Generation of safe plant operation sequences using reachability analysis

Thomas Cochard, David Gouyon, Jean-François Pétin

Generation of safe plant operation sequences using reachability analysis

Thomas Cochard*,†, David Gouyon*,† and Jean-François Pétin*,†
* Université de Lorraine, CRAN, UMR 7039,
2, avenue de la forêt de Haye,
54516 Vandœuvre-lès-Nancy cedex, France
† CNRS, CRAN, UMR 7039, France

***Abstract*—This article focuses on operation sequences engineering and preparation for complex and critical systems. The main objective is to safely operate some action sequences on the process devices (mainly actuators), according to safety requirements specifications. Based on a process formal model using communicating automata, this article shows both feasibility and limits of an automatic approach for the generation of safe operation sequences based on reachability analysis.**

***Index Terms*—Action sequences generation, reachability analysis, model checking, communicating automata**

I. Introduction

Industrial process control is a complex system that involves field devices (transmitters and actuators), automated reflex control system and plant operation control mainly supported by Supervisory Control And Data Acquisition (SCADA) systems or Manufacturing Execution Systems (MES). All these devices may be manually controlled and monitored by human field operators or by human control room operators.

Operating such complex systems is often based on predefined procedures, which are most of the time implemented in SCADA or MES. An operating procedure is a series of control and monitoring tasks aiming to modify the process state and to control its evolution.

In the context of critical systems, the predefined procedures must be qualified to ensure that all safety requirements are satisfied. It requires an important effort in terms of engineering to verify and check all the properties to be satisfied:

- control of physical variables range with regard to safety limitations,
- control constraints for actuating some devices, eventually taking into account actions that are operated on other devices,
- constraints on sub-systems availability to start an operation.

Moreover, critical systems are characterized by massive organic and functional redundancies that enable several procedures to reach a same goal.

This article focuses on action sequences which are part of operating procedures. An action sequence leads the system from an initial situation to a goal situation. A situation is characterized by the state (such as functioning features) and status (such as availability) of the system components and by a set of physical values. An action sequence consists in a set of ordered actions which may be performed manually by an operator or automatically by control devices. These actions induce a change in the state or the status of an equipment, leading to an evolution on the process physical variables. Once a sequence is executed, the system must have reached the predefined goal situation.

This article proposes the evaluation of a formal approach for generating safe action sequences. This approach is based on:

- a system model that is structured according to ISA88 standard and formalised using communicating automata [1]
- reachability analysis techniques (supported by model-checking tools) to exhibit some execution traces that represent the searched action sequences.

It is assumed that no distinction is made between manual and automated actions, a reasonable hypothesis considering that safety constraints to be ensured depend on what kind of action is done rather than which are the human or technical actors that execute it.

Section II presents both modeling and sequence generation existing approaches. Section III proposes the generation approach and its evaluation, applied on a case study described in section IV and discussed in section V.

II. Existing approaches for modeling and sequence generation

Industrial process control hierarchical structure, applicable for both batch and continuous processes, has been formalised in ISA88 [2] and IEC 61512 [3] standards. Among the different works on these processes [4], the object-oriented method ASTRID, based on the ISA88 standard (Figure 1), rely on a principle of hierarchical description of a facility and procedures decomposed in materials and functional subsets: devices, resources, functions and recipe.

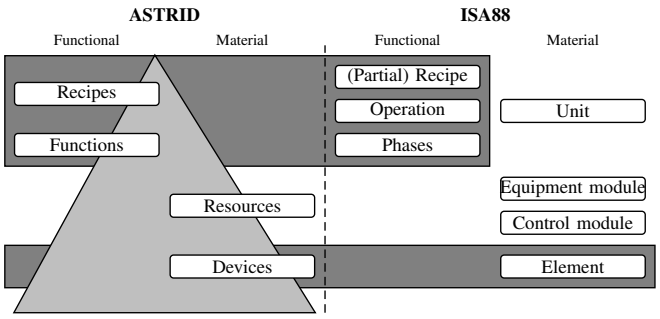


Figure 1: Levels of abstraction used in the ASTRID method and in the ISA88

Complementarily, works on batch processes operating procedures aim to propose graphical representations, as in [4]–[6] who introduce the Grafchart formalism, a Grafcet extension for hierarchical and sequential procedures modeling. If these works allow to represent industrial processes and their operating procedures, automatic generation of actions sequences *a priori* respecting safety constraints requires the use analytics mechanisms on formal models. Obtaining procedures for critical systems is a problem especially addressed by [7], who where among the first to propose an action generation approach for systems made of valves. This work has then been extended by [8] who proposed an automatic trajectory generation method suitable for more complex systems, based on the changes of state between initial and goal situations, and taking into account safety constraints. However it did not take into account structural elements of the process or physical variables. Other approaches, relying on the reachability of several intermediary goals before reaching the final goal situation, has been proposed by [9], [10] in an objective of procedures scheduling under constraints [11].

These works mostly use heuristics or operational research techniques that cannot cover modular aspects of modeling (hierarchical levels, equipments, functions, ...) and dynamic behaviours of operating architectures (materials consignment for example).

Discrete event systems based approaches [12] carry answers to the previously mentioned limits. They allow structural and behavioural formal representation of a system. Using these approaches, several works have been carried out for action sequences generation, as for example [13] using Petri nets to propose a sequence generation methodology, applied to the case study in [8]. No aggregation mechanism is however treated, despite an important amount of elements to consider.

To facilitate hierarchical organisation representation of the considered systems, statecharts, a visual formalism for complex systems representation [14], can be considered. Statecharts are an extension of state-transition diagrams using additional mechanisms: hierarchy, concurrence and communication. Hierarchy being one of the key characteristic of the systems of interest, it is there a strong point for modeling formalism choice, because it enables state aggregation representation. However, overlapping, that is to say the fact that an equipment can belong to more than one function, cannot be represented with statecharts while preserving modeling language semantic, as indicated in [15]. This problem seems to be resolvable using communicating automata [1], an extension of finite state automata, that include both concurrency and synchronisation notions.

Thanks to their syntax and semantic, formal languages previously cited allow model simulation and formal property analysis (such as reachability properties), by a state space exploration. Moreover, the integration of formal rules specification ensures a complete respect of safety constraints. If existing, a path reaching a goal situation in the state space corresponds to a feasible safe action sequence.

Supervisory control theory [16] is an approach allowing to get the set of all existing paths [17], and showed its interest in [18] and [19] for manufacturing systems reconfiguration. This technique however induces a problem regarding how to choose a path among the set of possibilities.

Model checking approaches also allow reachability properties verification, and then can be used to generate action sequences, as shown in [20] for cyclic operating procedures. Compared to supervisory control, this technique only generate one path, so that sequence choice is no longer a problem. This interest is reinforced by the work of [21] that shows the path find can be near of optimality.

III. Existing approach evaluation

A. Evaluation principles

This article aims to evaluate a methodology for automatic action sequence generation using process flow diagrams and safety constraints specifications, based on:

- a communicating automata model, using generic models specialised with specific knowledge (such as safety constraints on guards), and structured according to abstraction levels,
- a goal situation reachability formalisation as a CTL property,
- and a property verification mechanism.

B. Structural levels

In this article, a three-level decomposition (Figure 2), based on the ASTRID method principles, is used for the evaluation of the action sequence generation methodology. The "equipment" level is the lower abstraction level, corresponding to an aggregation of ASTRID "device" and "resource" levels, and representing the different handled equipment pieces (valves, pumps, ...). From a functional point of view, equipments are regrouped in "functions" enabling the evolution of one or more physical variables of φ, the set of all physical variables of the whole process. Finally, the "recipe" level, specific to the process, describes specified goals.

C. Modeling principles

1) Modeling using communicating automata: Communicating automata where first defined in [1], introduced as a subclass of the temporised automata. Communicating automata can be defined by a 7-uplet $A = (S, X, L, T, S_m, s_0, v_0)$ as:

- S is a finite set of locations;
- X is a finite set of variables;
- L is a set of events, decomposed in two disjoints subsets L_e and L_r, where
  - L_e is the set of emitted events;
  - L_r is the set or received events.

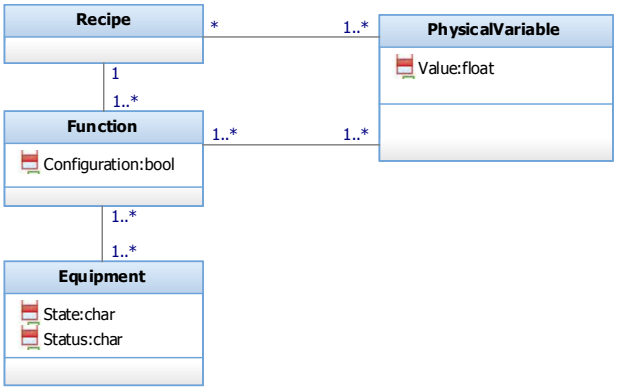


Figure 2: Abstraction levels used for action sequence generation

- T is a set of transitions $(s, l, g, m, s') \in S \times L \times G \times M \times S$, where
  - G is a set of guards (constraints on variables of X);
  - M is the set of the updates on variables values of X.
- $S_m \subseteq S$ is a set of marked locations;
- $s_0 \in S$ is the initial location;
- $v_0 : X \leftarrow \mathbb{N}$ is the initial variables values.

In the automata models presented in this paper, the following graphical conventions are used: location names are given in bold, initial location is given by a transition with no source, guards on transitions are between brackets, events are in italic and followed by "!" or "?", to represent respectively emission or reception, and updates are underlined.

2) Equipment generic models proposal: Equipments, mostly valves and pumps, are elements on which whether operator or control can operate on to change state. An equipment is characterised by a couple (*state*, *status*), defined as:

- *state* characterises a set of discrete values to represent an equipment state (e.g. for a valve: open/closed),
- *status* characterises an equipment operational configuration (e.g. padlocked, condemned...).

Two locations express equipment *state*, and a boolean variable is used to represent equipment *status*. This boolean variable has to be defined at the initialisation of the model (in v_0) and will not change during state space exploration, the goal being to determine the reachability of a given situation through different actions while taking into account the current equipments *status*.

A valve behavior can thus be modeled according to the pattern presented in Figure 3. To evolve from a location towards another, a valve must receive an opening / closing order, modeled by the *OpenValve* / *CloseValve* events. Safety constraints binding opening or closing actions are modeled by *Opening_Constraints* / *Closing_Constraints* guards on transitions. Moreover, a valve can only be operated if it is not condemned, which is taken into account by adding a boolean variable *Condemned* on the guards.

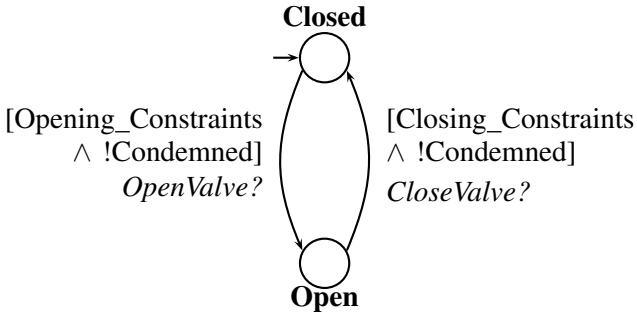


Figure 3: Generic model for valves

In a similar way, a pump behavior can be modeled according to the pattern presented in Figure 4. To evolve from a location towards another, a pump must receive a start / stop order, modeled by the *StartPump* / *StopPump* events. Safety constraints binding start or stop actions are modeled by Start_Constraints / Stop_Constraints guards on transitions. Moreover, a pump can only be operate if it is not condemned, which is taken into account by adding a boolean variable *Condemned* on the guards.

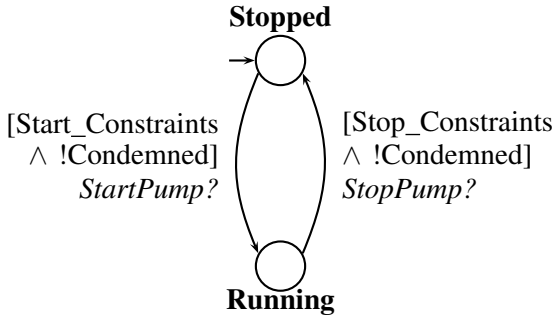


Figure 4: Generic model for pumps

3) Function generic model proposal: The configuration of a function (Figure 5) is characterised using two locations defined by:

- *Configured* if $\forall eqt \in EQT_f$ the equipment *state* fits to the awaited state for a given configuration of the function f, leading to an evolution on the set $\varphi_f \subseteq \varphi$ of the concerned physical variables;
- *Not configured* in other cases.

Transition from a location towards another is bound firstly to the validation of a guard that corresponds to a possible configuration for the function to be executed (condition on the *state* of equipments), and secondly to the occurrence of the *Function* event in charge of the synchronisation of different levels of models.

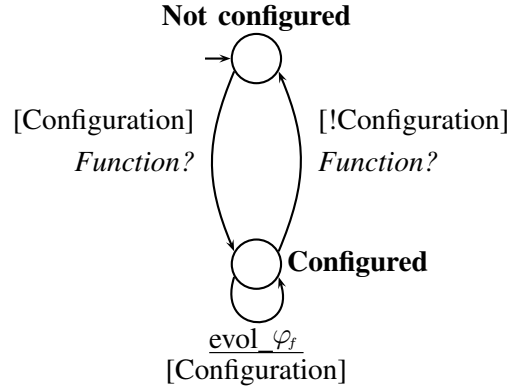


Figure 5: Generic model for functions

All configurations were previously found using a graph search methodology adapted to our needs. We transformed the process flow diagram to a graph, with equipments modeled by vertices, and pipes modeled by arcs [22]. Using this graph, a configuration, which we could define as a path leading a flow from a source to a destination, can easily be found [23].

4) Recipe modeling: The "recipe" level, which describes the set of situations, is specific to each process. Transition from a location towards another is bind to a guard modeling a set of constraints on physical variables and on equipments *status*. It is also conditioned to the occurrence of the event *Recipe*, that synchronises models of different levels.

5) Synchronization of different models: The whole model behaves according to the following description: one particular event l_{eqt_i} of the set $L_{eqt} \subseteq L_e$ of events associated with actions modifying the *state* of an equipment is emitted. This action has an effect on the function, thus impacting the process. This principle is modeled by a "generation" automata (Figure 6), that has two roles:

- To generate events (that correspond to actions on equipment status): it is necessary to explore the state space for the verification of the reachability property,

- To synchronise the execution of the different models: an occurrence of an event on an equipment ($l_{eqt_i}!$), is followed by an event to update functions models (*Function!*), and one final event to update recipe model (*Recipe!*).

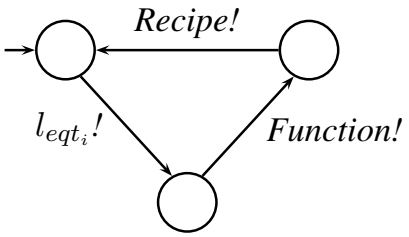


Figure 6: Events generator automata to synchronise models

6) Situation reachability analysis: The reachability of a state in the *Recipe* model means that at least one feasible sequence exists leading to this state. Therefore, finding a feasible sequence (satisfying all constraints) can be considered as a reachability problem.

A reachability property can be expressed in a formal way as a CTL formula "EF p", where the "E" quantifier means "Exists", the "F" quantifiers means "eventually, in the future", and p corresponds to the goal specification. In natural language, property "EF p" means that there exists a path such as property p holds in the future.

Such reachability problem can be addressed with automatic techniques, like model checking [24]. Model checking is an automatic technique that explores a model $\mathcal{M}$ state space to formally verify if $\mathcal{M}$ satisfies a property p (also expressed $\mathcal{M} \models p$). If reachability property holds, the execution trace that results represents a feasible action sequence.

IV. A CASE STUDY: CISPI

A. Case study presentation

For evaluation purposes, the considered case study is based on the CRAN laboratory platform named CISPI (French acronym for Safe and Interactive Operating of Industrial Processes (http://safetech.cran.univ-lorraine.fr/). Its process flow diagram is given in Figure 7.
Due to the various physical redundancies of the CISPI platform, there exist 8 different configurations to supply tank 002BA with water from tank 001BA.

On this process, the considered safety constraints are related to:

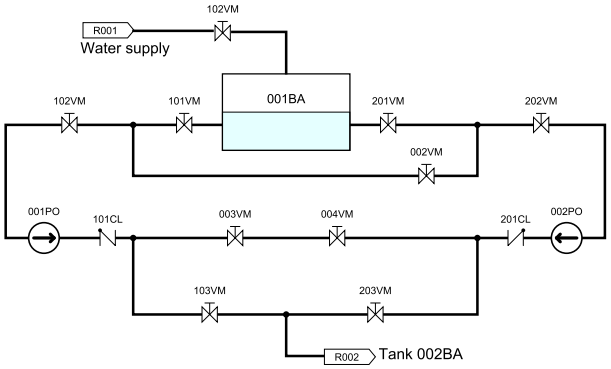
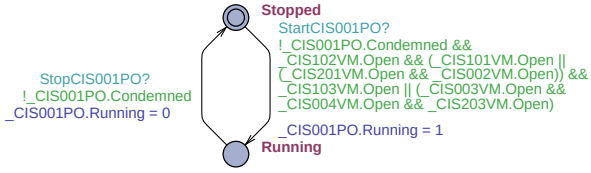


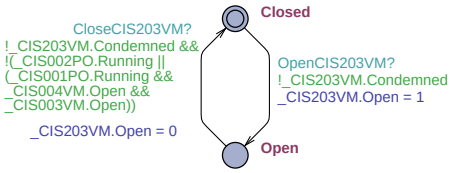
Figure 7: Process Flow Diagram of the CISPI platform case study

- pump start, which need both upstream and downstream valves to be opened,
- valves closing , which may required prior pumps stop.

The generic models previously presented in section III have been instantiated to model the 10 valves and pumps, and the 8 functions considered. Safety constraints in transition guards are manually derived from the plant topology. These models have been implemented using the Uppaal tool [25] (Figure 8). This tool provides both a graphical interface for automata models edition [1] and a model checker that allows verification of formal property written in CTL. Among all the existing model checkers, this particular tool has been chosen for its ease of handling, the main objective being to evaluate such type of approach.



(a) Pump 001PO



(b) Valve 203VM

Figure 8: Instantiation examples relative to CISPI

[1]Graphical conventions are: circles represents automata locations, two-circled location is initial state; in purple, location names; light blue is events, green is guards, and blue is updates on variables.

The partial recipe in Figure 9 shows two locations *SF0* and *SF1* corresponding to operating situations. Transition from one to another if conditioned by the reaching of a specified level in tank 002BA. CTL property that express the reachability of this second location is: *EF Recipe.SF1*.

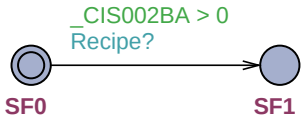


Figure 9: Partial recipe model

B. Sequence generation results

The methodology presented in section III, using formal models presented in section III-C, is implemented on the considered case study (Figure 10). The models are analysed using UPPAAL model checker, given the reachability property *EF Recipe.SF1*, to finally obtain, if existing, an action sequence.

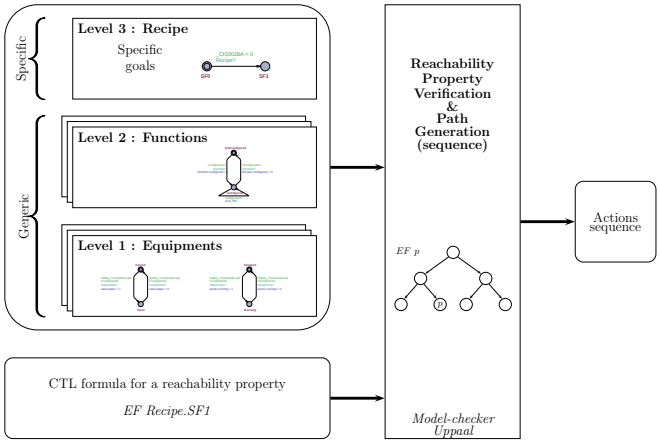


Figure 10: Implementation of sequence generation approach on CISPI case study

Let us consider a first hypothesis where the whole platform is in ”shut down” mode, which means that all valves are closed and all pumps are stopped. Additionally, we will consider that the water level in tank 001BA does not constitute a limit for tank 002BA water supply. A generated action sequence is given in Table I. It enables the water supply *via* the ”left side”. It should be noted that the order of the actions matters, as it takes into account safety constraints.

Let us consider a second hypothesis where valve 201VM is *condemned* and *closed*, and where a maintenance operation is scheduled on pump 001PO (implying, for safety reasons, condemnation of valve 102VM in state *closed*). An alternative action sequence to reach the predefined goal is given in Table II.

Table I: Actions sequence 1: no condemnation

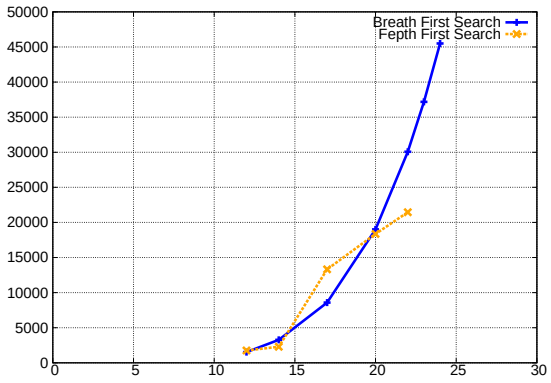
	Action	Equipment
1	Open	101VM
2	Open	102VM
3	Open	103VM
4	Start	001PO

Table II: Actions sequence 2: 001PO maintenance, valves 102VM and 201VM condemned closed

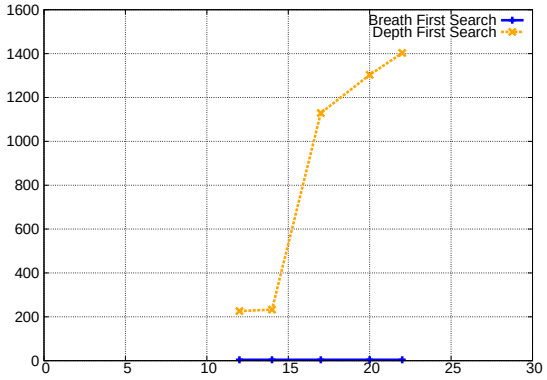
	Action	Equipment
1	Open	203VM
2	Open	202VM
3	Open	101VM
4	Open	002VM
5	Start	002PO

V. Discussion

The implementation on a laboratory case study showed the feasability of action sequence generation, with an approach based on existing methods. However, the increasing size of models needed to fit industrial scale puts in evidence limits of such an approach (Figure 11). Applications on larger scale examples show a rapid state space growth. Moreover, generated sequences are not optimal because of ”useless” actions, such as opening/closing successions of actions on a same equipment.



(a) State space size evolution



(b) Number of actions evolution

Figure 11: Evolutions of state space size and number of actions in relation to the number of equipments

Within model checking mechanisms, breadth first search limits the size of the generated sequence, and can even propose (almost) optimal sequences. Even though this technique gives promising results, the state space growth is exponential, rapidly limiting the size of the systems of interest which can be addressed.

Depth first search limits the growth of the state space, but this impacts the size of the generated sequence. Although the state space growth seems to be linear, the results of this exploration method are unusable for real applications.

VI. Conclusion and perspectives

This article proposes an evaluation of an automatic generation approach for safe action sequence. This approach is based on a goal situation reachability analysis in a structured communicating automata network. The obtained results eventually show the problems of this approach regarding its application to large scale systems.

Several axes are currently under study to reduce the combinatorial explosion caused by state space exploration, mainly focusing on modelling issues. On one hand, a stronger integration of physical constraints should also limit the size of the state space explored to find a feasible sequence. On the other hand, aggregation and multi-scale modeling techniques should reduce models size, as well as changing the lower level of modeling to the functions and their effects on variables values.

In particular, next work will be focused on abstraction techniques, more specifically on the verification that a particular abstract model satisfies the same properties, both from a behavioral and a safety point of view, as a set of detailed models. It is hoped that abstract models can then produce higher level action sequences, thus allowing large-scale systems to be considered.

Acknowledgment

This works is supported by the CONNEXION Cluster, financed through the "*Investissements d'Avenir / Briques Génériques du Logiciel Embarqué*".

References

[1] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[2] ISA, "Ansi/isa-88.01-1995 : Batch control. part 1 : Models and terminology," *The Instrumentation, Systems and Automation Society*, 1998.

[3] I. E. C. TC65, "Iec 61512-1: Batch control. part 1: Models and terminology," *International Electrotechnical Commission*, 1997.

[4] K.-E. Arzen and C. Johnsson, "Object-oriented sfc and isa-s88. 01 recipes presented at the world batch forum," *ISA transactions*, vol. 35, no. 3, pp. 237–244, 1996.

[5] S. Viswanathan, C. Johnsson, R. Srinivasan, V. Venkatasubramanian, and K. E. Ärzen, "Automating operating procedure synthesis for batch processes: Part i. knowledge representation and planning framework," *Computers & chemical engineering*, vol. 22, no. 11, pp. 1673–1685, 1998.

[6] ——, "Automating operating procedure synthesis for batch processes: Part ii. implementation and application," *Computers & chemical engineering*, vol. 22, no. 11, pp. 1687–1698, 1998.

[7] J. R. Rivas and D. F. Rudd, "Synthesis of failure-safe operations," *AIChE Journal*, vol. 20, no. 2, pp. 320–325, 1974.

[8] N. Foulkes, M. Walton, P. Andow, and M. Galluzzo, "Computer-aided synthesis of complex pump and valve operations," *Computers & Chemical Engineering*, vol. 12, no. 9, pp. 1035–1044, 1988.

[9] R. Fusillo and G. Powers, "A synthesis method for chemical plant operating procedures," *Computers & chemical engineering*, vol. 11, no. 4, pp. 369–382, 1987.

[10] ——, "Operating procedure synthesis using local models and distributed goals," *Computers & Chemical Engineering*, vol. 12, no. 9, pp. 1023–1034, 1988.

[11] H. S. Li, M. L. Lu, and Y. Naka, "A two-tier methodology for synthesis of operating procedures," *Computers & chemical engineering*, vol. 21, pp. S899–S903, 1997.

[12] C. G. Cassandras and S. Lafortune, "Introduction to discrete event systems," 2008.

[13] Y.-F. Wang, H.-H. Chou, and C.-T. Chang, "Generation of batch operating procedures for multiple material-transfer tasks with petri nets," *Computers & chemical engineering*, vol. 29, no. 8, pp. 1822–1836, 2005.

[14] D. Harel, "Statecharts: A visual formalism for complex systems," *Science of computer programming*, vol. 8, no. 3, pp. 231–274, 1987.

[15] D. Harel and C.-A. Kahana, "On statecharts with overlapping," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 1, no. 4, pp. 399–421, 1992.

[16] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM journal on control and optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[17] M.-L. Yeh and C.-T. Chang, "An automata-based approach to synthesize untimed operating procedures in batch chemical processes," *Korean Journal of Chemical Engineering*, vol. 29, no. 5, pp. 583–594, 2012.

[18] R. G. Qiu, "Virtual production line based wip control for semiconductor manufacturing systems," *International Journal of Production Economics*, vol. 95, no. 2, pp. 165–178, 2005.

[19] J.-F. Pétin, D. Gouyon, and G. Morel, "Supervisory synthesis for product-driven automation and its application to a flexible assembly cell," *Control Engineering Practice*, vol. 15, no. 5, pp. 595–614, 2007.

[20] J.-H. Li, C.-T. Chang, and D. Jiang, "Systematic generation of cyclic operating procedures based on timed automata," *Chemical Engineering Research and Design*, vol. 92, no. 1, pp. 139–155, 2014.

[21] P. Marangé, J.-F. Pétin, A. Manceaux, D. Gouyon *et al.*, "Contribution à la reconfiguration des systèmes de production: ordonnancement par recherche d'atteignabilité," *Journal Européen des Systèmes Automatisés*, vol. 45, no. 1/3, pp. 45–60, 2011, in French.

[22] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, vol. 45, no. 3. Citeseer, 2003, pp. 1–17.

[23] J.-C. Fournier, *Graphes et applications*. Hermès science publications-Lavoisier, 2007.

[24] E. M. Clarke, O. Grumberg, and D. Peled, "Model checking," 1999.

[25] G. Behrmann, A. David, K. G. Larsen, J. Hakansson, P. Petterson, W. Yi, and M. Hendriks, "Uppaal 4.0," in *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*. IEEE, 2006, pp. 125–126.