We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

5

Supervisory Control of Industrial Processes

Alexander A. Ambartsumyan
Institute of Control Sciences RAS,
Russia

1. Introduction

Modern production is complex, integrated and is constantly being adapted to the market requirements by means of the reconfiguration of equipment structure and process alteration. The development of such production is performed based on evolutionary strategy by successively engaging (eliminating) stand-alone technological systems.

Evolutionary developed technical systems and facilities presently make up a considerable share of technical systems. It is typical both for high-tech industries, namely: aviation, space exploration, military equipment, machine-building (Sujeet, 2005), and for applications based on large-scale interconnected production complexes (e.g. oil- and gas-producing industry, oil and gas transportation, city economy engineering etc) (Gilard, 1999; Van Brussel et al., 1999; Jo, 1999; Ambartsumyan, Prangishvili, Poletykin, 2003; Ambartsumyan, Kazansky, 2008; Ambartsumyan, Potehin, 2003; Ambartsumyan, Branishtov, 2006).

Evolutionary developed technical systems and facilities are featured by complex control system availability. The latter integrates into a single whole different, as to the purposes, automatic control loops (automatic control and regulation of physical process parameters, automatic shielding and blocking, logical configuration control) as well as the functions of supervisory control mainly aimed at coordination of different processes in a technical system.

Supervisory control (SC) is intrinsically logical and is to provide the required operational sequence and exclude mutual blocking and deadlocks for stand-alone components (operating according to their internal rules time scale). SC is discrete and asynchronous by its nature and most commonly reveals itself as the change of event flow as required by certain application (technical system functionality).

It is important to consider two "event" aspects: first, everything happens as the result of a certain event; second, the change of states is regulated by events – there is no physical time though the system is dynamic.

Though control systems are widely spread in the technical systems of such kind (Sujeet, 2005; Gilard, 1999; Van Brussel et al., 1999; Jo, 1999; Ambartsumyan, Prangishvili, Poletykin, 2003; Ambartsumyan, Kazansky, 2008; Ambartsumyan, Potehin, 2003; Ambartsumyan, Branishtov, 2006), presently there is no appropriate theoretical base to solve such supervisory control tasks as local control loops coordination, configuration of material flows structure and interaction with operations staff.

Most spread concept of practical engineering of such systems is based on the model of interacting ″black boxes″: a ″black box–control object″ and symmetrically connected with it as to inputs and outputs a ″black box–control system (device)″. (Fig. 1).

Source: Process Management, Book edited by: Mária Pomffyová,
ISBN 978-953-307-085-8, pp. 338, April 2010, INTECH, Croatia, downloaded from SCIYO.COM

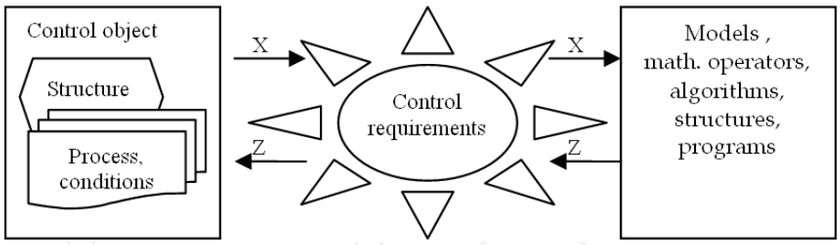

Fig. 1. The scheme of transfer from the object data base and control requirements to the mathematical description of the control

The first ″black box–control object″ is formed as a data base on the control object and technique at the stage of the object examination and includes the requirements of this object appropriate behaviour. The task of the required control search is tackled by the defining of a ″black box–control system″ able to monitor the behaviour – the event flow and, with the control purpose taken into account, to affect the object inputs in such a way that an appropriate behaviour of the object is achieved.

The question is how to search for a ″black box–control system″ with information on the first black box available. Common engineering practice shows that information on control object behaviour is only used indirectly.

What is the problem? We may speak about precise correspondence between a ″black box–control object″ and a ″black box–control system″ only as far as inputs and outputs are concerned, while behaviour is an approximate result of the designer's informal, speculative experiment with the initial data and limitations – the information the designer acquires considering the process physics peculiarities and the object structure properties. At that, there is not any confidence that a ″black box–control system″ can limit the behaviour of a ″black box–control object″ and provide its meeting the requirements since they, as a rule, are specified as models of another (not "event") nature and the extent they are taken into account depends on the designer's skills. The above leads to serious problems: designer's uncertainty in the fact that the designed system complies with the control tasks set; the necessity to make laborious verification of such compliance by computer simulation and the refinement of the designed system at facilities.

For the last 10–15, a sophisticated interaction among computer-driven actuating devices necessitates, when engineering, to analyze the design solutions safety and correctness, to validate technical systems implementation techniques, to take other approaches actually based on testing. It is a common knowledge that such approaches only can reveal a part of errors but cannot guarantee the system as a whole is error-free.

Different engineering approach than that based on two black boxes concept is declared in the theory of discrete event dynamic systems and supervisory control paradigm. The abbreviation is often simplified to DES. The distinctive features of supervisory control theory (all basic concepts and notions of this paper are borrowed from (Cassandras, Lafortune, 2008)) are as follows:

- The controlled object is represented in DES model by three components: generator *G* of *L(G)* language – proper control object, specification language *K* – limitations and *G* functionality required, supervisor *S* – control component in DES;
- Setting and solving the task of formal synthesis of *S* on *L(G)* and *K*.

The above, in its turn, creates a theoretical basis for machine control engineering fundamentally different from the deciphering of "black boxes" approximately fitting each

other. What does it give as compared with the classic procedure of discrete process control system synthesis according to two-black-boxes model?
First, the description of the object as *L(G)*-language generator *G*, limited by nothing, is more simple than the object description with all the admissible behaviour limitations taken into account. This work is performed as a separate stage – primary object examination and constructing a model "as it is".
Second, to form the required functionality (*K* specifications) basing on a generator *G* model already available is also easier than to consider all limitations and requirements in yet non-existing control system.
Third, control task is solved formally: a supervisor (provided the initial data is correct) is synthesized and does not require verification while the object and its behaviour are specified by object and know-how specialist and he is responsible for the data correctness, its verification and validation.
The present paper formulates the purpose of DES theory development, with the structural properties of technical systems taken into account, thus creating effective methods to synthesize a supervisor as an instrument to solve the task of consistency and co-ordination control of stand-alone components in a technical system.
Here below is given a brief survey of basic concepts and major noted results, as to DES and supervisory control, followed by the description of the present paper tasks and the results obtained.

2. Basic concepts and definitions

DES behaviour is considered generally as behaviour of a certain generator (source) of strings (sequences) of the events from a finite set of events *E*. The event $e \in E$ is an abstraction for a multitude of facts associated with DES "life". Events are instantaneous, occur spontaneously in unpredictable moments, therefore the only thing that can be observed is their sequences that are represented by strings. Event examples are: the facts of change in position and state of separate object components; commands to which the object reacts by the change of its state (position); characteristics of normal and abnormal states etc.
The main operation of strings forming is concatenation (we would like to remind that concatenation is the appending of separate events or entire strings of events on the right to the string, including ε – a space character). For the string, an integral function $\mu(s) = n$ is defined, where n is the number of characters in string s. If $n = 0$, $s = \varepsilon$. A set of all string of any finite length is designated by E^* (it is endless but countable). Let a string s consist of three parts: r, u, $t \in E^*$ connected by concatenation in such a way that $s = rut$, where r – a prefix, t – a suffix, and u – a substring of string s. Any subset of strings $L \subseteq E^*$ is called a language over E. If L includes ε and, jointly with any string s, contains all its prefixes, L is a prefix–closed language. As usual, conventional language operations are defined, namely: concatenation, prefix-closure and Kleene-closure.
In many constructions of DES theory, a couple of very important operations over languages are used: a projection P and a back projection P^{-1}. Let E_1, $E_2 \subset E$ be such that $E_1 \cup E_2 = E$ (possibly $E_1 \cap E_2 \neq \emptyset$). Projection P_i of any string from E^* on E_i is defined in three steps:
1. $P_i(\varepsilon) = \varepsilon$; 2. $P_i(e) = \varepsilon$ if $e \notin E_i$, otherwise $P_i(e) = e$; 3. $P_i(se) = P_i(s)\, P_i(e)$ for $s \in E^*$ and $e \in E$.
Conceptually, a projection of strings from larger alphabet E on smaller one E_i deletes from the string all characters from $E \setminus E_i$ (all characters outside E_i). Inverse function $P_i^{-1}(s) = \{t \in$

E^*: $P_i(t) = s\}$. $P_{i-1}(s)$ correlates every string $s \in E_i$ with some subset of strings E^* the projects of which on E_i equal s. Both operations are in natural manner extended to the languages $L \subseteq E^*$ and $L_i \subseteq E_i^*$. $P_i(L)=\{t \in L_i: (\exists s \in L)\ [P_i(s) = t]\}$; $P_{i-1}(L_i) := \{s \in E^*: (\exists t \in L_i)\ [P_i(s) = t]\}$.

In projection operation definition, instead of set indexes, for the sets, the events of which are excluded from the result of this operation, we shall use the designation of the set itself: P_{E_i} or $P_{E_i}^{-1}$.

Languages are a good instrument to observe DES behaviour but in order to perform analytical study and to set the task of providing the required dynamics (off-line behaviour), it is necessary to present a countable string set as a mathematical operator. There are many ways to present languages in the form of mathematical operators that generate or recognise the language. In DES theory, for these purposes, as a rule, finite state machines are used. A finite state machine is defined as $G = (Q, E, \delta, \Gamma, Q_m, q_0)$, where Q – a set of states; E – a set of events; δ – a transition function $Q \times E \to Q$; $\Gamma : Q \to 2^E$ – a function of admissible events in each state; Q_m – a set of marked states; q_0 – an initial state. We would like to note that in this definition the function of outputs is missing. For every state q_i the function of transitions is specified for the events admissible in this state (e.g. for $q_i \in Q$ and $e \in \Gamma_i$ the function $\delta(q_i, e) := q_j$). This definition can be naturally extended also for the following event strings: $\delta(q_i, \varepsilon) := q_i$, $\delta(q_i, se) := \delta(\delta(q_i, s), e)$ for $s \in E^*$ and $e \in E$. Let's denote by $\delta(q_i, s)!$ the fact that the function $\delta(q_i, s)$ is defined.

The function $\Gamma : Q \to 2^E$ is excessive in a model definition but it simplifies many examination schemes and algorithms development when analysing the languages presented by finite state machines, e.g. consistency definition. $Q_m \subset Q$ is a subset of marked states – the states corresponding to a certain functionality of G, with one of them necessarily being initiated in a specific variant of G use.

The language generated by G machine is designated as $L(G) := \{s \in E^* : \delta(q_0, s)!\}$. This is a set of all strings from E^* admissible in the initial state q_0. It is evident that $L(G) \subseteq E^*$. If the machine is completely defined, $L(G) = E^*$. It G is represented by a weighed graph of transitions, $L(G)$ is presented as a set of strings of the events weighing the edges of all the paths originated from the initial state q_0.

When a sophisticated DES is defined via components, two more operations on machines are often applied: Cartesian product and parallel composition. Product definition

$$G_1 \times G_2 = (Q_1 \times Q_2,\ E_1 \cap E_2,\ \delta_{1,2},\ \Gamma_{1\times 2},\ Q_{m1} \times Q_{m2},\ (q_0 := q_{01,\,02}))$$

is conventional but there is one nuance: a function of transitions is defined on common events for every pair of states. Isolated pairs and those unattainable from the initial state are discarded together with their associated transitions. From the definition it follows that the language $L(G_1 \times G_2)$ of the Cartesian product of two machines is equal to $L(G_1) \cap L(G_2)$ – the intersection of these machines languages.

Parallel composition (or just composition, let it be designated as $\oplus$) is defined on the union of events of both machines $G_1 \oplus G_2 = (Q_1 \times Q_2,\ E_1 \cup E_2,\ \delta_{1,2},\ \Gamma_1 \oplus_2,\ Q_{m1} \oplus Q_{m2},\ (q_{01}, q_{02}))$. At this, it is possible that $E_1 \cap E_2 \neq \varnothing$, then on common events, transition synchronization takes place in both components. If the event is individual, transition takes place in one component (provided for this pair this event belongs to the value area of the corresponding function Γ).

Formally:

$\delta((q_1, q_2), e) = \{(\delta_1(q_1, e), \delta_2(q_2, e))$ if $e \in \Gamma_1(q_1) \cap \Gamma_2(q_2)$ | $(\delta_1(q_1, e), q_2)$ if $e \in \Gamma_1(q_1) \setminus E_2$ | $(q_1, \delta_2(q_2, e))$ if $e \in \Gamma_2(q_2) \setminus E_1$ | and indeterminate in other cases}.

It is obvious that both operations are associative and, provided parentheses are places accordingly, may be easily generalized for n machines: a product – $G = \bigtimes_1^n G_i = G_1 \times ... \times G_n$; a composition – $G = \bigoplus_1^n G^i = G^i \oplus ... \oplus G^n$.

The initial stage of object study (modelling) is dedicated to prognostication of possible physical behaviour of the entire object or its subsystems, i.e. consideration of possible actions and possible variants of behaviour in the absence of any control and restrictive actions. At this stage, DES is represented by machine G as a language $L(G)$ generator. Thus, G generates event sequences of any kind reflecting control-free DES behaviour. In order to specify and provide control in DES, a set of events E is subdivided into two disjoint subsets: E_c – a subset of controllable events corresponding to the commands and E_{uc} – a subset of uncontrollable events for which the moments they occur are unpredictable.

The present-day view on DES was first worded in (Ramadge, Wonham, 1987) though then the term "discrete event systems" was not used but a new technique of discrete process modelling and control was stated. The term "discrete event systems (DES)" appears already in (Ramadge, Wonham, 1989), where DES is represented by generator G of different sequences of events from E. G is limited by nothing and therefore the sequences reflect the behaviour $L(G) \subseteq E^*$ unbounded by control. Any DES has some functionality to implement which are required not all possible sequences but only those providing this functionality and meeting the limitations specified. In order only to provide the required event sequences, G is term "supplemented" by supervisor S, built-in a "feedback" manner (Fig. 2).

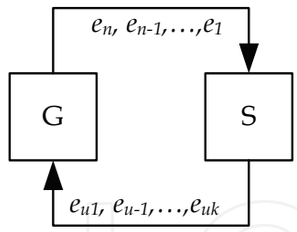

Fig. 2. The scheme of object – supervisor interaction

The scheme in Fig. 2 is no different from the conventional structure "control object – control system" but the behaviour is absolutely different. First, a generator event sequence covers all events in the system; second, a supervisor sequence includes only controlled events and third, controlled event e_k is incorporated into G output sequence conditioned to its presence also in S sequence. This allowed to define S transparently enough as a function of strings from the set $L(G)$: $S : L(G) \to 2^E$.

Supervisor S is equipped with a mechanism of G sequences blocking provided they do not meet limitations. For this purposes, S structure comprises one more component allowing for G "free" behavior restriction – a specification K. For the real object, a certain functionality (depending on G destination) must consider a multitude of all types of requirements and limitations $R = \{r_i \mid i=1,..,n\}$. As a rule, R is formed reasoning from physical, process and

design limitations imposed on joint behaviour of separate *G* components. The allowance for all restrictions *R* gives rise to $K \subseteq L(G)$ – a language of specifications – a subset of sequences dictated by *G* functionality. Actual control scheme stated in (Van Brussel et al., 1987) is presented in Fig. 3. It took the name of "Supervisory control theory" or RW approach (named after its authors J. Ramadge J. and W. Wonham W).

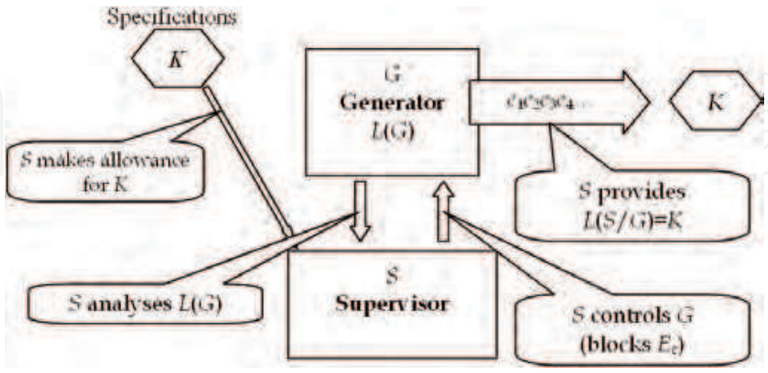

Fig. 3. Interrelationship of supervisory control components in DES

The functioning of *G* in the presence of *S* is denoted by *S/G* and a corresponding language – *L(S/G)*. The scheme symbolically shows that specification *K* is involved in *S* forming and in providing blocking. Supervisor is designed, with *K* taken into account, in such a way that, in accordance with *L(G)* observation results, *S* blocking mechanism provide the language *L(S/G)* = *K* at DES output. We would like briefly to dwell upon the way *L(S/G)* generation is realized. *G* is supposed to have its own controller that generates control events while a supervisor blocks the events the occurrence of which runs counter to the specification (Fig.4).

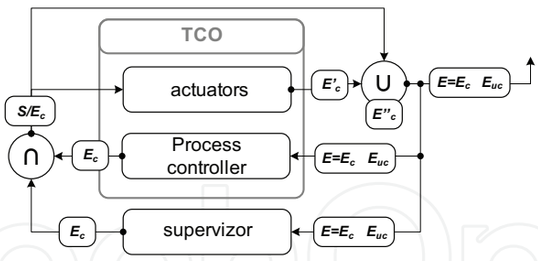

Fig. 4. Control scheme proposed in the paper (Ramadge, Wonham 1987)

Supervisor *S* monitors *G* output events and permits all E_{uc} events, while as to E_c events, it is "entitled" to permit or not permit them (to block by imposing limits on transition function $\delta(q_i, e_c) \coloneqq q_j$). For every string $s \in L(G)$ generated by *G* under *S* control, a supervisor only permits a set $\{S(s) \cap \Gamma(\delta(q_0, s))\}$ – a set of events admissible in *G* current state $\delta(q_0, s)$ and not conflicting with *K*. Hereinafter, $\delta(q_0, s)$ will mean a state *G* transfers to from q_0 as affected by *s*. In other words, *G* cannot realize the event from its current active event subset $\Gamma(\delta(q_0, s))$ unless this event is contained also in *S(s)*. However, making allowance for the fact that *E* is subdivided into **controllable** and **uncontrollable** subsets and the appearance of the latter is limited by nothing, supervisor *S* is called **admissible** if for all $s \in L(G)$, always $E_{uc} \cap \Gamma(\delta(x_0, s)) \subseteq S(s)$, i.e. *S* is specified in such a way that in all states it is impossible to block an

uncontrollable event and vice versa: *S* blocks the events not meeting limitations (irrelevant to *K*). Further on, only admissible supervisors will be considered.
For the modelling of DES with passive actuators in paper (Chalmers, Golaszewski, Ramadge, 1987) it is suggested that the model should be expanded with forced controllable events and a new control scheme (Fig. 5), with controllable events generated by supervisor, is developed. For such model, the terms of controllability for specification language are also defined.

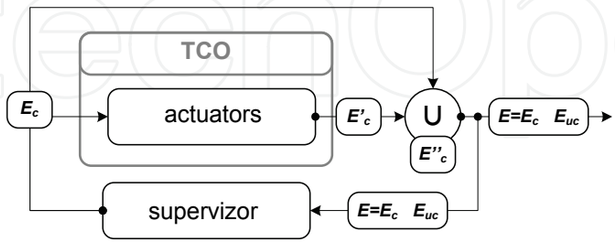

Fig. 5. Control scheme for DES with forced controllable events

For both models were developed the methods of supervisor synthesis as a finite state machine (FSM) with output converters regulating blocking (or generation) of E_c events. However, for the methods proposed the number of supervisor S states is less or equal to the product of the number of states for *G* and *K* (Cassandras, Lafortune, 2008).
DES dynamics is interpreted in the sense that the system (a pair of *G* and *S*), once set to the initial state, operates off-line, reacting to internal and external events, and provides a resulting flow relevant to *G* structure and *S* control.
Since 1987, there have been a lot of publications on DES subject-matter. At three last world IFAC Congresses, three sections on DES theory were working; IFAC Committee on DES theory was established; symposiums on this subject-matter are held. The paper scope limitation does not allow to survey the results on DES theory so we shall confine ourselves to listing the basic research trends. They are as follows:

- Study of DES as a dynamic system with a certain range of states and a structure of event transitions; the study of properties of the languages generating DES from the position of general control theory and the definition, in terms of language properties, of controllability, observability, attainability, safety (avoiding blocking situation) and some others;
- Study of different models of *G* and *K* specification (finite state machines, Petry nets etc) and the development of synthesis (engineering) methods for supervisor *S* on *G* and *K*;
- Assessment of supervisor complexity at synthesis with FSM models of *G* and *K* involved;
- Study of different modular presentations of supervisor *S* in the form of parallel generators of sub-languages with their subsequent combining via product operation (conjunctive scheme), via parallel composition operation (disjunctive scheme) and others;
- Development of programming methods for logical controllers in industrial systems with supervisor control theory applied;
- Creation of program verification methods for industrial systems with DES, as simulation instrument, applied;
- Development of the methods of industrial system state diagnostics using DES as a modelling instrument.

A detailed survey of the results obtained on DES can be found in (Cassandras, Lafortune, 2008); herein the major results on controllability from (Ramadge, Wonham, 1987; Ramadge, Wonham, 1989) are set forth:

- Is formulated the condition of controllability for the language: $K \subseteq L(G)$ is controllable if $\overline{K}E_{uc} \cap L(G) \subseteq \overline{K}$
- It is proved that if K is controllable, there exists a non-blocked S such that $L(S/G) = K$
- Are developed the methods to design supervisor S as a function of strings (Ramadge, Wonham, 1987; Cassandras, Lafortune, 2008).

However, the direct practical application of the proposed models and methods is confined to lab examples of dynamic DES engineering and supervisor synthesis. Such constraint is explained by high dimensionality of the object states set. To analyze for controllability, a complete DES specification of generator G is required. Even in the simple example given here below (a machine with four mechanisms) the number of states equals 4356. (The number can be considerably reduced with DES structural features taken into account).

Main direction of works focused on overcoming supervisor synthesis complexity is based on different kind of modularity. Methods of modular supervisor synthesis for G, as a single entity, are elaborated. At this, different control schemes are explored (disjunctive, conjunctive, hierarchical, generalized). Pioneer work (Ramadge, Wonham, 1989) that initiated the development of modularity, as applied to DES theory, was evolved and generalized in (Yoo, Lafortune, 2002). Later, different authors (De Queiroz, Cury, 2000; Gaudin, Marchand, 2003) developed the methods of modular supervisor synthesis on modular description $G=<G_1, G_2, \ldots, G_n>$ and modular specification $K=<K_1, K_2, \ldots, K_n>$ of modular S. However, the complexity of such synthesis and weak correspondence of the initial specification structure to the resulting supervisor make the methods proposed scantily attractive for practical implementation. Besides, controllability properties are verified on language models K and $L(G)$ defined for the object (Plant) as a whole, which makes it difficult to apply these results to real industrial facilities.

The present paper sets the task to develop a prototype of structured dynamic DES by structuring the object components according to their functionality. To operate the model, the paper proposes the methods that will allow to raise the dimension of supervisor control tasks and form a theoretical basis for a new supervisor control engineering technique. Structured are all three DES components but mainly object model and specification.

3. Structured Discrete Event Systems (SDES)

3.1 Base concept – the structuring of events and specifications

The author considers it promising to develop a supervisory control theory in the direction of structuring the events according to their role in production operations and in the required object behaviour specification. This research is based on two specific machinery features from DES-modelling point of view. The first feature relates to the fact that for discrete machinery a set of events is usually subdivided into three sets. These are sets of controllable and uncontrollable events E_c and E_{uc} (typical for DES theory) and E_w is a set of expected events. The events from E_w simulate states (positions) of actuator(s) or object components. Supervisor cannot block E_w events as those controllable from E_c and thus E_w events are traditionally referred to uncontrollable events as per Wonham's classification (Ramadge and Wonham, 1987). However, E_w events are expected to occur as a response to E_c events – a

confirmation of the fact that the commands sent to actuators were executed. So, the foregoing gives the ground to mark out E_w events as a separate set. The second specific feature is as follows: the behaviour of every actuator G^i is simulated by the language $L(G^i)$ of words over $E^i = \{E^i_w \cup E^i_c\}$ and the specification of desired behaviour is formulated as a language K over events $E_d = E_c \cup E_{uc}$, a totality of commands and conditions of their use. Making the allowance for these specifics, makes it possible to get numerous advantages both in defining DES and formulating controllability conditions and supervisor synthesis.

3.2 SDES definition

Definition 1: If the structure of DES is defined by: a collection of components $G=<G^1,G^2,\ldots,G^n>$; sets of E_i events, each being structured on $E^i = \{E^i_w \cup E^i_c\}$, and a set E_{uc} of general uncontrolled events; the behaviour of each DES component being defined by FSM $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q^i_m, q^i_0 \right\rangle$ and $L(G^i)$ language, then the DES with the above structure is called well structured.

A set of common events for $G=<G^1,G^2,\ldots,G^n>$ is defined through the union of subsets $E = \{E_w \cup E_c \cup E_{uc}\}$, where E_w and E_c each are the unions of appropriate component subsets.

Note 1: Sets E_w and E_c for various mechanisms do not intersect, since various mechanisms have their own actuators and their states are individual.

Note 2: Components of G^i define the behaviour of G that is not limited (controllable) by anything, e.g. from the successive operation of $<G^1,G^2,\ldots,G^n>$ in any order up to their independent work in parallel.

According to the theory of supervisory control, a parallel composition of all object components is implemented, and, as the result, a model of uncontrollable object behaviour is created (Ramadge & Wonham, 1987). The narrowing of free behaviour is carried out with the constraints of purposeful joint behaviour considered. This, in essence, is the procedure of adapting the initial unlimited behaviour i.e specifying the behaviour as required by application. We would like to remind that the implementation of all restrictions generates a language $K \subseteq L(G)$ called a language of specifications. Establishing the restrictions is a creative process that requires an experimental approach to achieve a reliable result. Such experiment is quite difficult to carry out as the number of states is increasing in the course of composing. There is a collision.

On the one hand, a system analyst needs to get a general picture of all the transitions to analyze their admissibility.

On the other hand, it is unreal to do it for complete composition, since the number of states in it is too high (for practical applications this number is about $n \cdot 103$). Sequent revealing of restrictions in the process of pair-wise composing, gives a ground to doubt of such restrictions completeness or, on the contrary, of their extreme strictness. At the same time, there is no possibility to consider the joint action of components with those absent in the composition.

At the same time, it is known from the practice of discrete process engineering that the efficient behaviour of discrete systems is achieved by solving two control tasks, namely: operation control and control of operation sequence. Operation control is provided by the execution of a certain command and monitoring the corresponding object response. Commands and their reactions once defined, are iterated in various places of the sequence of operations. In process modelling, it is important to set up the sequence of commands and

to evaluate the completeness and correctness of conditions. With the above in view, herein is proposed to create a specification of a well-structured DES with the events $E_d = \{E_c \cup E_{uc}\}$, i.e. combination of commands and conditions for their execution in sequence.

Definition 2: The language $K \subseteq E_d^*$ defined by FSM $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ as a set of strings defining the required specifications, is called a directive specification language (a process specification tapes language).

It is assumed that FSM *H* has no deadlocks (Fig. 6) and livelocks (liveloops, within which *H* fails to go out of a certain state subset and does not reach Q_m and then q_0), i.e. *H* is non-blocking.

It is worthy to be noted that if a graph is strongly connected and $q_0 \notin Q_m$, then q_0 transitions only as shown in Fig. 6 are possible.

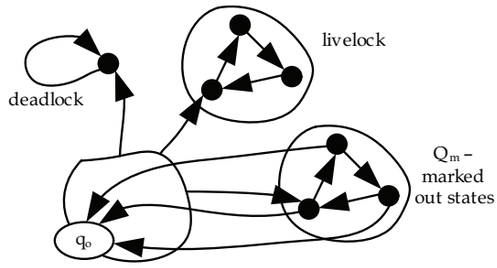

Fig. 6. Types of fragments in the machine *H*

The fact of non-blocking is easily verified. Contrary to the general DES theory (Cassandras, Lafortune, 2008), where deadlocks and livelocks result from the excessive general description via the product and composition, in SDES, there should be no hurry in cutting down "bad" states and transitions but, vice versa, it is necessary to check if any transition is missed to avoid deadlock or livelock situations.

Let's define a supervisor for *G* and *K*. It is conceptually evident, that supervisor is an operator that defins, for every string s, which of possible events, admissible for *G*, are suitable as the next event not conflicting with *K*. At this, supervisor remains admissible in terms of (Van Brussel et al., 1998) since it in no way limits E_{uc} occurrence and affects only E_c.

Definition 3. Supervisor *S* is a converter of strings admissible for the system $G = \langle G^1, G^2, ..., G^n \rangle$ initial state to the events $S(s) = \{\varepsilon \cup E_{uc} \cup \{e_c\}\}$ such that: first, these are any of uncontrollable events E_{uc} (i.e. S is admissible for *G*); second, these are controllable events e_c admissible for the current *G* state; third, these events do not cause blocking of *S* and $G = \langle G^1, G^2, ..., G^n \rangle$ composition.

Let's denote, as agreed, by *L(S/G)* the language generating *G* under *S* control. It is evident that $L(S/G) \subseteq L(G)$. Let's also give a definition of *L(S/G)* language generating *S/G*, that is consistent to the conventional definition of language generating *G* under *S* control.

Definition 4. The language *L(G/S)* generating $G = \langle G^1, G^2, ..., G^n \rangle$ under *S* control contains the following strings:

1. $\varepsilon \in L(S/G)$;
2. $\forall s, e(s \in L(S/G) \wedge e \in S(s)) : se \in L(G) \Leftrightarrow se \in L(S/G)$

In other words, any string *se* belongs to *L(S/G)* provided it also belongs to *L(G)* being at the same time the extension of string *s* which also enters *L(S/G)* by event e such that $e \in S(s)$. Possibly, $s = \varepsilon$.

Definition 5. A well-structured DES, for which the uncontrollable part is set up by definition 1, the desired behaviour is set by specification language $K \subseteq E_{d^*}$ $(K \neq \emptyset)$, and which is supplied with a supervisor S such that K is fulfilled, is called a structured dynamic discrete event system (SDES).

K fulfilment means that $P_{E_d}(L(S/G)) = K$, i.e. that K will be equivalent to the projection on E_d of $L(S/G)$ language that is generated by S/G.

3.3 Technical object modelling by structured DES

The events associated with real industrial objects, as a rule, are easily divided into groups (types) as proposed herein. Such event grouping is typical for process systems of many industrial spheres. Here below is the example which refers to the field of mechanical metal-working. We consider this example most interesting since it is close to illustrative examples frequently used in publications on DES (Ramadge, Wonham, 1987; Ramadge, Wonham, 1989; Chalmers, Golaszewski, Ramadge, 1987; Ambartsumyan, 2009).

The structuring of technical object (the first phase of study) includes as follows:

- enumerating actuators;
- defining for each of them the set of events necessary and sufficient for the outer supervisor to identify actuators behaviour;
- defining the classification of marked out events;
- defining the components and object behaviour in the compact-form languages, e.g. finite machine models.

In Fig. 7 a kinematical scheme of a small milling machine is presented. The machine consists of 4 mechanisms: "workpiece clutch" - G^1, "turntable" – G^2, "spindle" – G^3 and "cutter" – G^4

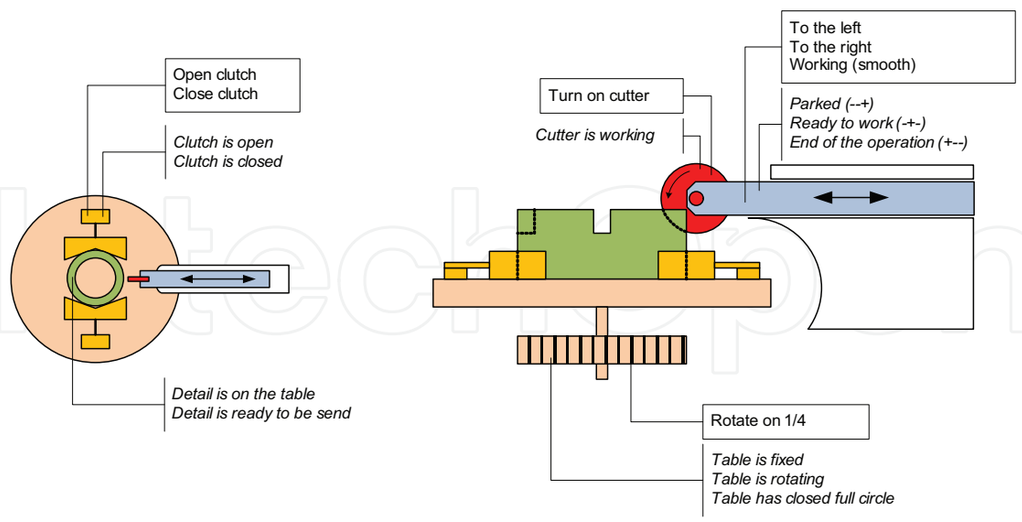

Fig. 7. Kinematical model of the machine

Let's enumerate the events and their semantics in the liveloop (behaviour) of each mechanism.

"Workpiece clutch" mechanism: $e_{1\text{-}1}$ – to clamp, $e_{1\text{-}2}$ – clutch closed, $e_{1\text{-}3}$ – to unclamp, $e_{1\text{-}4}$ – clutch closed, $e_{1\text{-}5}$ – clutch is moving.
"Turntable" mechanism: $e_{2\text{-}1}$ – to lock the table, $e_{2\text{-}2}$ – table locked, $e_{2\text{-}3}$ – to unlock the table, $e_{2\text{-}4}$ – table unlocked, $e_{2\text{-}5}$ – locker is moving, $e_{2\text{-}6}$ – to make a ¼ turn, $e_{2\text{-}7}$ – table is moving, $e_{2\text{-}8}$ – table is turned, $e_{2\text{-}9}$ – to switch off turning gear, $e_{2\text{-}10}$ – table stopped.
"Spindle" mechanism: $e_{3\text{-}1}$ – to move spindle fast to the left, $e_{3\text{-}2}$ – feed zone, $e_{3\text{-}3}$ – working position, $e_{3\text{-}4}$ – to move spindle to the left, $e_{3\text{-}5}$ – working zone, $e_{3\text{-}6}$ – operation finished, $e_{3\text{-}7}$ – to move spindle to the right, $e_{3\text{-}8}$ – to move spindle fast to the right, $e_{3\text{-}9}$ – parked.
"Cutter" mechanism: $e_{4\text{-}1}$ – to turn on cutter, $e_{4\text{-}2}$ – cutter working, $e_{4\text{-}3}$ – to turn off cutter, $e_{4\text{-}4}$ – cutter stopped, $e_{4\text{-}6}$ – cutter unstable spinning.
Mechanisms behaviour, as agreed here above, will be considered as sequences (strings) of possible events. These sequences will be defined as finite state machines (Fig. 8–11). Hereinafter they are called component finite machines (CFM). It is easily seen that CFM transition graphs and graph edges weighed by events, specify operation of each mechanism quite transparently.

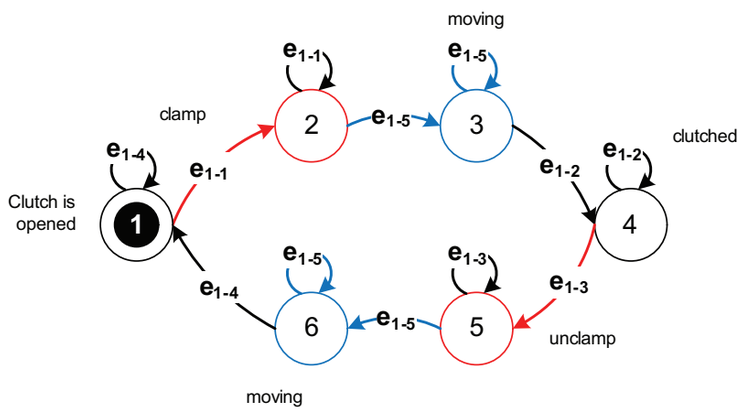

Fig. 8. G^1 CFM – a model of "Workpiece clutch" mechanism

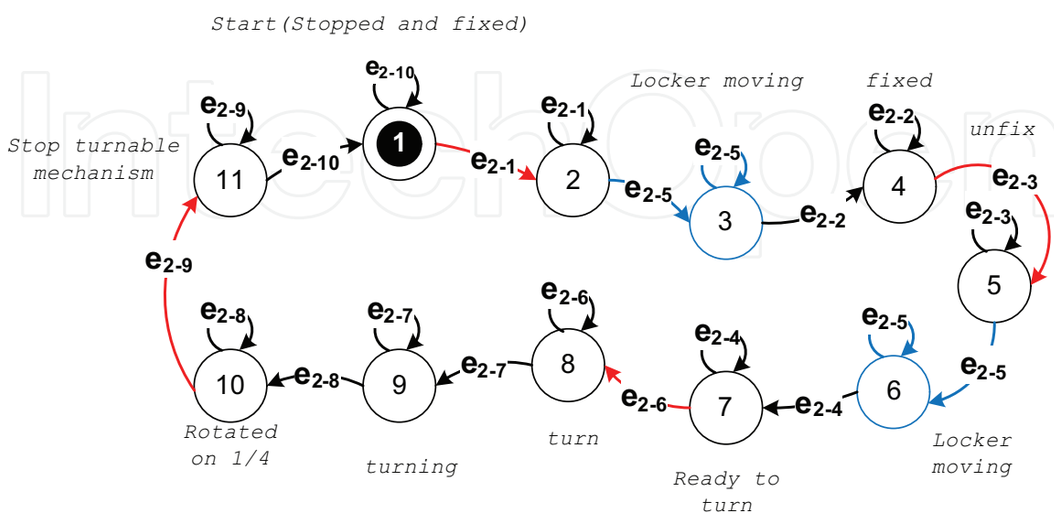

Fig. 9. G^2 CFM – a model of "Turntable" mechanism

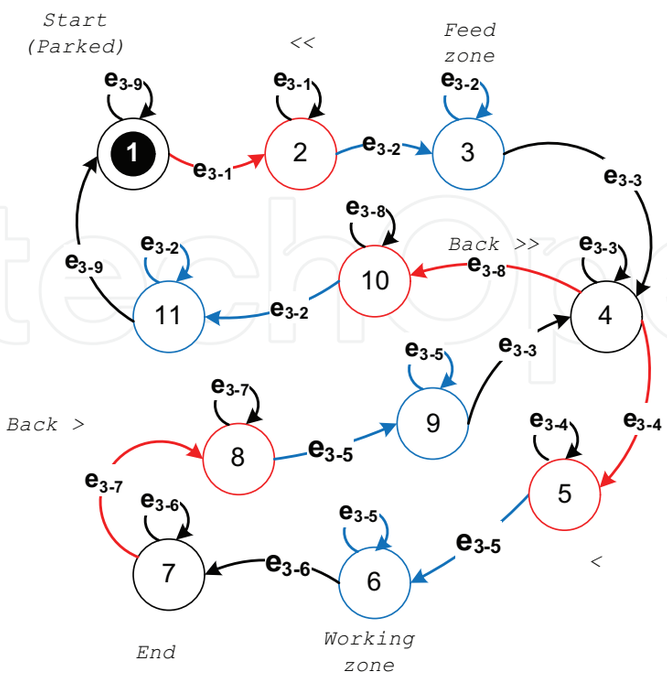

Fig. 10. G^3 CFM – a model of "Spindle" mechanism

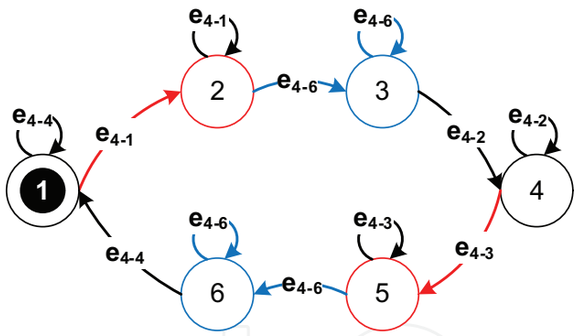

Fig. 11. G^4 CFM – a model of "Cutter" mechanism

It is easy to make natural event grouping in all the CFM, namely:

G^1 – $E_c^1 = \{e_{1-1}, e_{1-3}\}$, $E_w^1 = \{e_{1-2}, e_{1-4}, e_{1-5}\}$; G^2 – $E_c^2 = \{e_{2-1}, e_{2-3}, e_{2-6}, e_{2-9}\}$,

$E_w^2 = \{e_{2-2}, e_{2-4}, e_{2-5}, e_{2-7} e_{2-8}, e_{2-10}\}$; G^3 – $E_c^3 = \{e_{3-1}, e_{3-4}, e_{3-7}, e_{3-8}\}$,

$E_w^3 = \{e_{3-2}, e_{3-3}, e_{3-5}, e_{3-6}, e_{3-9}\}$; G^4 – $E_c^4 = \{e_{4-1}, e_{4-3}\}$, $E_w^4 = \{e_{4-2}, e_{4-4}, e_{4-6}\}$ and to see the events $E_{uc} = \{e_{ex-1}, e_{ex-2}, e_{ex-3}, e_{ex-4}, e_{ex\text{-}s}, e_{ex-w}\}$ common for all components (respectively: a workpiece is on the table; a workpiece is removed from the table; processing is over, clutch of s type , clutch of w type).

Note 3. Sets E_w and E_c for different mechanisms do not intersect.

It is evident, since different mechanisms have their own drivers and their positions for each mechanism are individual.

The next stage of a technical system SDES-modelling is the defining of the system behaviour specification based on the requirements to the system functionality and limitations. It is

done by forming the behaviour of *G* as an uncontrollable system, as a whole, followed by putting in limitations, thus "narrowing" *G* behaviour up to that required.
The traditional approach being applied, uncontrollable *G* behaviour is defined by component machines combination. Let's use two mechanisms of the above milling machine (Turntable and workpiece Clutch) to illustrate this.

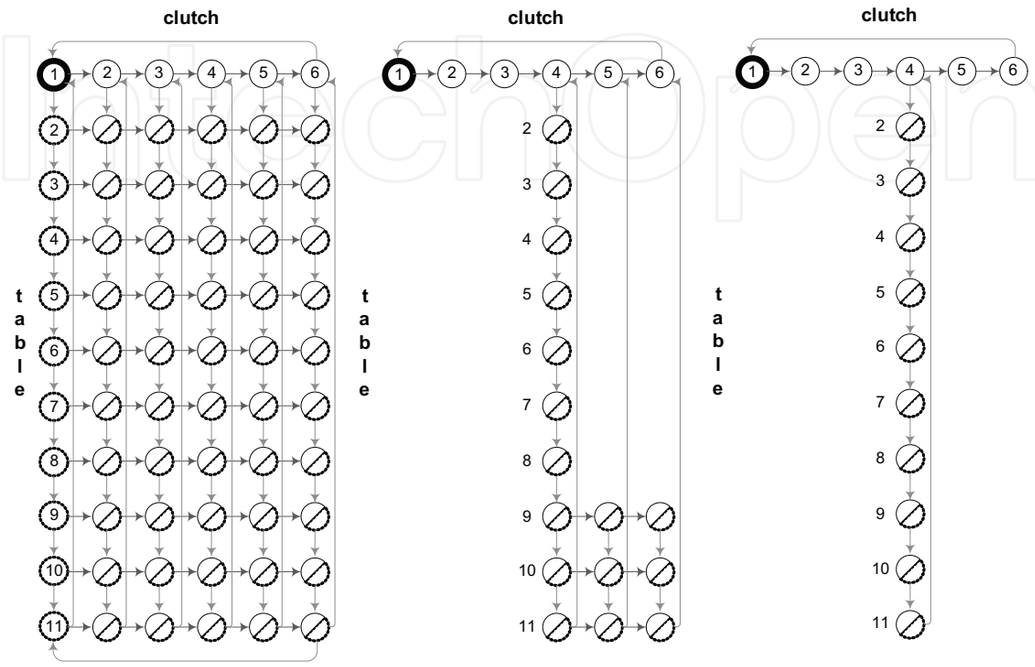

Fig. 12. CFM composition for G^1 and G^2: a) complete; b) with allowance for limitations r_1 and r_2; c) with allowance for limitations r_1, r_2, r_3

Pursuant to SC theory, we should make a composition of all machines to achieve "uncontrollable" *G* behaviour. DES, modelling "uncontrollable" behaviour of the first two mechanisms, is represented by $G^1 \oplus G^2$ composition, with relevant transition graph structure illustrated in Fig. 12-a. Here a structure of initial components transitions is shown: across - G^1 structure, down - G^2 structure, and relevant pairs are represented by nodes at arrows intersection. Edges weighing corresponds to weighing of transitions in the initial components.
Machine $G^{1\otimes 2}$ represents unlimited by anything, parallel operation of mechanisms G^1 and G^2 originating $L(G^{1\otimes 2})$ language.
In our example, the following restrictions as to joint behaviour of the mechanisms take place: r_1: "turning of G^2 "Turntable" mechanism is possible if a workpiece is clutched"; r_2: "if in the course of the table turning a workpiece unclasping begins , "Turntable" will only terminate turning".
The implementation of these technological restrictions are formally realized by banning the following state compositions: 1, 2, 3 of G_1 CFM and 2-9 of G_2 CFM. With these limitations applied, all pairs of states under verticals 1, 2, 3 and a number of pairs under verticals 5, 6 are excluded (Fig. 12-b). The same refers to their incident transitions. As the result, we get the machine K_1 as shown in Fig. 12-b. More detailed analysis of admissible transitions results in the necessity of one more limitation: r_3 – "at table turning, a workpiece unclasping is inadmissible", which makes specification more strict (K_2) as shown in Fig. 12-c.

Thus, we have DES of $G^{1\otimes 2}$ and it's necessary to provide its operation within the framework of language *K*. In what way is it possible to regulate a path choice in $G^{1\otimes 2}$ graph? In our example, for $G^{1\otimes 2}$ $E_c^{1\otimes 2} = \{e_{1-1}, e_{1\text{-}3}, e_{2-1}, e_{2\text{-}3}, e_{2-6}, e_{2-9}\}$, $E_w^{1\otimes 2} = \{e_{1-2}, e_{1\text{-}4}, e_{1-5}, e_{2-2}, e_{2\text{-}4}, e_{2-5}, e_{2-7} e_{2-8}, e_{2-10}\}$.

Graph transition trajectory can be regulated by a function of transitions $G_{1\otimes 2}$ by blocking or accepting the events from E_c set with the help of supervisor *S* (outer to *G*) which dynamically interacts with *G* in a feedback manner. The way it can be realized is illustrated by our example. In state, $q_{1,4}$ in cycles 1, 2 and 3 of the table operation, a supervisor each time enables $e_{2\text{-}1}$ and disables $e_{1\text{-}4}$, and, after the table returns to its initial position for the 4-th time, it is $e_{1\text{-}4}$ that is admitted and $e_{2\text{-}1}$ that is banned.

So, CFM sequential merging and the detection of limitations for CFM joint operation are quite a complicated procedure even in our case. We have already noted that the detection of limitations in the course of pairwise component combination, gives the ground to doubt about the completeness of such limitations or vice versa in their excessive strictness. Besides, there is no possibility to predict the consequences of joint operation with the components still absent in the composition. For example, should we start CFM merging with "Spindle" and "Turntable" mechanisms, it will in no way possible to make allowance for the fact that between their "activities" a locker actuation will take place.

At the same time, for technical objects, their required behaviour is always defined by their functionality that is specified, for example, by text description. The required machine behaviour is presented by informal specification in table 1.

1) on arrival, the piece is locked by clutch; 2) after clenching, the spindle moves from park position to work position (to the left); 3) the cutter is switched on; 4) smooth feed to the left utmost position (operation is over); 5) the spindle moves to the right back to work position;	6) positioner makes a ¼ table rotation; 7) after the table is fixed, the next operation is carried; 8) after the table makes a turnover, the spindle is parked, the clutch is unclamped, the signal of the piece readiness is sent; 9) prior to parking, to switch off the cutter and wait for a stop.

Table 1. Text description of initial specification

At SDES-modelling, at this stage, a specification of joint behaviour in *K* language is applied. A specification, compliant with the text specification, is presented by machine $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ shown in Fig. 13.

Since the verbal behaviour description, as a rule, is inaccurate, the resulting specifications may vary. The example of another interpretation of verbal description is presented in Fig. 14. The specification is described in conformity with verbal description. Basing on the information from table 1, it is possible to assume that at the beginning of operation, the table is fixed, since otherwise is not specified and thus, the operation relevant to the transition graph node 3 is omitted. However, should the order of operations as shown in Fig. 14 be accepted, already the processing of the second workpiece will start with the table unfixed since in the beginning of the large loop locker is not considered. The necessary operation is missing.

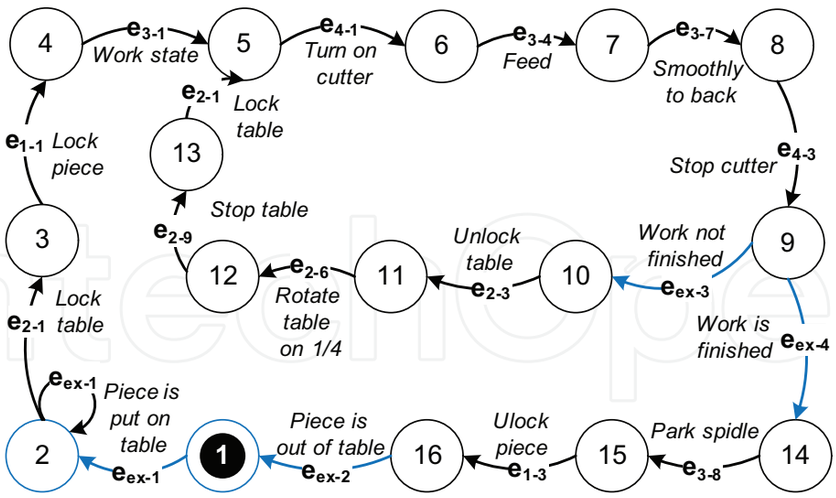

Fig. 13. The required machine behaviour in terms of directive specifications. The semantics is as follows: e_{ex-1} – a workpiece is on the table, e_{ex-2} – a workpiece is removed from the table, e_{ex-3} – processing is not over, e_{ex-4} – processing is over (other events semantics was given here above in the mechanisms description).

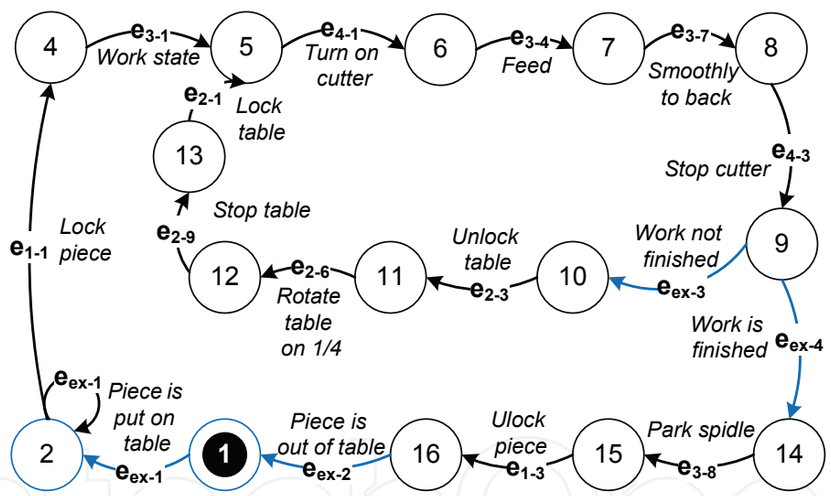

Fig. 14. *H* specification with erroneous missing of "Table locking" operation

Operation omission is far from being the only inconsistency in the required behaviour specification. Here below (Fig. 15) another text description interpretation is given. The specification is elaborated in accordance with the text but a "cutter halt" operation (node 8 of Fig. 15) is performed prior to cutter parking in the "large" loop, which follows from item 9 of the text description from Table 1. Cutter halt is performed in the "large" loop but on the processing termination, therefore, while processing the second piece position, the attempt will be made to switch on a working cutter.

Note3. The composition of modular hierarchic DES description of solely unblocked modules may result in DES blocked operation.

This stage of SDES-modelling reveals a principle difference of discrete control engineering with supervisor *S* on G and *K* given, as compared with a "black box" technique.

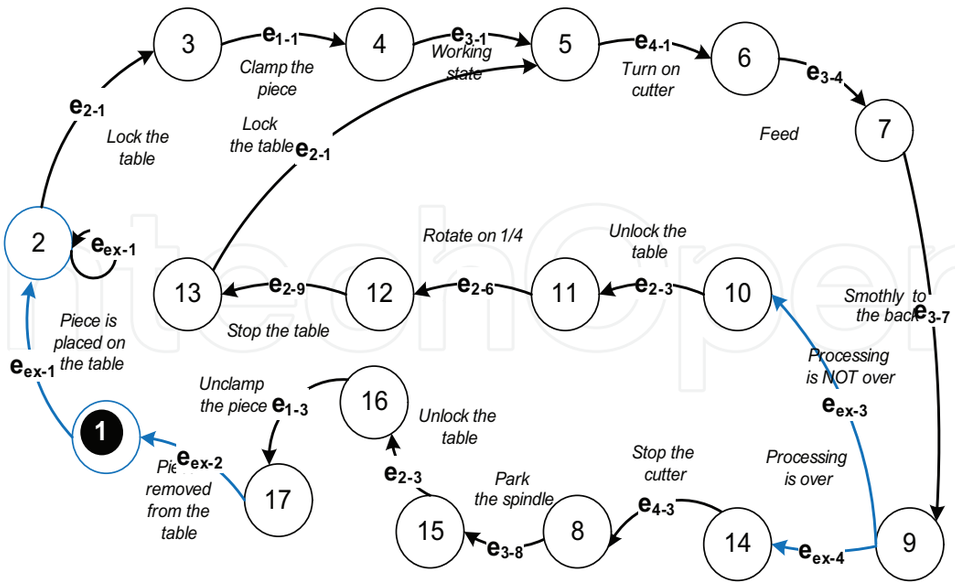

Fig. 15. Machine behaviour as described in the language of directive specifications, with a "Cutter halt" operation moved to the large loop

Indeed, if we make quite a transparent substitution of CFM operations in the transition graph of *H* specification and properly apply the functions of outputs (to be shown here below), we shall get a controlling finite state machine. This machine, provided inputs are independent (this being an indispensable condition for conventional logical control according to the "black box" scheme), will precisely perform the operation sequences specified. Note that substitutions can be made for each of three specifications and, thus, three different controlling machines will be obtained. Later on, it will be possible to carry out arbitrarily profound optimization applying all the methods used in the finite machine theory and logical synthesis. However, at the attempt to unite a control object and $G = \langle G^1, G^2, ..., G^n \rangle$ machines, obtained as per specifications presented in Fig. 15, 16, the errors, mentioned here before, will reveal themselves in blocking (non-fulfilment) of some commands and a "hanging" – an unforeseen cyclic operation interruption will occur. At the same time, with DES theory analytic methods applied, possible blocking situation will be revealed analytically. It is evident that once DES theory methods are applied, a "dimension damnation" will manifest itself: CFM parallel composition of the example in question already gives a machine with the number of states equal to 4356 and its composition with *H* machine results in the machine with dozens of thousands states.

So, we face the following problem: how to predict blocking situation without composition of G_i in *G* followed by general composition with *K*. To tackle this problem, let's continue considering the theory of SDES-modelling.

4. Features of the models of *G* components and H specification

We would like to point out a number of important features of the models of $G = \langle G^1, G^2, ..., G^n \rangle$ components and specifications of industrial objects. Model components, as a rule, simulate the behaviour of different actuators able to "perceive" events-commands,

react to them by the change in the position (location, speed, pressure, level, temperature, flow rate etc), with a set of space co-ordinates being split up into a number of intervals and presented by events. Since space, though presented by a set of events remains physical, the events in it may "happen" in a certain order.

Feature of expected events (F1). For the events $e \in E_w^i$ of one component, there exists ordering based on consecution of $e_{i1}, e_{i2}, \ldots, e_{in}$ such that in any chain of these events on graph, the events are arranged in direct or reverse order (this also refers to e_{i1} and e_{in}). Furthermore, this relation is also valid for neighbouring graph chains.

Feature of operations (**F2**)**.** The events $e \in E_w{}^i$ weigh on G^i graph the chains of transactions – transitions (edges and states), with one edge and state, weighed by $e \in E_c^i$ (event-command), adjoining to this chain on the left side, and on the right side, either an edge and state, also weighed by another command $e \in E_c^i$, or a fork with events $e \in E_{uc}^i$. This feature allows to unambiguously mark out process operations - the substrings relevant to the command and the reaction expected, on G^i graph (i.e. to "colour" graph). Then, uncoloured will be left only the edges corresponding to $e \in E_{uc}^i$.

Example 1. G^1 operations (Fig. 8) are as follows: To clench piece: states 1→4, chain – $\langle e_{1-1}, e_{1-5}, e_{1-2} \rangle$; to unclench piece: states 4→1, chain – $\langle e_{1-4}, e_{1-5}, e_{1-6} \rangle$.

Example 2. G^3 operations (Fig. 9): Quick feed to the left: states: 1→4, chain - $\langle e_{3-1}, e_{3-2}, e_{3-3} \rangle$; operational feed to the left: states: 4→7, chain - $\langle e_{3-4}, e_{3-5}, e_{3-6} \rangle$; slow retraction to the right: states: 7→4, chain - $\langle e_{3-7}, e_{3-5}, e_{3-3} \rangle$; spindle parking: states: 4→1, chain - $\langle e_{3-8}, e_{3-2}, e_{3-9} \rangle$.

Feature of forks separability in *G* and *H* (F**3**)**.** Any fork in the transition graph (both for G^i and H) is weighed by the events from $\{E^i{}_c \cup E^i{}_{uc}\}$ in a separate way, i.e., branching is always either on $e \in E_{uc}^i$ or on $e \in E_c^i$: $\forall j : [\left|\Gamma_j^i\right| \geq 2] \Rightarrow [\forall e \in \Gamma_j^i : e \in E_{uc}^i \mid e \in E_c^i]$.

Forks in transition graphs are limited, as a rule, to provide one-to-one description and implementation. For example, mixed branching (Fig. 16) is difficult to interpret. Since a transition, particularly in object, has some delay, then, when analysing q_i state (Fig. 16), it is expedient to introduce a new fact – an event $\hat{e}_{uc}$, negating the initial event e_{uc}, and to transform the initial specification in corresponding transitions as shown in Fig. 16.

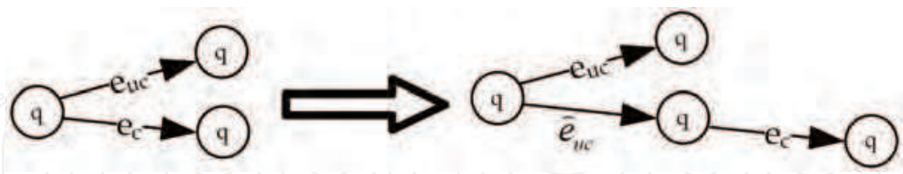

Fig. 16. Transformation of mixed branching

Feature (F4) of marked states $q_j \in Q_m^h$ for H. All edges leading to $q_j \in Q_m^h$ are weighed by $e_l \in E_c$. This is a feature of terminated fragments: specific action is performed last.

Feature of uniqueness in use of operations in *H* (F5). Every edge of H graph can be associated with one chain from G^i graph. Should in the description be any ambiguity, it can be easily eliminated by duplicating the corresponding fragment of H graph. For actuators, all operations of which are associated with different commands, this feature is always valid. If there are still same commands executed at different "path" sections (a fragment of sequence from E_w), they can be always described in different fragments of H graph by duplicating the initial paths.

The features presented are applied to choose the principle of role structuring, as a basis of two-level SDES, and are used in $\Re(s)$ algorithm of carrying out the experiment (refer to i. 5.1), actually, replacing the operation of component machine composition.

5. SDES study

It is natural to inquire, what properties the behaviour specification in language K should possess to provide a supervisor which ensures behaviour $G=<G^1,G^2,\ldots,G^n>$ according to the specification, and at the same time is admissible for G. The answer to the question is associated with controllability study (in terms of Ramadge and Wonham, 1987)) of the language K which is a specification of the required behaviour of G defined by a set of components $<G^1,G^2,\ldots,G^n>$. As the basic method to study joint G and H behaviour, it is proposed to experiment with $<G^1,G^2,\ldots,G^n>$ by strings $s\in K$ (such, that $\delta_h(q_0, s)!$, i.e. admissible for the initial state q_0). The experiment point is to simulate operation of component machines driven by events-commands from strings $s\in K$. The algorithm of such experiment is given here below.

5.1 Algorithm $\Re(s)$ of the experiment with SDES of $G=\left\langle G^1,G^2,\ldots,G^n\right\rangle$ by string s

The assignments and functions used in $\Re(s)$ algorithm are as follows: $\mu(s)$ – length of string s; $s(i)$ – current event in string s, with i being the symbol number in string s; $\theta(e)=\{c \mid uc \mid w\}$ – function of event type; $N(e)\in\{0,1,\ldots,n\}$ – number of component G^k in set $G=<G^1, G^2,\ldots, G^n>$ such that $e\in E^k$. If $N(e)=0$, then $e\in E_{uc}$ refers to common uncontrollable events of G. $VIS=\left\langle q^1_{r,1},q^2_{r,2},\ldots,q^n_{r,n}\right\rangle$ is called a n-dimension vector of initiated states of each of $G=<G^1,G^2,\ldots,G^n>$ components.

As you see, the algorithm executes the experiment on $<G^1, G^2,\ldots, G^n>$ collection which is set componentwise to the initial states by strings $s\in K$.

The algorithm has two kind of results:

1. Logical. $\Re(s)=True\,|\,False$. If, under the consecutive influence of symbols s, all components fulfil their transitions successfully, $\Re(s)=True$. If in the course of the experiment for a certain symbol $s(i)$ a component k $(k=v(s(i)))$ fails to fulfil its transitions, i.e. $s(i)\notin\Gamma^k_j$, then $\Re(s)=False$ (refer to step 2, table 1). The latter means that H is inconsistent with $G=<G^1,G^2,\ldots,G^n>$.
2. Constructive. If $\Re(s)=True$, $\Re O(s):=v$; $\Re r(s):=r$; $\Re g(s):=k$; $\Re b(s):=j$; $\Re e(s):=l$, where (if $k\neq 0$) k is a component number of G^k, j and l are numbers of its states at which the experiment ends successfully by string s, with j being the beginning and l - the end of a substring v corresponding to the last operation of component G^k as a reaction to string s, and r is a resulting string. If $k=0$, then the experiment is successful, but v, j and l point to the operation of the last component involved in the experiment. In any case, $r\in L(G)$ is a string admissible in G and is one of s prototypes, i.e. $r\in P^{-1}_{E_d}(s)$. (It should be reminded that $r\in P^{-1}_{E_d}(s)$ is a string r, with its projection upon events E_w being equal to s).

The examples of experiments on specifications:

1. For the experiment, let's choose a string s_1 covering the beginning and a small loop: states 1, 2, …, 13, 5 (Fig. 10). Here is the string:

Step	Operation	Comment
1	Set $i=1$; $r:=\varepsilon$; $VIS=\langle q_0^1, q_0^2, ..., q_0^n \rangle$	Initial setting
2	$k=N(s(i))$; If $k=0$, then $r:=rs(i)$; go to item 5, else choose from VIS the state of component k – q_j^k ;	steps 2-5: moving along string s
	If $s(i) \notin \Gamma_j^k$, then $\Re(s)$ =False, go to item 6;	Current event of string is not admissible for G^k; the experiment failed
	Let $\delta^k(q_j^k, s(i)) = q_l^k$; If $\theta(s(i)) = uc$, then $r:=rs(i)$, go to step 4, otherwise $v := s(i)$;	Uncontrollable event injected into output string. set command from $s(i)$ to operation
3	Choose $e_r \in \Gamma_l^k$ such that $\{\delta^k(q_l^k, e_r) = q_p^k$ and $l \neq p\}$; If $\theta(e_r) = w$, then $v:=ve_r$, $l:=p$ go to step 3	Scrolling by e_w – the expected events of operation performed as the reaction to command in G^k
4	place q_l^k in VIS k-position; $r:=rv$; $v:=\varepsilon$;	Current operation is over
5	$i=i+1$; if $i \le \mu(s)$, then go to step 2; $\Re(s) :=$ True; $\Re O(s) := v; \Re r(s) := r; \Re g(s) := k;$ $\Re b(s) := j; \Re e(s) := l;$	Checking for the string end and assigning of output experiment results
6	The end	

Table 2. Algorithm $\Re(s)$

$s_1 = e_{2-1}, e_{1-1}, e_{3-1}, e_{4-1}, e_{3-4}, e_{3-7}, e_{4-3}, e_{ex-3}, e_{2-3}, e_{2-6}, e_{2-9}, e_{2-1}$. It is easy to trace that all transitions in component models will operate since the commands are given correctly. The experiment result is $\Re(s_1) = True$. We shell not adduce the resulting string but the last string operation is offered in full: $\Re O(s) := e_{2-1}, e_{2-5}, e_{2-2}$; $\Re g(s_1) = 2$; $\Re b(s_1) = 1$; $\Re e(s) = 4$. String r will include similar extensions for all the events of sting s_1.

For the experiment on erroneous graph (Fig. 13), let's choose string

$s_2 = e_{1-1}, e_{3-1}, e_{4-1}, e_{3-4}, e_{3-7}, e_{4-3}, e_{ex-3}, e_{2-3}$. It is easy to notice that G^2 is addressed first at the last event but this attempt fails since the transition from state 1 of G^2 component is not specified for the event e_{2-3} (Fig. 5). Therefore, $\Re(s_2) = False$.

5.2 Main SDES result

It is natural to ask, what properties a specification of behaviour in language K should possess to provide a supervisor making $G = \langle G^1, G^2, .., G^n \rangle$ behave in conformity with specification and not blocking G.

The answer as to the supervisor existence can be obtained using the following theorem.

Theorem of SDES controllability. Let a well-structured $G=<G^1, G^2, \ldots, G^n>$ be given, where $E = \{E_w \cup E_c \cup E_{uc}\}$, $E_d = E_c \cup E_{uc}$, and $K \subseteq E_d^*$ $(K \neq \emptyset)$. Non-blocking supervisor S, such that $P_{E_d}(L(S/G)) = K$, exists if and only if for any $s \in K$ (such that $\delta_k(q_0, s)!$) $\Re(s) = True$ with respect to $G=<G^1, G^2, \ldots, G^n>$.

In other words, the theorem asserts, that for a well-structured $G=<G^1, G^2, \ldots, G^n>$ and a given specification K, there exists a non-blocking supervisor S such, that the projection on E_d of the

language, generated by *G* under *S* control, coincides with *K* provided that for any line *s*, specified for the initial state of *H*, which defines the language of specification *K*, the experiment on $\Re$ *(s)* algorithm is positive.

Proof The necessity is proved by contradiction: the theorem terms are satisfied, unblocking supervisor *S* such that $P_{E_d}(L(S/G)) = K$ exists but for a certain string $s \in K$ (such that δ_h *(q0, s)!)* the experiment $\Re$ *(s)* = *False* with respect to $G = \langle G^1, G^2, .., G^n \rangle$. There can be a lot of such strings, but let *s* be the shortest of them. Let $\omega(s) = e$ and *s* := *ue*. It is evident that for all prefixes of string *u* and string *u* itself, the experiment is positive, i.e. $\Re$ *(u)* = *True*, but, in case *u* is extended, $\Re$ *(ue)* = *False* and $\theta(e)$ = *c* (*e* – control event). It is revealed at 2.2 (Table 1). At this, G^k is in state q_j^k and $e \notin \Gamma_j^k$ (*e* is inadmissible for G^k in its current state q_j^k).

On the other hand, as *u*, $ue \in K$ and supervisor *S*, such that $P_{E_d}(L(S/G)) = K$, exists, let's choose from $L(S/G)$ strings *u'*, *u'e* for which $P_{E_d}(u') = u$ and $P_{E_d}(u'e) = ue$. Since the sets of CFM events do not intersect (refer to note 1), $N(e) = k$, from *u'* admissibility for G^k it follows that at u' generation G^k will be transferred to state q_j^k, and then from *u'e* admissibility for G^k it immediately follows that $e \in \Gamma_j^k$ (i.e. e is admissible for G^k in its current state q_j^k). This comes into conflict with the assumption that $\Re$ *(s)* = *False*. The necessity is proved.

Sufficiency. Let's $K \neq \varnothing$ be a language such that for any $s \in K$ $\Re$ *(s)* = *True* with respect to $G = \langle G^1, G^2, .., G^n \rangle$. We shall show that in such case there exists a supervisor unblocking for *G* and providing $P_{E_d}(L(S/G)) = K$.

Let's define language *M* on *K* in the following way:

$$M = \left\{ \varepsilon \cup \bigcup_{s \in K} \Re r(s) \right\} \tag{1}$$

For any $u \in M$ let's define:

$$S(u) = \left\{ e : \begin{cases} (\theta(e) = uc \wedge \Re(P_{Ed}(u)e) = True), \\ (\theta(e) = w \wedge e \in \Re O(P_{Ed}(u)e)), \\ (\theta(e) = c \wedge \Re(P_{Ed}(u)e) = True), \\ \text{other } e \text{ not included in codomain } S \text{ for } u \end{cases} \right\} \tag{2}$$

The designed converter admits as follows:

- all E_{uc} possible (as to transition function for *G*) after *u* (string 1 from (2));
- all E_w if they fall into the definition area of corresponding transition in a certain G^k component (string 2 from (2));
- all controlled events E_c for which the experiment on $P_{E_d}(u)e$ is positive (string 3 from (2)).

Thus, the converter is a non-blocking supervisor such that $L(S/G) = M$ and $P_{E_d}(L(S/G)) = K$ (this follows from *M* definition (1) and option 3, step 2 of $\Re$ *(s)* algorithm on which $\Re r(s)$ is formed). Since finite state machine *H* (generating *K*) does not contain deadlocks and liveloops, then *S* by construction also cannot contain deadlocks and liveloops, thus, *S* is non-blocking. The theorem is proved.

Comments to the theorem. A natural question may arise: how this result is correlated with the controllability condition by Wonham? First of all, it is quite correlated. If for any $s \in K$ $\Re$ *(s)* = *True* then the language *M*, built as per the algorithm (refer to (1)), will be controllable, i.e. for it, a controllability condition by Wonham is satisfied.

The controllability condition derived in the paper is formed with respect to specification language K outside $L(S\backslash G)$. Therefore, K is controllable with respect to $<G^1, G^2,\ldots,G^n>$ if it is prefix-closed and $\Re$ experiment is positive on all $s \in K$. This requirement is more strict then Wonham's but it relates to the language K that is more expressive then $L(S\backslash G)$. The example in section 4 illustrates SDES blocking by supervisor (in case the experiment is false). At the same time, this result and, which is most important, the procedure of its verification (algorithm $\Re(s)$) are pragmatic, i.e. the number of checks cannot exceed the number of simple paths to every edge of graph H) and the result is given in terms of conditions and transitions of all the components involved in the experiment.
Our example is illustrated in Table 3, with supervisor S designed as a function of strings as per the algorithm defined above in the theorem proof.

Q^h	Operation	$s = s \bullet u$	$S(s)$
1	e_{ex-1} (Put piece on table)	$s = s \bullet e_{ex-1}$	e_{2-1}
2	$e_{2-1}, e_{2-5}, e_{2-2}$	$s = s \bullet e_{2-1}, e_{2-5}, e_{2-2}$	e_{1-1}
3	$e_{1-1}, e_{1-5}, e_{1-2}$	$s = s \bullet e_{1-1}, e_{1-5}, e_{1-2}$	e_{3-1}
4	$e_{3-1}, e_{3-2}, e_{3-3}$	$s = s \bullet e_{3-1}, e_{3-2}, e_{3-3}$	e_{4-1}
5	$e_{4-1}, e_{4-6}, e_{4-2}$	$s = s \bullet e_{4-1}, e_{4-6}, e_{4-2}$	e_{3-4}
6	$e_{3-4}, e_{3-5}, e_{3-6}$	$s = s \bullet e_{3-4}, e_{3-5}, e_{3-6}$	e_{3-7}
7	$e_{3-7}, e_{3-5}, e_{3-3}$	$s = s \bullet e_{3-7}, e_{3-5}, e_{3-3}$	e_{4-3}
8	$e_{4-3}, e_{4-6}, e_{4-4}$	$s = s \bullet e_{4-3}, e_{4-6}, e_{4-4}$	e_{ex-3} / e_{ex-4}
9	e_{ex-3} (processing is not finished) e_{ex-4} (processing is not finished)	$s = s \bullet e_{ex-3}$ $s = s \bullet e_{ex-4}$	e_{2-3} e_{3-8}
10	$e_{2-3}, e_{2-5}, e_{2-4}$	$s = s \bullet e_{2-3}, e_{2-5}, e_{2-4}$	e_{2-7}
11	$e_{2-7}, e_{2-8}, e_{2-9}$	$s = s \bullet e_{2-7}, e_{2-8}, e_{2-9}$	e_{2-10}
12	e_{2-10}	$s = s \bullet e_{2-1}, e_{2-5}, e_{2-2}$	e_{2-1}
13	$e_{2-1}, e_{2-5}, e_{2-2}$	$s = s \bullet e_{2-1}, e_{2-5}, e_{2-2}$	$\rightarrow 5$
14	$e_{3-8}, e_{3-2}, e_{3-9}$	$s = s \bullet e_{3-8}, e_{3-2}, e_{3-9}$	e_{1-3}
15	$e_{1-3}, e_{1-5}, e_{1-4}$	$s = s \bullet e_{1-3}, e_{1-5}, e_{1-4}$	e_{ex-2}
16	e_{ex-2}	$s = s \bullet e_{ex-2}$	$\rightarrow 1$

Table 3. Specifying $S(s)$ as a function of strings

6. Method of direct supervisor synthesis on the basis of SDES model for real-time automation systems (RTAS)

The investigations set forth in sections 4, 5 were carried out for SDES with off-line components that were controlled via blocking mechanism as pee the scheme shown in Fig.

4. At the same time, RTAS has a number of features that are useful to apply for control modelling and engineering.

- First, RTAS is featured by control subdivision into two sublevels of control: the level of actuators that executes operations control and a process control level that provides operation sequences.
- Second, actuators are passive but can receive operative commands, execute them autonomously and provide feedback.
- Third, while RTAS engineering, a technologist defines specifications (the required operation sequences), and it is advisable that in a synthesized supervisor, the structure of sequences was preserved and the synthesis result, as to its complexity, was linearly dependent on initial specification.

The papers on the synthesis of logical devices (Kuznetsov, 1975; Ambartsumyan, Potekhin 1977) contained similar requirements and synthesis methods were called standard realization. According to the papers on standard realization methods, such approach has the following advantages:

- The obtained result is always "recognizable" by the author of initial specifications;
- The result complexity is proportional to the scope of initial data;
- The number of operations in the synthesis procedures is also linearly dependent on initial data.

Basic paradigm of standard realization is the synthesis of object control system (device) by syntactic transformation of this object behaviour specification. Therefore, standard realization is the engineering method that guarantees the engineering result of acceptable complexity and for acceptable time, provided there is the initial description of the object behaviour

With the above mentioned RTAS features and standard realization idea taken into account, the present section pursues the objective to develop a supervisor synthesis method providing dependability – acceptable complexity of the result (supervisor) achieved for acceptable time (the number of operations).

This section is dedicated to the study of SDES with passive actuators. In such SDES, all controlled events are forced from the point of view of operation (Chalmers, Golaszewski, Ramadge, 1987) and the control is performed as per the scheme similar to that shown in Fig. 17.

Definition 6. A well-structured DES, for which the composition of uncontrollable part is defined as per Definition 1, all are forced, the required behaviour is defined by the specification language $K \subseteq E_d^*, (K \neq 0)$, and which is provided by supervisor S generating unambiguously controlled events E_c in such a way that K is fulfilled, will be called a **structured discrete event system with forced controlled events (SDESf)**.

Comments to the definition. SDESf should meet the condition of determinacy, i.e. for any string s admissible for the initial state, if its extension by a controlled event is possible, such extension for this string is unique. It is suggested that a structured DES with forced events should be realized according to the scheme (Fig. 17) in which supervisor "perceives" all the events generated by G but initiates only controlled events.

Based on introduced notions, let's specify the tasks of this section.

For SDESf specified by G component set and K specification, they are as follows:

- Define a condition of K specification controllability.
- Examine the matter of a supervisor existence.

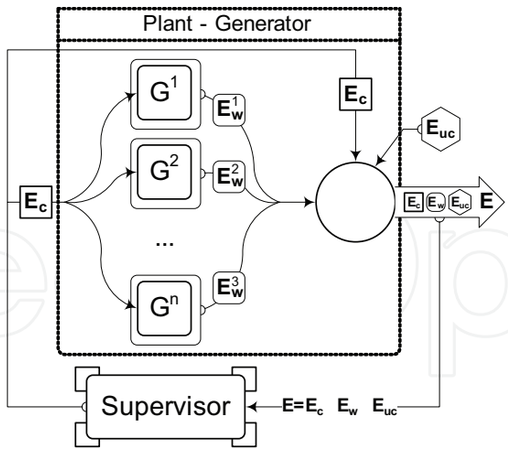

Fig. 17. The scheme of supervisory control for SDES with forced controlled events

- Elaborate the method of specification realizability analysis that will indicate if *G* and *K* are consistent.
- Develop the method of synthesis of supervisor *S* (if *K* specification is realizable) providing control in *G* in such a way that *K* is fulfilled, with the method synthesising *S* for acceptable time and *S* complexity having linear dependence on *K* complexity.

6.1 Study of SDES with forced events

In order to unambiguously define the behaviour of SDES represented by $G = \langle G^1, G^2, ...G^n \rangle$ collection, it is necessary to specify all the system states and their admissible transitions (structure and weighing functions). A traditional tool used for such tasks in discrete systems is the building of attainability tree. In section 5 of the present paper, as a basic instrument to study SDES behaviour, it is proposed to use the algorithm $\Re$ that actually is a procedure of *H* graph traversal. At this walk, for any reached state of *H*, is formed $VIS = \langle q^1_{r,1}, q^2_{r,2}, ..., q^n_{r,n} \rangle$ - a vector of initiated states of each of $G=<G^1,G^2,\ldots,G^n>$ components, which appears sufficient to build a tree of attainability. It would be logical to use intermediate results of algorithm $\Re$ at supervisor synthesis. The way to this is set forth below. Important is the fact that the synthesis task can be divided into two main subtasks: the analysis of *H* graph structure and the analysis of complete states.

Structure analysis For further study, we shall need to examine states q_i in which the selection (fork) in the transition graph, defining machine *H*, takes place. Without the loss of generality, we assume that there are no mixed forks in the transition graph of *H*. In other words, if more than one edge originates from q_i, these edges are always weighed either only by uncontrollable events or, on the contrary, only by those controllable (feature F4 – forks separability feature worded in section 4).

The last condition in the defining set definition $\left[\omega(s) = \tau(q_i \to q_k)\right]$ states the fact of string *s* termination on $(q_i \to q_k)$ transition.

Definition 7. Let $\tau(q_i \to q_k)$ be a function with its value equal to the event weighing the transition $(q_i \to q_k)$, and $\omega(s)$ – the last event in string *s*. Then $\Phi^i_k = \{s | [\delta^h(q_0, s)!] \wedge [\omega(s) = \tau(q_i \to q_k)]\}$ will be called a **defining set** of strings of *k* direction in fork q_i .

The last condition in the above definition $\omega(s) = \tau(q_i \to q_k)$ states the fact of the string s end on $(q_i \to q_k)$ transition.

Definition 8. Let O_i be a set of subscripts of states q_j to which there is a direct transition from q_i ; let $\theta(e) = \{c | uc | w\}$ be a function of event type; let $q_i \in Q^H$; $\lfloor O_i \rfloor \geq 2$. Then q_i is called a **correct selection** (fork), if only one of the following conditions is fulfilled:

- For all the transitions incidental to q_i, $\theta(\tau(q_i \to q_k)) = uc$ is fulfilled (the selection on uncontrollable events);
- For all the transitions incidental to q_i $\theta(\tau(q_i \to q_k)) = c$ is fulfilled (the selection on controllable events), with the defining sets on any pair of directions not intersecting, i.e. $\forall m,n : [m,n \in O_i \Rightarrow \Phi_i^m \cap \Phi_n^i = \varnothing]$.

The answer to the question, as to the existence of supervisor for SDES with forced controlled events, is given by the following theorem.

Theorem of SDESf controllability. Let a well-structured $G = \langle G^1, G^2, .., G^n \rangle$ be given, for which $E = \{E_w \cup E_c \cup E_{uc}\}$, all E_c are forced $E_d = \{E_c \cup E_{uc}\}$ and $K \subseteq E_d^*, (K \neq \varnothing)$. Non-blocking supervisor S such that $P_{E_{uc}}(L(s/G)) = K\}$ exists then and only then when for any $s \in K$ (such that $\delta^h(q_0, s)!$) $\Re(s)$ = *True* as respects $G = \langle G^1, G^2, .., G^n \rangle$ and all selections in the transition graph of H are correct.

Theorem proof Necessity. The first part of condition: $\Re(s)$ = *True* is valid as shown in the proof of the theorem of controllability in section 5. Let's prove the necessity and sufficiency of the second condition: branching correctness. Proof is made by contradiction. Let there exist an unblocking supervisor S but for H the condition of branching correctness is not met. Then two options are possible, namely:

- Branching is mixed, i.e. for the transitions incident to q_i $\theta[\tau(q_i \to q_k)] = [c \wedge uc]$ is fulfilled. It is impossible as conflicting branching limitation.
- For q_i branching, the condition of empty intersection of defining sets is not fulfilled. Let s^i transfer H to q_i state, than there exists at least one event e^{i+1}_c that simultaneously weighs two different edges originating from q_i. On the other hand, since a supervisor exists, it is determinate and is defined as a function of strings, thus, for this option, two different values e^{i+1}_c and $[e^{i+1}_c]$, weighing the next pair of edges, must fit the same argument s^i. We have arrived at a violation.

The necessity is proved.

Sufficiency is proved constructively. Let $K \neq \varnothing$ be a language such that for any $s \in K$ $\Re(s)$ = *True* with respect to $G = \langle G^1, G^2, .., G^n \rangle$. We shall show that in such case there exists a supervisor unblocking for G and providing $P_{E_d}(L(S/G)) = K$.

Let's define language M on K in the following way:

$$M = \left\{ \varepsilon \cup \bigcup_{s \in K} \Re r(s) \right\} \tag{3}$$

Any string $u \in M$ is admissible for $L(G)$ as per construction in $\Re$. For this reason, $M \subseteq L(G)$.

For any $u \in M$ let's define:

$$S(u)=\begin{cases} e:\theta(e)=c\wedge\Re(P_{E_d}(u)e)=True \\ e:\theta(e)\neq c\wedge\Re(P_{E_d}(u)e)=True \\ \text{other } e \text{ not included in codomain } S \text{ for } u \end{cases} \quad (4)$$

The designed converter admits the following:

- all E_c possible (as to transition function for G) after u (string 1 from (4));
- ε instead of any event of $e\in\{E_w\cup E_{uc}\}$, if this event enters the definition area of corresponding transition in H or a certain component of G^k (string 2 from (4)).

Since machine H (generating K) does not contain deadlocks and liveloops, $L(S/G)=P_{E_d}^{-1}(K)$ by designing, and all $s\in P_{E_t}^{-1}(K)$ are admissible for G (as $\Re$=*True*), than S by construction also cannot contain deadlocks and liveloops, thus, S is non-blocking. The theorem is proved.

Comments to the theorem.

1. The proposed condition of controllability is formulated with respect to specification language K that is more expressive and compact then $L(S\backslash G)$, at least, because it is a projection of $L(S/G)$ on E_d.
2. For controllability, besides the requirement of positive experiment, it is necessary that all forks in the transition graph of H should be correct, which is effectively verified by the graph nodes review.
3. The condition of controllability for SDESf is worded as a limitation imposed only on specification language K but does not restrict language $L(S/G)$. Nevertheless, K is controllable as relates $G=\langle G^1,G^2,..,G^n\rangle$, provided it is prefix-closed, algorithm $\Re$=*True* for all $s\in K$, and all forks are correct. This condition is more strict than that in paper (Chalmers, Golaszewski, Ramadge, 1987) as it admits branching on controllable events, in case the selection is correct.

Let's consider possible branching variants on $e\in E_c$ - controlled events. Practically, the following situations are possible:

- Logical substantiation for choosing the continuation is in the pre-history.
- There is no logical substantiation in the past (the defining sets for both directions intersect but, at this, sequences are admissible for both branches.

This situation will be illustrated by the structure of transition graph shown in Fig. 18. Semantics of events, states and sequences will be described later in section 6.4. Herein we shall discuss a few peculiarities of forks in the transition graph. The edges of forks originating in states q_1,q_4 and q_{22}, in Fig. 18, are outlined by firm ellipses. The events: $e_{ex\text{-}1}$ – a round piece or $e_{ex\text{-}2}$ – a hexahedral piece, took place in the first outlined fragment, but in the situation of the following firm ellipses, there are no longer such events and a clamp choice should be made from memory of those events.

Another branching variant is referred to in the description of cutter-type choice – in Fig. 18, corresponding forks are marked by dashed ellipses. From the point of view of event sequence, both variants are admissible for $G=\langle G^1,G^2,..,G^n\rangle$ and there is no data to choose the variant of the process continuation. In principle, the second situation, in conformity with the theorem condition, testifies that K is not coordinated with $G=\langle G^1,G^2,..,G^n\rangle$ and SDES is not controllable with K. However, for practical tasks, such situation is settled by the addressing of algorithm to the external, as relates to given SDES, system (e.g. to operator).

Thus, when analysing forks (selection) of H graph on controllable events, two aspects, important for supervisor S engineering, were revealed. First, for every branching on

controllable events $q_i \begin{cases} \rightarrow & q_i{}' \\ \rightarrow & \dots \\ \rightarrow & q_i{}' \end{cases}$, it is a unique extension $(s'_{q_i} \neq s'_{q_n})$ that corresponds to string s'_{q_i}, and this provides determinacy of S. Second, the events defining the condition of selection (direction) either happened in the past or lie outside SDES structure.

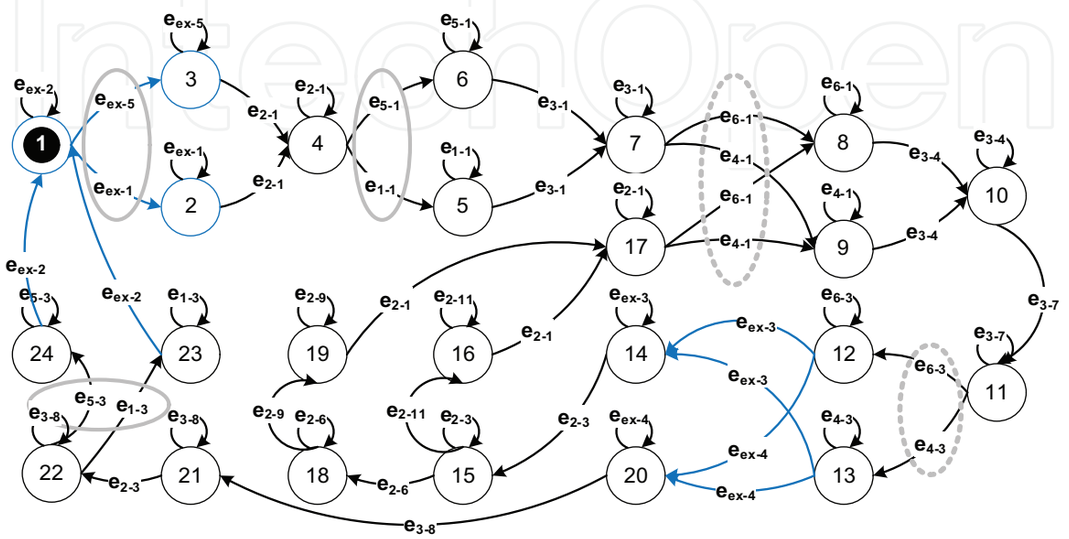

Fig. 18. Graph of H specifications

In any case, as of the moment of the analysis of situation with branching, the events, conditioning it, either happened, and the selection should be requested as a new event, or the selection is impossible to define by prehistory and then it should be requested from external sources. For providing such request, let's design machines of special kind: selection agents. For all states of branching origin q_i, an individual agent – a machine of $A_{qi}=\langle Q_a, E_c, \delta_a, \Gamma_a, \lambda_a, q_0 \rangle$ kind is built on controllable variables. A transition graph A_{q_i} is shown in Fig.19.

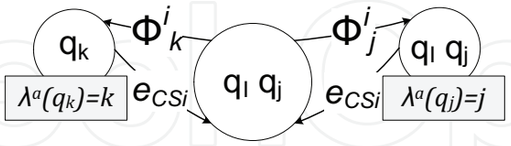

Fig. 19. Machine-agent of selection

The discipline of interconnection of supervisor S with branching machine-agent is as follows: on S reaching state q_i, after which it is necessary to make choice (arrive at a decision), S, through its special output, sends a request to A_{q_i}. In accordance with the discipline accepted, A_{q_i} replies issuing a direction index $a_{q_i} = \{k | j\}$, (not necessarily 1 of 2, possibly 1 of many). At this, a_i will be used as index. Fig. 20 illustrates a control scheme.

Analysis of complete states To answer the question about the consistency of supervisor S and object G, SDES state analysis is required. The state of the set of primary components $VIS = \langle q_{j1}^1, q_{j1}^2, \dots, q_{jn}^r \rangle$ reflects the state of control object $G = \langle G^1, G^2, .., G^n \rangle$. The current state

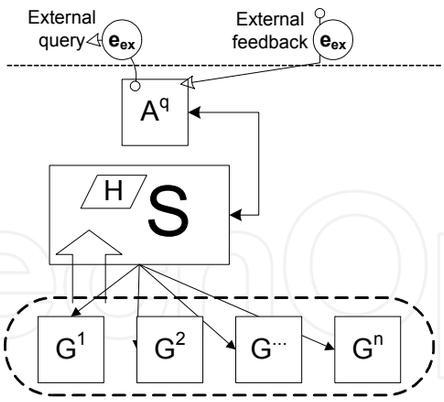

Fig. 20. Control scheme with selection agent

of H machine - q_r^h reflects the intentions to control (to limit free behaviour of G) for the purpose of solving some technological tasks. Thus, a complete SDES state is $V_{FS} = \langle q_r^h, q_{j1}^1, ..., q_{jm}^r \rangle$. A conventional technique of listing the complete attainable states in similar situations is evident: it is necessary to build a tree of attainability and, guided by the tree, to carry out the experiment on H machine and a set of $\langle G^1, G^2, .., G^n \rangle$ component machines. In the paper, it is proposed to analyze complete states by their "projection" on H transition graph, in parallel with the experiment on controllability.

For this purpose, let's set H and all the component machines to their initial states. We shall get a vector of initial states. Let's weigh H initial state by this vector. Then we shall make an experiment with a string composed of events E_d (let it be e_c^i), weighing H graph edges, and of the events corresponding to the reaction from one of the components $\langle G^1, G^2, .., G^n \rangle$ of a definite component machine G^i. At this, H will transit to another state q_r^h, with the following states adjoined: first, all the states of component machines $\langle G^1, G^2, .., G^n \rangle$ that are unchanged at this transition; second, by turns, all G^i component machine states corresponding to its reaction to e_c^i. These vectors form a block of states that weighs q_r^h states. The experiment fragment is shown in Table 4. From the fragment of the table of complete states, it is clear that one H node is associated with blocks of complete states and this results not only from including states and components "put into action" but also from the availability of multiversion attainment of the given state. For example, in state 10, there are 4 blocks corresponding to different state combinations for components G^1 and G^2. It is important that all the states inside blocks should successfully operate on subsets of events admissible in this state.

Reasoning from the above, the requirements to the method of complete states design and analysis are set forth as follows:

1. Complete states are formed as they are required in H and operations are processed in G;
2. The method should provide for the structure of data on H complete states and transitions. This structure should provide easy access to complete states at H transition graph traversal; the advancement and distribution of complete states along H must be accompanied by H and G consistency analysis.

The major novelty of this idea is that a set of complete states V_{FS} moves and spreads along H graph structure in compliance with the flow of possible events. It is important to note the following:

q^h	G^1	G^2	G^3	G^4	G^5	G^6
1	1	1	1	1	1	1
4	1	1-4	1	1	1	1
	1	1-4	1	1	1	1
10	4	4	4-7	1	1	4
	4	4	4-7	4	1	1
	1	4	4-7	1	4	4
	1	4	4-7	4	4	1

Table 4. Full states (fragment)

1. If, first, the experiment was made for every complete state in every node and it, at least once, was positive, and, second, all *H* edges were walked through, than the set of all attainable complete states was obtained. This is equivalent to the building of the tree of attainability.
2. If all transitions came into action, than the specification defined by *H* is controllable and the data acquired is sufficient to form *S* basis.

It is suggested that a supervisor should be synthesised via weighing *H* graph edges by new operations and output functions $\lambda_s(q_i, t_{i,j}) = \{e_c, \varepsilon\}$ defined on pairs of states and transitions. Output functions will "return" either controllable event e_c or empty symbol ε and they must keep their value over the whole transition $q_i \to q_j$.

6.2 Synthesis of supervisor for SDESf

Supervisor engineering will be made based on FSM of special type.

Definition 9. A **machine active at transitions (TAM)** is a finite machine $S = (Q_s, E, \delta_s, \rho_s, \lambda_s, \Gamma^s, \Gamma^\tau, Q^s_{,}, q_0)$ in which the set of states Q^s, set of events *E*, transition functions δ_s, functions of admissible events Γ^s a set of necessarily attainable states Q^s_m and the initial state q_0 are defined conventionally. Control functions $\Gamma^\tau, \rho_s, \lambda_s$ of TAM are defined in the following manner:

- $\Gamma^t_i : Q^s \to \{t_{i,j} \,|\, \{j : \delta_s(q_i, \Gamma_i)\} = q_i\}$ is a function of possible transactions (transitions). This function associates every edge originating from q_i with string $t_{i,j} \in E^*$ which, when performed, initiates in *S* a transition $q_i \to q_j$; thus, at *S* operation, both states and transitions are active.
- $\rho_s(q_i) = e_{q_i}$ Moor-type function of outputs defined only at forks on controllable events ($|O_i| \geq 2; \theta(q_i) = c$); for the rest of q_i, $\rho_s(q_i) = \varepsilon$.
- $\lambda_s(q_i, t_{i,j}) = \{e_c \,|\, \varepsilon\}$ - a machine output function, defined on a pair [state– string of transition], equal either to controlling event or to empty symbol and keeps its acquired value over the whole transition $q_i \to q_j$

Apparently, a machine, active at transitions (TAM), has a number of destinations:

- "Language generator" *L(S)*. Let's present any path incident to the initial state q_0 in *S*, as iteration of concatenations of pairs [state–outgoing edge] expressed as $r_{0,l} = \bullet_{i=0}^{l}[q_i r_{i,j}]$, than $L(S) = \{s \,|\, s = \bullet_{i=0}^{l}[q_i := \lambda^s(q_i, r_{i,j} \bullet t_{i,j})]\}$ is a set of strings obtained from the set of paths by substitutions of corresponding events and strings.

- "Direction pointer" – Milly-type output function successfully relevant to the definition of operator able to admit or turn down controllable events. However, it is very important that the function is defined on states and substrings $S(s_{0,l}) = \begin{cases} \varepsilon \text{ for } q_0 \\ \lambda^s(q_i, t_{i,l}) \end{cases}$.
- "Internal interconnections". Provides interconnection with selection machines-agents that interface SDESf with external media - a supplier of incontrollable events of e_{uc} type.

From the point of view of the theory of finite machines, *S* is a machine of mixed– type: it has output $\rho_s(q_i) = e_{q_i}$ depending only on states, which is typical for Moor machine, as well as output $\lambda_s(q_i, t_{i,j}) = \{e_c | \varepsilon\}$ typical for Milly machines; furthermore, the latter is defined at transition.

Supervisor *S* is designed, basing on specification *H*, by the special algorithm of syntactical transformation $\Im(G, H) \to S$. Machines-agents are supposed already defined, so, the transformation is made according to the following scheme:

$$H = (\ Q^h,\ E_d,\ \delta_h,\ \Gamma^h,\ Q_m^h,\ q_0\ )$$
$$\downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow\ \ \downarrow$$
$$S = (\ Q^s,\ E,\ \delta_s,\ \rho_s, \lambda_s\ \Gamma^s,\ \Gamma^t,\ Q_m^s,\ q_0\ )$$

Thus, supervisor synthesis method must solve the following tasks:

- form the vectors of complete states for every state Q^h;
- put in action all the transitions (edges) of *H*;
- should the experiment be positive, to build for *S* main missing constructions $\rho_s, \lambda_s, \Gamma^\tau$.

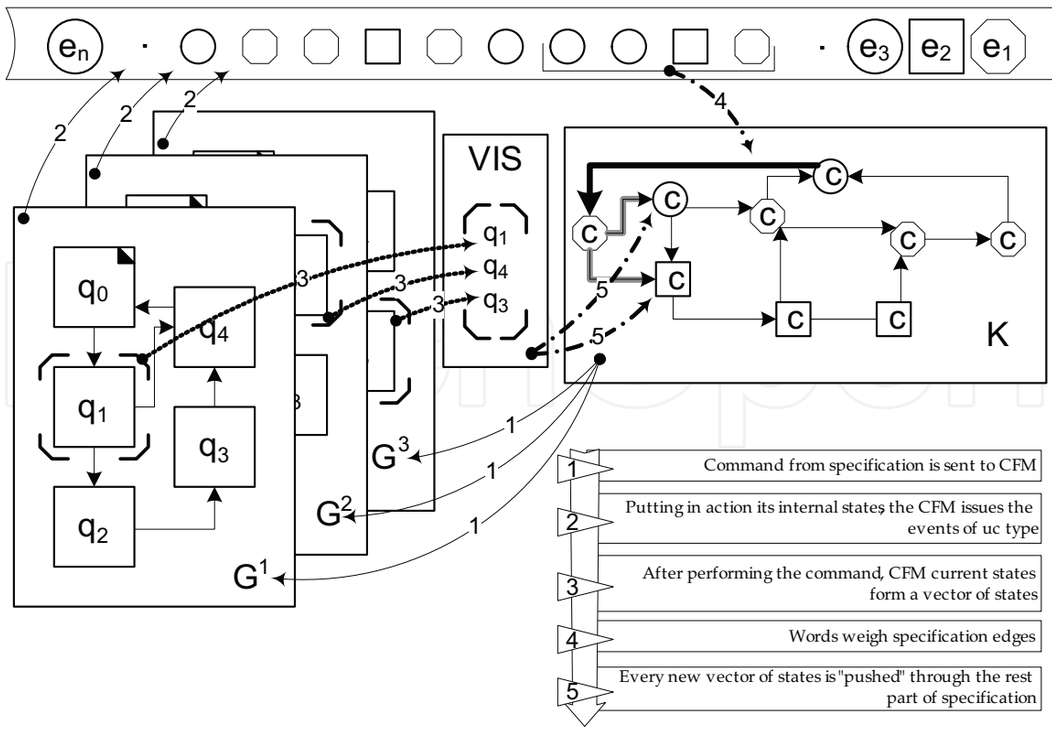

Fig. 21. Data scheme of algorithm ℑ

The proposed algorithm ℑ which fulfils the method of direct supervisor engineering, for the purpose of clearness, will be depicted graphically as data structure and the algorithm diagram (Fig. 21). Algorithm ℑ processes specification K, defined by a transition graph, carries out the simulation of control command operation (arrows 1 in Fig. 21) in component machines G^i. The component machines "put into action" the transitions, in conformity with the command. As the result, algorithm ℑ forms:

- word $e_c e_{w1} \ldots e_{w_n}$ representing operation fulfilled (arrows 2 in Fig. 21);
- next complete state $VIS := \langle q^1_{j1}, q^2_{j1}, \ldots, q^r_{jm} \rangle$ (arrows 3 in Fig. 21).

Specification graph is processed as shown in Fig. 22. Block inscriptions correspond to algorithm steps. We would like to draw your attention to the fact that the algorithm is constructed as the traversal (block 4) of graph with unprocessed complete states (the first unprocessed complete state $\langle q^h{}_0, q^1{}_0, q^2{}_0 \ldots q^n{}_0 \rangle$ is created at initialization – block 1). For every

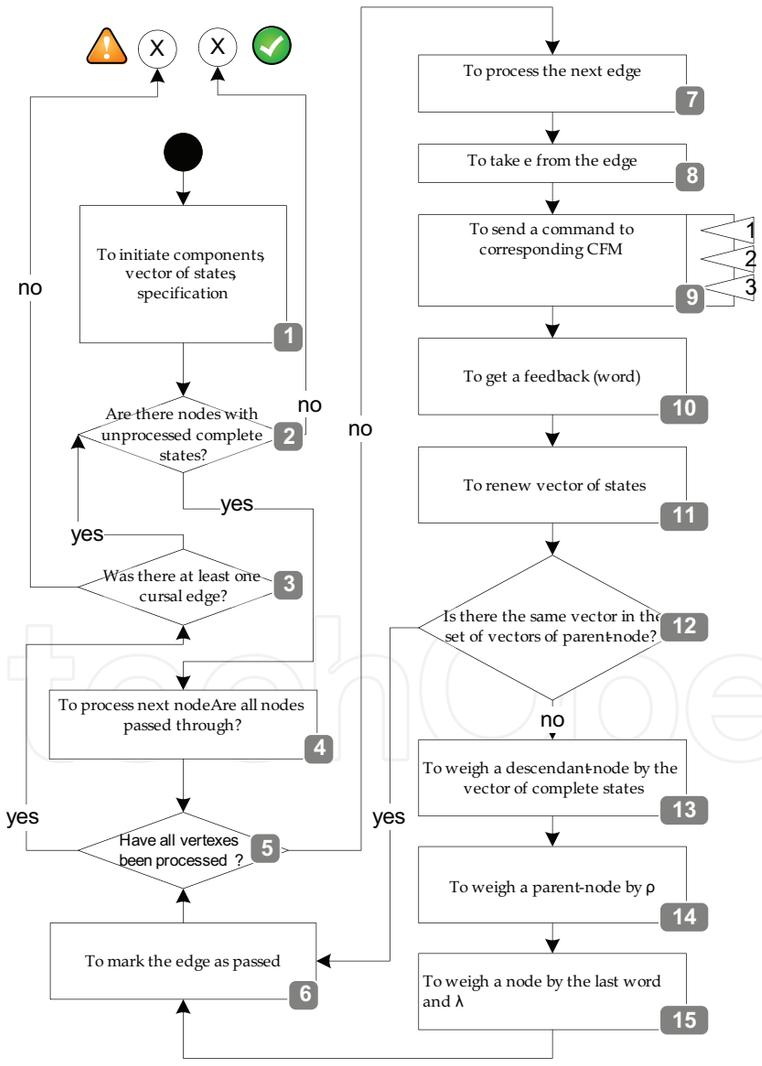

Fig. 22. Block-diagram of algorithm ℑ

state, a loop on outgoing edges (block 5) is formed. For every edge, the weighing by functions ρ, λ and operation $e_c e_{w1} ... e_{w_n}$ takes place (blocks 14, 15) once. On termination of the traversal of all H nodes and edges, we get the set of all admissible states. If all the transitions took place, than H is controllable (ticked-off output) and the acquired data define the weighing of machine - supervisor S. If graph traversal is terminated ahead of schedule (the output marked with "!" symbol) than specification K and object $G=<G^1, G^2, \ldots, G^n>$ are incompatible.

6.3 Study of algorithm $\Im$

The purpose of the method study is to assess its complexity and time characteristics. The main question is whether the main features of standard realisation methods, are preserved, namely: linear dependence of the result complexity and spent time on the initial data scope. Partially, the answer to this question may be given by the following theorem:

Theorem on standard realization. Given: $G = \langle G^1, G^2, .., G^n \rangle$, language K specified by machine $H = (Q_h, E_d, \delta^h, \Gamma^h, Q_m^h, q_0)$ for which the set of transition graph edges is designated by R. Than algorithm $\Im$, based on $G = \langle G^1, G^2, .., G^n \rangle$ and $H = (Q_h, E_d, \delta^h, \Gamma^h, Q_m^h, q_0)$ specification, constructs (as per the scheme of transition-active machine) supervisor S such that project $P_{Ed}(L(S/G)) = K$ and, at this, S complexity on the **data scope** $\leq O(MAX(|Q^h|, |R|))$, and the **number of operations** to design $S \leq O(|R|)$

Proof. The fact that the first theorem part is true follows from the method, though the proof by induction on string length, can be easily developed. The validity of the second part – the complexity of S presentation (specification), namely: $\leq O(MAX(|Q^h|, |R|))$ [1] follows from the fact that all S constructions are obtained by redefining H constructions and by including new constructions (block 6 of the algorithm) associated with states and edges of H graph transitions and limited by the data sets from G and H. From the above, it follows that the complexity of $S \leq O(MAX(|Q^h|, |R|))$. The validity of the last theorem statement on the number of operations follows from the fact that the total number of the executed algorithm blocks from 2 to 6 on all states $\sum_1^n |O_i|$ equals the number of H edges - $|R|$. Which required. Thus, for the case, when the complete states one by one take their places in H states, the linearity of dependence of the number of operations on the number of edges, is maintained but in more complicated cases, the occurrence of "additional" complete states in H weighing structure results in the iteration of transitions analysis.

6.4 Example 2

The proposed method will be illustrated by the example of supervisor synthesis for a milling machine with 6 mechanisms: a clamp for round pieces (1), turntable (2), spindle (3), rectangular cutter (4), angle clamp (5), round cutter (6). Kinematics of this machine is similar to that of the machine from section 3.3 (Fig. 4) but in the considered machine, the processing of 2 types of pieces with different fastening and by different tools (mechanisms G^5 and G^6) are foreseen. Each mechanism is simulated by corresponding CFM $\{G^n\}$. The nodes

1 The formula runs as follows: S has the order of magnitude maximal of two values: the cardinal number of the set of states or the cardinal number of the set of edges.

correspond to the space position and edges – to events. Event semantics is presented in Appendix 1.

Respective CFM is shown in Fig. 23, by colour, in transitions, are marked the events of e_c type, ticked-off is the initial state q_0.

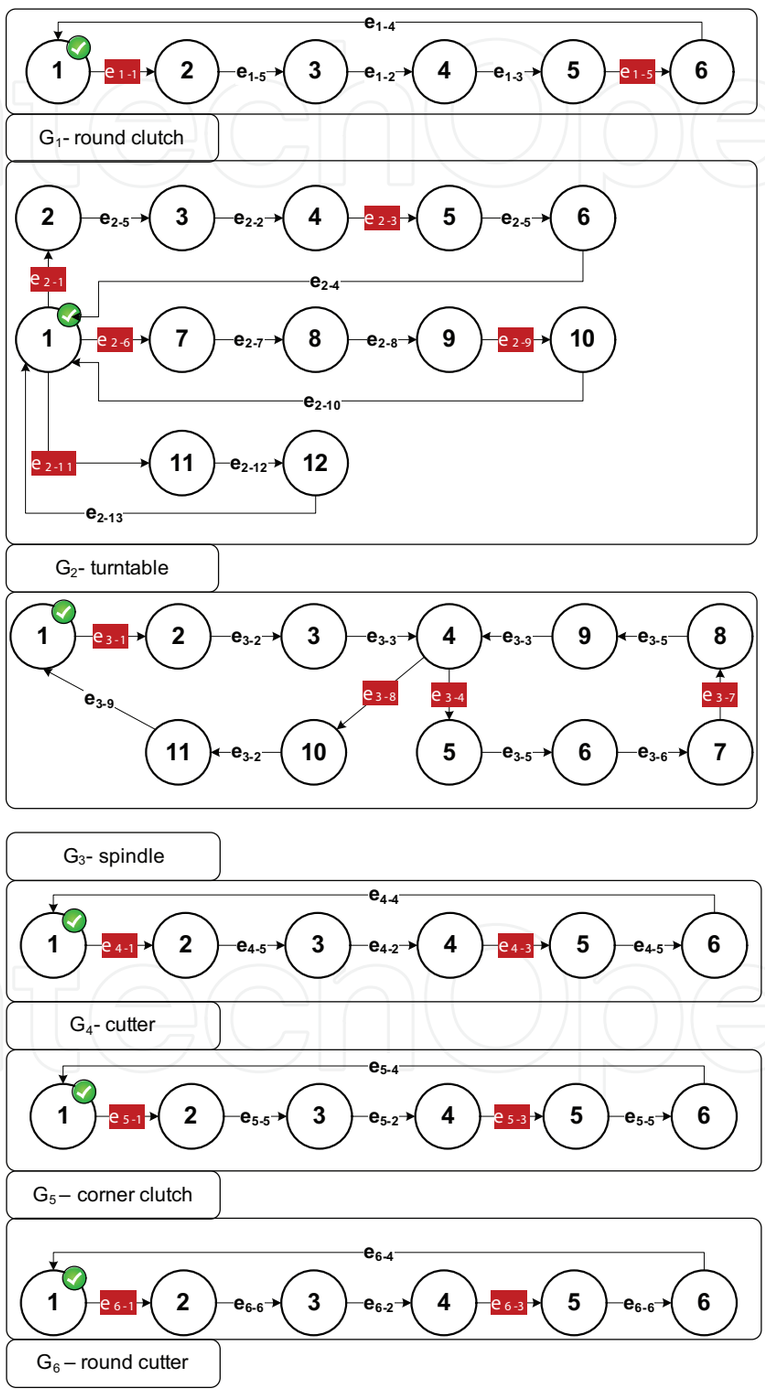

Fig. 23. Component finite machines $G^1, G^2, \ldots, G^6$

The required machine behaviour is informally presented by text specification in Appendix 2. This behaviour is formalized by finite machine $H = (Q_h, E_d, \delta^h, \Gamma^h, Q_m^h, q_0)$, with a graph of transitions shown in Fig. 18, section 6.1. The states of machine $Q^h = \{q_1, ..., q_{23}\}$ correspond to the steps of processing and the edges – to the operations of component machines.

The proposed method was applied to analyse G=<G^1, G^2, …, G^n> object and H specification. The experiment $\Re$ showed the consistency of H and G. Then, the graph of supervisor (Fig. 24) was obtained. Entering a node, the edges relevant to the same operation have common marking (e.g. the edges entering node 7). In the supervisor graph, every edge is weighed by G^i CFM component operation and supplemented by the events of relevant reaction. Events e_{ex} are incontrollable, the edges with these nodes do not change. The comparison of two graphs reveals their structure identity. This illustrates that the complexity of supervisor designed by the proposed method, linearly depends on the complexity of initial specification.

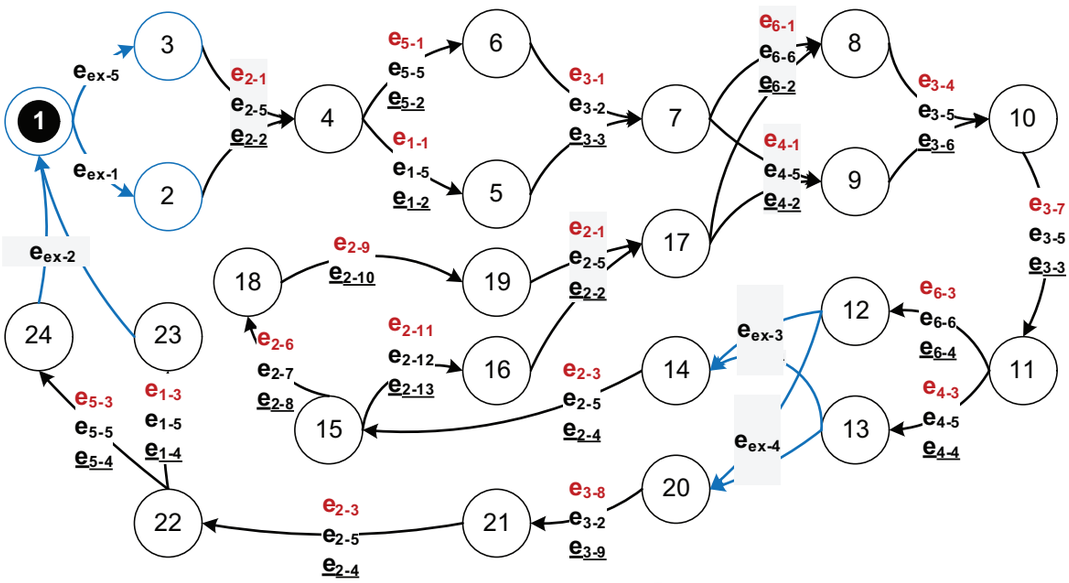

Fig. 24. Supervisor graph

7. Conclusions

SDES model proposed herein, does not use component composition in the explicit form but operates $\langle G^1, G^2, ..., G^n \rangle$ set and $K \subseteq E_d^*$ specification language specified by $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$ machine (recall that $E_d = E_c \cup E_{uc}$ is a language over a set of commands and conditions). Such approach to the description model is more economical, than that in $L(G)$ language and is much more expressive than the one based on parallel composition $\bigoplus_1^n G^i = G^i \oplus ... \oplus G^n$ and $K \subseteq L(G)$.

Thus, the proposed SDES model and the procedure of its operation take maximum account of the SDES (real-time automation system) peculiarities mentioned in the introductory part. There is a ground to believe that thereby it will be possible to avoid the «explosion of states» at supervisor synthesis. The proved theorem of controllability for SDES builds a theoretical

basis for further studies and a base for programming and experiments on the stream of real tasks.

Thus, (turning back to the problem stated in Introduction) it can be declared that herein is developed a theoretical basis for a new technique of machine control engineering that excludes ambiguity and mistakes in the initial specification of a control object as a "black box".

As the result of research pursued, the conditions of SDES and SDESf controllability were formulated, the matter of supervisor existence was studied, the method of specification realizability verification was shown.

The condition of controllability was worded with respect to specification language K that is more expressive and compact (being a project of $L(S/G)$) than language $L(S/G)$ traditionally used in the models with parallel composition.

The paper suggests the structure of supervisory control, contains the study of the method of supervisor S synthesis based on the object model and specification (G and K). It also illustrates a linear dependence of supervisor S complexity on the number of edges of H machine.

At the same time, the number of synthesis operations (time complexity) remains linear only for the specification in which complete states, one at a time, are disposed on H, i.e. the number of operations for the designing of $S \le O(|R|)$. Generally, the appearance of "second" complete states in the structure of H weighing, results in the repeated analysis of transitions and the linearity is violated. However, practically, for real tasks, this phenomena, reflecting a designer's aspiration to specify commands sent to aggregates (actuators) more economically, does not lead to a considerable growth of the number of operations.

8. References

Ambartsumyan A.A. (2009) Supervisory Control of Dynamic Discrete-Event Systems. *Automation and Remote Control*, No.8 (August 2009), pp. 156-166

Ambartsumyan, A.A., Kazansky, D.L. (2008). Complex automation of technological processes with the event model involved. *Procedings of 17th IFAC World Congress*

Ambartsumyan A.A., Bronishtov S.A. (2006) *Event models of process control directed at the protection against personnel's erroneous actions.* «Greenwich LTD », Moscow

Ambartsumyan A.A., Potehin A.I. (2003) Development of control mechanisms of objects with continuous technology on base of channel event models. *Automation and Remote Control*, No 4 (April 2003)

Ambartsumyan, A.A., Prangishvili, I.V., Poletykin, A.G. (2003). Power plants: the analysis of state and the proposal of enhance automation. *Problems of control*, No.2 (March 2003)

Ambartsumyan A.A., Potekhin A.I. (1977) Standard realization of asynchronous machine *Automation and Remote Control;* No.5, (May 1977) pp.67-83

Cassandras C.G., Lafortune S. (2008) *Introduction to discrete event systems* « Springer Science+Business Media, LLC ». USA.

Chalmers Golaszewski C. H., Ramadge P. J. (1987) Control of discrete-event processes with forced events. *Proc. 28th Conf. Decision Control.* p. 247-251, Los Angeles

De Queiroz, M.H., Cury, J.E.R. (2000) Modular supervisory control of large scale discrete-event systems. Discrete Event Systems: Analysis and Control. *Proc. WODES'00*, pp. 103-110.

Gaudin, B., Marchand, H. (2003) Modular supervisory control of asynchronous and hierarchical finite state machines. *Proc. ECC*, pp. 13-25

Gilard Langer. HoMuCS (1999) *A methodology and architecture for Holonic Multicell Control Systems Preface*. KPB -2-99; ISBN 87-90855-00.

Jo Wyns. (1999) *Reference architecture for Holonic Manufacturing Systems - the key to support evolution and reconfiguration.* ISBN 90-5682-164-4, K.U.Leuven

Kuznetsov O.P. (1975) Logical machine graphs and their transformations *Automation and Remote Control,* No. 9, (September 1975) pp.149-158

Ramadge J. G., Wonham W. M. (1989) The control of discrete-event systems *IEEE Trans. Automat. Control*.. 77(1). p. 81-98.

Ramadge J. G., Wonham W. M. (1987) Supervisory control of a class of discrete event processes *SIAM J. Control Optimization*. 25(1). p. 206-230.

Sujeet Chand. (2005) From electric motors to flexible manufacturing: control technology drives industrial automation. *IFAC*

Van Brussel H., Valckenaers P., Bongaerts L.,Peeters P. (1998) Reference Architecture for Holonic Manufacturing Systems: *PROSA. Computers in Industry.* p.37- 47.

Yoo, T.-S., Lafortune, S A (2002) General architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory & Applications* – No. 12(3), pp. 335-377

Appendices

Appendix 1

event	operation	event	operation
e_{1-1}	To clench	e_{3-8}	Feed ->>
e_{1-2}	Clamp is closed	e_{3-9}	Parked
e_{1-3}	To unclench	e_{4-1}	To switch on
e_{1-4}	Clamp is open	e_{4-2}	Working
e_{1-5}	clamp is moving	e_{4-3}	To switch off
e_{2-1}	To fix	e_{4-4}	Stopped
e_{2-2}	Table is fixed	e_{4-5}	Unstable rotation
e_{2-3}	To unfix	e_{5-1}	To clench
e_{2-4}	Table is unfixed	e_{5-2}	clamp is closed
e_{2-5}	Locker is moving	e_{5-3}	To unclench
e_{2-6}	To make a ¼ turn	e_{5-4}	Clamp is open
e_{2-7}	Table is moving	e_{5-5}	Clamp is moving
e_{2-8}	Table is turned through ¼	e_{6-1}	To switch on
e_{2-9}	To switch off turning mechanism	e_{6-2}	Working
e_{2-10}	Table is stopped	e_{6-3}	To switch off
e_{2-11}	To tilt plane	e_{6-4}	Stopped
e_{2-12}	Is tilting	e_{6-5}	Unstable rotation
e_{2-13}	Angle is achieved	e_{ex-1}	Piece is on the table (round)
e_{3-1}	<<- to feed	e_{ex-2}	Piece is removed from the table
e_{3-2}	Feed zone	e_{ex-3}	Processing is not finished
e_{3-3}	Operating position	e_{ex-4}	Processing is over
e_{3-4}	<- to feed	e_{ex-5}	Piece is on the table (hexahedral)
e_{3-5}	Operational zone	e_{ex-6}	To choose usual cutter
e_{3-6}	Operation is over	e_{ex-7}	To choose round cutter
e_{3-7}	Feed ->		

Table of system events

Appendix 2

1	On arrival, a piece is clenched by a clamp
2	After clamp operated, spindle is transferred from parking to working position (to the left)
3	After spindle shifted to working position, cutter is turned on
4	After cutter is turned on, a smooth feed to the left utmost position takes place (end of operation)
5	After operation is over, spindle is fed back to the right up to working position
6	After spindle occupied working position, a positioner turns the table through ¼.
7	After the table is fixed, the next operation is executed until the processing is over
8	After the round is made, a spindle parks, a clamp is unclenched, a signal of piece readiness is sent
9	Before parking, to switch off cutter, to wait until it stops
10	Operator chooses a cutter type and a cutting angle

Table of text specifications

Process Management

Edited by Maria Pomffyova

ISBN 978-953-307-085-8

Hard cover, 338 pages

Publisher InTech

Published online 01, April, 2010

Published in print edition April, 2010

The content of the book has been structured into four technical research sections with total of 18 chapters written by well recognized researchers worldwide. These sections are: 1. process and performance management and their measurement methods, 2. management of manufacturing processes with the aim to be quickly adaptable after real situation demands and their control, 3. quality management information and communication systems, their integration and risk management, 4. management processes of healthcare and water, construction and demolition waste problems and integration of environmental processes into management decisions.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Alexander A. Ambartsumyan (2010). Supervisory Control of Industrial Processes, Process Management, Maria Pomffyova (Ed.), ISBN: 978-953-307-085-8, InTech, Available from: http://www.intechopen.com/books/process-management/supervisory-control-of-industrial-processes

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen