COMPLEXITY OF THE PROBLEM OF VERIFYING THE COORDINATION MECHANISM IN A SYSTEM OF SOFTWARE SUPPORT OF NETWORK COLLABORATION

N. N. Glibovets[a†] **and D. K. Hlomozda**[a‡]

The problem of verifying the coordination mechanism in a system of software support of network collaboration is considered. This problem is demonstrated to be similar to the agent verification problem. It is proved that the problem of verifying the coordination mechanism is co-complete.

Keywords: *coordination, Petri net, computational complexity, co-NP-completeness, program agent.*

We consider the problem of verification of the coordination mechanism (CM) in a collaboration system constructed within the framework of a project that was first mentioned in [1] and is devoted to the creation of a model of a software system for supporting a collaboration environment in the language of Petri nets. The network model proposed in [2] for a collaboration system is based on the structure of a collaboration environment whose components are sessions, users, shared resources, and levels. Using such levels, a protocol for accessing resources is realized. The scheme of a network model of the system is presented in Fig. 1.

The collaboration system consists of N users, M sessions, L resources, and its coordination mechanism composed of a collection of places and transitions of a Petri net that connect users U, sessions S, and resources R. In the model being considered, the CM is a block consisting of controllers of levels (to each level corresponds one resource) and a mechanism for providing mutual exclusion during creating a session. It includes the network model places described below.

1. Block for providing mutual exclusion in creating a session for each user and each session. Its structure includes the following places: "A user U_i is not the leader of a session S_j," "A user U_i is the leader of a session S_j," and "A user U_i requests permission to create a session S_j." Transitions of the Petri net are as follows: "A user U_i wants to begin a session S_j," "A user U_i wants to complete a session S_j," "A user U_i is allowed to create a session S_j," and "A user U_i is not allowed to create a session S_j" (in this case, $i \in 1\ldots N$ and $j \in 1\ldots M$).

2. Level controller. Its places are as follows: "A level F_k is unallocated," "A level F_k processes a request," "A level F_k is in wait state," "Allocate a resource R_k," "A resource R_k is allocated," "Unallocate a resource R_k," "A resource R_k is unallocated," "Eliminate a resource R_k," and "A resource R_k is eliminated" (in this case, $k \in 1\ldots L$).

3. Block for providing mutual exclusion of access of each user to each level. Its places are as follows: "A user U_i is not the holder of a level F_k," "A user U_i requests access to a level F_k," "A user U_i is the holder of a level F_k," "A level F_k is actively used by a user U_i," and "A level F_k is passively used by a user U_i." Its transitions are as follows: "A user U_i wants to access a level F_k," "Access of a user U_i to a level F_k is denied," "Access of a user U_i to a level F_k is allowed," "A level F_k is occupied by a user U_i," "A user U_i suspends the use of a level F_k," "A user U_i resumes the use of a level F_k," "A user U_i wants to eliminate a resource R_k," "A resource R_k is eliminated by a user U_i," "A user U_i wants to unallocate a level F_k," and "A user U_i has unallocated a level F_k."

In the general case, a Petri net modelling the operation of the coordination mechanism of the collaboration system being considered has

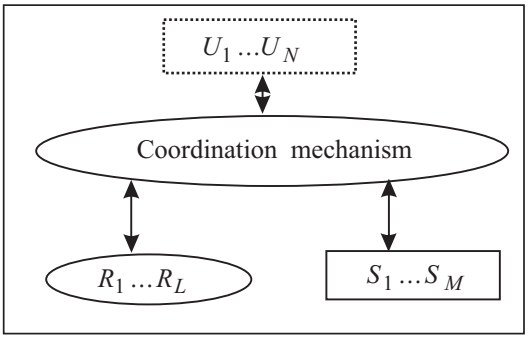

Fig. 1. Structure of a network model for a collaboration environment.

— $3 \cdot N \cdot M$ places and $4 \cdot N \cdot M$ transitions regulating mutually exclusive creation of sessions;

— $9 \cdot L$ places corresponding to level controllers;

— $5 \cdot N \cdot L$ places and $10 \cdot N \cdot L$ transitions corresponding to the connections between level controllers and users.

We will consider the problem of verification of the CM and determine its computational complexity. At an informal level, the problem of verification of the CM consists of determining the suitability of using this CM in the collaboration system or, in other words, in checking the CM for the correspondence to definite specifications regulating the principles and results of operation of the system. Before proceeding to a formal statement of this problem, we introduce the definitions listed below.

We call inadmissible the system states in which two or more users are simultaneously the leaders[1] of the same session and/or the holders of the same level. We call all the other states admissible. Then, at a formal level, the problem of verification of the CM can be stated as follows.

We have N users, L levels (to each level corresponds one resource), M sessions, and the coordination mechanism.

The solution is as follows: if, for each achievable variant of marking the Petri net corresponding to the CM, the marking admissibility condition is satisfied (i.e., a marking corresponds to an admissible state), then the CM is suitable. Otherwise, it is unsuitable.

In this statement, the problem of verification of the CM is similar to the problem of verification of agents that is considered in [3] and in which the dependence of the problem complexity on the characteristics of the environment in which an agent operates and on the complexity of the specification of a problem Ψ stated in the form of a predicate over the set of runs $\mathbf{R}$ of the agent in the system,

$$\Psi: \mathbf{R} \to \{\text{true, false}\}.$$

The run of an agent in the system is a sequence of environment states and actions of the agent that lead to a change in environment states. The CM can be associated with an agent that solves the support problem, i.e., retains one of admissible environment states.

In a collaboration environment, there are infinitely many possible runs. But, to check the reliability of this coordination mechanism, it suffices to successively determine the possibility of each user to get access to each available resource as the leader of each possible session. In this case, it makes sense to introduce an additional condition according to which only the leader of a session can access a resource since the attachment to an already activated session is not mutually exclusive. Thus, the entire set of possible runs is reduced to one finite run (the termination condition is as follows: all users as leaders of all possible sessions have used all the available resources). Such a run $\mathbf{R}'$ can be conveniently represented in the form of a sequence of definite stages. To a run stage corresponds a system state in which the use of all environmental components is maximally complete. This means that if, for example, we have $N = M = L$, then each user controls his unique concrete session and holds one concrete level. In actual fact, all other variants in which the numbers of users, sessions, and levels are different ($N = n$, $M = m$, $L = l$; $(n \neq m) \vee (l \neq n) \vee (m \neq l)$) cannot necessarily be separately considered since they are reduced to the case when $N = M = L = \min(n, m, l) = n$ (when $n \geq 2$). We explain this as follows.

[1] The leader of a session is considered to be the user who has created or activated the session, i.e., who has switched it from the state Idle to the state Active.

1. The competition between users for the control over redundant sessions and/or levels is absent and, hence, there is no need to specially coordinate their use.

2. If the number of users exceeds that of sessions, then the operation of the CM at each stage is reduced to the coordination of actions of m users (one user per session). The users who are still lacking sessions are outside of the system since, according to the additional condition, resources are available only to the leaders of sessions and, hence, there is no need to coordinate actions of these users.

3. The condition $n \geq 2$ should be satisfied since, otherwise, as a result of reduction, a degenerate case can be obtained with one user, one session, and one level and, in this case, any competition and coordination cannot be considered. If there is only one session and/or a level, then the number of users is equated to two (the least number that allows for a competition for a resource).

To construct the specification Ψ, we introduce the following auxiliary predicates:

— $\mathrm{SL}(i, j)$ assumes the value "true" ↔ there is a token at the place "A user U_i is the leader of a session S_j" in the Petri net and, otherwise, "false;"

— $\mathrm{LH}(i, k)$ assumes the value "true" ↔ there is a token at the place "A user U_i is the holder of a level F_k" in the Petri net and, otherwise, "false;"

— $\mathrm{CSME}(j) \equiv \forall\, i_1\ \forall\, i_2\ (\neg(\mathrm{SL}(i_1, j) \wedge \mathrm{SL}(i_2, j) \wedge (i_1 \neq i_2)))$, where $i_1, i_2, j \in 1 \ldots n$. "True" means that the control of the corresponding session is mutually exclusive. We denote an expression without quantifiers by $\mathrm{CSME}'(j)$;

— $\mathrm{HLME}(k) \equiv \forall\, i_1 \forall i_2\ (\neg(\mathrm{LH}(i_1, k) \wedge \mathrm{LH}(i_2, k) \wedge (i \neq i_2)))$, where $i_1, i_2, k \in 1 \ldots n$. "True" means that the holding of a level is mutually exclusive. We denote an expression without quantifiers by $\mathrm{HLME}'(k)$;

— $\mathrm{Addit\_state}(\mathrm{CM}) \equiv \forall j\ \forall\, k\ (\mathrm{CSME}(j) \wedge \mathrm{HLME}(k)) \equiv \forall\, i_1\ \forall\, i_2\ \forall\, j\ \forall\, k\ (\mathrm{CSME}'(j) \wedge \mathrm{HLME}'(k))$.

Then the predicate $\Psi(\mathbf{R}')$ assumes the form

$$\Psi(\mathbf{R}') = \begin{cases} \text{true if the predicate Addit\_state (CM)} \\ \qquad \text{is satisfied for all markings of the run;} \\ \text{false otherwise.} \end{cases} \tag{1}$$

In what follows, it makes sense to pass from a predicate to a quantified Boolean formula (QBF) as follows:

— $\mathrm{SL}(i, j)$ is replaced by the Boolean variable $x_{i,j}$;

— $\mathrm{LH}(i, k)$ is replaced by the Boolean variable $y_{i,k}$;

— $\mathrm{CSME}(j)$ is replaced by the formula

$$\forall\, x_{1,1}\ \forall\, x_{1,2} \ldots \forall\, x_{1,n}\ \forall\, x_{2,1} \ldots \forall\, x_{n,n} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,1} \wedge x_{3,1}) \wedge \ldots]; \tag{2}$$

— $\mathrm{HLME}(k)$ is replaced by the formula

$$\forall\, y_{1,1}\ \forall\, y_{1,2} \ldots \forall\, y_{1,n}\ \forall\, y_{2,1} \ldots \forall\, y_{n,n}\ [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,1} \wedge y_{3,1}) \wedge \ldots]; \tag{3}$$

— Addit_state (CM) is replaced by the formula

$$\forall\, x_{1,1}\ \forall\, x_{1,2} \ldots \forall\, x_{1,n}\ \forall\, x_{2,1} \ldots \forall\, x_{n,n} \forall\, y_{1,1} \forall\, y_{1,2} \ldots \forall\, y_{1,n} \forall\, y_{2,1} \ldots \forall\, y_{n,n} [\neg(x_{1,1}$$
$$\wedge x_{2,1}) \wedge \neg(x_{1,1} \wedge x_{3,1}) \wedge \ldots] \wedge [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,1} \wedge y_{3,1}) \wedge \ldots]. \tag{4}$$

For example, when $n = 2$, the corresponding formulas are of the form

$$\forall\, x_{1,1}\ \forall\, x_{1,2}\ \forall\, x_{2,1}\ \forall\, x_{2,2} [\neg(x_{1,1} \wedge x_{2,1}) \wedge \neg(x_{1,2} \wedge x_{2,2})];$$

$$\forall\, y_{1,1}\ \forall\, y_{1,2}\ \forall\, y_{2,1}\ \forall\, y_{2,2}\ [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,2} \wedge y_{2,2})];$$

$$\forall\, x_{1,1}\ \forall\, x_{1,2}\ \forall\, x_{2,1}\ \forall\, x_{2,2}\ \forall\, y_{1,1}\ \forall\, y_{1,2}\ \forall\, y_{2,1}\ \forall\, y_{2,2} [\neg(x_{1,1} \wedge x_{2,1})$$
$$\wedge \neg(x_{1,2} \wedge x_{2,2})] \wedge [\neg(y_{1,1} \wedge y_{2,1}) \wedge \neg(y_{1,2} \wedge y_{2,2})].$$

According to [3, pp. 120–121], the computational complexity of the problem of verification of agents for $\sum_u^p$-complete specifications of a problem is $\prod_{u+1}^p$-complete. Worthy of mention are polynomial hierarchies of classes $\sum_u^p$ and $\prod_u^p$. They cover an uncountable set of classes to each of which corresponds a definite value $u \in N$ (for $u < 0$,

$\sum_u^p = \varnothing$). It is assumed that the specification Ψ can be represented by a Turing machine T_Ψ that accepts at its input only the runs satisfying this specification. Any $\sum_u^p$-specification can be described by an alternating Turing machine that solves the problem in polynomial time using no more than u places between existence and generality states (corresponding to existence and generality quantifiers). Thus, we have $\sum_0^p \equiv P$ and $\sum_1^p \equiv NP$.

Each complexity class $O(t(n))$ is a set of languages that can be accepted in time $O(t(n))$ by the corresponding Turing machine. In particular, the class *NP* consists of languages that are accepted in polynomial time by nondeterministic Turing machines. In the general case, we have $\prod_u^p \equiv \text{co-}\sum_u^p$. However, the result from [3] cannot be directly applied to our problem since the specification of this problem is co- *NP*-complete (or $\prod_1^p$) since it contains only generality quantifiers.

According to [4, Chapter 10], co-*NP* is the complexity class that includes problems with succinct disqualification in contrast to the class *NP* uniting problems with succinct certificate. Since the Boolean validity problem is co-*NP*-complete, our specification of problem (4) is also co-*NP*-complete. Therefore, to determine the complexity of the problem of verification of the CM, the lemma formulated below should be proved.

LEMMA 1. The computational complexity of the problem of verification of agents for co-*NP*-complete specifications is co-*NP*-complete.

Proof. Co-*NP*-complexity directly follows from the complexity of the problem specification. To prove its co-*NP*-completeness, we reduce the problem of determination of the truth of a QBF containing only generality quantifiers to the problem of verification of agents. As a result, we obtain

$$\forall \bar{x}_1 \; \forall \bar{x}_2 \; \ldots \; \forall \bar{x}_n \; \chi(\bar{x}_1, \ldots, \bar{x}_n), \tag{5}$$

where each $\bar{x}_i$ is a finite collection of Boolean variables and $\chi(\bar{x}_1, \ldots, \bar{x}_n)$ is a propositional logic formula over a set of Boolean variables $\bar{x}_1, \ldots, \bar{x}_n$.

The problem of verification of agents on the basis of formula (5) is constructed as follows. Let $\bar{x}_1 = x_1^1, x_1^2, \ldots, x_1^m$ be the outermost collection of quantified variables. To each of these variables x_1^i will correspond the following two possible states of the environment: $e_{x_1^i}$ and $e_{\neg x_1^i}$ that correspond to the values "true" and "false" of the variable x_1^i. We denote the initial environment state by e_0. The environment allows an agent to execute only the action a_0 and responds to the ith execution of this action by the transition to the state $e_{x_1^i}$ or $e_{\neg x_1^i}$. After the mth execution of the action a_0, the corresponding run terminates. Thus, each run assigns the corresponding value to each generally quantified variable from the set $\bar{x}_1 = x_1^1, x_1^2, \ldots, x_1^m$. In this case, the set of all possible runs corresponds to the set of all possible combinations of values of these variables. For a given run r, $\chi(\bar{x}_1, \ldots, \bar{x}_n)[r / \bar{x}_1]$ is a Boolean formula obtained from the formula $\chi(\bar{x}_1, \ldots, \bar{x}_n)$ as a result of the replacement of each variable x_1^i by its value ("true" or "false") assigned to this variable within the framework of the run r. Then the specification of the problem Ψ assumes the form

$$\Psi(r) = \begin{cases} \text{truth if the co-}NP\text{-complete formula} \\ \quad \forall \bar{x}_2 \ldots \forall \bar{x}_n \; \chi(\bar{x}_1, \ldots, \bar{x}_n)[r / \bar{x}_1] \\ \quad \text{assumes the value truth;} \\ \text{false otherwise.} \end{cases} \tag{6}$$

Input formula (5) assumes the value "true" when all the runs of an agent satisfy specification (6). Since the reduction is polynomial, the computational complexity of the problem of verification is co-*NP*-complete, which is what had to be proved. ■

Next, let us determine the complexity of the problem of verification of the CM.

THEOREM 1. The computational complexity of the problem of verifying the CM is co-*NP*-complete.

Proof. The computational complexity of the specification of the problem of verifying CM (1) is co-*NP*-complete. According to Lemma 1, the computational complexity of the problem of verification of agents that is similar to the problem of verification of the CM is co-*NP*-complete in the case of the co-*NP*-complete specification of the problem. Hence, the computational complexity of the problem of verification of the CM is co-*NP*-complete, which is what had to be proved. ■

Thus, the obtained theoretical result allows one to range the problem of verifying the coordination mechanism of a collaboration system in the already well-known and investigated class of co-*NP*-complete problems. This result is important for investigations of a general model of software systems supporting collaboration networks with an arbitrary number of users, sessions, and resources, in particular, cooperative collaboration in the Internet.

REFERENCES

1. M. M. Glibovets and D. K. Hlomozda, "Formal model of a coordinationally founded network for a training collaboration system," Probl. Progr., No. 2–3, 402–412 (2006).
2. M. M. Glibovets, "Models and methods of creation and support of a high-performance distributed training environment," Synopsis of Doctoral Thesis in Physical and Mathematical Sciences, V. M. Glushkov Institute of Cybernetics of NASU, Kiev (2006).
3. M. Wooldridge and P. E. Dunne, "The computational complexity of agent verification," in: Proc. 8th International Workshop (Intelligent Agents VIII) on Agent Theories, Architectures, and Languages (ATAL-2001), Springer, Berlin (2002), pp. 115–127.
4. C. H. Papadimitriou, Computational Complexity, Addison-Wesley, New York (1994).