23

Formal Methods in Factory Automation

Corina Popescu and Jose L. Martinez Lastra
Tampere University of Technology
Finland

1. Introduction

The world market share for European industries targeting capital intensive products and equipment for manufacturing is only 22% (Manufuture, 2004). This position need not only be secured, but improved – to have the world standards of manufacturing made and approved in Europe.
In EU, each job in manufacturing is linked to two jobs in services. To support competitiveness of its industries in the global economy, Europe must be a leader in manufacturing technologies, at both process control and coordination control level. Having the highest-tech equipment in factories is necessary but not sufficient to achieve high production effectiveness. Research is needed to assist the devices cooperate (optimally) reducing waste caused by loss of energy/material and inefficient processes, while ensuring correct design and execution of standalone processes. Formal methods have an important potential to assist the development of feasible solutions in this sense.
This chapter is an introduction to formal methods in factory automation. Far from being an extensive review of the state of the art, this work provides a structured start-point for the newcomers to the field, stressing pointers to some of the most relevant works in the area. The chapter is organized as follows: Section 2 presents and compares significant features of three formalisms, leaving out specific usages of these formalisms in particular scenarios. Section 3 describes the use of formal methods in factory automation for two main purposes: verification/validation/synthesis of software control, and coordination of manufacturing activities. Section 4 presents a summary of the discussed topics and conclusions.

2. Formalisms: Overview and comparison

This section is focused on discussing relevant features and main strengths/weaknesses of three formalisms: Timed Automata, Process Algebras and Petri Nets. The intention is to leave out the specific usages of these formalisms in particular scenarios.
A timed automaton (Alur & Dill, 1994) is a finite automaton with a finite set of real-valued clocks. The clocks can be reset to zero independently of each other. The role of each clock is to keep track of the time elapsed since the last reset. The choice of the next state transition depends on the input symbol and its time relative to the times of the previously read symbols. The complexity of describing concurrent systems (especially their interactions)

with automata is high. In UPPAAL (UPPAAL), for instance, interactions can be represented through synchronization channels or guards.

Fig. 1. Conveyor-robot transfer

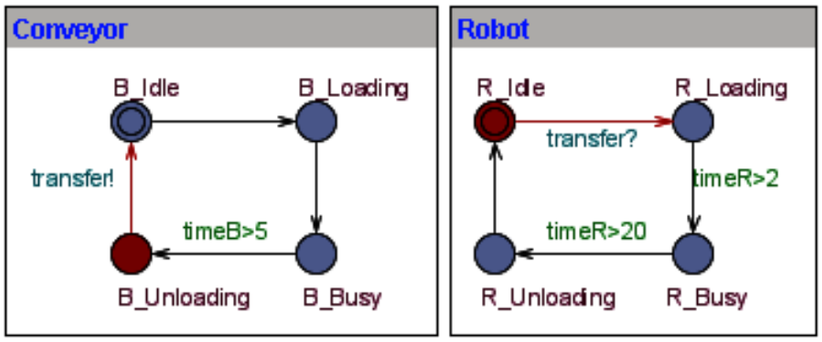

Fig. 2. Automata example: Conveyor-robot transfer

Figure 2 illustrates a simple automata representation of the transfer of a part from a buffer/conveyor of one location and a robot (Figure 1). State transitions depend on elapsed times (modelled through the clocks timeB and timeR). The interactions (e.g. the conveyor-robot transfer) are expressed through the synchronization channel 'transfer'.

The verification problem is an inclusion problem of the languages accepted by the implementation and the specification automata. The automata-theoretic approaches to verification have drawbacks related to the needed computational space and time. Even if there is enough space to store the specification and implementation automata separately, after computing the synchronized product the size of the representation can become too large. The size of the representation influences proportionally the execution time as well.

A process algebra is the study of the behaviour of a system by algebraic/axiomatic means. It is similar to the notion of a group, in the sense that both are mathematical structures having operators that satisfy a set of axioms (the equational theory) of the structure. The main derivatives of process algebra are CCS (Calculus of Communicating Systems) (Milner, 1980), CSP (Calculus of Sequential Processes) (Hoare, 1978) and ACP (Algebra of Communicating

Processes) (Bergstra & Klop, 1984). An important extension of CCS is Pi-calculus (Milner et al., 1989), which was developed to address mobility and dynamic link configuration between processes.
The details on some of the operators within the specification stand at the core of the distinctions between the various derivatives of process algebras (Philippou & Sokolsky, 2008). For instance, CCS, CSP and ACP all incorporate a different view of the synchronization-related data of the parallel composition operator. Hiding an action in ACP prevents the action from taking place altogether. Applying the same operator on a set of actions in CCS prevents the actions from taking place on the interface with the environment, but not within the system.
The view on the concurrency relation is another aspect that distinguishes certain process algebras from other formalisms. For instance, CCS approximates parallel behavior by interleaving executions. It is assumed that a system is fully described from the point of view of an external observer. Observation is made possible through the communication that takes place between the observer and the observed system. Since communication can only take place in a sequential order between the participants, the external observer can make only one observation at a time. This implies that when composing two agents in CCS, their actions are treated as occurring in arbitrary order but not simultaneously. This idea is reflected mathematically in the expression of the expansion law for CCS.
As opposed to model checking, the verification technique supporting process algebras is equational reasoning. The axioms of the algebra are used to determine the equivalence of two processes.
A Petri Net (PN) (Murata, 1989) consists of places, transitions, and flowarcs that connect places with transitions. Figure 3 illustrates a simple PN representation of the transfer of a part from a buffer/conveyor of one location and a robot (Figure 1). The elements of this net are: places (P={p1,p2,p3,p4,p5}), transitions ({t1,t2,t3,t4}) and flowarcs ({(p1;t1), (p2,t2), (p3,t2), (p4,t3), (p5,t4), (t2, p1), (t2, p4), (t1, p2), (t3, p5), (t4, p3)}). The idle statuses of the buffer and of the robot are modeled through places p1 (B_idle) and p3 (R_idle). Places p4 and p5 model the start and stop of robot processing. Place p2 (B_busy) represents the situation in which a part is available in the buffer/on the conveyor.

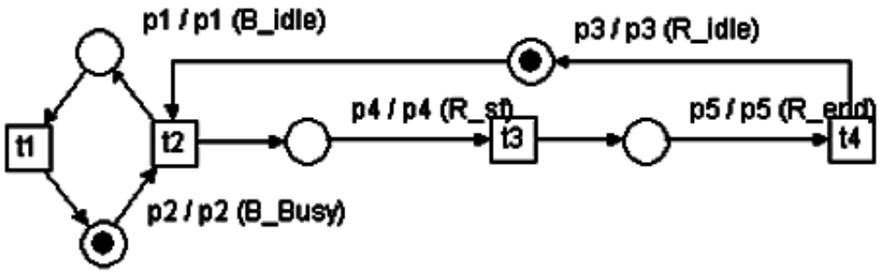

Fig. 3. PN Example: Conveyor-robot transfer

Each place may contain tokens. In the shown example, tokens are available in p2 and p3 (i.e. the marking of each of these places is 1: m(p2)=1 and m(p3)=1). A transition is enabled if each of its input places contains at least one token (e.g. in Figure 3, transition t2 is enabled as all its input places - p2 and p3 - hold one token).

If enabled, transitions act on input tokens by a process known as firing. The firing of a transition results in consumption of the tokens from its input places and the processing of some task. At the same time, a specified amount of tokens is added into each of its output places. In the shown example, the firing of t2 corresponds to the transfer of pallet between the buffer/conveyor B and the robot R. After firing, m(p2)=m(p3)=0 (i.e. a part is no longer available in buffer, and the robot is no longer free to receive part) and m(p4)=1 (i.e. the robot starts processing). State (marking) evolution is reflected in the firing of transitions.
The flow of tokens within a PN can be fully described algebraically. Table 1 illustrates the incidence matrix W for the PN example of Figure 3.

t1	t2	t3	t4	W
-1	+1	0	0	p1
+1	-1	0	0	p2
0	-1	0	+1	p3
0	+1	-1	0	p4
0	0	+1	-1	p5

Table 1. Incidence Matrixof the PN in Figure 3

This is a marking-independent description of the structure of the net. Columns correspond to places and rows correspond to transitions. Negative matrix elements are associated with place-transition flowarcs (e.g. W[p1][t1]=-1: there exists a flowarc from p1 to t1 in the described PN). Positive matrix elements are associated with transition-place flowarcs (e.g. W[p2][t1]=-1: there exists a flowarc from t1 to p2 in the described PN).
The marking M obtained by firing of a pre-specified sequence of transitions Stransitions from an initial marking M_{start} is mathematically describable through the fundamental equation:

$$M=M_{start}+W\cdot S \tag{1}$$

where S is the characteristic vector (of size equal to the number of transitions in the PN) associated with the sequence $S_{transitions}$. A firing sequence $S_{transitions}$={t1,t3,t2,t3} in a net with four transitions corresponds to a characteristic vector S=[1,1,2,0]. The first element of the vector corresponds to t1, which appears once in $S_{transitions}$. The third element of S corresponds to t3, which appears twice in $S_{transitions}$. t4 $\notin S_{transitions}$, therefore S[4]=0. Finally, S[2]=1 as transition t2 does appear once in $S_{transitions}$.
For the PN example of Figure 3, the firing effect of t2 can be calculated based on the fundamental equation:

$$\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\\0\\0\end{bmatrix} + \begin{bmatrix}-1&+1&0&0\\+1&-1&0&0\\0&-1&0&+1\\0&+1&-1&0\\0&0&+1&-1\end{bmatrix} \cdot \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$$

All states of the system can be derived algebraically. Figure 4 illustrates the reachability graph (state space) of the conveyor-robot system shown in Figure 1.

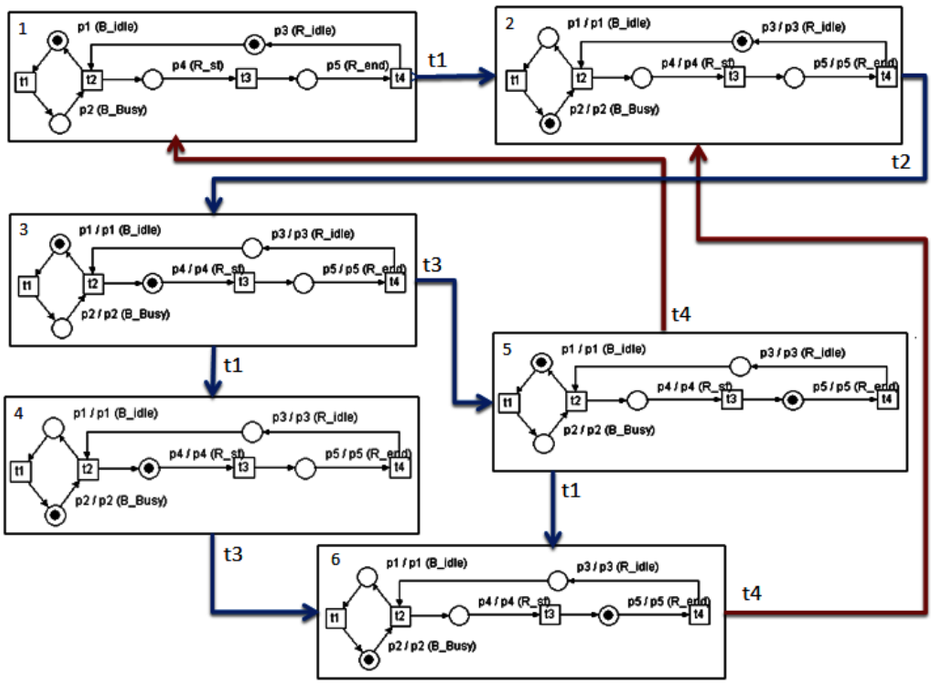

Fig. 4. Reachability graph example, shows the dynamic behaviour expressible through a PN

Several qualitative properties can be checked with PNs: liveness, boundedness, safeness, reversibility, etc. (Murata, 1989). If satisfied, the liveness property expresses potential fireability in all future markings (i.e. for every reachable state, the model can evolve in such a way that every transition can always fire in the future). A system described by a live PN is therefore a system in which every activity can ultimately be carried out.

The firm mathematical foundation confers the PN formalism a powerful set of analysis tools. Analysis methods of Petri Nets are either enumeration-based or net-driven:

Enumeration techniques rely on computation of the reachability graph. State of the art model checkers can handle state spaces up to 10^9 states with explicit state enumeration (Baier & Katoen, 2008). However, the size of the state space grows exponentially in the number of represented objects because of concurrency and interleaving semantics used to represent any sequence of possible actions. This key problem is addressed through clever algorithms and data structures (for some specific problems, state spaces of sizes 10^{20} up to 10^{476} have been handled successfully (Baier & Katoen, 2008)). Several types of methods have been researched to make enumeration-based analysis applicable in an industrial context. State-based techniques (BDDs and on-the-fly verification [Clarke et al., 2001]) aim to efficiently manage the construction of the reachability graph. Partial-order methods (sleep sets, stubborn sets and unfoldings (Girault & Valk, 2003)) make use of the dependency relations between system events to compact the state space.

Net-driven techniques aim to obtain useful information about system behaviour, reasoning from the structure of the net and the initial marking. Generative families of flows characterizing PNs are assessable based on graph theory and linear algebra techniques. Linear invariants (computable only for underlying PN models) enable certain properties of the reachable markings and firable transitions to be characterized irrespective of evolution.

Petri Nets have formal semantics, a graphical nature and an explicit representation of states (Aalst, 1998). Performance measures such as response/waiting times and occupation rates can be easily computed for PN-based models. Unlike other formalisms (e.g. CCS), PNs are capable of expressing simultaneous execution and non-determinism easily. Finally, although the verification stage does not have to be PN-based, it can benefit from the PN-nature of the model (Girault & Valk, 2003).
Table 2 summarizes the main outlined points of this section.

		Formalism		
		Timed Automata	**Petri Nets**	**Process Algebras**
Modelling issues	Describing concurrency	High complexity	**++**	**Interleaving approximation (CCS)**
	Graphical nature	+	**++**	**-**
	Explicit representation of states	-	**+**	**-**
Verification issues	Verification methods	Crossproduct; checking language inclusion	Model checking; verification does not need to be PN-based	**Equational reasoning**
	Verifiable properties, specific to the formalism		**Invariance**	

Table 2. Distinctive features of formalisms, from the modelling and verification viewpoints

3. Using formal methods in factory automation

In factory automation, formal system representations are used for two main purposes: to verify/validate/synthesize software control, and to coordinate manufacturing activities.

3.1 Verification/Validation/Synthesis of software control

The utilization of formal methods for the synthesis and verification of process logic control has arisen as an alternative to the testing of direct implementations of control realizations against informal specifications. The formalized descriptions of the control objectives, the synthesized/reinterpreted control algorithm and (sometimes) the formal model of the uncontrolled plant are input to verification and validation procedures (Figure 5).
Formal verification aims at investigating whether the design satisfies the identified standard requirements ("Are we building the product right?"). Unlike testing, verification can prove that a system has a certain property. Formal validation investigates whether the formal model is consistent with the informal conception of the design ("Are we building the right product?"). Unlike verification, validation cannot be fully automated, because it implies investigation of informal specification.

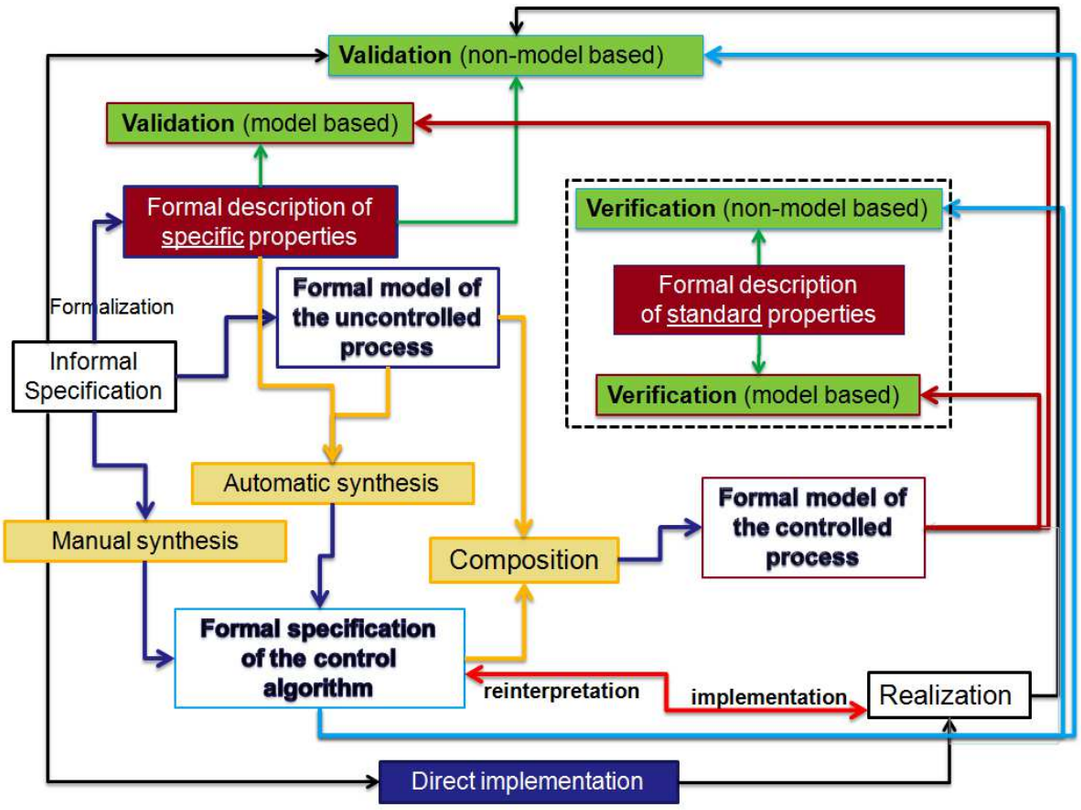

Fig. 5. Formal methods in PLC programming (adapted from (Frey & Litz, 2000))

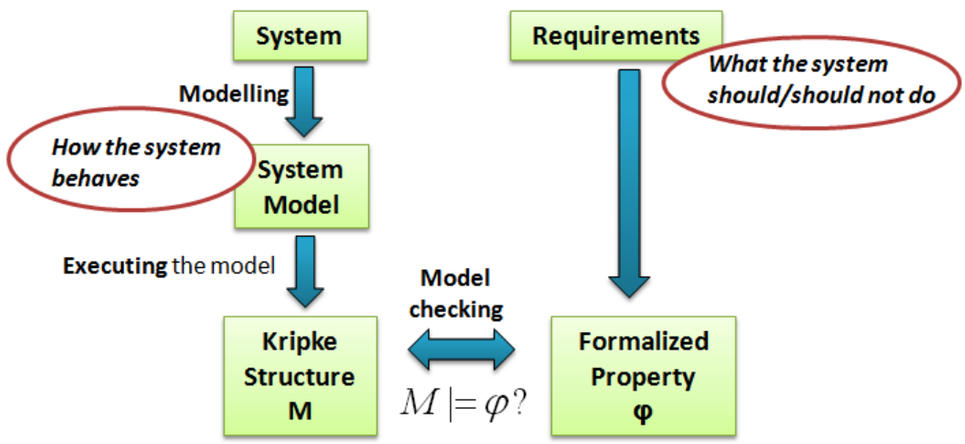

Fig. 6. Model checking

There are two main formal verification techniques: model checking (Clarke et al., 2001) and theorem proving (Duffy, 1991). In model checking (Figure 8) specifications of the system behavior (typically formulated in a temporal logic) are checked automatically on a finite model of the system (based on Petri Nets, automata, UML, etc.). The properties are investigated for given states or successions of states corresponding to the system model.

Theorem proving assumes that both the system and its expected properties are formalized in a mathematical logic. Inference rules are then applied to prove the properties from the axioms of the system description.

3.2. Coordination Control: (Re)scheduling and deadlock handling

Coordination refers to obtaining a system-level functionality based on functionalities provided by each individual component of the system. The inputs to the Coordination Control level are given by the activities to be achieved in the system (e.g. process flows, activity charts, etc.).

Planning and scheduling

Planning is deciding what actions to use to achieve some set of objectives.
A production schedule is a specification, for each resource required for production, of the planned start time and end time of each job assigned to that resource.
Scheduling is the process of creating a production schedule for a given set of jobs and resources, while optimizing some performance measure (increase of productivity, minimization of operation costs, etc.). Based on production schedules, the release of jobs to the shop can be controlled, for a better overall coordination of the activities in the manufacturing line.
Rescheduling is the process of updating an existing production schedule in response to disruptions such as machine failures and repairs, urgent job arrival, job cancelation, due date change or change in job priority.
Three main types of rescheduling strategies have been identified in the literature: completely reactive scheduling, predictive-reactive scheduling and robust pro-active scheduling:
Completely reactive rescheduling methods do not generate firm schedules in advance, but use heuristic dispatching rules to assist real time execution. Such rules are defined based on experience and are assessed through simulation, with respect to various performance criteria (e.g. tardiness, flow time, etc.). The choice of policies is problem specific, and no rule performs well for all performance criteria (Abumaizar & Svetska, 1997). Dispatching rules are used extensively in multi-agent architectures (Lee & DiCesare, 2007; Wang et al., 2008), where overall system behavior is influenced by concurrent local decisions taken by networks of individual problem solvers that cooperate. Here, heuristic guidelines come in response to the traditional drawbacks of central and hierarchical scheduling (e.g. high system complexity and cost, low fault tolerance and flexibility). Comprehensive reviews and comparative studies of such regulations in dynamic job shops and flow shops have been provided in (Rajendran & Holthaus, 1999) and (Panwalkar & Iskander, 1977).
Predictive/Reactive scheduling is an iterative process of repairing previously-created schedules (Abumaizar & Svetska, 1997; Jain & ElMaraghy, 1997) or completely regenerating schedules (Church & Uszoy, 1992). Depending on the implemented rescheduling policy, the revisions may be triggered in response to unexpected events altering the system status (event-driven), periodically, or in a hybrid manner.
Robust pro-active scheduling refers to the construction of predictive schedules which satisfy performance requirements predictably in a dynamic environment.
A wide variety of dynamic scheduling techniques have been discussed in the literature (Shukla & Chen, 1996; Stoop & Weirs, 1996; Zhou, 1995; Zhou, 1999).

Heuristics are schedule repair methods that target the finding of reasonably good solutions in short time. The main problem associated with these techniques is the difficulty to predict system performance because decisions are taken locally.
Mathematical programming techniques ignore practical constraints such as material handling capacity and complex resource sharing/routing and therefore have only a few real applications in industry (Zhou, 1995, 1999).
Meta-heuristics seek to avoid entrapment in poor local optimums obtained through local neighborhood search methods. The most popular meta-heuristic techniques include tabu search, simulated annealing and genetic algorithms (Ouelhadj & Petrovic, 2008).
Knowledge based systems, genetic algorithms (e.g. Jain & ElMaraghy,1997), fuzzy logic, case-based reasoning and neural networks have also been regarded as potential solutions to the scheduling problem.
Petri Nets can finely describe shared resources, synchronization, lot sizes and routing flexibility (Lee & DiCesare, 1994; Zhou & Jeng, 1998). PN-based scheduling implies a search for a sequence of feasible transition firings that can bring the system from an initial state to a goal state. The found schedule is deadlock free (one of the main advantage of Petri Nets over the other discussed dynamic scheduling techniques). Additionally, it is event-driven, which makes this type of scheduling perfectly suitable for real time implementation.

Deadlock handling

Deadlocks (Figure 7) are situations in which a (part of a) system remains indefinitely blocked and cannot terminate its task (Fanti & Zhou, 2004). These phenomena are caused by the inappropriate allocation of resources to concurrent executing processes.

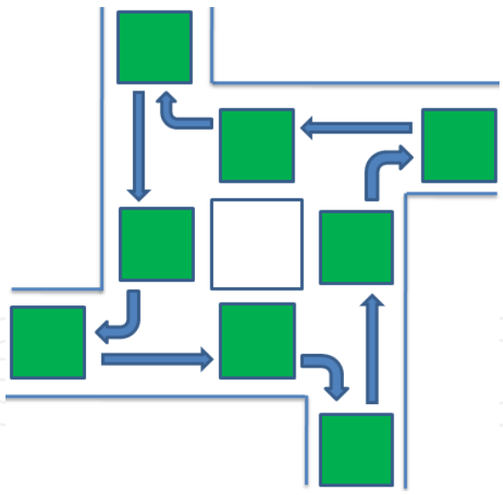

Fig. 7. Deadlock condition example (Fanti & Zhou, 2004)

Deadlocks were extensively studied first in the field of computer science and several deadlock handling techniques were originally developed for this domain. However, direct application of these methods to manufacturing systems is not possible. Computer applications only require that the bounds on the total number of resources needed by each process are known. In factory automation the information concerning the order of resource (de)allocation is also requisite (Wysk et al., 1991).

Four conditions are identified in the literature for a deadlock to occur. First, tasks claiming exclusive control of the resource they acquire may lead to deadlock (the **mutual exclusion condition**). Second, deadlock may occur when resources cannot be forcibly removed from the tasks holding them until the resources are used to completion (the **no-preemption condition**). Third, processes holding resources allocated to them while waiting for additional ones may prevent proper termination of all tasks (the **wait-for condition**). Fourth, circular claims of tasks, such that each task holds one or more resources that are being requested by the next task(s) in the claim (the **circular wait condition**), will cause indefinite blockage of a system.

In Automated Manufacturing Systems, the first three conditions always hold true. Orchestrators do claim exclusive control of the resources (machines/robots/conveyors) they acquire. Once acquired, a resource must complete the processing it was originally contracted for: a device cannot be forcibly stopped while processing in order to start machining for a different requestor. Last but not least, orchestrators hold resources allocated to them until (some of the) needed future (transportation) devices become available. Therefore, deadlocks can be excluded only if the circular wait condition is falsified.

Three main strategies have been identified for resolving deadlock problems: prevention, detection & recovery, and avoidance. **Deadlock prevention** is an offline technique involving static resource allocation policies for eliminating deadlocks. Knowledge of the system state is not required to realize the control. However with this method the utilization of resources is low and production flexibility is limited. The **detection and recovery** technique aims at resolving blockages after they have occurred. The recovery process is assisted by special buffers reserved for breaking deadlocks. This solution enables higher resource utilization, however it should be used only when deadlock is rare and detection & recovery cost is low. **Deadlock avoidance** is an online method that uses look-ahead strategies and operational control of part flow to falsify the circular wait condition. Track of the current system state and possible future states is needed. This technique is considered to yield better performance from the viewpoint of resource utilization than the first two.

Deadlock analysis and handling approaches seek the circular waits within models of process-resource interactions (job mix). The interactions between jobs and resources are traditionally represented through graphs (Wysk et al., 1991; Cho et al., 1995; Kim & Kim, 1997; Zhang & Judd, 2007) or Petri Nets (Banaszak & Krogh, 1990; Viswanadham et al., 1990; Wu et al., 2008):

Wysk, Yang and Joshi (Wysk et al., 1991) consider the deadlock problem for direct address Flexible Manufacturing Systems (FMS) during design phase. They use a graph representation of all wait relations between the input job mix and resources. All circuits within, together with their interactions , are investigated. A circuit is considered to be a deadlock if the number of jobs occupying the nodes of the cycle is equal to the numbers of nodes and edges of the cycle. The circuits are identified through a string multiplication procedure that uses one distinct character to encode each machine/node in the graph. Circuit detection is computed only upon the introducing of a new part into the system.

Cho and colleagues (Cho et al., 1995) develop graph theoretic deadlock handling procedures that are suitable for the real time control of manufacturing systems. The complete part routings of all the parts in the circuit are needed to detect impending part flow deadlocks. A system status graph is virtually updated for every part movement

before the parts move physically to the next destination. The deadlock detection and resolution procedures are based on the defined notion of 'bounded circuit' and its derivatives for this graph. A circuit becomes a sufficient condition for part flow deadlock if the number of edges in the circuit is equal to the number of parts and machines. The circuit type and its degree of node occupation characterize both part flow deadlocks and impending part flow deadlocks.

Kim (Kim & Kim, 1997) approach the deadlock avoidance problem from the graph theoretic viewpoint. Deadlock avoidance is rephrased as the problem of inserting´/deleting edges to/from the resource allocation graph while keeping it acyclic. Cycle detection on this graph is employed via a method originally developed by Belik (Belik, 1990). This technique is enriched with a resource allocation policy, effective in Automated Manufacturing Systems, to ensure superior resource utilization and productivity.

Banaszak and Krogh (1990) model concurrent job flow and dynamic resource allocation in an FMS with Petri Nets. A policy to restrict transition enabling in this model is used to avoid possible deadlocks.

Viswanadham, Narahari and Johnson (Viswanadham et al., 1990) describe a set of deadlock prevention policies that utilize look-ahead procedures on the reachability graph of the system. All behavioral characteristics of an FMS (including deadlocks) are captured offline, at modeling phase. The feasibility of the method for large systems is questionable as the entire state space of the system must be computed in its initial phase. Another important drawback concerns adaptability: if any change is made in the system the corresponding modifications have to be translated into the formal model.

Zhang and Judd (Zhang & Judd, 2008) propose a deadlock avoidance algorithm (DAA) for FMS which allows free choices in part routing. They calculate the effective free space of circuits in the digraph model of all wait relations between the resources involved in all process plans. The presented DAA runs in polynomial time once the set of necessary circuits of the digraph is computed offline.

4. Summary

This chapter provides a short introduction to the topic of formal methods in factory automation. The discussion covers the differences between two formalisms widely used in the considered application domain (Petri Nets and timed automata), and process algebras (commonly used in the field of computer science). Details are given on how formal methods are used in factory automation for verification and synthesis of process logic control and for coordination control. Pointers to relevant studies in the field are given, to provide the newcomers to the field with initial guidelines for further investigations.

5. References

Aalst, Van der, W.M.P., 'The Application of Petri nets to workflow management' Journal of Circuits, Systems and Computers, vol.8, pp. 21-66.

Abumaizar, R.J. and Svetska, J.A., 'Rescheduling job shops under random disruptions', International Journal of Production research, 1997, vol. 35(7), pp. 2065-2082

Alur, R. and Dill, D.L.(1994), A Theory of Timed Automata, Theoretical Computer Science , vol. 126, pp.183-236.

Baier, C., Katoen, J.-P., 'Principles of Model Checking', MIT Press, ISBN 978-0-262-02649-9, 2008.

Banaszak, Z.A., Krogh, B.H.(1990) Deadlock avoidance in Flexible Manufacturing Systems with Concurrently Competing Process Flows, IEEE Transactions on Robotics and Automation, vol. 6, no.6, 724-734.

Belik, F. (1990), 'An efficient deadlock avoidance technique' IEEE Transactions on Computers, vol. 39, no.7, 882-888.

Bergstra, J.A., and Klop, J.W.(1984),'The algebra of recursively defined processes and the algebra of regular processes', Lecture Notes in Computer Science 172, pp.82-95.

Cho, H., Kumaran, T.K., Wysk, R.A.(1995).Graph-Theoretic Deadlock Detection and Resolution for Flexible Manufacturing Systems. IEEE Transactions on Robotics and Automation, vol. 11, no.3, 413-421.

Church, L.K., Uszoy, R., 'Analysis of periodic and event-driven rescheduling policies in dynamic shops', International Journal of Computer Integrated Manufacturing , vol. 5(3), 1992, pp. 153-163.

Clarke, E.M., Grumberg, O., Peled, D.A., "Model Checking", MIT Press, 2001, ISBN 0262032708, 9780262032704

Duffy, D. A., "Principles of Automated Theorem Proving", John Wiley & Sons, 1991.

Fanti, M.P., Zhou, M.C. (2004). Deadlock control methods in automated manufacturing systems. IEEE Transactions on Systems, Man, and Cybernetics –Part A: Systems and Humans, vol. 34, 5-22.

Frey, G., Litz L., 'Formal Methods in PLC Programming', Proceedings of SMC, pp. 2431-2436, October 2000,.

Girault, C., and Valk, R. (2003), 'Petri Nets for Systems Engineering,' Springer, ISBN 3-540-41217-4.

Hoare, C.A.R, 'Calculus of Sequential Processes' Communications of the ACM , vol. 21, pp. 666-677.

Jain, A.K, ElMaraghy, H.A., 'Production scheduling/rescheduling in flexible manufacturing', International Journal of Production Research, vol.35, no.1, pp.281-309

Kim, C.O., Kim, S.S. (1997). An efficient real-time deadlock-free control algorithm for automated manufacturing systems. Int.J. Prod. Res., vol. 35, no.6, 1545-1560.

Lee, D.Y., DiCesare, F., 'Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search', IEEE Transactions on Robotics and Automation, vol. 10, no.2, April 1994, pp.123 -132

'Manufuture: A vision for 2020', Report of the High Level group, November 2004, Directorate-General for Research, European Commission, Brussels, Belgium

Milner, R. (1980), 'A Calculus of Communicating Sequences', Lecture Notes in Computer Science, vol. 92.

Milner, R., Parrow, J. and Walker, D.(1989), 'A Calculus of Mobile Processes - Part I,' LFCS Report 89-85. University of Edinburgh.

Murata, T., 'Petri nets: Properties, analysis and applications', Proceedings of the IEEE, vol.77, no. 4, pp. 541-580, April 1989.

D. Ouelhadj and S. Petrovic (2008), 'A survey of dynamic scheduling in manufacturing systems', Journal of Scheduling

Panwalkar, S.S., Iskander, W., 'A survey of scheduling rules', Operations Research , vol. 25, no.1, Jan.-Feb. 1977, pp. 45-61

Philippou, A. and Sokolsky O. (2008), 'Process-Algebraic Analysis of Timing and Schedulability Properties', Handbook of Real-Time and Embedded Systems

Rajendran, C. and Holthaus, O., 'A comparative study of dispatching rules in dynamic flow shops and job shops', European Journal of Operational Research, 116 (1), 156-170

Shukla, C.S., Chen, F.F., 'The state of the art in intelligent real-time FMS control: a comprehensive survey', Journal of Intelligent Manufacturing , 1996, vol. 7, pp. 441-455

Stoop, P.P.M., Weirs, V.C.S., The complexity of scheduling in practice, International Journal of Operations and Production Management, vol. 16(10), pp.37-53.

UPPAAL, www.uppaal.com

Viswanadham, N., Narahari, Y., Johnson, T.L. (1990). Deadlock prevention and deadlock avoidance in Flexible Manufacturing Systems using Petri Net models. IEEE Transactions on Robotics and Automation, vol. 6, no.6, 713-723.

Wang, C., Ghenniwa, H., Shen, W., 'Real time distributed shop floor scheduling using an agent-based service-oriented architecture', International Journal of Production Research, vol. 46(9), pp. 2433-2452

Wu, N., Zhou M.C., Li, Z.W. (2008). Resource-oriented Petri Net for deadlock avoidance in Flexible Assembly Systems. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, vol. 38, no.1, 56-68.

Wysk, R.A., Yang, N.S., Joshi, S. (1991). Detection of Deadlocks in Flexible Manufacturing Cells. IEEE Transactions on Robotics and Automation, vol. 7, no.6, 853-859.

Zhang, W., Judd, R.P. (2007). Deadlock avoidance algorithm for flexible manufacturing systems by calculating effective free space of circuits. Int.J. Prod. Res., vol. 46, no.13, 3441-3457.

Zhou, M.,(1995) Petri Nets in Flexible and Agile Automation, Kluwer Academic Publishers

Zhou, M., Jeng, M.D., 'Modeling, Analysis, Simulation, Scheduling and Control of Semiconductor Manufacturing Systems: A Petri Net Approach', IEEE Transactions on Semiconductor Manufacturing , vol.11, no. 3, August 1998, pp.333-357.

Zhou, M., Venkatesh, K.: Modeling , Simulation and Control of Flexible Manufacturing Systems – A Petri Net Approach, World Scientific Publishing , 1999.

Factory Automation

Edited by Javier Silvestre-Blanes

ISBN 978-953-307-024-7
Hard cover, 602 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Factory automation has evolved significantly in the last few decades, and is today a complex, interdisciplinary, scientific area. In this book a selection of papers on topics related to factory automation is presented, covering a broad spectrum, so that the reader may become familiar with the various fields, and also study them in more depth where required. Within various chapters in this book, special attention is given to distributed applications and their use of networks, since it is one of the most relevant subjects in the evolution of factory automation. Different Medium Access Control and networks are analyzed, while Ethernet and Wireless networks are looked at in more detail, since they are among the hottest topics in recent research. Another important subject is everything concerning the increase in the complexity of factory automation, and the need for flexibility and interoperability. Finally the use of multi-agent systems, advanced control, formal methods, or the application in this field of RFID, are additional examples of the ideas and disciplines that experts around the world have analyzed in their work.

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