Here \( Q \) is the set of states \( q \); \( \Sigma \) is the set of events; \( \delta: \Sigma \times Q \rightarrow Q \) is the transition function; \( q_0 \in Q \) is the initial state; \( Q_m \subset Q \) is the set of marked states. \( G \) is also called a plant in the automatic control theory.

Let \( \Sigma = \Sigma_c \cup \Sigma_{uc}, \Sigma_c \cap \Sigma_{uc} = \emptyset \) where \( \Sigma_c \) is a controllable event set. \( \Sigma_{uc} = \Sigma \setminus \Sigma_c \). Let \( \Sigma^* \) denote a Kleene closure, \( \varepsilon \) is an empty string. \( \delta \) is easily extended on strings from \( \Sigma^* \). Language generated by \( G \) is \( L(G) = \{ w : w \in \Sigma^* \text{ and } \delta(w, q_0) \text{ is defined} \} \), while language marked by \( G \) is \( L_m(G) = \{ w : w \in L(G) \text{ and } \delta(w, q_0) \in Q_m \} \). 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 \( G \) represented by a supervisor [20]. The supervisor switches control patterns in such a way that the supervised discrete event system achieves a control objective described by some regular language \( K \). As a rule, a supervisor is designed as another automaton, and the desired control is implemented by constructing the parallel composition of the automata of the plant and the supervisor. Let a language generated by the closed-looped behavior of the plant and the supervisor be denoted by \( L(J/G) \). 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.

In this paper we focus on the blocking issue that may arise in the system when the marked states are distinguished from the set of all states. Marked strings usually denote completed tasks in the system, such as finished assembly process on the manufacturing factory or prescribed
actions sequence, forming a mission for mobile robot. Given \( L_m(\gamma) \), the language marked by the supervisor is defined as \( L_m(\gamma/\gamma) = L(\gamma/\gamma) \cap L_m(\gamma) \), i.e. it is a set of marked strings of the system which survived in the presence of the supervisor.

**Definition 1.** [20] A supervisor \( \gamma \) is called nonblocking if \( L(\gamma/\gamma) = L_m(\gamma/\gamma) \).

Nonblockingness means that every prefix of the string of \( L(\gamma/\gamma) \) may be completed to a marked string.

**Nonblocking supervisory control problem (NSCP).** Given the DES \( \gamma \), specification \( K \), design a minimally restricting supervisor \( \gamma \), such that \( L(\gamma/\gamma) = K \) and \( L_m(\gamma/\gamma) = K \).

**Definition 2.** \( K \) is controllable (with respect to \( L(\gamma) \) and \( \Sigma_{uc} \)) if \( K \Sigma_{uc} \cap L(\gamma) \subseteq K \).

If \( K \) represents the admissible behaviour of the system, \( K \) is controllable if occurring of any uncontrolled event after a prefix of a word from \( K \) leads to a word from \( K \), i.e., still admissible. Only controllable languages may be exactly achieved by the joint behaviour of the plant and a supervisor.

**Definition 3.** A language \( K \) is called \( L_m(\gamma)-\)closed if \( K = K \cap L_m(\gamma) \).

**Controllability theorem** (the existence criterion of the solution to NSCP). Given the DES \( \gamma \), specification \( K \), a nonblocking supervisor \( \gamma \) as a solution of NSCP exists iff

(i) \( K \) is controllable;

(ii) \( K \) is \( L_m(\gamma)\)-closed.

Considering the second requirement of the Controllability theorem, it should be noted that the inclusion \( K \subseteq \overline{K} \cap L_m(\gamma) \) holds if \( K \) is chosen as \( K \subseteq L_m(\gamma) \). Thus, the opposite inclusion \( \overline{K} \cap L_m(\gamma) \subseteq K \) should be ensured to design a nonblocking supervisor. Insofar \( L_m(\gamma)\)-closedness is achieved with natural choice \( K \subseteq L_m(\gamma) \), to build a supervisor, it is required to ensure the fulfillment of the controllability property of \( K \), including for languages that are not prefix-closed.

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. Let a set \( \mathcal{C}_m(K) = \{ L \subseteq K : \mathcal{T} \Sigma_{uc} \cap L(\gamma) \subseteq \mathcal{T} \} \) be a set of all controllable sublanguages of a given language \( K \) [21]. 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. Following [21], we denote this language \( K^{\uparrow C} \):

\[
K^{\uparrow C} := \bigcup_{L \in \mathcal{C}_m(K)} L.
\]

Note that in the worst case \( K^{\uparrow C} = \emptyset \), while \( K^{\uparrow C} = K \) if \( K \) is controllable.

**Example 1.** Consider a DES model, depicted in figure 1, of autonomous underwater vehicle (AUV) functioning regimes switching while following a predefined reference path in some unknown area. Here \( Q = \{ PF, OA, PC, NP \} \) where the state \( PF \) corresponds to the main mode of AUV, path following. \( OA \) is the obstacle avoidance mode which is on when it is impossible to follow the path without collision with an obstacle. The state \( PC \) corresponds to the path calculating mode. It switches on after obstacle avoiding and at the other moments. \( NP \) corresponds to navigation to reference path. Set \( \Sigma = \{ a, b, c, d, e, f, g, h, i, j \} \) includes all events that appear in the model. They correspond to possible modes switching due to environmental or navigational circumstances. For example, after avoiding an obstacle, there are three possible variants to continue functioning: continue following the reference path, turn on regime of navigating to the previous path, or calculate a new trajectory. Let \( \Sigma_{uc} = \{ a, d \} \), that is, obstacle avoiding manoeuvre is uncontrollable. All events are observable. \( Q_m = \{ PF, PC \} \) as a mobile robot may finish its movement if path calculation results in an unrealizable trajectory.
Let the specification language $K$ be a language marked by the automaton $H$ in Fig. 2. This specification means that computation of a new path becomes possible only after obstacle avoiding manoeuvre which required navigating to previous path afterwards. Such situation may lead to growing navigational errors so in this case it worth spending resources on computations. In other case, computations are not performed. Moreover, before occurring of the event $c$, events $b$, $i$ and $d$ are forbidden. Note that condition $K \subseteq L_m(G)$ holds. Observe that such $K$ is not controllable. Indeed, the uncontrollable event $d$ is forbidden before occurring of $c$ by the specification but can not be prevented from occurring.

Let the specification language $K$ be a language marked by the automaton $H$ in Fig. 2. This specification means that computation of a new path becomes possible only after obstacle avoiding manoeuvre which required navigating to previous path afterwards. Such situation may lead to growing navigational errors so in this case it worth spending resources on computations. In other case, computations are not performed. Moreover, before occurring of the event $c$, events $b$, $i$ and $d$ are forbidden. Note that condition $K \subseteq L_m(G)$ holds. Observe that such $K$ is not controllable. Indeed, the uncontrollable event $d$ is forbidden before occurring of $c$ by the specification but can not be prevented from occurring.

3. Representation of DES in the calculus of PCFs

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 $\Phi = \{F_1, \ldots, F_n\}$ be a set of FOFs. The following formulas $\forall x_1 \ldots \forall x_k(A_1 \& \ldots \& A_m) \rightarrow (F_1 \lor \ldots \lor F_n)$ and $\exists x_1 \ldots \exists x_k(A_1 \& \ldots \& A_m) \& (F_1 \& \ldots \& F_n)$ are denoted as $\forall x_1, \ldots, x_k A_1, \ldots, A_m \{F_1, \ldots, F_n\}$ and $\exists x_1, \ldots, x_k A_1, \ldots, A_m \{F_1, \ldots, F_n\}$. They can be abbreviated as $\forall_X A \Phi$ and $\exists_X A \Phi$ respectively, keeping in mind that the $\forall$-quantifier corresponds to $\rightarrow \Phi^\lor$, where $\Phi^\lor$ means disjunction of all the formulas from $\Phi$, and $\exists$-quantifier corresponds to $\& \Phi^\land$, where $\Phi^\land$ means conjunction of all the formulas from $\Phi$. Any of sets $X$, $A$, $\Phi$ may be empty, and in this case they could be omitted in formulas.
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 \( P = \forall \{ F_1, \ldots, F_n \} \) and \( F_i = \exists X_i B_i \{ Q_{i1}, \ldots, Q_{im} \}, \ i = 1, n \), then \( F_i \) is called base subformula of \( P \), \( B_i \) is called base of facts or just base, \( Q_{ij} \) are called question subformulas, and roots of question subformulas are called questions to the base \( B_i, i = 1, n \). A question of a form \( \forall X \ A \) (without any children) is called goal question.

Example 2. The general form of PCF representing some automaton consists of the single base \( B = \{ I(S), L(\varepsilon, S), L^m(\varepsilon, S), \delta(S_1^i, \sigma^i, S_2^j), \delta^m(S_1^i, \sigma^i, S_2^j), \Sigma_{\text{uc}}(\sigma^i) \}, \ i = 1, n, \ j \in 1, \ldots, k, \ n \) is the number of transitions, \( k \) is the number of events, and two questions shown in Fig. 3. Here \( L(s, S) \) denotes “\( s \) is a current sequence of events in the state \( S \)” and \( L^m(s, S) \) 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 \( \delta(S_1^i, \sigma, S_2^j) \) will be interpreted as the automaton transition from a state \( S_1 \) to a state \( S_2 \) with an event \( \sigma \). If the target state of a transition is marked, then delta atoms with an index \( m \) are used, i.e., \( \delta^m(S_1^i, \sigma^i, S_2^j) \) 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 \( \Sigma_{\text{c}}(\_ ) \) and \( \Sigma_{\text{uc}}(\_ ) \), respectively. As usual, the function symbol “\( \cdot \)” denotes strings concatenation, and the “\( \varepsilon \)” symbol corresponds to the empty string.

Definition 4. [Answer] Consider some base subformula \( \exists X A \Psi \) of a PCF. A question of the subformula \( Q = \forall Y B \Phi, Q \in \Psi \) has an answer \( \theta \) if and only if \( \theta \) is a substitution \( Y \rightarrow H^\infty \cup X \) and \( B\theta \subseteq A \), where \( H^\infty \) is Herbrand universe based on constant and function symbols that occur in the corresponding base subformula.

Definition 5. Let \( P_1 = \exists X A \Psi \) and \( P_2 = \exists Y B \Phi \), then \( \text{merge}(P_1, P_2) = \exists X, Y A \cup B \Psi \cup \Phi \).

Definition 6. Consider some base subformula \( B = \exists X A \Psi \). A question subformula \( Q \in \Psi \) has the form \( \forall Y D \{P_1, \ldots, P_n\} \), where \( P_i = \exists Z, C_i, \Gamma_i, i = 1, n \), then \( \text{split}(B, Q) = \{\text{merge}(B, P'_1), \ldots, \text{merge}(B, P'_n)\} \), where \( \cdot' \) is a variable renaming operator. We say that \( B \) is split by \( Q \), and \( \text{split}(B, Q) \) is the result of the split of \( B \). Obviously, \( \text{split}(B, \forall Y D) = \emptyset \).

Definition 7. [The inference rule \( \omega \)] Consider some PCF \( F = \forall \Phi \). If there exists a base subformula \( B = \exists X A \Psi, B \in \Phi \) and there exists a question subformula \( Q \in \Psi \), and the question of \( Q \) has an answer \( \theta \) to \( B \), then \( \omega(F) = \forall \Phi \setminus \{B\} \cup \text{split}(B, Q\theta) \).

Note, that if the set \( \Phi \) becomes empty after applying the \( \omega \) rule, and the PCF becomes just \( \forall \), then the negation of the original statement is unsatisfiable; therefore, the statement itself is true.

Example 3. The generator \( G \) depicted in Fig. 1 may be represented by the PCF shown in Fig. 3 with \( B \) being the conjunct \( \{I(PF), L(\varepsilon, PF), \Sigma_{\text{uc}}(b), \ldots, \Sigma_{\text{uc}}(a), \Sigma_{\text{uc}}(d), \delta(PF, a, OA), \ldots, \delta^m(PF, f, PC), \ldots\} \). 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.

Note that a realization of a generator via PCF becomes possible due to the endlessness of the process of inference search. Obviously, in the PCF calculus, the absence of the goal question in the formula means its validity, i.e. the inference can only terminate with the exhaustion of all
substitutions necessary to apply the inference rules. Our formalization is structured in such a way that functional symbols are used to extend languages, which makes it impossible to exhaust all substitutions.

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. Computation of the language $K^{\uparrow C}$

With the help of the PCF calculus, we can construct $K^{\uparrow C}$ during the checking controllability of $K$, thus combining these two procedures. As a rule, controllability of $K$ is verified by comparing uncontrollable transitions in automata $G$ and $H$. We proceed in the same way. The inference is applied to the product $G \times H$ of $G$ and $H$ that merges the corresponding states of the automata $G$ and $H$. Due to the space limitation, the PCF-based construction of product of automata is omitted and may be found in [9].

Example 4. Fig. 4 represents the product of the automaton $G$ shown in Fig. 1 with the automaton $H$ shown in Fig. 2.

\[\text{Figure 4. The product of } G \text{ and } H.\]

To construct $K^{\uparrow C}$, we use PCF $F_S$ with the base $B_S = B_{G \times H} \cup \{\delta_3(S_{11}^{h_1}, S_{21}^{h_2}, \sigma^k, S_{12}^{h_1} \cdot S_{22}^{h_2})\}$, $k = 1, n_3$, that contains predicate descriptions of states and transitions of $G$, $H$, and transitions of the function $\delta_3$ of $G \times H$. The questions to the base of $F_S$ are depicted in Fig. 5 and have the following meaning:

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

$Q_2^S$ adds to the base the atom $Chk(p_1, \sigma, 0)$ which means that there is an uncontrollable transition $\sigma$ from the state $p_1$ of the automaton $G$. If there is no such transition from the state $p_1$ of $H$, $K$ is not controllable. This is checked by the next rule.
Given a specification language $K$, to construct $K^{\uparrow C}$ 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 $Q_5^S$. 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 $\text{Del}(\_)$ 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. 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

$$Q_3^S : \forall s_1, s_2, p_1, p_2, \sigma I_1(s_1), I_3(p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) \rightarrow \exists N(s_1, p_1)$$

$$Q_4^S : \forall s_1, s_2, p_1, \sigma N(s_1, p_1), \delta_1(s_1, \sigma, s_2), E_{uc}(\sigma) \rightarrow \exists Chk(p_1, \sigma, 0)$$

$$Q_5^S : \forall p_1, p_2, \sigma Chk^*(p_1, \sigma, 0), \delta_3(p_1, \sigma, p_2) \rightarrow \exists Chk(p_1, \sigma, 1)$$

$$Q_6^S : \forall s_1, s_2, p_1, p_2, \sigma N(s_1, p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) \rightarrow \exists N(s_2, p_2)$$

$$Q_7^S : \forall p_1, p_2, \sigma Chk(p_1, \sigma, 0) \rightarrow \exists Del(p_1)$$

$$Q_8^S : \forall s_1, s_2, p_1, p_2, \sigma N(s_1, p_1), \delta_1(s_1, \sigma, s_2), \delta_3(p_1, \sigma, p_2) \rightarrow \exists N(s_2, p_2)$$

$$Q_9^S : \forall p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{10}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{11}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{12}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{13}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{14}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{15}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{16}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{17}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{18}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{19}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

$$Q_{20}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 0)$$

$$Q_{21}^S : \forall s_1, s_2, p_1, p_2, \sigma Chk(p_1, \sigma, 1)$$

Figure 5. PCF $F_S$, constructing $K^\uparrow C$. 
Consider this strategy with an example.

**Example 5.** Table 1 shows the inference of the PCF, which constructs the supremal controllable sublanguage of the specification \( K \) represented by the automaton shown in Fig. 2. It is assumed that the product automaton (Fig. 4) has been already built, and the base from which the inference starts includes the subset of atoms \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \sigma_3(PF \cdot 1, a, OA \cdot 2), \ldots \} \). For convenience, in the table, we omit the superscript \( S \) in questions names.

![Image](https://via.placeholder.com/150)

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

| Q       | Base atoms used                                                                 | Substitution                                                                 | Atoms added                                                                 | Step Result                                                                 |
|---------|--------------------------------------------------------------------------------|------------------------------------------------------------------------------|----------------------------------------------------------------------------|----------------------------------------------------------------------------|
| 1       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( NP(PF, PF \cdot 1) \)                                                  | The pair of states \( PF \) and \( PF \cdot 1 \) in the automata \( G \) and \( G \times H \) is to be checked for violation of controllability |
| \( Q_1 \) | \( \delta_1(PF, a, OA), \delta_2(PF \cdot 1, a, OA \cdot 2) \)               | \( \{ p_1 \rightarrow PF \cdot 1, p_2 \rightarrow OA \cdot 2, \} \)         | \( OA \cdot 2, \sigma \rightarrow a \)                                   |                                                                             |
| 2       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( Chk(PF \cdot 1, a, 0) \)                                               | In automaton \( G \) an uncontrolled transition labeled by the event \( a \) was found, i.e., the check of the corresponding transition in the automaton \( G \times H \) is necessary |
| \( Q_2 \) | \( \delta_1(PF, a, OA), \delta_2(PF \cdot 1, a, OA \cdot 2) \)               | \( \{ p_1 \rightarrow PF \cdot 1, \sigma \rightarrow a \} \)               | \( N(PP, PP \cdot 1) \)                                                  | Analagetical to the first step of this inference                           |
| 3       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( N(OA, OA \cdot 4) \)                                                  | The check assigned in the step 2 is passed, i.e., violation of controllability is not found |
| \( Q_3 \) | \( \delta_1(PF \cdot 1, a, OA \cdot 4) \)                                    | \( \{ p_1 \rightarrow PF \cdot 1, \sigma \rightarrow a \} \)               | \( Chk(PF \cdot 1, a, 1) \)                                               | Analagetical to the first step of this inference                           |
| 4       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( N(PP, PP \cdot 1) \)                                                  | In automaton \( G \) an uncontrolled transition labeled by the event \( d \) was found, i.e., the check of the corresponding transition in the automaton \( G \times H \) is necessary |
| \( Q_4 \) | \( \delta_1(PF \cdot 1, a, OA \cdot 4) \)                                    | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( N(PP, PP \cdot 1) \)                                                  | In automaton \( G \) an uncontrolled transition labeled by the event \( d \) was found, i.e., the check of the corresponding transition in the automaton \( G \times H \) is necessary |
| 5       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( Chk(PF \cdot 1, a, 1) \)                                               | Analagetical to the first step of this inference                           |
| \( Q_5 \) | \( \delta_1(PF \cdot 1, a, OA \cdot 4) \)                                    | \( \{ p_1 \rightarrow PF \cdot 1, \sigma \rightarrow a \} \)               | \( N(OA, OA \cdot 4) \)                                                  | The check assigned in the step 2 is passed, i.e., violation of controllability is not found |
| 6       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( NP(PP, PP \cdot 2) \)                                                 | Analagetical to the first step of this inference                           |
| \( Q_6 \) | \( \delta_1(PF \cdot 1, a, OA \cdot 4) \)                                    | \( \{ p_1 \rightarrow PF \cdot 1, \sigma \rightarrow a \} \)               | \( NP(PP, PP \cdot 2) \)                                                 | In automaton \( G \) an uncontrolled transition labeled by the event \( d \) was found, i.e., the check of the corresponding transition in the automaton \( G \times H \) is necessary |
| 7       | \( \{ \sigma_1(PF), \sigma_2(PF \cdot 1), \} \)                              | \( \{ s_1 \rightarrow PF, s_2 \rightarrow OA, \} \)                          | \( Del(PF \cdot 2) \)                                                   | The check assigned at the previous step was failed; therefore, the specification corresponding to the automaton \( H \) 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.

**Diagram**: Two diagrams illustrating the inference process with transitions labeled according to the table.
5. Supervisory control implementation

Once the supremal controllable sublanguage for the given uncontrollable specification was found, a solution of the problem of nonblocking supervisory control may be constructed. Employing the fact that if \( K \subseteq L_m(G) \) is \( L_m(G) \)-closed, then so is \( K^{\uparrow_C} \), the solution of NSCP is to choose \( J \) such that \( L(J/G) = K^{\uparrow_C} \) and \( L_m(J/G) = K^{\uparrow_C} \) [21].

Since a closed-looped behaviour of the plant and the supervisor is usually realized by the parallel composition of the corresponding automata, i.e., \( L(J/G) = L(J||G) \), the supervisor may be realized by taking a recognizer of \( K^{\uparrow_C} \). It may be extracted from the base of PCF \( F_S \) after the inference termination (atoms \( \delta_2(_,_,_) \)).

In PCF formalization, the joint work of the system and the supervisor is carried out using the PCF \( F_{SC} \) below.

\[
\exists B, B_{SC} \quad \exists \sigma, s, \sigma', s' L(\sigma, S), \delta^1(s, \sigma', s'), \delta^2(s, \sigma', s') \quad \exists L(\sigma \cdot \sigma', s')
\]

\[
\exists \sigma, s, \sigma', s' L(\sigma, S), \delta^1_m(s, \sigma', s'), \delta^2_m(s, \sigma', s') \quad \exists L_m(\sigma \cdot \sigma', s')
\]

Here bases \( B \) and \( B_{SC} \) are the sets of atoms corresponding to the transitions of the plant and the supervisor, correspondingly. Questions of \( F_{SC} \) may be interpreted as follows. If the system is at the state \( s \) and an event \( \sigma \) occurs, then according to the \( \delta^2 \), the system is switched to the specified state \( s' \), and \( \sigma \) is added to the current chain of events stored as the first argument of the predicate \( L(_,_,_) \). That is, any transition corresponding to the language \( L(G) \) (implemented by the atom \( \delta^1(_,_,_) \)), is simultaneously traced 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., atom \( \delta^2(_,_,_) \) limits the answers that could be generated in the presence of the atom \( \delta^1(_,_,_) \) 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. 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.

Conclusion

In this paper, we presented the PCF calculus-based approach to solving the problem of nonblocking supervisory control for DES as a generator of formal language. Given uncontrollable specification on DES behavior, the way of supremal controllable sublanguage constructing is illustrated.

Implemented in Rust language, the prover utilized to solve problems above is specialized for PCFs without unconfined variables (an unconfined variable is a variable controlled by an universal quantifier, which does not enter in its type condition). In the prover, an approach is implemented that allows for non-monotonic logical inference, i.e. at the next step of the inference, a command to remove a fact from the base may be issued. In addition, strategies have been implemented that increase the efficiency of the inference search, such as saving memory due to sharing common parts of the PCF and context usage and quick backtracking to any previous state of the inference. A strategy for selecting a question at the next step of the inference may be separately configured by the user to regulate the priority of selecting questions. To remove atoms from the base employed a mechanism for executing additional commands in case of a successful answer to the question. The removal of atoms is implemented as setting the removal mark so
that one can return to the previous state of the inference and return the removed atom back to the base. The prover was tested on the problems from the tptp.org library [22]. Problems of automated theorem proving software in the PCF calculus design and implementation are briefly discussed in [23].

The authors’ future study supposes design modular and decentralized nonblocking supervisors via the PCF calculus. Results obtained will be embedded at the different levels of the hierarchical control system for mobile robots and robot groups.

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] Mohajerani S, Malik R and Fabian M 2014 IEEE Transactions on Automatic Control 59 150–162 ISSN 1558-2523
[3] Mohajerani S, Malik R and Fabian M 2016 Discrete Event Dynamic Systems 26 33–84
[4] Vassiliev S N 1990 The Journal of Logic Programming 9 235–266
[5] Zherlov A K, Vassiliev S N, Fedosov E A and Fedunov B E 2000 Intelligent control of dynamic systems (Moscow: Fizmatlit) in Russian
[6] Davydov A, Larionov A and Cherkashin E 2011 Automatic Control and Computer Sciences 45 402–407
[7] 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
[8] Davydov A, Larionov A and Nagul N 2018 AIP Conference Proceedings 2046 020021
[9] 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
[10] Bychkov I, Davydov A, Nagul N and Ul’yanov S 2018 Information Technology in Industry 6 20–26
[11] 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
[12] Jayasiri A, Mann G and Gosine R 2011 IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41 1224 – 1238 ISSN 1083-4419
[13] Torrico C R, Leal A B and Watanabe A T 2016 IFAC-Papers OnLine 49 240 – 245 ISSN 2405-8963
[14] Dai X, Jiang L and Zhao Y 2016 Applied Intelligence 45 18–29 ISSN 1573-7497
[15] Tsatslanis A, Yalcin A and Valavanis K P 2012 Robotics 30 721—730
[16] Hill R C and Lafortune S 2017 2017 American Control Conference (ACC) pp 3840–3847 ISSN 2378-5861
[17] 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
[18] Lopes Y K, Trenkwalder S M, Leal A B, Dodd T J and Groß R 2016 Swarm Intelligence 10 65–97 ISSN 1935-3820
[19] Mendiburu F J, Morris M R A and Lima A 2016 IEEE International Conference on Automatica (ICA-ACCA) Oct. 2016 1 – 7 ISSN 2405-8963
[20] Ramadge P J and Wonham W M 1987 SIAM Journal on Control and Optimization 25 206–230
[21] Cassandras C G and Lafortune S 2008 Introduction to Discrete Event Systems (Springer US)
[22] Sutcliffe G 2017 Journal of Automated Reasoning 59 483–502
[23] Larionov A, Davydov A and Cherkashin E 2013 International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija 2013 1023–1028