The construction of controllable sublanguage of specification for DES via PCFs based inference

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 considers how methods of a logical inference search in the calculus of positively constructed formulas may be applied to represent and study discrete event systems. The formalisms of discrete event systems and positively constructed formulas are briefly described. A method for constructing a product of automata using constructive inference in the positively constructed formulas calculus is proposed. Based on the given specification on the behaviour of the system, a method for constructing a supremal controllable sublanguage of the specification is presented.

1. Introduction

In many applications, logical tools proved to be a mighty instrument for solving complex problems. For example, automated theorem proving (ATP) is used in various areas, including verification problems ([1, 2]), supporting research and education ([3]), and robotics ([4, 5]). In this paper, we show how ATP based on the *calculus of positively constructed formulas* (PCF) helps to solve one of the problems arising in supervisory control theory (SCT) for discrete event systems (DES).

The term *discrete event system* appeared in the early 1980s and denoted a wide class of systems with discrete states whose evolution is driven by the occurrence of some discrete events. The most common and convenient way to represent DES is finite state automata. In the seminal work of P. Ramage and W. Wonham [6] the theory for DES control was established. Some events of automata-based DES are supposed to be prohibited from occurring to restrict system behaviour within constraints given by some specification. The means of such control is called a supervisor. The problem of designing a proper supervisor for fully or partially observed DES is the main problem of SCT. Controllability of specification is a crucial property which must be verified to start building a supervisor. It may be compared to the safety property of the controlled system: no uncontrolled event which is not allowed by specification may occur. In the case when the specification is not controllable, controllable sublanguage of this specification may be found as a new admissible specification. Although there is a well-known algorithm of building controllable sublanguage (see, e.g. [7]), we suggest a new approach to this problem, based on the ATP in the PCF calculus. This approach is a part of a new method of dealing with

DES, developed in authors' previous works [8, 9, 10]. It allows one to solve SCT problems more effectively via the use of additional information accumulated in the process of the inference.

The calculus of PCFs was introduced in [11] and further developed in [12] and [13] as a complete method for ATP with functional symbols. The most important features of the PCF calculus and its implementation in the prover are the following: 1) large block data structures for representing formulas and inference rules; 2) no need of removing existence quantifiers by the skolemization procedure, which increases the complexity of the formula; 3) compatibility with specific application heuristics, as well as with general inference control heuristics; 4) logical inference is easy to read, which helps to find errors in formalization more quickly; 5) support for equality; 6) modifiability of semantics, support for non-classical logics. A detailed discussion of characteristics and features of the calculus may be found in [13].

The rest of the paper is organized as follows. The introductory notes on DES and SCT are provided in the next section. The third section contains a brief description of PCFs and the PCF calculus. In Section 4, a way of automata-based DES representation as PCF is described. In Section 5, a method for a supremal controllable sublanguage of a given language construction is presented. Some remarks on the results obtained are made in Conclusion.

2. Supervisory control of DES

Consider discrete-event system (DES) in the form of a generator of a formal language [6] $\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$, also called a *plant* in the automatic control theory. 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. Let Σ^* denote the set of all strings over Σ, including the empty string ε. 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 $\overline{K} = K$.

Suppose some events of $\mathcal{G}$ are controllable, i.e. may be prohibited from occurring. Let Σ_c be a controllable event set, $\Sigma_{uc} = \Sigma \setminus \Sigma_c$, $\Sigma_c \cap \Sigma_{uc} = \emptyset$. The Ramadge–Wonham supervisory control framework assumes the existence of a means of control $\mathcal{G}$ presented by a *supervisor* [6]. 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. Denote $L(\mathcal{J}/\mathcal{G})$ a language generated by the closed-looped behavior of the plant and the supervisor. 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 plays a crucial role in characterizing those languages that can be generated by the closed-loop structure plant–supervisor.

Definition 1 [6] K is controllable (with respect to $L(\mathcal{G})$ and Σ_{uc}) if

$$\overline{K}\Sigma_{uc} \cap L(\mathcal{G}) \subseteq \overline{K}.$$

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 supervisor. 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 specification and the plant. The controllability checking using PCFs is illustrated in Section 5 below.

If the specification under consideration happen to be not controllable, a controllable part of it may be used for designing a supervisor. Let [7]

$$\mathfrak{C}_{in}(K) = \{L \subseteq K : \overline{L}\Sigma_{uc} \cap L(\mathcal{G}) \subseteq \overline{L}\}$$

be a set of all controllable sublanguages of a given language K. 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 [7], denote this language $K^{\uparrow C}$. To obtain $K^{\uparrow C}$ for a prefix-closed K, a quotient structure of automata is used.

Definition 2. [7] The quotient operation for languages $L_1, L_2 \subseteq \Sigma^*$ is defined as

$$L_1/L_2 := \{s \in \Sigma^* : (\exists t \in L_2)[st \in L_1]\}.$$

The exact formula for computing $K^{\uparrow C}$ for a prefix-closed K is then as follows:

$$K^{\uparrow C} = K \setminus [(L(\mathcal{G}) \setminus K)/\Sigma^*_{uc}]|\Sigma^*.$$

This formula means that if the string with the prefix belonging to K has an uncontrollable continuation leading it out of K, it should be excluded from K. In the next sections, we construct $K^{\uparrow C}$ with the help of logical inference in the PCF calculus using the PCF representation of automata.

3. The calculus of positively constructed formulas

In the process of reasoning in the natural language, one often proves some statement by refuting the contradictory statement. We intend to proceed similarly. The main idea of the PCF calculus is to refute the negation of a formula $\mathcal{F}$ to prove that $\mathcal{F}$ is valid.

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.), possibly 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 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 formula formalization. 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 3. 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,

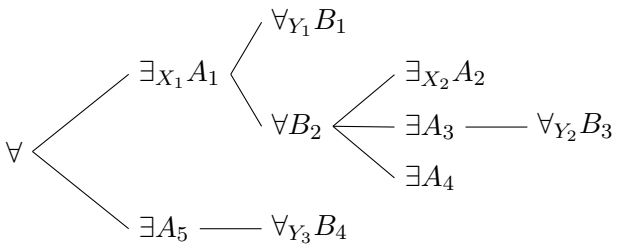
(vi) There are only PCFs of a form $\exists$-PCF and $\forall$-PCF.

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.

Definition 4. [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 5. 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 6. 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 7. [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 when the set Φ becomes empty after applying an ω rule, and the PCF becomes just $\forall$, then it can be concluded that the negation of an original formula is unsatisfiable.

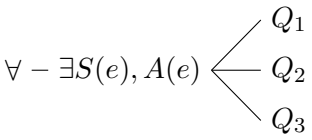
Any finite sequence of PCFs $\mathcal{F}, \omega\mathcal{F}, \omega^2\mathcal{F}, \ldots, \omega^n\mathcal{F}$, where $\omega^s\mathcal{F} = \omega(\omega^{s-1}\mathcal{F}), \omega^1 = \omega, \omega^n\mathcal{F} = \forall$, is called *an inference* of $\mathcal{F}$ in PCF calculus (with the axiom $\forall$).

Suppose that a search strategy does not use repeated application of ω to a question with the same θ (question-answering method of automated inference).

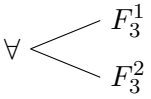
Example 1. [A refutation in PCFs]. Consider the PCF $\mathcal{F}_1$

$$\forall \cdot \exists S(e) \begin{cases} \forall x\ S(x) \text{ - } \exists A(x) \\ \forall x, y\ C(x, y) \\ \forall x\ A(x) \begin{cases} \exists y\ C(y, f(x)) \\ \exists - \forall x\ S(x), A(x)\text{-}\exists C(x, f(x)) \end{cases} \end{cases}$$

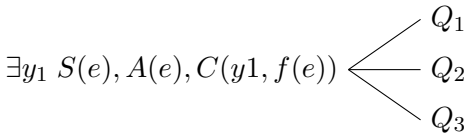
At the first step of the inference there is only one answer $\{x \to e\}$ to the first question (top to bottom numbering, name it Q_1). After applying rule ω with this answer to the only base of $\mathcal{F}_1$, the formula will be converted to the following form, $\mathcal{F}_2$:



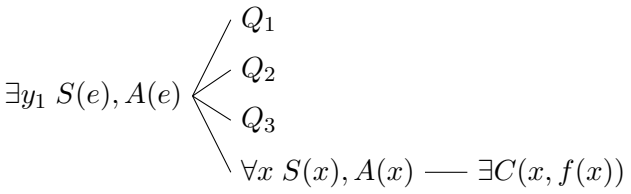
At the second step there is also only one answer $\{x \to e\}$ to question Q_3. After applying ω with this answer, formula is split, because Q_3 has disjunctive branching. The formula will have the following form, $\mathcal{F}_3$:



where F_3^1 is:



and F_3^2 is:



At the third step the first base can be refuted by answering on Q_2 (the goal question) with $\{x \to y_1; y \to f(e)\}$. The refuted base (and its whole base subformula) is to be deleted from the list of the base subformulas.

At the fourth step there is the answer $\{x \to e, y \to e\}$ to the fourth new question of the base F_3^2 which adds the atom $C(e, f(e))$ to it.

At the fifth step, the only base can be refuted by answering on Q_2 (the goal question) with the answer $\{x \to e; y \to f(e)\}$, finishing refutation because all the bases were refuted.

One of the essential features of the calculus of PCFs which will be used later is that we can build *a nonmonotonic* 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 always ends.

4. PCF representation of automata

To formalize a finite state machine as a generator of a formal language, the following predicates will be used. Let $L(s, S)$ denote "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 $\delta(S_1, \sigma, S_2)$ 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.

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

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

Figure 1. General form of PCF representation of some automaton.

Suppose that behaviour of $\mathcal{G}$ should be constrained within the specification language K, and let automaton $\mathcal{H}$ generates, or recognizes, $\overline{K}$. In order to check the controllability of K, we first build a product of automata $\mathcal{G}$ and $\mathcal{H}$, on the basis of which further inference will be arranged. The PCF representation of constructing the product of two automata may also be written using a single formula where indices distinguish atoms corresponding to different automata. The inferences will be based on the reasoning about the structures of automata which are involved in the product. The PCF $\mathcal{F}_{\mathcal{G}\times\mathcal{H}}$ constructing the product of automata will consist 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 first automaton and transitions δ_2, $k = \overline{1, n_2}$, of the second one. The questions of $\mathcal{F}_{\mathcal{G}\times\mathcal{H}}$ are as follows:
$1 : \forall s, p\ I_1(s), I_2(p) - \exists I_3(s \cdot p)$
$2 : \forall s, p\ I_1(s), I_2(p) - \exists \delta_3(\xi \cdot \rho, \varepsilon, s \cdot p)$
$3 : \forall \sigma, s_1, p_1, s_2, p_2, \sigma', s'\ \delta_1(s_1, \sigma, s_2), \delta_2(p_1, \sigma, p_2), \delta_3(s', \sigma', s_1 \cdot p_1) - \exists \delta_3(s_1 \cdot p_1, \sigma, s_2 \cdot p_2)$

Index 1 in the subscripts of atoms of this formula corresponds to the atoms of the automaton $\mathcal{G}$, 2 to the $\mathcal{H}$, and 3 to the atoms corresponding to $\mathcal{G} \times \mathcal{H}$. 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.

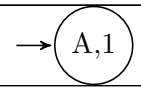
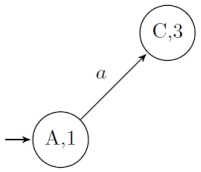
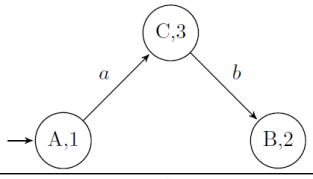
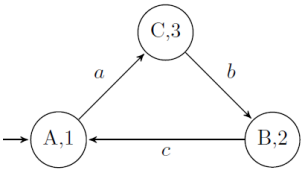
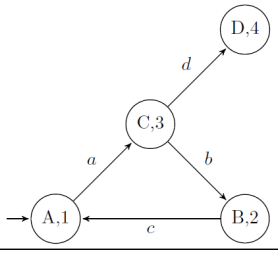
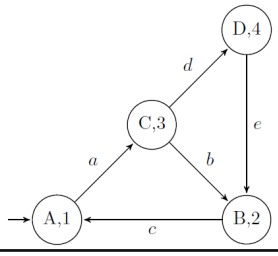
Example 2. [Product construction] The steps of the inference that constructs the automaton depicted in figure 4 is shown in table 1. It is assumed that the base contains atoms corresponding to transitions of the automata depicted in figure 2 and figure 3.

Table 1. Constructive inference for the product of two automata.

Q	Base atoms	Substitution	Atoms added	Result
1	$I_1(A), I_2(1)$	$\{s \to A,\ p \to 1\}$	$I_3(A \cdot 1)$	→ (A,1)

Continued on the next page

Table 1 – continued from the previous page

Q	Base atoms	Substitution	Atoms added	Result
2	$I_1(A), I_2(1)$	$\{s \to A, p \to 1\}$	$\delta^3(\xi \cdot \rho, \varepsilon, A \cdot 1)$	
3	$\delta^3(xi \cdot \rho, \varepsilon, A \cdot 1)$, $\delta^1(A, a, C)$, $\delta^2(1, a, 3)$	$\{s_1 \to A, p_1 \to 1, \sigma \to a, s_2 \to C, p_2 \to 3\}$	$\delta^3(A \cdot 1, a, C \cdot 3)$	
3	$\delta^3(A \cdot 1, a, C \cdot 3)$, $\delta^1(C, b, B)$, $\delta^2(3, b, 2)$	$\{s_1 \to C, p_1 \to 3, \sigma \to b, s_2 \to B, p_2 \to 2\}$	$\delta^3(C \cdot 3, b, B \cdot 2)$	
3	$\delta^3(C \cdot 3, b, B \cdot 2)$, $\delta^1(B, c, A)$, $\delta^2(2, c, 1)$	$\{s_1 \to B, p_1 \to 2, \sigma \to c, s_2 \to A, p_2 \to 2\}$	$\delta^3(B \cdot 2, c, A \cdot 1)$	
3	$\delta^3(A \cdot 1, a, C \cdot 3)$, $\delta^1(C, d, D)$, $\delta^2(3, d, 4)$	$\{s_1 \to C, p_1 \to 3, \sigma \to d, s_2 \to D, p_2 \to 4\}$	$\delta^3(C \cdot 3, d, D \cdot 4)$	
3	$\delta^3(C \cdot 3, d, D \cdot 4)$, $\delta^1(D, e, B)$, $\delta^2(4, e, 2)$	$\{s_1 \to D, p_1 \to 4, \sigma \to e, s_2 \to B, p_2 \to 2\}$	$\delta^3(D \cdot 4, e, B \cdot 2)$	

5. Supremal controllable sublanguage construction using a PCF inference

In this section, with the help of logical inference in the PCF calculus, on the base of $\mathcal{G} \times \mathcal{H}$, we construct $K^{\uparrow C}$ during the checking controllability of K. Note that in the worst case $K^{\uparrow C} = \emptyset$ while in the best way $K^{\uparrow C} = K$.

Suppose that the base of PCF $\mathcal{F}$ contains predicate descriptions of transitions of $\mathcal{G}$, $\mathcal{H}$ and $\mathcal{G} \times \mathcal{H}$. Let the questions to the base of $\mathcal{F}$ be as depicted in figure 5. In these rules, states which are simultaneously achieved from the states of $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ by the same event are called the *neighbouring* states and stored as the arguments of the predicate $N(\_, \_)$. The rules are to be interpreted as follows.

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. Although such s_1, p_1 are not neighboring states in our sense, $N(s_1, p_1)$ with them is necessary for further inference.

Q_2 checks if there is an uncontrollable transition from some state of the automaton corresponding to the plant. Upon successful answer to this question, the atom $Chk(p_1, \sigma, 0)$ is added to the base. The first argument of this atom denotes a state p_1 of $\mathcal{H}$, in which the

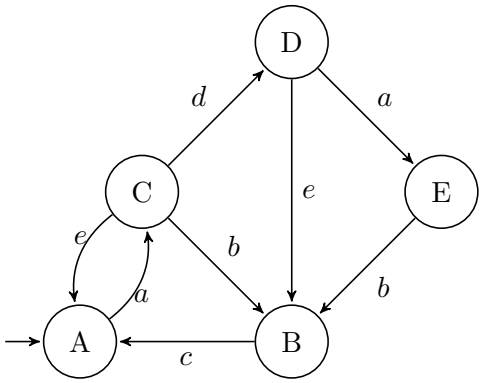


Figure 2. Generator $\mathcal{G}$.

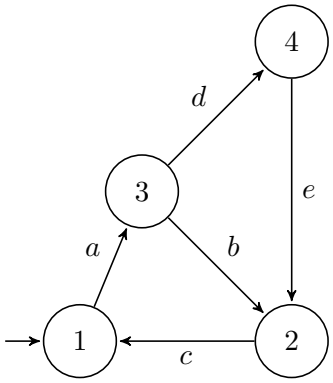


Figure 3. Recognizer $\mathcal{H}$ for the specification K.

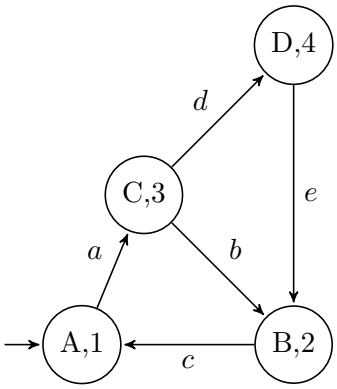


Figure 4. The product of $\mathcal{G}$ and $\mathcal{H}$.

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

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

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

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

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

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

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

Figure 5. Questions of the PCF $\mathcal{F}$.

transition labelled by the uncontrollable event σ should exist for K to be controllable. By default we suppose that the transition does not exist, so the third argument in $Chk()$ is 0 before checking by the next rule.

Q_3 is aimed to check 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 $\mathcal{H}$. With the help of * operator, we delete the atom $Chk(p_1, \sigma, 0)$ from the base.

Q_4 checks if the atom $Chk(p_1, \sigma, 0)$ is present in the base. The presence of $Chk(p_1, \sigma, 0)$ in the base means that an uncontrollable event is not allowed by the specification, i.e., K is not controllable. So the atom $Del(p_1)$ is added to the base upon a successful answer. It contains information about the state (p_1) which has the event (σ) that violate the controllability condition. This atom is further used in questions Q_6, Q_7 to delete transitions associated with the state p_1.

Q_5 adds the next checked pair of states to continue the inference search. A pair of states s_2, p_2, which are simultaneously achieved from the previously checked states of $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ by the same event, is added to the base as the arguments of the predicate $N(s_2, p_2)$.

The rules Q_6, Q_7 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 removes all transitions leaving this state and Q_7 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 to Q_7. The rules Q_1 - Q_5 check whether the specification is controllable or not. The rules Q_6, Q_7 remove transitions associated with the state in which an event occurs that violates the controllability condition. If during the application of the Q_1 - Q_5 rules the answer to question Q_4 was found, an atom $Del()$ denoting uncontrollability of K is added to the base. Then we apply the rules Q_6, Q_7, ignoring the rules from the Q_1 - Q_5, until possible substitutions for applying the inference rule to Q_6 and Q_7 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 equal to the empty string alone, then during the inference all transitions in the automata corresponding to the specification will be deleted. Consider this strategy with the example.

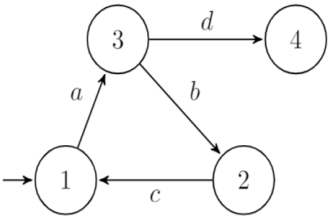
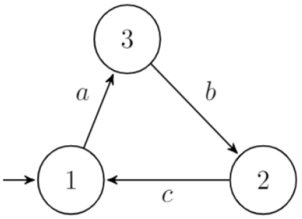
Example 3. Table 2 shows the inference of the PCF, which constructs the sublanguage of the specification represented by the automaton shown in figure 3 with respect to the plant shown in figure 2. It is assumed that the inference of the product automaton (figure 4) is already built, and the base from which the presented inference starts includes the next subset of atoms:

$$\{I_1(A), I_3(A \cdot X), \Sigma_{uc}(a),$$
$$\delta_1(A, a, C), \delta_1(C, e, A), \delta_1(B, c, A), \delta_1(C, b, B), \delta_1(C, d, D), \delta_1(D, e, B), \delta_1(D, a, E), \delta_1(E, b, B),$$
$$\delta_2(1, a, 3), \delta_2(2, c, 1), \delta_2(3, b, 2), \delta_2(3, d, 4), \delta_2(4, e, 2),$$
$$\delta_3(A \cdot 1, a, C \cdot 3), \delta_3(B \cdot 2, c, A \cdot 1), \delta_3(C \cdot 3, b, B \cdot 2), \delta_3(C \cdot 3, d, D \cdot 4), \delta_3(D \cdot 4, e, B \cdot 2)\}$$

Table 2. Constructive inference building the controllable sublanguage of the specification.

Q	Base atoms used	Substitution	Atoms added	Step Result
Q_1	$I_1(A), I_3(A \cdot 1),$ $\delta_1(A, a, C),$ $\delta_3(A \cdot 1, a, C \cdot 3)$	$\{s_1 \to A, s_2 \to C,$ $p_1 \to A \cdot 1, p_2 \to$ $C \cdot 3,$ $\sigma \to a\}$	$N(A, A \cdot 1)$	The pair of states A and $(A, 1)$ in the automata $\mathcal{G}$ and $\mathcal{G} \times \mathcal{H}$ is to be checked for violation controllability
				Continued on the next page

Table 2 – continued from the previous page

Q	Base atoms used	Substitution	Atoms added	Step Result
Q_2	$N(A, A \cdot 1)$, $\delta_1(A, a, C)$ $\Sigma_{uc}(a)$	$\{s_1 \to A, s_2 \to C,$ $p_1 \to A \cdot 1, \sigma \to a\}$	$Chk(A \cdot 1, a, 0)$	In automaton $\mathcal{G}$ an uncontrolled transition label by the event a was found, i.e. the check of the corresponding transition in the automaton $\mathcal{G} \times \mathcal{H}$ is necessary
Q_3	$Chk(A \cdot 1, a, 0)$ $(deleted)$, $\delta_3(A \cdot 1, a, C \cdot 3)$	$\{p_1 \to A \cdot 1,$ $p_2 \to C \cdot 3,$ $\sigma \to a\}$	$Chk(A \cdot 1, a, 1)$	The check assigned in the previous step is passed, i.e. violation of controllability is not found
Q_5	$N(A, A \cdot 1)$, $\delta_1(A, a, C)$, $\delta_3(A \cdot 1, a, C \cdot 3)$	$\{s_1 \to A, s_2 \to C,$ $p_1 \to A \cdot 1, p_2 \to$ $C \cdot 3, \sigma \to a\}$	$N(C, C \cdot 3)$	The same as at the first step of this inference
Q_5	$N(C, C \cdot 3)$, $\delta_1(C, b, B)$, $\delta_3(C \cdot 3, b, B \cdot 2)$	$\{s_1 \to C, s_2 \to B,$ $p_1 \to C \cdot 3, p_2 \to$ $B \cdot 2, \sigma \to b\}$	$N(B, B \cdot 2)$	...
Q_5	$N(C, C \cdot 3)$, $\delta_1(C, d, D)$, $\delta_3(C \cdot 3, d, D \cdot 4)$	$\{s_1 \to C, s_2 \to D,$ $p_1 \to C \cdot 3, p_2 \to$ $D \cdot 4, \sigma \to d\}$	$N(D, D \cdot 4)$	...
Q_2	$N(D, D \cdot 1)$, $\delta_1(D, a, E)$ $\Sigma_{uc}(a)$	$\{s_1 \to D, s_2 \to E,$ $p_1 \to D \cdot 4, \sigma \to a\}$	$Chk(D \cdot 4, a, 0)$	The same as at the second step of this inference
Q_4	$Chk(D \cdot 4, a, 0)$	$\{p_1 \to D \cdot 4,$ $\sigma \to a\}$	$Del(D \cdot 4, a)$	The check assigned in 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.				
Q_6	$Del(D \cdot 4)$, $\delta_2(4, e, 2)$, $\delta_3(D \cdot 4, e, B \cdot 2)$	$\{p_{11} \to D, p_{12} \to 4,$ $p_{21} \to B, p_{22} \to$ $2, \sigma \to e\}$	$Deleted(4, e, 2)$	
Q_7	$Del(D \cdot 4)$, $\delta_2(3, d, 4)$, $\delta_3(C \cdot 3, d, D \cdot 4)$	$\{p_{11} \to C, p_{12} \to 3,$ $p_{21} \to D, p_{22} \to$ $4, \sigma \to d\}$	$Deleted(3, d, 4)$	

Conclusion

Above we described the approach to the representation of automata, which underlie DES in the Ramadge-Wonham SCT framework, using first-order logical formulas of the PCF language. The method for building a product of automata in the process of the constructive inference of the PCF is proposed. Based on the specification on the behavior of DES, given in the form of a regular language, a method for constructing its supremal controllable sublanguage is presented. The knowledge, accumulated in the base of the PCF during the inference, may be employed for solving various problems of SCT, including modification of specifications, choosing controllable event set, and supervisors reduction. These and other issues will be studied in our future works.

Acknowledgments

The research is supported by the Russian Science Foundation (project no. 16-11-00053).

References

[1] Lockhart J, Purdy C and Wilsey P A 2017 *2017 IEEE National Aerospace and Electronics Conference (NAECON)* pp 358–361

[2] Shiraz S and Hasan O 2018 *Formal Methods: Foundations and Applications* ed Massoni T and Mousavi M R (Cham: Springer International Publishing) pp 74–89 ISBN 978-3-030-03044-5

[3] Frank M and Kreitz C 2018 *Electronic Proceedings in Theoretical Computer Science* **267** 59–69

[4] Erdem E and Patoglu V 2018 *KI - Künstliche Intelligenz* **32**

[5] Nalepa G J 2018 *Designing Robot Control Logic with Rules* (Cham: Springer International Publishing) pp 381–401 ISBN 978-3-319-66655-6

[6] Ramadge P J and Wonham W M 1987 *SIAM Journal on Control and Optimization* **25** 206–230

[7] Cassandras C G and Lafortune S 2008 *Introduction to Discrete Event Systems* (Springer US)

[8] 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

[9] 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`

[10] 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`

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

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

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