In: Proc. of the IEEE European Workshop on Computer-Intensive Methods in Control and Signal Processing
Sep. 7-9, 1994, Prague, Czech Republic, pp. 229–236

Dealing with Complexity in the Discrete Event System Control Methods[1]

Petr Kozák

Czech Academy of Sciences, Institute of Information Theory and Automation
Pod vodárenskou věží 4, 182 08 Prague 8, Czech Republic, E-mail: kozak@utia.cas.cz

Abstract

Complexity issues related to a discrete event system control problem called the Supervisory Control Problem are discussed and a simple introduction to the problem is provided. The methods for dealing with complexity are surveyed.

1. Introduction

Discrete event models and methods have great potential for solving important problems in development of control systems for manufacturing, traffic systems, communication and database systems, to mention few application areas. Nevertheless, there is still a relatively considerable gap between the discrete event control theory and requirements of applications. It can be characterized using two main aspects:

- It is required to increase expressive power of formalisms (e.g. description of more complicated timed features of discrete event systems, incorporation of certain "continuous" system features into the discrete event systems designed originally for solving "logical" problems),
- It is necessary to deal effectively with problems of enormous size which often arise in applications.

The present paper focuses on the latter issue in a subarea of discrete event system theory, i.e. supervisory discrete event control originated from [1] and more specifically one of the problems called the Supervisory Control Problem (SCP). The results in solving the untimed SCP are discussed in detail (Sections 2 and 3) but the untimed extensions are mentioned only briefly (Section 4). An attempt to survey the methods for dealing with complexity is made (Section 3). For the sake of brevity, the definitions of only some concepts are included. Recent surveys of the supervisory control are in [2, 3]. The detailed treatment or other relevant references can be found in the referenced literature.

[1] The work was partially supported by the AVČR grant #27558.

1.1. Preliminaries

The set of finite sequences over an arbitrary finite set Σ is denoted Σ^*. The length (number of symbols) of a sequence v is $|v|$. The set of all prefixes of $L \subseteq \Sigma^*$ is

$$\mathrm{pref}(L) = \{v \in \Sigma^* \mid \exists v' \in \Sigma^*, vv' \in L\} \ .$$

Eg.: Let $L = \{\mathrm{ab}, \mathrm{abc}\} \subseteq \{\mathrm{a}, \mathrm{b}, \mathrm{c}\}^*$ be a set of two sequences. In this case $\mathrm{pref}(L) = \{\epsilon, \mathrm{a}, \mathrm{ab}, \mathrm{abc}\}$, where ϵ is the empty sequence.

For sets Σ and Σ_o such that $\Sigma_o \subseteq \Sigma$, the *(natural) projection* $P : \Sigma^* \to \Sigma_o^*$ is defined iteratively as follows:

$$\begin{aligned} P(\epsilon) &= \epsilon \\ \forall v \in \Sigma^*, \forall \sigma \in \Sigma, P(v\sigma) &= \begin{cases} P(v)\sigma & \text{if } \sigma \in \Sigma_o \\ P(v) & \text{otherwise.} \end{cases} \end{aligned}$$

The projection is extended to a set of sequences $L \subseteq \Sigma^*$, with $P(L) = \{P(v) \mid v \in L\}$.

The power set of a set Σ is denoted $\mathcal{P}(\Sigma)$.

2. Supervisory Control Problem

The *uncontrolled behaviour of the plant* is given as a nonempty set L of finite sequences over a finite *set of event labels* Σ such that

$$L = \mathrm{pref}(L) \ . \qquad (1)$$

The set L represents all sequences of events the plant can generate if it is not controlled by any controller. Property (1) is interpreted such that the plant can stop the generation of events at any time. This model of system behaviour does not give exact time moments of event occurrences. The time is represented here only by the order of events.

The set L can be visualized using a tree[2] (Fig. 1) with the set of nodes equal to the set of sequences, edges labeled by event labels, the root equal to the empty sequence, and the depth given by the length

[2] The tree is suggested here only for the visualization purpose not for effective implementation.

of sequences. E.g.: the node $v \in L$ has depth 3. Note that the depth of the bottom line is equal to ∞, i.e. infinite sequences would be shown at this level of the tree. As L contains only finite sequences (but of an arbitrary length) the nodes of the tree may be arbitrarily close to the bottom line. The subtree corresponding to each node represents all sequences which may be generated in the future, e.g. the shaded area for v in Fig. 1:

$$\mathcal{C}_L(v) = \{v' \in L \mid v \in \mathrm{pref}(v')\} \ .$$

Let $\Sigma_c \subseteq \Sigma$ be a set of *controllable event labels.*

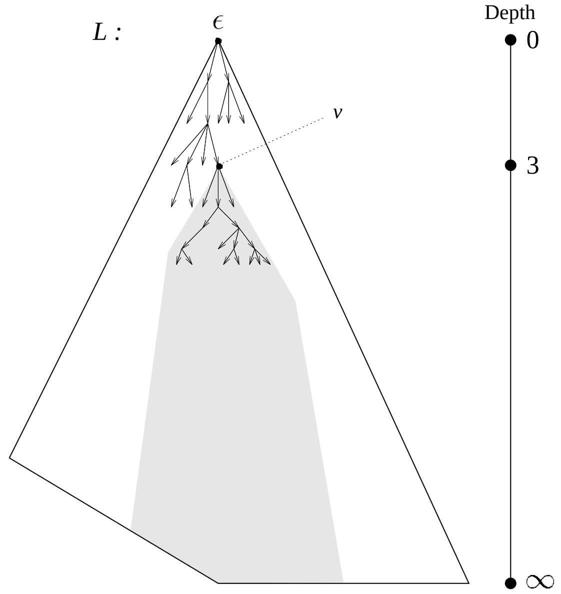

Figure 1: Uncontrolled behaviour of the plant L

It represents the labels of events which may be dynamically disabled and enabled by a controller. The *centralized controller* for the given plant is defined as the mapping $\gamma : L \to \mathcal{P}(\Sigma_c)$. The sets $\gamma(v)$ (for $v \in L$) are interpreted as the sets of disabled events after observation of v.

The *closed-loop system* generated by the feedback interconnection of L and γ is defined as

$$\begin{aligned}\mathcal{S}(L/\gamma) \;\; = \;\; & \{v \in L \mid \forall v' \in \mathrm{pref}(v), \forall \sigma \in \Sigma, \\ & v'\sigma \in \mathrm{pref}(v) \;\Rightarrow\; \sigma \notin \gamma(v')\} \ \ ,\end{aligned}$$

i.e. the behaviour of the closed-loop system is constructed from the tree L by removals of disabled edges.

Consider the sets $\emptyset \neq A \subseteq E \subseteq L$ and $L_m \subseteq L$ such that $A = \mathrm{pref}(A)$ and $E = \mathrm{pref}(E)$. The *Supervisory Control Problem (SCP)* [1] is to find a centralized controller γ for the given plant such that the closed-loop system satisfies the following two conditions

$$A \subseteq \mathcal{S}(L/\gamma) \subseteq E \ \ , \qquad (2)$$

$$\mathrm{pref}(\mathcal{S}(L/\gamma) \cap L_m) \;=\; \mathcal{S}(L/\gamma) \ . \qquad (3)$$

It is visualized on Fig. 2. The set E *(legal language)* represents legal sequences, i.e. entering of the dark shaded area of the tree ($\sim L \setminus E$) must be avoided. The set A *(minimal acceptable language)* represents sequences which generation must be possible, i.e. no event can be disabled in the light shaded area. The set L_m *(marked language)* can be interpreted as the set of sequences representing completed tasks. Specification (3) means that at least one task must be completable at any time, i.e.

$$\forall v \in \mathcal{S}(L/\gamma),\ \mathcal{C}_{\mathcal{S}(L/\gamma)}(v) \cap L_m \neq \emptyset \ .$$

In other words, the system is never blocked to do "something reasonable". It is shown [1] that if the

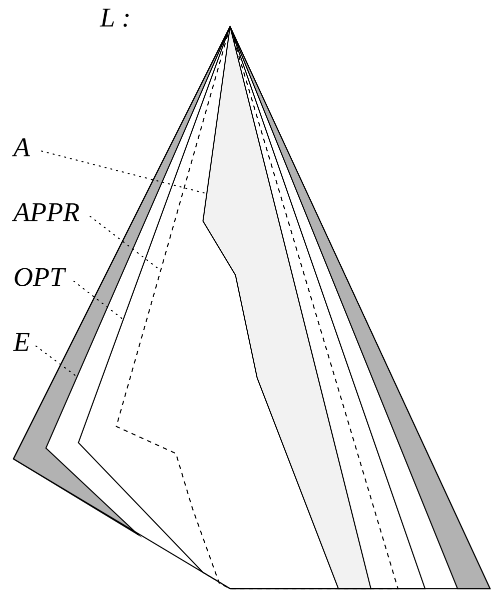

Figure 2: Supervisory Control Problem

problem is solvable then there exists a unique supremal (in the sense of set inclusion) solution of the problem. It is denoted *"OPT" (optimal)* in Fig. 2. It can be effectively computed under certain additional assumptions. In this paper, it is supposed (unless it is stated otherwise) that the sets A, E, L_m, and L are regular languages, i.e. generated by finite automata. In this case, the computation [4, 5] of OPT has complexity $O(m^2\, n^2)$, where m (n) is the number of states of automaton generating L (E), respectively. If $\mathrm{pref}(E \cap L_m) = E \cap L_m$ then the computation has complexity $O(m\, n)$.

The control problem often has additional requirements, e.g. a particular structure of controller is required (only certain events can be observed by the controller, or the controller consists of a set of subcontrollers each with limited means of observation and control) etc. The resulting controller often provides a solution ($APPR$ in Fig. 2) which is strict subset of OPT and is in fact only an *approximation* of OPT with respect to the additional requirements. Typically there is no supremal approximation in the sense of set inclusion.

The following text discusses two main difficulties and gives a survey of research related to:

1. **Size of Controller Implementation.** The size of a "naive" controller implementation is proportional to the multiplication of numbers m and n which can be enormous (10^4 to $10^{1000000}$ [6]). Therefore some "concise" representation of controller is required.
2. **Computational Complexity.** The "naive" computation of the centralized controller is also not tractable for such big numbers.

It may seem difficult to build an automaton of such size, but it is easy using composition of automata [7], e.g. for parallel interconnection of automata *(product systems)* [15] the size of the global automaton is given by multiplication of subautomata sizes.

3. Current Approaches to the Complexity Issue

The approaches are organized according to the kind of additional assumptions into the following groups:

1. Decentralized and modular controllers,
2. Assumptions related to the legal behaviour,
3. Model reduction and approximation,
4. Effective methods fo representation of models in computers and parallel algorithms,
5. Structured Formulation of Plant Model.

3.1. Decentralized and Modular Controllers

The controller can be composed from several decentralized subcontrollers [8, 9]. Controllers with partial observability of events can be regarded as a special case of decentralized controllers [10, 11, 12].

Suppose that I is the set of controller names and that each controller $i \in I$ can observe only event labels $\Sigma_{o,i} \subseteq \Sigma$ and control only $\Sigma_{c,i} \subseteq \Sigma_c$. Each subcontroller $i \in I$ is defined as the mapping $\gamma_i : P_i(L) \to \mathcal{P}(\Sigma_{c,i})$, where $P_i : \Sigma^* \to \Sigma^*_{o,i}$ is a projection. The sets $\gamma_i(v)$ (for $v \in P_i(L)$) are interpreted as sets of events disabled by the ith subcontroller after observation of v. The decentralized controller γ is defined for all $v \in L$ such that

$$\gamma(v) = \bigcup_{i \in I} \gamma_i(P_i(v)) \quad .$$

The reasons for design of the decentralized controllers are:

1. It is required by the given application that the controller must be divided into several parts with limited means of observation and control. This is e.g. the case of communication protocols where one part of the protocol observes only events on the sender site and another on the receiver site.
2. It is hoped that the decentralized controller implementation will require less memory space than the centralized implementation.

Unfortunately, given I, $\Sigma_{o,i}$, $\Sigma_{c,i}$, $(i \in I)$ the SCP has no optimal solution (in the sense of the set inclusion) in general, i.e. there are often several maximal solutions not comparable using set inclusion. Moreover, the minimal solutions are also not unique. The computation of a solution has exponential complexity in general.

If $\mathrm{pref}(E \cap L_m) = E \cap L_m$ and the problem is solvable then there exists a unique minimal solution, i.e. a unique decentralized controller which gives the minimal closed-loop behaviour (in the sense of the set inclusion). This solution is studied in [9].

Another solution *(fully decentralized controller)* is proposed in [13] and for special case of single subcontroller in [12]. Each subcontroller of the fully decentralized controller is supposed to have the following form:

$$\gamma_i(v) = \{\sigma \in \Sigma_{c,i} \mid \exists w \in L, P_i(w) = v \wedge \sigma \in \gamma(w)\},$$

for all $v \in P_i(L)$, where γ is the optimal centralized controller.

The main reason for enormous complexity of computations of the decentralized controllers is computation of projections. In [12, 13], it is shown that under the assumptions that $A = \{\epsilon\}$ and $\mathrm{pref}(E \cap L_m) = E \cap L_m$ the controllers can be implemented such that the formulas for subcontrollers are evaluated on-line and it is not necessary to calculate the projections. In this case, the size of each subcontroller and the complexity of the decision step of each subcontroller are proportional to the size of the optimal controller. The off-line part of design procedure consists of the design of the optimal controller. The overall complexity is $O(m\,n)$, i.e. equal to the complexity of the centralized case.

Modular design is proposed in [14, 5]. The assumption is that the subcontrollers do not have limited observation and control capabilities. As it requires also some structure of the legal behaviour it will be discussed in the following subsection.

3.2. Assumptions Related to Legal Behaviour

3.2.1 Formulation using Subtasks:

The legal behaviour can be naturally given in the form

$$E = \bigcap_{j \in J} E_j \ ,$$

where $E_j = \mathrm{pref}(E_j) \subseteq L$ ($j \in J$ for some index set J). The sets $E_j \cap L_m$ represent subtasks to be done. Suppose in this subsection that the complexities of E_j are bounded by some n.

The idea of modular design [5] is to design for each subtask E_j ($j \in J$) a subcontroller and then to test that the resulting closed-loop behaviours K_j of these subproblems ($j \in J$) are not conflicting (this is necessary to guarantee the nonblocking specification (3)), i.e. they must satisfy the condition

$$\mathrm{pref}(\bigcap_{j \in J} K_j \cap L_m) = \bigcap_{j \in J} K_j \ .$$

The test has complexity $O(m^p\, n^p)$ and the overall computation has complexity $O(p\, m^2\, n^2 + m^p\, n^p)$, where p is the cardinality of J. The complexity of the centralized computation is $O(m^2\, n^{2p})$

Supposing that $A = \{\epsilon\}$ and $\mathrm{pref}(E_j \cap L_m) = E_j \cap L_m$ (for all $j \in J$) the complexity is $O(p\, m\, n)$ and no additional test is necessary. The complexity of the centralized design is $O(m\, n^p)$.

3.2.2 Problems for Product Systems:

This subsection presents some results of [15] related to product systems composed of p non-interacting (parallel) components[3].

Let $I = \{1, 2, \ldots, p\}$, Σ_i ($i \in I$) be pairwise disjoint finite sets of event labels, $\Sigma_{c,i} \subseteq \Sigma_i$ be sets of controllable event labels,

$$\Sigma \ = \ \bigcup_{i \in I} \Sigma_i \ ,$$

$L_i \subseteq \Sigma_i^*$ ($i \in I$), and $P_i : \Sigma^* \to \Sigma_i^*$ ($i \in I$) are projections.

The output behaviour of the product system is given by

$$L \ = \ \{v \in \Sigma^* \mid \ \forall i \in I,\ P_i(v) \in L_i\} \ .$$

Suppose that $A = \{\epsilon\}$ and that $L_m = L$. Let the size of automata generating L_i are bounded by some n.

Mutual Exclusion Problem. Let $Y_i \subseteq L_i$ ($i \in I$) be the sets of sequences representing the "critical section", which cannot be entered by all components simultaneously and let each Y_i be recognized by a subset of the state space of the automaton generating L_i.

The legal language E is defined such that

$$E \ = \ \{v \in L \mid \ \exists i \in I,\ P_i(v) \notin Y_i\} \ .$$

This variation of the SCP is solvable in $O(p\, n^2)$.

String Exclusion Problem. Let $y_i \in L_i$ ($i \in I$) and $y \in \Sigma^*$ such that for all $i \in I$, $P_i(y) = y_i$. The legal language E is defined such that

$$E \ = \ \{v \in L \mid \ \forall v', v'' \in L,\ v' y v'' \neq v\} \ .$$

The complexity of this problem is bounded by a polynomial in n, p, and $|y|$.

3.3. Model Reduction and Approximation

In [16], it is supposed that the controller knows the plant model only for the window of N-events ahead *(limited lookahead policy)*. Suppose that the global plant model is L. For all $v \in L$ the controller knows only the tree

$$\mathcal{L}_{N,v} \ = \ \{v' \in \mathcal{C}_L(v) \mid \ |v'| \leq |v| + N\} \ ,$$

i.e. $\mathcal{L}_{N,v}$ is the plant model after observation v. This requires to perform the centrallized controller design for the plant $\mathcal{L}_{N,v}$ after observation of each event.

Several design policies are proposed according to the transformation of problem specifications at each step. The *conservative policy* formulates the control problem at each step with the legal language

$$\{v' \in E \cap \mathcal{L}_{N,v} \mid \ |v'| < |v| + N\} \ ,$$

minimal acceptable language $A = \{\epsilon\}$, and marked language $L_m \cap \mathcal{L}_{N,v}$.

The *optimistic policy* considers the legal language $E \cap \mathcal{L}_{N,v}$, minimal acceptable language $A = \{\epsilon\}$, and marked language

$$(L_m \cap \mathcal{L}_{N,v}) \ \cup \ \{v' \in \mathcal{L}_{N,v} \mid \ |v'| = |v| + N\} \ .$$

It is shown that if the centralized problem for the plant L is solvable then the conservative policy gives a solution of the problem. However, this solution need not be optimal. The limited lookahead controllers can be used in applications to reduce significantly the complexity. The size N influences strongly how the closed-loop behaviour is close to the optimal solution OPT. The minimal size of the window such that the optimal behaviour is guaranteed can be found. But this computation has the worst case complexity same as the computation of the optimal controller itself.

[3] The results are presented here in a simplified way using only finite sequences. In [15], infinite sequences are used and also the problems are formulated in a more general way.

The optimistic policy can give excellent results for some applications but again the computation to guarantee the optimal solution has the worst case complexity of the centralized case.

The limited lookahead controller design has several modifications. One of them considers the variable size N of the window [17] *(variable limited lookahead policy)*. The size is recalculated at each step to guarantee the optimal behaviour. The worst case complexity is again equal to the complexity of the centralized case but in practical situations it is often far smaller.

3.4. Effective Methods for Representation of Models in Computers and Parallel Algorithms

The *binary decision diagrams (BDD)* are efficient data structures for boolean function representation [18]. The BDD is an acyclic directed finite graph labeled by $\{0,1\}$. The leave nodes have values in $\{0,1\}$. The names of other nodes correspond to variable names of the boolean function. The evaluation of a variable assignment starts at the root and proceeds from a node named after a variable by the edge labeled by the value assigned to this variable. The value of a leave node reached during the evaluation gives the function value. This represent a very effective way of encoding the truth table of a boolean function. However, the efficiency depends significantly on the chosen variable ordering.

The plant and the specifications can be represented as boolean functions and implemented as BDDs. This method is proposed in [18], where an example of size 10^7 and detailed explanation can be found.

Another useful technique is implemented in the Boolean analysis software XBOOLE presented in [19]. It uses *ternary vector lists* to represent solutions of Boolean functions. The ternary vector list is a table of vectors over the set $\{0, 1, \text{don't\_care}\}$. Each vector can represent one or more binary vectors depending on the number of "don't_care" elements.

To evaluate efficiency of the representation techniques, it has to be taken into account not only the efficiency of representation but also efficiency of operations performed with the data structures, efficiency of creation and maintenance of the data structures when the data are changed, flexibility with respect to the special requirements in decentralized, hierarchical control etc., and possibility to incorporate also numerical information often required for timed problems. E.g. the suitability of BDDs for mixed (logical-numerical) data structures is questionable. At present time, it is not possible to give any general conclusion because there is only little experience with real-world problems and only little research is done in this respect.

Parallel implementations of both design and run-time algorithms are investigated in [20]. Two cases are considered for the simplified design procedure of the centralized controller (Recall that the single-processor complexity is $O(m\,n)$.) with $m\,n$ and $m^3\,n^3$ processors. The complexity of the latter case is $O(\log^2(m\,n))$. It is suggested that utilization of massively parallel computers (16,000 processors are commercially available) can help significantly to solve large problems. Parallel algorithms for some other supervisory control tasks are also described [20].

3.5. Structured Formulation of Plant Model

3.5.1 Hierarchical Supervisory Control:

Hierarchical supervisory control is presented in [21]. Consider only two level hierarchy and the low-level plant equal to the plant L of Section 2. The low-level plant is controlled by the low-level controller which accepts commands from the high-level controller and controls the low-level plant in the same fashion as the centralized controller. The low-level plant sends events to the low-level controller and via a mapping θ to the high-level controller. Let T be the event label set of the high-level plant. The mapping $\theta : L \rightarrow T^*$ is defined such that for all $v, v' \in L$

$$v \in \mathrm{pref}(v') \;\Rightarrow\; \theta(v) \in \mathrm{pref}(\theta(v')) \;.$$

This property is called *prefix preserving* or *causality*.

Each event $\sigma \in T$ of the high-level plant replaces a set of sequences $\mathcal{L}_\sigma \subseteq \Sigma^*$ of the low-level plant, i.e. in other words it informs by its occurrence the high-level controller about the occurrence of these sequences. The controllability of σ is determined such that σ is controllable if each sequence in $\mathcal{L}_\sigma$ contains at least one low-level controllable event, otherwise it is uncontrollable.

The consistency of the low and high level plants, controllers, and control problems is investigated [21]. The automatic decomposition of the centralized problem into the hierarchical problem has exponential complexity in general. However, the plant model and specifications can be given in a hierarchical way by the human designer. This enables to formulate large problems in a relatively easy way and also the synthesis procedures are more effective than for the centralized design in general. This forms one the most promising directions of the current research.

In [22], the problem of reachability is studied for a class of discrete event plants modeled as hierarchical state machines with possibility of parallelism. An efficient method for testing reachability is provided. The optimal centralized controller for the case $\mathrm{pref}(E \cap L_m) = E \cap L_m$ which uses also the idea of on-line synthesis is presented.

3.5.2 Vector Discrete Event Systems:

Supervisory control theory of systems with state space equal to the integer vector space is developed in [23, 24]. The model is called *vector discrete event systems (VDES)* and it is closely related to Petri nets. The main idea is to enhance the abstract language (automaton) based supervisory control by exploiting algebraic regularity of internal system structure given by arithmetic additivity over the integers.

Let X be N dimensional vector space of integers for some natural N. The *state transition mapping* is $\delta : \Sigma \times X \rightarrow X$ defined for all $\sigma \in \Sigma$ and $x \in X$ only if

$$F_\sigma \leq X$$

such that

$$\delta(\sigma, x) = x + E_\sigma ,$$

where $F_\sigma \in X$ is the *occurrence-condition vector* and $E_\sigma \in X$ is the *displacement vector*. The VDES starts at a initial state $x_0 \in X$ and generates events according to the state transition mapping. It is supposed that $F_\sigma = -E_\sigma$ for all $\sigma \in \Sigma$. The controller is defined in a similar way as in Section 2 as the mapping $\gamma : X \rightarrow \mathcal{P}(\Sigma_c)$ and gives for each state the set of disabled events. The closed-loop behaviour is defined according to this semantics.

The control problem is formulated for the given plant such that the legal language E is defined using a subset of the state space which should be made unreachable by means of control. It is shown that if the VDES is loop-free then the reachable set of states can be characterized by a system of linear inequalities [24]. The controller design procedure based on linear integer programming is presented.

The supervisory control formulated using Petri nets is studied in [25].

The VDES and Petri nets can model non-regular languages (generated by infinite automata). Therefore the complexity of computations based on these models cannot be expressed in the same way as for the regular language models. There is no comprehensive complexity study of controller design for these models. The models can describe very efficiently plants with buffers, where the automaton model is often too big because cannot express the regularity of such devices. On the other hand, the transformation of some automata into VDES gives no advantage in terms of efficiency or readability. Therefore it can be useful to combine these two models in one approach.

4. Timed Supervisory Control Problem

The timed supervisory control problem can be formulated similarly as the SCP such that the sets of (untimed) sequences are replaced by sets of timed traces. The timed trace has two components: the untimed sequence of event labels and a sequence of time moments which the events occur at.

Modelling of timed discrete event systems offers far more possibilities for development of different approaches. The properties of systems include simple lower and upper bounds on delays between two subsequent output discrete events [26, 27], communication delays between the plant and the controller [28, 29], and also complex time dependent properties which require uncountable state space as e.g. hybrid controllability [30].

A typical operation which is performed with such model is the calculation of distances in a labeled oriented graph. It has cubic complexity in terms of the number of nodes. So the calculation of distances in the model including the plant and the legal language with an additional labeling by real numbers is $O(m^3\, n^3)$, where m and n are as in Section 2. As the complexity is even higher than for the untimed problem the methods for dealing with complexity are urgently required for the timed problems. The ideas for complexity reduction used for the SCP have been exploited also for the timed SCP. However, the research is in an early stage in this area comparing to the untimed models.

The development of timed models also enables to answer rigorously the question when it is possible to use simplified untimed models instead of timed ones. In [31], this question is studied for strongly causal timed discrete event systems.

5. Conclusion

Most of the computations mentioned in the present paper have polynomial complexity. Some complexities are linear in terms of multiplication of the size of plant and specifications. Unfortunately, even linear dependency of complexity on the plant size can be regarded as too restrictive for some applications (recall the number $10^{1000000}$). Only complexity of some problems (e.g. the mutual exclusion problem for product systems), which depends on the size and number of components, can be regarded as satisfactory for such application. At present time, three possibilities how the complexity can be further reduced can be identified:

1. Models with greater expressive power (e.g. VDES) can express certain classes of problems very efficiently.

2. Hierarchical methods (where the hierarchy is introduced by a human designer) can utilize some additional information which is not expressible using a single automaton.
3. Additional restrictive assumptions on the plant model and the form of specifications. However, these assumptions could cause that no solution is found even if the non-restricted problem is solvable.

The supervisory control theory has been used in several applications, e.g. [32]. It is shown that some of the ideas for dealing with complexity developed within the theory were intuitively used by engineers in some applications. E.g. some database management protocols for the concurrency problem can be designed as the fully decentralized controllers [33].

The complexity of real-world systems is typically much higher than the present control design software packages can deal with. However the need for reliable and effective software and hardware systems is well recognized [34, 35] and gives good motivation for the research. The theory of discrete event system control can provide solutions to some of these problems.

The issue of complexity includes also the complexity (friendliness) of man-computer interfaces of design packages. This is very important for applicability but only few results have been reported so far, cf. [36, 34].

References

[1] P.J. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM J. Control Optim.*, vol. 25, pp. 206–230, Jan. 1987.

[2] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems", *Proc. of the IEEE*, vol. 77, pp. 81–98, Jan. 1989.

[3] J.G. Thistle, "Logical aspects of control of discrete-events systems: A survey of tools and techniques", In Cohen and Quadrat [36], pp. 3–15.

[4] W.M. Wonham and P.J. Ramadge, "On the supremal controllable sublanguage of a given language", *SIAM J. Control Optim.*, vol. 25, pp. 637–659, May 1987.

[5] W.M. Wonham and P.J. Ramadge, "Modular supervisory control of discrete event systems", *Mathematics of Control, Signals and Systems*, vol. 1, pp. 13–30, 1988.

[6] R.P. Kurshan, "Automata-theoretic verification of coordinating processes", In Cohen and Quadrat [36], pp. 16–28.

[7] W.M. Wonham, "A control theory for discrete event systems", in M.J. Denham and A.J. Laub, editors, *Advanced Computing Concepts and Techniques in Control Engineering*, vol. 47 of *Computer and Systems Sciences*, pp. 129–169. Springer Verlag, 1988.

[8] F. Lin and W.M. Wonham, "Decentralized control and coordination of discrete-event systems with partial observation", *IEEE Trans. Autom. Control*, vol. 35, pp. 1330–1337, Dec. 1990.

[9] K. Rudie and W.M. Wonham, "Think globally, act locally: Decentralized supervisory control", *IEEE Trans. Autom. Control*, vol. 37, pp. 1692–1708, Nov. 1992.

[10] F. Lin and W.M. Wonham, "On observability of discrete-event systems", *Information Sciences*, vol. 44, pp. 173–198, 1988.

[11] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations", *IEEE Trans. Autom. Control*, vol. 33, pp. 249–260, Mar. 1988.

[12] M. Heymann and F. Lin, "On-line control of partially observed discrete event systems", Technical Report #9310, Center for Intelligent Systems, Technion – Israel Inst. of Technology, Haifa, Israel, Mar. 1993.

[13] P. Kozák and W.M. Wonham, "Fully decentralized solutions of supervisory control problems", Technical Report # 9310, Systems Control Group, Dept. of El. Eng., Univ. of Toronto, Canada, Aug. 1993.

[14] P.J. Ramadge and W.M. Wonham, "Modular feedback logic for discrete event systems", *SIAM J. Control Optim.*, vol. 25, pp. 1202–1218, Sep. 1987.

[15] P.J. Ramadge, "Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata", *IEEE Trans. Autom. Control*, vol. 34, pp. 10–19, Jan. 1989.

[16] S.-L. Chung, S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems", *IEEE Trans. Autom. Control*, vol. 37, pp. 1921–1935, Dec. 1992.

[17] N.B. Hadj-Alouane, S. Lafortune, and F. Lin, "Variable lookahead supervisory control with state information", Technical Report # CGR-93-9, Dept. of Electrical Engineering and Computer Science, Univ. of Michigan, USA, July 1993.

[18] G.J. Hoffmann and H. Wong-Toi, "Symbolic supervisory synthesis for the workshop exercise", In Balemi et al. [37], pp. 189–197.

[19] R. Scheuring and H. Wehlan, "Control of discrete event systems by means of the boolean differential calculus", In Balemi et al. [37], pp. 79–93.

[20] V.K. Garg, "Parallel and distributed algorithms for supervisory control of discrete event

systems", In IEEE Conference on Decision and Control [38], pp. 2236–2241.

[21] H. Zhong and W.M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems", *IEEE Trans. Autom. Control*, vol. 35, pp. 1125–1134, Oct. 1990.

[22] Y. Brave and M. Heymann, "Control of discrete event systems modeled as hierarchical state machines", *IEEE Trans. Autom. Control*, vol. 38, pp. 1803–1819, Dec. 1993.

[23] Y. Li and W.M. Wonham, "Control of vector discrete-event systems I – the base model", *IEEE Trans. Autom. Control*, vol. 38, pp. 1214–1227, Aug. 1993.

[24] Y. Li and W.M. Wonham, "Control of vector discrete-event systems II – controller synthesis", *IEEE Trans. Autom. Control*, vol. 39, pp. 512–531, Mar. 1994.

[25] L.E. Holloway and B.H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets", *IEEE Trans. Autom. Control*, vol. 35, pp. 514–523, May 1990.

[26] J.S. Ostroff and W.M. Wonham, "A framework for real-time discrete event control", *IEEE Trans. Autom. Control*, vol. 35, pp. 386–397, Apr. 1990.

[27] B.A. Brandin and W.M. Wonham, "Supervisory control of timed discrete-event systems", *IEEE Trans. Autom. Control*, vol. 39, pp. 329–342, Feb. 1994.

[28] Y. Li and W.M. Wonham, "On supervisory control of real-time discrete event systems", *Information Sciences*, vol. 46, pp. 159–183, 1988.

[29] P. Kozák, "Control of elementary discrete event systems: Synthesis of controller with non-zero decision time", in D. Franke and F. Kraus, editors, *Proc. of 1st IFAC Symp. on Design Methods of Control Systems*, pp. 457–462, Zurich, Switzerland, Sep. 1991. Pergamon Press, Oxford, UK.

[30] P. Kozák, "Control of real-time discrete event systems with hybrid controllability of events", In IEEE Conference on Decision and Control [38], pp. 223–228.

[31] P. Kozák, "Supervisory control of discrete event processes: A real-time extension", Technical Report # 9301, Systems Control Group, Dept. of El. Eng., Univ. of Toronto, Canada, Jan. 1993.

[32] S. Balemi, G.J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G.F. Franklin, "Supervisory control of a rapid thermal multiprocessor", *IEEE Trans. Autom. Control*, vol. 38, pp. 1040–1059, July 1993.

[33] P. Kozák and W.M. Wonham, "Synthesis of database management protocols", Technical Report # 9311, Systems Control Group, Dept. of El. Eng., Univ. of Toronto, Canada, Aug. 1993.

[34] A. Benveniste and K.J. Åstrom, "Meeting the challenge of computer science in the industrial applications of control: An introductory discussion to the special issue", *IEEE Trans. Autom. Control*, vol. 38, pp. 1004–1010, July 1993.

[35] P.G. Neumann, "Illustrative risks to the public in the use of computer systems and related technology", *ACM SIGSOFT Software Engineering Notes*, vol. 19, pp. 16–29, Jan. 1994.

[36] G. Cohen and J.P. Quadrat, editors, *11th Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems*, vol. 199 of *LNCS*, Berlin, Germany, June 1994. Springer-Verlag, (INRIA Conf., Sophia-Antipolis, France).

[37] S. Balemi, P. Kozák, and R. Smedinga, editors, *Discrete Event Systems: Modeling and Control*, vol. 13 of *Progress in Systems and Control Theory*. Birkhäuser Verlag, Basel, Switzerland, 1993, (Proc. of the Joint Workshop on Discrete Event Systems (WODES'92), Aug. 26–28, 1992, Prague, Czechoslovakia).

[38] *Proc. of 32nd IEEE Conf. Decision and Control*, San Antonio, TX, USA, Dec. 1993.