8

PLC-Based Implementation of Local Modular Supervisory Control for Manufacturing Systems

André B. Leal[1], Diogo L. L. da Cruz[1,3] and Marcelo da S. Hounsell[2]
[1]*Department of Electrical Engineering, Santa Catarina State University*
[2]*Department of Computer Science, Santa Catarina State University*
[3]*Pollux Automation*
Brazil

1. Introduction

Developing and implementing control logic for automated manufacturing systems is not a trivial task. Industrial production lines should be able to produce many types of products that go through a growing number of processes given the needs of the market, and there is an ever growing flexibility demand because of it. To keep up with it a faster way to develop control logic automation for the production lines is required. And this should be done in such a way to easy development and to guarantee that the control is correct in terms of making the system to behave as it should. To this end, the use of formal modelling tools seems to help raise the abstraction level of specifying systems' behaviour at the same time that it provides ways to test the resulting model.

The Supervisory Control Theory (SCT) of Ramadge and Wonham (1987, 1989) is an appropriate formal tool for the control logic synthesis of automated systems because it ensures the achievement of an optimal control logic (minimally restrictive and nonblocking), and that meets control specifications. Regardless of its advantages for automated manufacturing systems control and troubleshooting, this theory and its extensions have not been broadly used in industrial environments so far. The main reason for this resides in some difficulties that exist in dealing with complex problems. According to Fabian and Hellgren (1998) another important reason is the difficulty in implementing a pragmatic solution obtained from SCT theoretical result, i.e., bridging practice and theory.

This chapter presents a methodology, named DECON9, that aims to reduce the gap between this promising theory and real world applications, i. e., it presents a methodology for the implementation of the SCT into Programmable Logic Controllers (PLCs). The local modular approach (Queiroz & Cury, 2000a, 2000b) is used for the supervisors' synthesis and the implementation in PLC is performed in the ladder diagram language. Local Modular approach is used because systems of greater complexity that have a big amount of machines (and then, events) usually can be modelled as many concurrently interacting and simpler subsystems.

PLC implementation of supervisory control was also discussed in (Ariñez et al., 1993; Lauzon, 1995; Leduc & Wonham, 1995; Leduc, 1996; Qiu & Joshi, 1996; Lauzon et al. 1997; Fabian & Hellgren, 1998; Dietrich et al., 2002; Hellgren et al., 2002; Liu & Darabi, 2002; Music &

Matko, 2002; Queiroz & Cury, 2002; Chandra et al., 2003; Hasdemir et al., 2004; Manesis & Akantziotis, 2005; Vieira et al., 2006, Morgenstern & Schneider, 2007; Noorbakhsh & Afzalian 2007a&b; Afzalian et al., 2008; Hasdemir et al., 2008; Noorbakhsh, 2008; Silva et al., 2008; Leal et al., 2009; Possan & Leal, 2009; Uzam et al., 2009). In most of these works the monolithic approach (Ramadge & Wonham, 1989) for the supervisors' synthesis is used, in which a single and usually large supervisor is computed to control the entire plant. According to (Queiroz & Cury, 2002), this approach is not adequate for most real problems because they involve a large number of subsystems. In order to overcome this problem, in some works the synthesis of supervisors is performed according to the local modular approach (Queiroz & Cury, 2000a), which reduces the computational complexity of the synthesis process and the size of supervisors by exploiting specifications modularity and the decentralized structure of composite plants. Thus, instead of a monolithic supervisor for the entire plant, a modular supervisor is obtained for each specification, taking into account only the affected subsystems.

In almost all these work the implementation is held on ladder diagram, which is a well-known PLC programming language in industrial environments. But most existing proposals can only tackle one event per PLC scan cycle, which represents a problem when handling large scale plants (Vieira et al. 2006). Just a few of those proposals, at the best situation, can process one event per supervisor at each PLC scan cycle, a situation that can certainly be improved. Finally, just a few of them proposed solutions for the broad spectrum of problems that arise when implementing supervisory control in a PLC-based control system, as will be detailed later.

A contribution of the DECON9 methodology is that it allows dealing with various events at each PLC scan cycle, regardless if these events are controllable (can be disabled by control action) or not. Moreover, DECON9 provides a standardized approach and solution to many problems that arise while implementing SCT into PLCs.

The remaining of this chapter is organized as follows: In section 2, basic notations of the Supervisory Control Theory (SCT) for Discrete Event Systems (DES) control are introduced altogether with Monolithic and Local Modular approaches; Section 3 details the problems that arise while implementing SCT into a PLC; Section 4 presents the general assumptions behind the proposed methodology as well as its step-by-step detailed functioning; Section 5 presents a case study and how it can be solved by the methodology, and; Finally, section 6 concludes this chapter.

2. Supervisory control of discrete event systems

In the solution of manufacturing automation problems through the Supervisory Control Theory (SCT), the shop floor plant can be modelled as a Discrete Event System (DES) and finite-state automata are used to describe plant, specifications and supervisors. In this section, we introduce basic SCT notations. More details can be found in (Wonham, 2011) and in (Cassandras & Lafortune, 2008).

2.1 Discrete event systems

A Discrete Event System (DES) is a dynamic system that evolves in accordance with the abrupt occurrence of physical events at possibly unknown irregular intervals (Ramadge & Wonham, 1989). Application domains include manufacturing systems, traffic systems, software engineering, computer networks and communication systems, among others.

According to (Cassandras & Lafortune, 2008) and (Ramadge & Wonham, 1987, 1989) the free behaviour of a DES can be described through automata. An automaton can be represented by the 5-tuple $(Q,\Sigma,\delta,q_0,Q_m)$, where Q is the set of states, Σ is the alphabet of events, $\delta: Q\times\Sigma\to Q$ is the (partial) state transition function, q_0 is the initial state and $Q_m\subseteq Q$ is the set of marked states (Vieira et al., 2006). Σ^* is used to denote the set of all finite length sequences of events from Σ. A string (or trace) is an element of Σ^* and a language is a subset of Σ^*. A prefix of a string s is an initial subsequence of s, i.e. if r and s are strings in Σ^*, u is a prefix of s if $ur=s$. For a language L, the notation $\overline{L}$, called the prefix-closure of L, is the set of all prefixes of traces in L. L is said to be prefix-closed if $L=\overline{L}$ (Afzalian et al., 2010).

Consider that an automaton G represents the free behaviour of the physical system. Two languages can be associated with it: the closed behaviour $L(G)$ and the marked behaviour $L_m(G)$. The language $L(G)$ is the set of all sequence of events that can be generated by G, from the initial state to any state of G. Thus, $L(G)$ is prefix-closed because no event sequence in the plant can occur without its prefix occurring first. It is used to describe all possible behaviours of G. The language $L_m(G)\subseteq L(G)$ is the set of all sequence of events leading to marked states of G, each of them corresponding to a completed task of the physical system. A DES represented by G is said to be nonblocking if $\overline{L_m(G)}=L(G)$, i.e., if there is always a sequence of events which takes the plant from any reachable state to a marked state (Afzalian et al., 2010).

The concurrent behaviour of two or more DESs is captured by the synchronous composition of them. Thus, for two DES, G_1 and G_2, the synchronous composition is given by $G=G_1\|G_2$. This expression can be generalized for any number of DES by $G=\|_{\forall i\in I}G_i$.

The automata can also be represented by transition graphs (see Figure 1), where the nodes represent the states and the arcs labelled with event names represent transitions. Usually, the initial state is identified by an ingoing arrow whereas a marked state is denoted by double circles.

2.2 Supervisory control theory

In the Ramadge & Wonham (1989) framework, the set of plant events Σ is partitioned into $\Sigma=\Sigma_c\dot\cup\Sigma_u$, two disjoint sets where Σ_c is the set of all controllable events and Σ_u is the set of all uncontrollable events. An event is considered to be controllable if its occurrence can be disabled by an external agent (named supervisor), otherwise it is considered uncontrollable. The necessary and sufficient conditions for the existence of supervisors are established in (Wonham, 2011).

A supervisor, denoted S, determines the set of events to be disabled upon each observed sequence of events. It is a map from the closed behaviour of G to a subset of events to be enabled $S: L(G)\to 2^{\Sigma}$. The controlled system is denoted by S/G (S controlling G) and is modelled by the automaton $G\|S$. The closed and the marked behaviour of the system under supervision are respectively represented by the following languages: $L(S/G)=L(S\|G)$ and $L_m(S/G)=L(S\|G)\cap L_m(G)$.

Further, S is said to be nonblocking if $L(S/G)=\overline{L_m(S/G)}$, i.e., if each generated trace of the controlled plant can be extended to be a marked trace of the controlled plant. Consider that a language $K\subseteq L_m(G)$ represents a control specification over the plant G. K is said to be controllable with respect to G (or simply controllable) if its prefix-closure $\overline{K}$ doesn't change

under the occurrence of uncontrollable events in *G*. In other words, *K* is controllable if and only if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. Given a discrete event plant *G* and a desired nonempty specification language $K \subseteq L_m(G)$, there exists a nonblocking supervisor S such that $L_m(S/G) = K$ if and only if *K* is controllable with respect to *G* (Wonham, 2011).

However, if K is not controllable, the *supremal controllable sublanguage of K* with respect to G, denoted by $SupC(K,G)$, must be computed. In this case $L_m(S/G) = SupC(K,G)$ (Ramadge & Wonham, 1989).

In order to differentiate the controllability of events in the graph representation of automata, usually the state transitions due to controllable events are indicated by a short line drawn across the transitions (Chandra et al. 2003). Figure 1 represents an automaton, where the event *A* is controllable and the event *B* is uncontrollable.

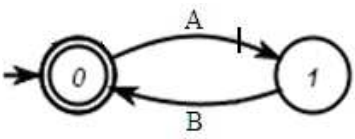


Fig. 1. A graph representation of an automaton

2.3 Monolithic approach

In the monolithic approach for the supervisors' synthesis (Ramadge & Wonham, 1989), the objective is to design a single supervisor that will coordinate the plant behaviour. Thus, all subsystems models G_i (where *i* is related to the number of subsystems), are composed in order to compute an automaton (generator) *G* that represents the free behaviour of the entire plant. In the same way, all control specifications E_j (where *j* is related to the number of control specifications) are composed into a global control specification *E*. From these models, one obtains the closed loop desired behaviour (known as target language) computing $K = G \| E$ and, consequently, obtaining a single supervisor *S* that marks the *supremal controllable sublanguage of K*, that is, $L_m(S) = L_m(S/G) = SupC(K,G)$.

According to (Queiroz & Cury, 2000a), in the monolithic approach the number of states of *G* grows exponentially with the number of subsystems. So this approach has the following drawbacks: the amount of computational effort when performing asynchronous composition of several automata, and; the use of supervisors with too many states in control platforms (usually a PLC) may generate extensive programs, where understanding, validation and maintenance will therefore, become difficult. In some cases the size of the program can render them unfit to be used in certain platform, either because of the storage or processing capacity.

In order to resolve these difficulties, Queiroz & Cury (2000a) propose using the local modular approach, as introduced in the next subsection.

2.4 Local modular approach

This approach is an extension to the monolithic approach and explores both the modularity of the plant and of the control specifications. It allows determining rather than a single and usually large supervisor, many local supervisors whose joint action guarantees the

attendance of all the control specifications. Figure 2 illustrates the structure of local modular supervisory control for two supervisors.

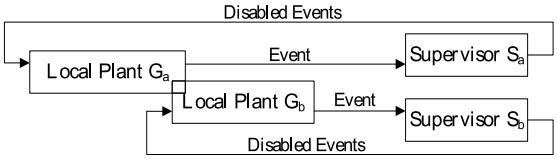


Fig. 2. Local modular supervisory control architecture (Queiroz & Cury, 2002)

In this approach, the physical system behaviour is modelled by a Product System (PS) representation (Ramadge & Wonham, 1989), *i.e.*, by a set of asynchronous automata $G_i = (Q_i, \Sigma_i, \delta_i, q_{0_i}, Q_{mi})$, with $i \in N = \{1,2,...,n\}$, all of them having disjoint alphabets Σ_i. In turn, each specification imposed by the designer is represented by an automaton *Ej* with an alphabet $\Sigma j \subseteq \Sigma$, $j \in \{1,..., m\}$, where *m* is the number of specifications. For each specification *Ej* a local plant G_{locj} is obtained, which is computed by the synchronous composition of all subsystems that share some events with the associated specification.

After determining all local plants it should be calculated the so-called local specification, which consists of performing synchronous composition between a given specification with its own local plant, *i.e.*, $K_{loc_j} = E_j \| G_{loc_j}$. Thus, the supremal controllable local sublanguages of K_{loc_j}, denoted by $SupC(K_{loc_j}, G_{loc_j})$, can be computed. Finally, it is possible to perform the synthesis of a local supervisor S_{loc_j} for each specification defined in the project. If at least one local supervisor disables the occurrence of an event, then the occurrence of this event is disabled in *G* (Vieira et al., 2006). To ensure that the system under the joint action of local supervisors is nonblocking, it should be guaranteed that the supervisors are nonconflicting, what is verified when the $\left\| _{j=1}^{m} \overline{L_m(S_{loc_j} / G_{loc_j})} = \overline{\left\| _{j=1}^{m} L_m(S_{loc_j} / G_{loc_j}) \right.} \right.$ test holds. According to (Queiroz & Cury, 2000b), this condition ensures that the joint action of local supervisors is equivalent to the action of a monolithic supervisor that addresses all specifications simultaneously. Some computational tools can be used to assist in the synthesis of supervisors. For each one of them the models of subsystems and control specifications should be introduced in order to obtain synthesized supervisors, automatically. Among these tools *IDES* (Rudie, 2006), *TCT* (Feng & Wonham, 2006) and "*Grail for Supervisory Control*" (Reiser et al., 2006) can be mentioned.

3. Supervisory control implementation

3.1 Problems

According to (Fabian & Hellgren, 1998), "*the supervisor implementation is basically a matter of making the PLC behave as a state machine*". However, this is not trivial task and can lead to many problems (Fabian & Hellgren, 1998):

Causality: SCT assumes that all events are spontaneously generated by the plant and that supervisors should only disable events generated by the plant. However, controllable events

on practical applications are not spontaneously generated by the physical plant, but as responses to given PLC commands. Thus, for implementation purposes, "who generates what?" must be answered.

Avalanche Effect: occurs when a change on the value on a given PLC input signal is registered as an event that makes the software jump over an arbitrary number of states within the same PLC scan cycle. This may occur particularly if a specific event is used to trigger many successive state transitions, thus producing an avalanche.

Simultaneity: Due to the cyclical nature of the PLC processing in which input signals readings are performed only at the beginning of each scan cycle, the occurrence of uncontrollable events from the plant is recognized by the PLC once there are changes in the input signals values. Therefore, if in between successive scan cycles two or more signals change, they will all be recognized as simultaneous uncontrollable events regardless of their exact timing. As a result, the PLC is unable to recognize the exact order of uncontrollable events that happen in between scan cycles.

Figure 3 shows an automaton that is sensitive to the sequence of *B* and *C* events. Notice that depending on the order *B* and *C* events happens, the control decision is different, which highlights the problem that if *B* and *C* are recognized altogether in the same scan cycle, we would not be able to determine the actual state and, what should be the control action: *E* or *F*.

In order to avoid the simultaneity problem, the system must present the "interleave insensitivity" property (Fabian & Hellgren, 1998), which requires that after any interleaved uncontrollable events the "control decision" must necessarily be the same.

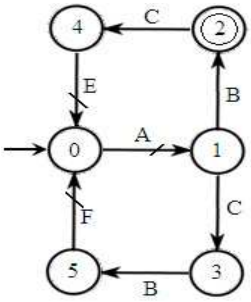


Fig. 3. Automaton that attempts to distinguish between the interleaving of events

Choice Problem: the supervisors obtained by the SCT are required to be "minimally restrictive", which means that the supervisors might provide alternative paths for the plant to choose from. Often a supervisor presents more than one possible controllable event from a single state. Thus, before producing a signal-change in the PLC outputs it may have to choose only one among them because according to Fabian & Hellgren (1998), generating more than one controllable event in a scan cycle can be contradictory and catastrophic.

Inexact Synchronization: during the program execution a change in any PLC input signal may occur and, this change will only be recognized at the beginning of the next scan cycle. The control reasoning is always performed on old frozen data. Therefore the communication between the PLC and the plant is subject to delays due to periodic reading of the input

signals (Balemi, 1992). This inexact synchronization (Fabian & Hellgren, 1998) can be a problem when a change in a PLC input signal invalidates a control action (the choice made by the program, which corresponds to the generation of a controllable event).

3.2 Related work

Many researches have dealt with producing PCL programs from TCS. Some attempts (Fabian & Hellgren, 1998) did not propose a methodology but focused on solving particular situations which is far from a generic approach such as (Hasdemir et al., 2008). In the following, we briefly discuss some of these proposals.

In order to solve the choice problem, the solutions adopted in the literature follow the idea that for practical PLC implementation purposes, a deterministic controller must be statically extracted from the supervisor. Fabian & Hellgren (1998) also show that if a choice is not taken, the sequential execution of the program within the PLC will choose and the chosen transition will always be the same in a particular state according to the ordering of the rungs.

Moreover, Malik (2002) shows that depending on the choice taken (or deterministic controller extraction) the controlled behaviour may be blocking, even when the supervisor is nonblocking, which in that work is named *determinism problem*. In (Dietrich et al., 2002) three properties are given which, when satisfied, ensure that any controller for the system is necessarily nonblocking. In (Malik, 2002) a more general property is introduced and an algorithm is given which checks whether every deterministic controller generated from a given model is nonblocking. However, no controller can be constructed from those works in case the DES model does not satisfy these conditions. In particular, a valid controller may exist, even if the conditions of (Malik, 2002) and (Dietrich et al., 2002) do not hold. But, in (Morgenstern & Schneider, 2007) another approach to generate deterministic controllers from supervisors is presented and a property named *forceable nonblocking* is introduced.

In (Queiroz & Cury, 2002), the authors introduce a general control system structure based on a three level hierarchy that executes the modular supervisors' concurrent action and interfaces the theoretical model with the real system. They also propose a ladder-based PLC implementation of TCS but do not discuss the above mentioned problems.

In (Hasdemir et al., 2004) the authors propose the use of two bits for each state in order to solve the avalanche effect, but none of the other problems are discussed. In addition, only a single event is processed per supervisor at each PLC scan cycle.

Vieira (2007) presents a methodology that considers some of the problems but the program is structured as Sequential Function Charts (SFC), which is not so widespread among PLC programmers so far. Also, this methodology requires to change the automaton model in order to remove self-loops and there is no solution to the choice problem as well.

Most of the proposals found in the literature (Leduc, 1996; Hellgren et al., 2002; Queiroz & Cury, 2002) implements SCT in the ladder language. They have the same drawback: they deal with one single event per scan cycle. Thus, if between two scan cycles "n" changes occur in the PLC inputs, the program will take "n" scan cycles to deal with them. The best proposals so far handle one event per supervisor at each PLC scan cycle. This constraint help ensure existing approaches to deal with the avalanche effect and the choice problem (*determinism problem*). However, in this way the supervisor's update rate and actions will be lower than that obtained via traditional solution, without the use of the SCT.

The related work presented above shows that a ladder-based PLC complete methodology that solves recurring problems on implementing TCS supervisory control is still missing. Below we present a methodology that fills this gap.

4. DECON9 methodology

This section presents DECON9 (which comes from the main idea: DEcomposing the CONtrol Implementation DEpending on the CONtrolability of the events), a nine steps ladder-based PLC implementation of SCT supervisory control methodology that treats multiple events in the same scan cycle and also solves the avalanche effect and the choice problem. The methodology was inspired by the work of Queiroz & Cury (2002) but the Product System (PS, the asynchronous sub-plant models) and supervisors' implementations are decomposed into blocks of events according to their controllability.

At the beginning of each PLC scan cycle, all signal changes in the PLC inputs are registered as uncontrollable events in the PS level, and state transitions due these events are processed in the PS automata. Immediately after that, the supervisors also perform the state transitions due uncontrollable events. In this way the treatment of uncontrollable events are prioritized, and PS and supervisors are maintained in synchronization with the plant.

From the current state of the supervisors all events that are still disabled are verified through a disabling map. Thus, if at least one local supervisor disables a certain event, then the occurrence of this event is globally disabled. Thereafter, from the list of non-disabled events the choice problem (determinism problem) is inferred. If there is more than one enabled event in a current local supervisor state an event is randomly selected. In opposition to the other proposals in which a deterministic controller is statically extracted out of the supervisor (offline procedure) before being implemented, in our methodology the supervisors are fully implemented and the decisions on which controllable event may be executed are dynamically performed on the fly (during the program execution). So all alternative paths in the supervisor are preserved and the system behaviour under supervision is ensured to be nonblocking.

All enabled controllable events which are likely to occur in the plant are generated in the PS and the states are updated in the subsystems and supervisors models. Finally, these events are mapped to PLC outputs and another scan cycle begins.

Notice that the coherence of control actions is guaranteed because before defining the set of disabled events and generating the controllable events in the PS, the states of the subsystems and supervisors are all updated (this means that the supervisors know in which state they are in and which events are enabled).

4.1 Solving the avalanche effect: Damming

To avoid the avalanche effect the methodology indicates to use two auxiliary memories for each uncontrollable event: the first one is used to store the events generated by the plant and; the second is used to enable PS and supervisors to transit states. Every time the second memory is used, it is reset (deleted).

In this way, an event is used only once and multiple transitions are hold up. In any case the initial state of the event is required, the first memory can be used. As long as the PS is

composed of asynchronous subsystems, once it is updated due to a given uncontrollable event, this information is not used any more until this same event is generated by the plant.

However, the information that a given uncontrollable event is active can be used somewhere in de program, especially to update supervisors states, once an uncontrollable event can be part of many supervisors simultaneously. Therefore, it should be possible to recover all information regarding uncontrollable events generated by the plant before updating supervisors. That's the reason for the second auxiliary memory.

4.2 Solving the choice problem: Random choice

To deal with the choice problem, one should analyse each supervisor at a time, to look after states where the problem may occur and which events are involved in each case. The simpler case is when there are only two controllable events to choose from, but situations involving a handful of choices are not rare in real applications. Figure 4 presents a flowchart for dealing with the choice problem. It helps identifying corresponding states and events.

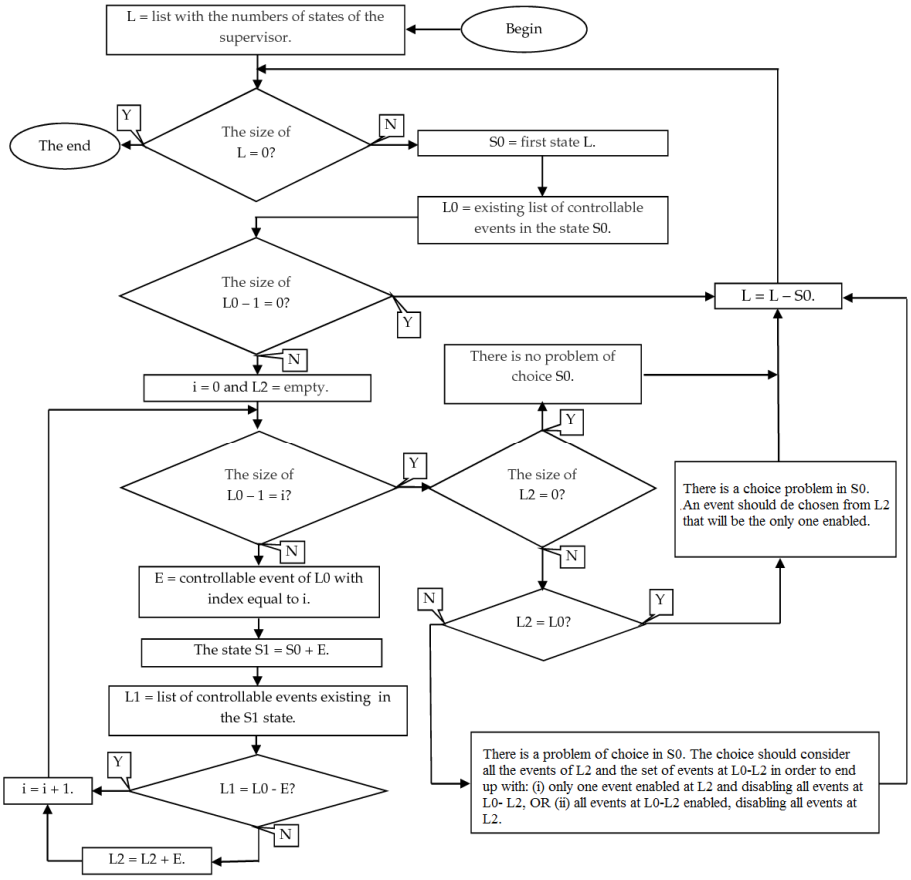


Fig. 4. A flowchart to identify a possible choice problem

To solve the choice problem, states that present multiple control alternatives need to be identified. But, if just after the disabling routine more than one controllable event is enabled at an active state, a routine is called that performs a random selection between pairs of these events. Randomness happens because an auxiliary memory is used to help perform the selection. This memory changes state at every scan cycle and, because there is no deterministic way to predict the number of scan cycles before the choice problem occurs, it acts as if it is random.

4.3 Solving simultaneity and inexact synchronization: Hardware interruption

For the simultaneity and inexact synchronization problems the solution adopted is the use of hardware interruption. Then, the uncontrollable events that may cause these problems must be associated with that type of PLC input. Thus, when a change in one of these PLC inputs occurs, the program is interrupted and the event is registered at the moment of its occurrence. It is important to notice that many PLCs do not have this kind of input and, in that case, there is no particular solution in DECON9.

It should be pointed out that these are not problems that arise exclusively while implementing SCTs into PLCs; they could happen in any given conventional approach that did not even use SCTs. Nevertheless, it can be said that an advantage of using SCTs is that these problems can be identified and we could be aware of them at the very beginning stage of designing the control system. On conventional approaches however, they can only be identified, if ever, after an extensive trial-and-error validation experiments.

On the other hand, not all systems' models will present this kind of problem. Thus, for the local modular supervisory control structure to be implemented without such problems the model must abide to some properties:

- To be sure that no problem regarding the "inability to identify event's order" problem will happen, it should be ensured that all automata that model every supervisor show the "interleave insensitivity" property (Fabian & Hellgren, 1998);
- Complementary, to be sure that no problem regarding "inexact synchronization" will happen, it should be ensured that the resulting language from every supervisor's automata, and their corresponding supervisors, show the "delay insensibility" (Balemi & Brunner, 1992).

5. DECON9 methodology step-by-step

This section will detail all nine steps of DECON9 methodology. DECON9 organizes the resulting program as a set of subroutine calls for the sake of better understanding, code reuse and reduction and, easy maintenance. Subroutine calls is a common feature available in almost every PLC.

Ten subroutines are created to fulfil all steps of DECON9 and they must follow a specific order. Figure 5 presents a complete flowchart where one can see all subroutine calls on the left and all nine steps on the right. It should be noticed that the third and fourth steps deal with uncontrollable events and the fifth to eighth steps deal with controllable ones. Also, "Make *Mx.0* = *Mx.1*" routine is called twice for every scan cycle.

First step is to be performed at the very first scan cycle only because it sets initial states of all auxiliary memories that store the initial state of all supervisors and PS subsystems. The remaining states are reset.

Second step reads all PLC inputs and identify events coming from the plant according to signal changes that corresponds to uncontrollable events.

Third step promotes the state transitions for the whole PS altogether with all just identified uncontrollable events. For this end, only transitions belonging to PS that involves uncontrollable events are transited. At this point it should be reminded that each event transition performed produces the information on the event to be erased in order to avoid the avalanche effect.

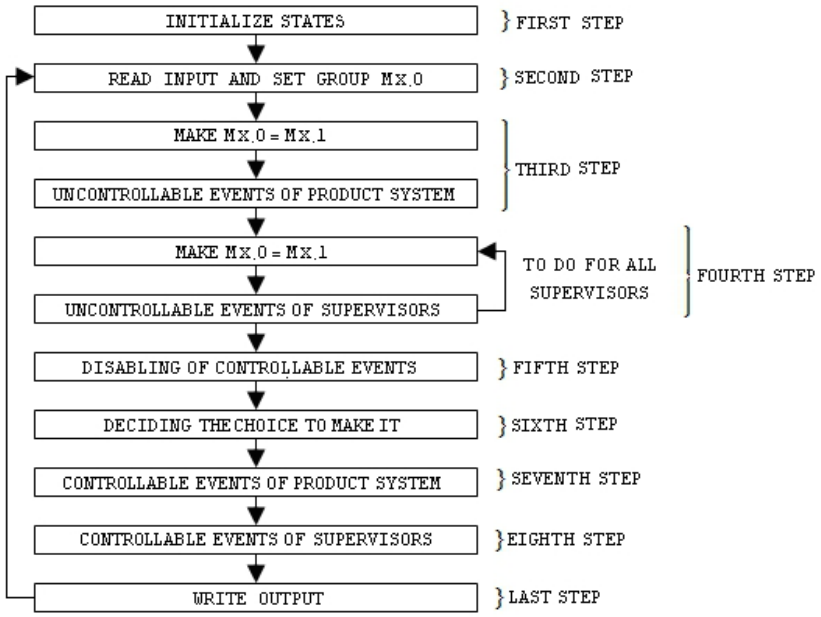


Fig. 5. Complete flowchart of the main routine

During the **fourth** step all supervisors must perform their state transitions considering uncontrollable events (and only these). The structure of the PLC program to be implemented for supervisors is the same as previously used for the PS system.

Because the information that a given event was active was erased during the transitions of the PS system, before updating the supervisors, the information of which uncontrollable events were generated by the plant must be recovered. For this end, all uncontrollable events use a pair of auxiliary memories: one of them to store which events were generated by the plant and the other is used for state transition and is discarded immediately after.

As a consequence, the methodology gives priority to uncontrollable events but do not neglect the synchronization of PS and supervisor states with the physical plant, therefore avoiding the avalanche effect.

The **fifth** step do not differ from what was proposed by Queiroz & Cury (2002) where, from the current state of each supervisor, the events disabled by any supervisor is disabled by the whole set of supervisors.

Sixth step starts off taking into account the status of all supervisors and a list of all still enabled controllable events. From these, if any supervisor shows the choice problem, it is resolved at this step. The program structure to solve this problem depends on the number of choices available (as presented earlier) but if no supervisor presents this problem, the sixth step does not produce any code.

As for the **seventh** step, it relates to the generation of controllable events that were not disabled beforehand and could possibly occur on the plant. This event generation is done at the PS level and is followed by the PS state transitions update due to the just generated events. Therefore, step seven is responsible for all state transitions related to PS´s controllable events and completing the implementation of the PS in the PLC.

Eighth step updates all controllable events generated in preceding step in all supervisors. Therefore, the remaining transitions not dealt with at the third step are done here completing the implementation of all supervisors in the PLC.

Thus, as a result of the last two steps, even before a physical control at the PLC output is issued due to controllable events, the PS and supervisors will be anticipating the state of the physical plant.

It should be noticed that there is also the possibility of the avalanche effect problem for controllable events. However, DECON9 establishes no particular procedure to deal with them because it is understood that in a practical application this problem will not occur. In any case, if it ever happens, it can be treated the same way it was done for uncontrollable events.

Ninth, and last step, sends controllable events generated by the control logic to the physical plant. This is done by mapping events to specific drives at the PLC outputs. This action generates new events from the physical plant and another scan cycle begins.

6. Case study

In order to illustrate DECON9, a complete solution for supervisory control problem is presented. The case study covers the plant and specifications modelling, the synthesis of supervisors, up to the coding of the control logic in ladder, ready to be implemented in a PLC.

6.1 Description of the physical system

The problem to be studied consists of a transfer line with six industrial machines M_X (where $X = 1,..., 6$) connected by four buffers B_Y (where $Y = A, B, C, D$), capable of storing only one part, as shown in Figure 6. This problem was studied by Queiroz & Cury (2000a) and was chosen because it produces simple automata, is easy to understand, is fairly possible to occur as part of bigger layouts and presents some of the problems DECON9 deals with.

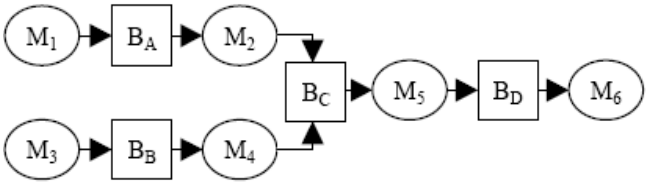


Fig. 6. Industrial transfer line case study

Start operation of the machines are controllable events, and the end operation are uncontrollable events. The transfer line should work in order to transport the parts through the machines, but a machine can't start operation if there is no part in its input buffer. Since the machines M_1 and M_3 have no input buffer, it should be considered that there will always be parts available for these machines. Similarly, M_6 can release as many parts as it is capable of producing.

6.2 Plant modelling

Table 1 presents a list of events associated with the operation of each machine as well as the type of event according to its controllability, the description of the event and, which PLC input (I) or output (Q) it is associated with.

DEVICE	EVENT	EVENT TYPE	DESCRIPTION	I/O
Machine 1	A_1	Controllable	Machine 1 start operation	Q0.0
	B_1	Uncontrollable	Machine 1 stop operation	I0.0
Machine 2	A_2	Controllable	Machine 2 start operation	Q0.1
	B_2	Uncontrollable	Machine 2 stop operation	I0.1
Machine 3	A_3	Controllable	Machine 3 start operation	Q0.2
	B_3	Uncontrollable	Machine 3 stop operation	I0.2
Machine 4	A_4	Controllable	Machine 4 start operation	Q0.3
	B_4	Uncontrollable	Machine 4 stop operation	I0.3
Machine 5	A_5	Controllable	Machine 5 start operation	Q0.4
	B_5	Uncontrollable	Machine 5 stop operation	I0.4
Machine 6	A_6	Controllable	Machine 6 start operation	Q0.5
	B_6	Uncontrollable	Machine 6 stop operation	I0.5

Table 1. Machine-related events for the case study

The behaviour of each *M*x (where x = 1, .., 6) machine is represented by a *G*x automaton shown in Figure 7. Each machine has only two states: in the state 0 the machine is stopped, waiting to work, and the state 1 the machine is working on a part. According to Table 1, the start operation is a controllable event *A*x, and the stop operation is uncontrollable event *B*x.

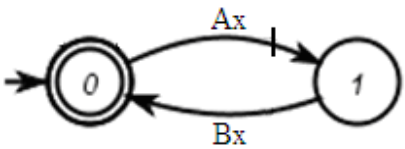


Fig. 7. G_X automaton for each machine

It is important to observe that passive devices need not be modelled, i.e., devices that don't have proper events, such as the buffers in the transfer line, for instance.

6.3 Control specifications modelling

Control specifications are models that describe the desired behaviour for the closed loop system.

The automaton presented at the left-side of Figure 8, shows the control specification of the buffers to avoid their overflow and underflow. It represents the working specification of B_A if x = 1, B_B if x = 3 and, B_D if x = 5. For all buffers, state 0 represents an empty buffer while state 1 represents a full buffer. The specification represented by the E automaton at the right-side of Figure 8 prevents the B_C buffer overflow and underflow. B_C will be full (state 1) if either B_2 or B_4 events occur and will be emptied (state 0) if an A_5 event happen. Therefore, machine M_5 will only be able to start operation after machine M_2 or M_4 produce a part in their output buffers. Once a part is deposited on a buffer, another part can only be deposited after a subsequent machine start operation (which signals that it collected a part from the buffer). Note that randomness must be guaranteed to prevent the machine M_5 from working with parts coming from only one of M_2 or M_4 machines.

Fig. 8. "E" specifications for B_A, B_B and B_D, buffers (on the left) and Bc (on the right)

6.4 Synthesis of local modular supervisors

From the devices (G_x) and operating specifications (E_Y) models, a synchronous composition between these models must be performed (as required by Queiroz & Cury, 2000b).

Firstly, you must determine the Product System (PS). To do this, the synchronous composition of all sub-plants that present common events should be performed. It should be looked for the biggest amount of asynchronous subsystems possible. For the present case study no common events between any sub-plants exist, therefore the models previously presented are the set of subsystems of PS.

Then the set of local plants must be determined. To do this, a synchronous product between the subsystems that are affected directly or indirectly by a particular specification must be done.

The most practical way to identify common events is through a table, like Table 2, so the local plants (those that share specifications) are: $G_{locA} = G_1 \| G_2$, $G_{locB} = G_3 \| G_4$, $G_{locC} = G_2 \| G_4 \| G_5$ and $G_{locD} = G_5 \| G_6$. From the common events between specifications and sub-plants analysis, it should be verified if some specification can be grouped together. For the present case study this grouping does not occur.

Following, a synchronous composition of local plants with the specifications should be performed to generate local specifications: $K_{locA} = G_{locA} \| E_A$, $K_{locB} = G_{locB} \| E_B$, $K_{locC} = G_{locC} \| E_C$, and $K_{locD} = G_{locD} \| E_D$.

Finally, the maximum controllable sublanguage of K_{locY} is calculate, which is denoted $SupC(K_{locY} \| G_{locY})$, where Y = {A, B, C, D}. The results are the local supervisors: S_{locA}, S_{locB}, S_{locC} and S_{locD}, that are presented in Figure 9 where the left side shows the supervisors S_{locA} if z=1, S_{locB} if z=3 and S_{locD} if z=5 and; at the right side the S_{locC} supervisor is shown.

	M_1		M_2		M_3		M_4		M_5		M_6	
	A_1	B_1	A_2	B_2	A_3	B_3	A_4	B_4	A_5	B_5	A_6	B_6
E_A		X	X									
E_B						X	X					
E_C				X				X	X			
E_D										X	X	

Table 2. Common events between sub-plants and specifications

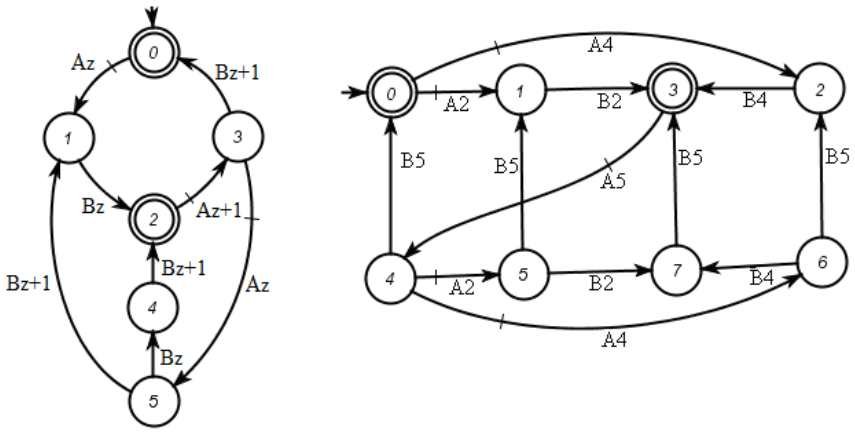


Fig. 9. Local modular supervisors

However, it is necessary to ensure the local modularity of local supervisors altogether so the joint action of all supervisors is nonblocking, as demonstrated by Queiroz & Cury (2000b). To verify local modularity, the synchronous composition of all local supervisors must be performed, as follows: $S = S_{locA} \| S_{locB} \| S_{locC} \| S_{locD}$. The resulting automaton from this composition should be checked for blocking states. If no blocking states can be found, then it can be said that local supervisors are modular to each other.

In order to reduce the amount of memory used in the implementation of these supervisors some tools to reduce the supervisors, these tools keep the control action that disable controllable events, but the supervisors lose information about the plant (Su & Wonham, 2004). However, as the product system will be implemented together with supervisors in the PLC, the information that was lost by reducing the supervisors will be preserved in the product system. Figure 10 shows the same supervisors of Figure 9, but in reduced form where the left-side show the S_{locA} (if z=1), S_{locB} (if z=3) and S_{locD} (if z=5) supervisors, and the S_{locC} supervisor is presented at the right-side.

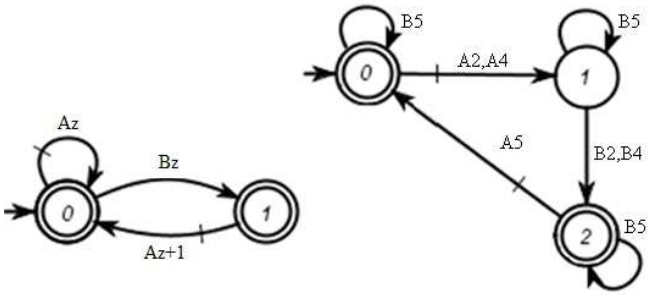


Fig. 10. Reduced supervisors automata

6.5 Following DECON9´s methodology

a. Main Routine

To easy understand, the PLC program is organized as a main routine that calls subroutines for every single block in the flowchart of Figure 5. Figure 11 shows DECON9's main routine. The sequence of calls should be followed in such a way that this main routine works like a template for all systems. Therefore, there will be 10 (ten) subroutines that will be detailed following. Notice that some abbreviations were considered in order to simplify the PLC code. Thus, S_C is the abbreviation for S_{locC} and Sc.0 means state 0 of Supervisor S_C. Moreover, dAx is used to indicate the disabling of the Ax event. Thus, in Figure 11 dA2 and dA4 are used to indicate the disabling of A2 and A4, respectively.

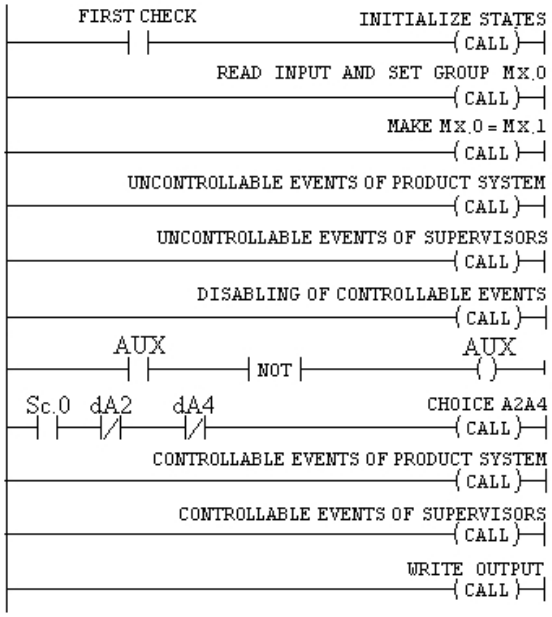


Fig. 11. DECON9 main routine

Figure 12 shows the subroutines in the order they will be called by the main routine. In order to facilitate understanding, the code for each subroutine is shown just below the line that promotes the corresponding call. Each of the subroutines is presented in sequence.

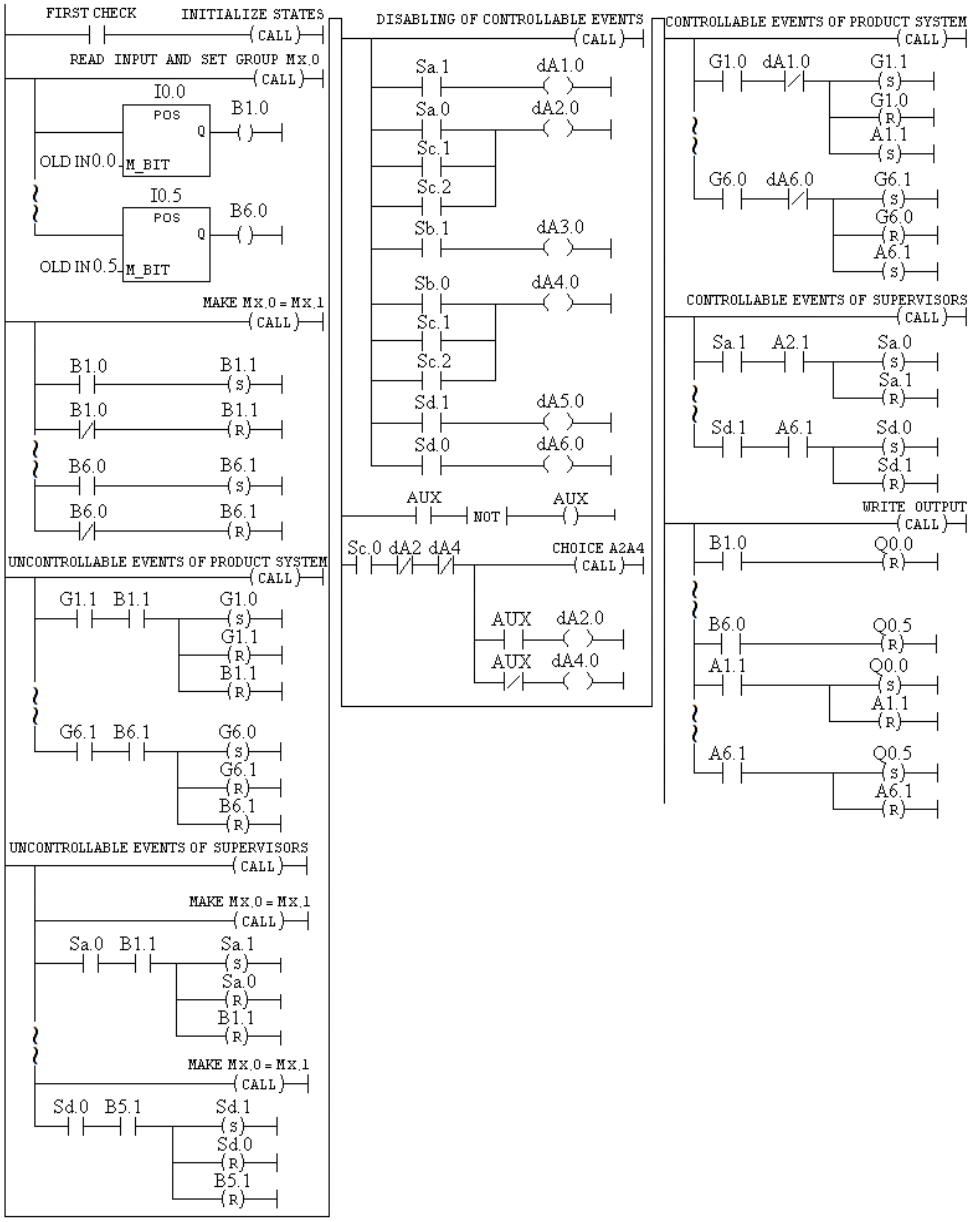


Fig. 12. PLC implementation for the case study

b. State Initialization

The first subroutine initializes all states of the Product System (PS) and supervisors. Thus, the memory that corresponds to the initial state of all automata is set to 1. Remaining memories that represents all other states are set to 0. This should be done only on the very first scan cycle (that's why a memory flag called "first check" is used alongside it) so the automata do not lose its evolutionary feature during a sequence of scan cycles.

c. Reading Inputs

Second subroutine reads PLC inputs and identifies controllable and uncontrollable events that came from the plant. This subroutine is called at the beginning of every scan cycle to verify if there is any positive transition at any PLC input. If so, there is an uncontrollable event being generated that corresponds to that input. The correspondence of inputs and uncontrollable events for the case study is in Table 1. It should be noticed that the "*POS*" PLC function (see Figure 12) ensures that the uncontrollable event will be identified only at the scan cycle immediately after the corresponding input signal changes (positive edge) and that this function is available to the RockWell PLC family.

d. Rescuing Uncontrollable Events

Every uncontrollable event uses two memories, *Mx.0* and *Mx.1*. The first group of memories, *Mx.0*, is responsible for storing the information of all events that actually have been produced by the plant. Therefore, there is a subroutine that updates the second set of memories (*Mx.1*) with the information stored at the other set (*Mx.0*) so the second group is used to promote the state transitions at PS as well as at supervisors.

As long as the information of each event that have been issued by the plant is stored in *Mx.0*, *Mx.1* can eventually lose its information because it can always be recovered from *Mx.0* by calling "Make *Mx.0* = *Mx.1*" subroutine, as shown in Figure 12.

e. Updating Product System with Uncontrollable Events

Next subroutine deals with PS uncontrollable events and is responsible for performing PS state transitions due to these events. It can be interpreted as an "automata player". There is no restriction on the number of events issued by the plant that this automata player is able to deal with. Therefore, at each scan cycle, PS automata can transit states regardless the number of uncontrollable events coming from the plant.

The current state of all subsystems is updated. This can be seen by observing for instance that when G_1 sub-plant is in state 1 and B_1 event happens, the state transition to 0 will occur and, to avoid the avalanche effect, *B1.1* memory is reset to 0 (see Figure 12). It should be noticed that if any other sub-plant is able to promote state transitions, it will be possible to promote it as well.

Because PS is composed of asynchronous subsystem, an event that is dealt with in one subsystem will not occur in another. Thus, there is no problem of erasing its information when its state transition occurs.

f. Updating Supervisors with Uncontrollable Events

Another automata player is implemented here but only to promote transitions for uncontrollable events of the supervisors. For these supervisors, that are not necessarily

asynchronous, the same event can produce state transitions in more than one supervisor. Therefore, once the information on every event that promotes a transition is erased to avoid the avalanche effect, this information should be recovered before executing the automata player for each supervisor. That's why "Make *Mx.0 = Mx.1*" subroutine is called once before each supervisor in the case study, as illustrated in Figure 12.

Note that the program structure used to update the supervisors is the same used for the PS but the supervisors' states are considered instead.

g. Disabling Controllable Events

According to the current state of all supervisors, all controllable events that should be disabled are determined.

Once PS and supervisors states are updated with the transitions promoted by the uncontrollable states issued by the plant, it can be said that all automata implemented into the PLC are in synchrony with the plant, i. e., they are all at the same states as the physical plant.

Therefore, it is possible to identify events that need to be disabled by the conjunction of the supervisors. It is possible that a single event became disabled by the action of many supervisors.

h. The Choice Problem

This subroutine should be called only if necessary and, depending on the state of a given supervisor and on the events involved in the choice problem. For each choice problem that appears, a different subroutine must be created to deal with it.

It is possible that two or more controllable events became disabled by the supervisors. If they belong to the same supervisor a choice problem may occur. In the case study at hand, M_2 and M_4 machines cannot start operation at the same time because they share the same output buffer and thus, once one machine issue a part to that buffer, the other cannot issue another one. In other words, when supervisor S_C is in state 0, A_2 e A_4 events are enabled but cannot be issued at the same time (neither at the same scan cycle) because issuing one means disabling the other. This is a clear choice problem whereby the Product System must decide which one to issue.

According to the flowchart shown in Figure 4 that ensures a solution to the choice problem at the same time that it avoids rendering a blocking system, a "Choice A2A4" subroutine is called (see Figure 12). This subroutine randomly enables only one event at a time, either A_2 or A_4 for the supervisor S_C when it is in its 0 state, for the present case study. An auxiliary memory, called "AUX", is used which changes its state (from 0 to 1, and vice-versa) at every scan cycle. Therefore, when AUX holds 1, A_2 event is disabled and, when AUX holds 0, A_4 is disabled.

i. Issuing Controllable Events from PS

Another automata player is implemented but only controllable events of the Product System are dealt with. Thus, each controllable event that has not been disabled and ready to occur would make PS to transit states and an event to be issued from PLC which means that a controllable event occur at the physical plant.

It is important to observe that the choice problem happens among non-disabled controllable events at a particular state of some supervisor and not among events of different supervisors. Thus, DECON9 allow that many controllable events can be issued at the same scan cycle. For instance, at the present case study, M_1 and M_3 start operation can happen at the same time and, as a consequence, A_1 e A_3 events can also be issued at the same scan cycle.

It should be noticed that every state transition that occur at PS corresponds to signalling a particular event that must be issued.

j. Issuing Controllable Events from Supervisors

As can be noticed in Figure 12, controllable events of PS might promote state transitions on the supervisors. Therefore, another automata player is implemented here but only for supervisors' controllable events.

k. Writing Outputs

Finally, at the end of the scan cycle, PLC outputs are updated. It should be noticed that all output reset conditions are implemented first and just afterwards, output signals are issued according to controllable events.

7. Conclusions

Supervisory Control Theory (SCT) of Discrete Event Systems (DES) has become a major player in the next step manufacturing system automation once it brings formality, predictability and a higher abstract level of specification to the analysis of complex layouts. Some of the advantages of using SCT include: plant and supervisors are high level models; testing of resulting control program is not required once it is produced from a sound theoretical background; equipment or plant behaviour models can be easily reused and; better control programs can be achieved by engineers focusing on the modelling instead of the intricacies of implementing it.

But widespread use of SCT has been hold up by the fact that Programmable Logic Controllers (PLCs) are the basic devices that can be found in the shop-floor. Implementing SCT in PLCs is not a trivial task because many problems and constraints arise while attempting to do it. Many researches have dealt with producing PCL programs from TCS. Some attempts did not propose a methodology but focused on solving particular situations which is far from a generic approach. Most of the existing proposals are based on the monolithic approach for the supervisors' synthesis, and the implementation is performed in ladder language. In some works the synthesis of supervisors is performed according to the local modular approach, which reduces the computational complexity of the synthesis process and the size of supervisors by exploiting specifications modularity and the decentralized structure of composite plants. But almost all of them can only tackle one event per PLC scan cycle, which represents a problem when handling large scale plants. Moreover, in this way the supervisor's update rate and actions will be lower than that obtained via traditional solution, without the use of the SCT. Finally, just a few of them proposed solutions for the broad spectrum of problems that arise when implementing supervisory control in a PLC-based control system.

In this chapter a nine step methodology, named DECON9, was presented. DECON9 is a methodology to implement SCT into PLCs in standardized, efficient and robust ways, closer to real size plants. It is a standardized approach because represents a complete methodology for the whole process, and is divided into simple sub-routines. It can deal with large scale plants because it uses the local modular approach for the supervisors' synthesis. It is an efficient solution because can tackle more than one event per scan cycle. It is robust because can predict problems and solves some of the most common ones. It turns PLCs into a state-machine where supervisors and plant events are explicitly represented and their control reasoning depends on their controllability.

This chapter also reviewed the basics of DES and the problems of implementing SCT into a PLC, presented detailed functioning and implementation of DECON9 and gave an example on how to apply it.

The local modular approach was used for the synthesis of supervisors and their implementation in PLC was programmed using the well-known ladder language. DECON9 use is exemplified by the implementation of the supervisory control of an industrial transfer line case study found in the literature. Using this case study, it is demonstrated that DECON9's advantages include: (*i*) it allows the control logic to deal with many events at each scan cycle, which improves existing approaches that are constrained to only one event at a time; (*ii*) a nonblocking property is achieved thanks to the random selection of controllable events approach that solves the choice problem; (*iii*) there is no fear of an avalanche effect thanks to the use of auxiliary memories, and; (*iv*) uncontrollable events are prioritized.

A computational tool for automatic generation of PLC programs obtained through the Supervisory Control Theory (SCT) is under development. This tool will comply with DECON9. With this tool, the gap between theory and practice will be reduced even further thanks to the automatic procedure based on a sound methodology.

8. Acknowledgment

The authors would like to thank *Pollux Automation Company* and *Santa Catarina State University* (*UDESC*) for their support to pursue this work.

9. References

Afzalian, A.; Noorbakhsh, M. & Navabi, A. (2008). PLC implementation of decentralized supervisory control for dynamic flow controller. *Proceedings of the 17th IEEE International Conference on Control Applications (CCA'08),* pp. 522-527, San Antonio, Texas (USA), September, 2008.

Afzalian, A. A.; Noorbakhsh, S. M. & Wonham, W. M. (2010). Discrete-Event Supervisory Control for Under-Load Tap-changing Transformers (ULTC): from synthesis to PLC implementation, In: *Discrete Event Simulations*, Aitor Goti (Ed.), pp. 285-310, InTech, ISBN 978-953-307-115-2, Retrieved from <http://www.intechopen.com/download/pdf/pdfs_id/11550>

Ariñez, J.F.; Benhabib, B.; Smith, K.C. & Brandin, B.A. (1993). Design of a PLC-Based Supervisory-Control System for a Manufacturing Workcell, *The Canadian High Technology Show and Conference*, Toronto, 1993.

Balemi, S. (1992). *Control of Discrete Event Systems: Theory and Application*, Ph.D. thesis, Swiss Federal Institute of Technology, Zürich, Switzerland.

Balemi, S. & Brunner, U. A. (1992). Supervision of discrete event systems with communication delays, *Proceedings of the American Control Conference*, pp. 2794-2798, Chicago, IL, USA, June, 1992.

Cassandras, C. G. & Lafortune, S. (2008). *Introduction to Discrete Event Systems. (2nd edition)*, Springer, ISBN: 978-0-387-33332-8, New York, USA.

Chandra, V.; Huang, Z. & Kumar, R. (2003) Automated Control Synthesis for an Assembly Line Using Discrete Event System Control Theory. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*, Vol. 33, No. 2, (may 2003), pp. 284-289, ISSN 1094-6977.

Dietrich, P.; Malik, R.; Wonham, W. M. & Brandin, B. A. (2002). Implementation considerations in supervisory control. In: *Synthesis and Control of Discrete Event Systems*, Caillaud, B.; Darondeau, P.;Lavagno, L.; Xie, X. (Eds), pp. 185-201, Kluwer Academic Publishers.

Fabian, M. & Hellgren, A. (1998). PLC-based implementation of supervisory control for discrete systems, *Proceedings of the 37th IEEE Conference on Decision and Control*, Vol. 3, pp. 3305-3310.

Feng, L. & Wonham, W. M. (2006). TCT: a computation tool for supervisory control synthesis, *Proceedings of the 8th International Workshop on Discrete Event Systems – WODES*, pp. 388-389, Ann Arbor, Michigan, USA, July 2006.

Hasdemir T., Kurtulan, S., & Gören, L. (2004). Implementation of local modular supervisory control for a pneumatic system using PLC. *Proceedings of the 7th International Workshop on Discrete Event Systems (WODES'04)*, pp. 27-31, Reims, France.

Hasdemir, T.; Kurtulan, S.; & Gören, L. (2008). An Implementation Methodology for Supervisory Control Theory. *International Journal of Advanced Manufacturing Technology*, Vol. 36, No. 3, (March 2008), pp. 373-385. ISSN 0268-3768.

Hellgren, A.; Lennartson, B. & Fabian, M. (2002). Modelling and PLC-based implementation of modular supervisory control, *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02),* pp. 1-6. ISBN: 0-7695-1683-1. Zaragoza, Spain, October 2002.

Lauzon, S. C. (1995). *An implementation methodology for the supervisory control of flexible-manufacturing workcells*, M.A. Sc. Thesis, Mechanical Engineering Dpt. University of Toronto, Canada.

Lauzon, S. C.; Mills, J. K.; & Benhabib, B. (1997). An Implementation Methodology for the Supervisory Control of Flexible Manufacturing Workcells, *Journal of Manufacturing Systems*, Vol. 16, No. 2, pp. 91-101.

Leal, A. B.; Cruz, D.L.L.; Hounsell, M.S. (2009). Supervisory Control Implementation into Programmable Logic Controllers. In: *Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA*, pp. 899-905, Palma de Mallorca, 2009.

Leduc, R. J. (1996). *PLC Implementation of a DES supervisor for a manufacturing testbeb: an implementation perspective*, M.A.Sc. Thesis, Dept. of Electrical and Computer Engineering, Univ. of Toronto, Canada.

Leduc, R .J. & Wonham W. M. (1995). PLC implementation of a DES supervisor for a manufacturing test bed. *Proceeding of Thirty-Third Annual Allerton Conference on Communication, Control and Computing*, pp. 519-528, University of Illinois.

Liu, J. & Darabi, H. (2002). Ladder logic implementation of Ramadge-Wonham supervisory controller, *Proceedings of the 6th International Workshop on Discrete Event System (WODES'02),* pp. 383-389. ISBN: 0-7695-1683-1. Zaragoza, Spain, October 2002.

Malik, P. (2002). Generating Controllers from Discrete-Event Models. In: F. Cassez, C. Jard, F. Laroussinie, M. D. Ryan (Eds.), *Proceedings of the Summer school in MOdelling and VErification of Parallel processes (MOVEP),* pp. 337-342.

Manesis, S. & Akantziotis, K. (2005). Automated synthesis of ladder automation circuits based on state-diagrams. *Advances in Engineering Software*, 36, pp .225–233.

Morgenstern, A. & Schneider, K. (2007). Synthesizing Deterministic Controllers in Supervisory Control In: *Informatics in Control, Automation and Robotics II*. Filipe, J.; Ferrier, J-L.; Cetto, J.A.; Carvalho, M. (Eds), pp. 95-102, Springer, ISBN 978-1-4020-5626-0, Netherlands.

Noorbakhsh, M. & Afzalian, A. (2007a). Design and PLC Based Implementation of Supervisory Controller for Under-load Tap-Changer. *Proc. of the 2007 IEEE Int. Conf. on Control, Automation and Systems (ICCAS'07),* pp. 901-906, Seoul, Korea.

Noorbakhsh, M. & Afzalian, A. (2007b). Implementation of supervisory control of DES using PLC. *15th Iranian Conf. on Electrical Engineering (ICEE'07),* (in Farsi), Tehran, Iran.

Noorbakhsh, M. (2008). *DES Supervisory Control for Coordination of Under-Load Tap-Changing Transformer (ULTC) and a Static VAR Compensator (SVC)*. M.A.Sc Thesis, Dept. of Electrical & Computer. Eng., Shahid Abbaspour University of Technology, (in Farsi), Tehran, 2008.

Noorbakhsh, M. & Afzalian, A. (2009). Modeling and synthesis of DES supervisory control for coordinating ULTC and SVC. *Proceedings of the 2009 American Control Conference (ACC' 09)*, pp. 4759-4764, Saint Louis, Missouri USA, June 10-12, 2009.

Possan, M. C.; Leal, A. B. (2009). Implementation of Supervisory Control Systems Based on State Machines. In: *Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation – ETFA*, pp. 819-826, Palma de Mallorca, 2009.

Queiroz, M. H. de & Cury, J. E. R. (2000a). Modular supervisory control of large scale discrete event systems, In: *Discrete Event Systems: Analysis and Control*. 1st Ed. Massachusetts: Kluwer Academic Publishers, pp. 103-110. Proc. WODES 2000.

Queiroz, M. H. de & Cury, J. E. R. (2000b). Modular control of composed systems. In: *Proc. of the American Control Conference*, pp. 4051-4055, Chicago, USA, 2000.

Queiroz, M. H. de & Cury, J. E. R. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell, *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES)*, pp. 1-6. ISBN: 0-7695-1683-1. Zaragoza, Spain, October 2002.

Qiu, R. G. & Joshi, S. B. (1996). Rapid prototyping of control software for automated manufacturing systems using supervisory control theory. ASME, *Manufacturing Engineering Division*, 4, pp. 95-101.

Ramadge, P. J. & Wonham, W. M. (1987). Supervisory control of a class of discrete event processes, *SIAM Journal on Control and Optimization*, Vol. 25, No. 1, pp. 206 - 230.

Ramadge, P.J. & Wonham, W.M. (1989). The control of discrete event systems, *Proceedings of the IEEE*, Vol. 77, No. 1, pp. 81-98.

Reiser, C.; Cunha, A. E. C. da & Cury, J. E. R. (2006). The Environment Grail for Supervisory Control of Discrete Event Systems, *Proceedings of the 8th International Workshop on Discrete Event Systems* – WODES, pp. 390-391, Ann Arbor, Michigan, USA, July 2006.

Rudie, K. (2006). The Integrated Discrete-Event Systems Tool, *Proceedings of the 8th International Workshop on Discrete Event Systems – WODES'06*, pp. 394-395, Ann Arbor, Michigan, USA, July 2006.

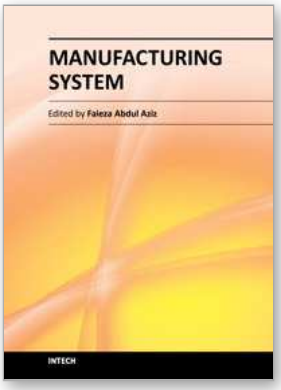
Silva, D. B.; Santos, E. A. P.; Vieira, A. D. & Paula, M. A. B. (2008). Application of the supervisory control theory in the project of a robot-centered, variable routed system controller, *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation – ETFA'08*, pp. 1-6.

Su, R. & Wonham, W. M. (2004). Supervisor reduction for discrete-event systems, *Discrete Event Dynamic Systems*, Vol. 14, No. 1, pp. 31-53.

Uzam, M.; Gelen, G. & Dalci, R. (2009). A new approach for the ladder logic implementation of Ramadge-Wonham supervisors, *Proceeding of the XXII International Symposium on Information, Communication and Automation Technologies (ICAT'09)*, pp. 1-7. ISBN: 978-1-4244-4220-1, Bosnia 2009.

Vieira, A. D.; Cury, J. E. R. & Queiroz, M. (2006). A Model for PLC Implementation of Supervisory Control of Discrete Event Systems. *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation - ETFA'06*, pp. 225–232. Czech Republic, September 2006.

Wonham, W. M. (2011). *Supervisory Control of Discrete-Event Systems*, The University of Toronto, available from: http://www.control.utoronto.ca/DES.



Manufacturing System

Edited by Dr. Faieza Abdul Aziz

ISBN 978-953-51-0530-5
Hard cover, 448 pages
Publisher InTech
Published online 16, May, 2012
Published in print edition May, 2012

This book attempts to bring together selected recent advances, tools, application and new ideas in manufacturing systems. Manufacturing system comprise of equipment, products, people, information, control and support functions for the competitive development to satisfy market needs. It provides a comprehensive collection of papers on the latest fundamental and applied industrial research. The book will be of great interest to those involved in manufacturing engineering, systems and management and those involved in manufacturing research.

How to reference

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

André B. Leal, Diogo L. L. da Cruz and Marcelo da S. Hounsell (2012). PLC-Based Implementation of Local Modular Supervisory Control for Manufacturing Systems, Manufacturing System, Dr. Faieza Abdul Aziz (Ed.), ISBN: 978-953-51-0530-5, InTech, Available from: http://www.intechopen.com/books/manufacturing-system/plc-based-implementation-of-local-modular-supervisory-control-for-manufacturing-systems

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