Supervisory Control of Discrete-Event Systems: A Brief History – 1980-2015

W.M. Wonham * **Kai Cai** ** **Karen Rudie** ***

* *Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4 Canada (e-mail: wonham@ece.utoronto.ca).*
** *Department of Electrical and Information Engineering, Osaka City University, Osaka, 558-8585 Japan (e-mail: kai.cai@eng.osaka-cu.ac.jp)*
*** *Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6 Canada (e-mail: karen.rudie@queensu.ca)*

Abstract: This brief history summarizes the 'supervisory control of discrete-event systems' as it has evolved in the period 1980-2015. Overall, the trend has been from centralized or 'monolithic' control to more structured architectures, and from 'naive' to symbolic computation. Like any 'history' this one represents the perspective of the authors; inevitably some important contributions will have been overlooked or short-changed.

1. BACKGROUND AND EARLY MOTIVATION c.1980

The year 1980 is a convenient point of departure for supervisory control of discrete-event systems (SCDES) in the current sense. By that time the broad field known as 'modern' systems control that had evolved over the previous 20 years was well-established, on the basis of the five fundamental concepts of feedback, stability, controllability, observability, and quantitative optimality [AF66]. [1] Of particular relevance later to SCDES were geometric concepts for regulator synthesis by feedback in linear multivariable systems [Won85], [BM91], namely the lattice of subspaces of a linear vector space, including controlled invariant subspaces, controllability subspaces, supremal (unique maximal) elements in the sense of partial order by subspace inclusion, and the concomitant notion of qualitative optimality. The dynamical systems to which systems control applied were generally those described by ordinary or partial differential equations and their discrete-time sampled-data counterparts, the main application drivers being the industrial process industries and various national space programs.

By contrast, discrete-event systems (DES) was an area apart, concerned with systems usually discrete in time and state space, driven by instantaneous events other than (or in addition to) the tick of a clock, and 'nondeterministic' in the sense of making state-transitional choices by internal chance or other mechanisms not necessarily modeled by the system analyst. Such systems were generally not amenable to the differential techniques of systems control. The application drivers included manufacturing, traffic, database management, and logistic systems. Owing to model complexity and analytical intractability, system simulation played a major role in analysis and optimization; indeed the term 'DES' seems to have originated with the simulation community and computer languages like SIMULA and SIMSCRIPT [Fis78]. Theoretical analysis rested on queues [EVW80], Markov chains [How60], Petri nets [Pet81], and boolean transition structures [Ave74]; while design approaches exploited semaphores [Dij65], path expressions [Shi79], and computer program representation by pseudo-code along with 'by-hand' cut-and-try alternating with informal verification [BAri82]. Formal attacks evolved in response, especially in software science, either expressed in process algebras such as *communicating sequential processes* (CSP) [Hoa85], the *calculus of communicating systems* (CCS) [Mil89], and work of the emerging Dutch school [Bae04]; or proposed from a language perspective in terms of process behaviors [BN80], [AN80]. [2]

In this literature control problems were certainly implicit, but formal synthesis (in the style of systems control) was broadly lacking. No standard paradigm existed analogous to optimal control, and there was often no clear separation of controller and uncontrolled 'plant'. The need (or at least the interest) was therefore apparent of a DES control theory which would (1) be discrete in time and space, asynchronous, event-driven as well as (possibly) clock-driven, and nondeterministic (supporting autonomous transitional choices); (2) rest on a simple control 'technology' and exploit standard control concepts; (3) be amenable to computation and applicable to the DES drivers (such as

⋆ This work was supported in part by the Natural Sciences and Engineering Research Council, Canada, Grant no. 7399 and RGPIN-2015-05699; JSPS KAKENHI Grant no. JP16K18122.

[1] For the reader's convenience, in place of original sources textbooks or monographs may be cited where references to the former can be found.

[2] While in several ways this paper foreshadowed the publications [RW82], [RW87a] cited below, it was not until 1992 that the respective authors became aware of each others' work. We thank Prof. Arnold for helping to clarify this connection, and retrospectively Dr. Angelo Bean for mediating between our two communities which, at that time, were mutually rather isolated.

manufacturing); and (4) be accessible to practitioners and students of control engineering.

2. LANGUAGE CONTROLLABILITY AND MONOLITHIC SUPERVISORY CONTROL – 1981-1987

The DES control theory which appeared in response [RW82] was not startling. The plant was modeled internally as a finite state machine (FSM) for ease of physical interpretation and explicit computation;[3] plant external behavior was thus a regular language, which could optionally be considered the conceptual starting point independently of representation. Control specification was also modeled as an FSM, the corresponding regular language providing an upper bound on acceptable controlled behavior. The key advantages of this setting were its flexibility, broad expressiveness, and the technical feature of regular language closure under the boolean operations. The proposed control technology was simply a partition of the language alphabet (of *events*) into *controllable* and *uncontrollable*, the former amenable to disablement (prevention from occurrence) by a hypothetical external agent dubbed *supervisor*, the latter events not capable of direct disablement but presumed liable to occur (by chance, or internal system 'volition') whenever defined at the system's current state.[4]

The *supervisory control synthesis problem* was then formalized as that of designing a finite-state supervisor which, on observing the string of events generated by the plant, would at each state disable a suitable subset of controllable events to ensure that the generated controlled behavior (regular language) continued to satisfy the control specification, namely remained within the specification or 'legal' language. As in standard systems control, the approach was thus to separate the issues of problem definition and explicit computation.

Inasmuch as disabling all controllable events (in a sense, allowing as little as possible) could often yield a (trivial) formal synthesis, a concept of qualitative optimality was introduced requiring that the controlled behavior be as rich as possible (*maximally permissive* or *minimally restrictive*) subject to the specification constraint. Therefore the need arose to identify the subclass of (regular) languages which, for given plant and specification, could be synthesized as just described, and within which an optimal behavior could be shown to exist. Thus the final and key ingredient of the theory [RW82], [RW87a] was the concept of *controllable language*, and the crucial fact that the controllable sublanguages of a given (specification) language admitted a *unique* maximal (or *supremal*) element. More technically, the legal controllable sublanguages form an upper semilattice under the partial ordering of language (i.e. string subset) inclusion. The solution of the formal problem of optimal control is thus precisely the *top* element of this semilattice, or *supremal controllable sublanguage* (of the legal language). The latter was shown to be effectively computable by an algorithm we shall later call 'Supcon'. The conceptual debt to previous geometric regulation theory was evident.

Not uniquely in the annals of interdisciplinary research [Gol89], the community response to this proposal ranged from indifference to hostility. Computer specialists dismissed the engineering application of FSM and regular languages as trivial and/or nothing new, while control specialists regarded FSM as impractical and/or irrelevant: "Finite automata," declared one anonymous reviewer for a leading journal, "have no place in control engineering." Eventually the archival paper was accepted by a third journal as a putative contribution to 'optimal control' [RW87a].

3. CONFRONTING THE COMPUTATIONAL CHALLENGE

Attempts to apply the new theory to industrial problems encountered the barrier notorious as *exponential state space explosion*. Thus a workcell with N machines each having k states would be modeled as a plant with *a priori* state size $\sim k^N$, so 10 machines each with 5 states would result in a global model with state count $5^{10} \sim 10$ million. Naive *extensional* representation of such systems (whereby the transitions are all listed and stored explicitly) rapidly becomes infeasible. As a first response, researchers therefore turned to 'smart architectures' involving horizontal and vertical modularity, or in systems terms decentralized [RW87b] and hierarchical [ZW90] decompositions, later including distributed control by supervisor localization [CW10a] (described below).

In their basic conceptual form these approaches depend on first computing the 'global' centralized or 'monolithic' control. Indeed, while control authority may ultimately be 'local', namely decomposable into specialized controllers with authority over just a few plant components, to guarantee that these entities interact without mutual conflict (which could result in system blocking, or even deadlock) means solving the global nonblocking problem, subject yet again to exponential computational effort. In other words, modular control typically requires global coordination, which threatens as before to be computationally infeasible.

Several approaches, used singly or in combination, emerged to grapple with this issue. Essentially they sought to combine efficient system architecture with 'smart' computation. Thus the computational model of *state charts* [Har87] was adapted for control purposes in the version *state tree structures* (STS) [MW05]. These are layered (or hierarchical) models which make essential use of *intensional* (as distinct from extensional) representation of control functions using *boolean decision diagrams* (BDDs) [Bry86].[5] With intensional representation a computable entity is stored not by tabling its values but instead providing an algorithm by which they are computed explicitly just when needed (as in the decimal representation of numbers, where '123' is stored instead of, say, a string of 123 1's). Another efficient model class to be introduced was *extended state*

[3] That the two areas of control theory and automata theory shared ideas in common had been recognized for some time (see e.g. [Arb65]).

[4] The term *supervisor* for such a 'disabling agent' was adopted to distinguish it from *controller* (conventionally a 'forcing agent'). Nevertheless, forcing action can effectively be modeled within the theory when needed.

[5] For apparently the earliest application of symbolic computation to supervisory control see the report [Gun97].

machines (ESM) [CL00], [YG05], [SAF07], namely FSM parametrized by boolean and integer variables for logic elements and buffers, plus logic-based transition guards and variable assignments for succinct representation of state transitions. Other model types found useful included (bounded) *Petri nets* [Kro87] or *vector DES* [LW94] (either of these usually equivalent to a synchronous product of buffers), especially when processed using FSM algorithms of Supcon type to achieve maximal permissiveness with nonblocking [CL13].

4. LANGUAGE OBSERVABILITY AND MONOLITHIC CONTROL WITH PARTIAL OBSERVATIONS

Successful formalization of 'local' control structures rests on some notion of 'local observability'. In systems control *observability* is a property of a plant together with its output or observation structure, which guarantees that enough data about plant behavior (or current state) are available for implementation of a given class of controls (such as arbitrary state feedback controls). Specialized to our DES model, 'observation' has been modeled by a channel for transmission of the generated language strings from plant to supervisor. The simplest type of channel has zero memory, transmitting selected alphabet symbols one-by-one without change (in the case of *observable events*) or else erasing them altogether (the *unobservable events*). Formally, the channel is modeled by a *natural projection* from a language over a given alphabet to its image over a specified *observable subalphabet*. A language is then said to be *observable*, with respect to a given plant and natural projection if, for every plant-generated string already in the language, its projection determines consistently whether a putative one-step (event) extension of the string remains a member of the language or not. It is shown that a controllable language can be synthesized (in a feedback loop with the plant) on observing only the projected generated strings (i.e. strings 'output by the channel') if and only if the language is observable ([LW88], [CDFV88]). In the regular language framework observability is decidable in time polynomial in the state size of the targeted language [Tsi89].

Conceptually, at least, the foregoing developments brought SCDES into the mainstream of systems control. Unfortunately, observability has turned out to be intractable for practical synthesis, the technical reason being that, unlike controllability, this property fails to be closed under language union; the upper semilattice algebraic structure that holds for controllable languages alone therefore fails; hence no optimal (unique maximally permissive) solution to the problem of *supervisory control under partial observations* (SCOP) need generally exist. Relaxing optimality to require only a 'maximal' solution is of no particular help inasmuch as a designer would generally have no idea where such a maximal element might be located in the landscape of solutions; in any case it is currently unknown how to compute even an 'adequate' solution to SCOP on the sole assumption that one happens to exist.

In mitigation, conditions stronger than full observability have been proposed that are tractable and, while a large catalog of realistic applications has yet to emerge, in many instances yield a useful result. The earliest and simplest was *normality* [LW88], namely that a language is determined essentially by its inclusion in the plant and specification languages together with its image under the given natural projection for partial observation. It is shown that normality implies observability, and that the family of controllable normal languages admits a supremal element that is often tractably computable despite exponential worst-case complexity. The main shortcoming of this normality solution to SCOP is that an event can be considered controllable (i.e. subject to possible disablement) only if it is observable. In general this means that the supremal normality solution may be empty even though some observability solution is not, albeit whether or not the latter exists will in general be problematic. More recently, however, an improved condition of *relative observability* has been proposed [CZW15b], stronger than observability but strictly weaker than normality and with the same desirable property of closure under language union. Thus the family of controllable and relatively observable languages admits a supremal element, yielding a 'relatively' optimal solution to SCOP, which happily places no restriction on the disablement of unobservable controllable events. As with normality, this solution can be tractably computable; examples have shown its practical utility as well as its generally greater permissiveness than the normality solution to SCOP.

5. DECENTRALIZED AND DISTRIBUTED CONTROL WITH PARTIAL OBSERVATIONS

Language observability was first applied to decentralized control via the extended concept of *co-observability* [RudW92]. The setup envisaged a global plant, together with a team of several isolated 'agents' each with an assigned subset of observable events (i.e. channel with corresponding natural projection) and an assigned subset of controllable events; an agent may share its observable and controllable events with other agents (i.e. agents' alphabets may possess elements in common). For a given controllable language meeting the (global) control specification, each agent is assumed independently to decide whether or not each 'next' controllable event should be enabled or disabled; it then communicates its decision to a central controlling authority. Employing one of several possible rules for 'decision fusion' [YL02] the latter implements the collective control decision. The given language is defined to be *co-observable* if this decision is always correct; in the case of just one agent, co-observability reduces to observability. With several agents, a controllable language can then be synthesized in the described decentralized architecture; for this the co-observability property is both sufficient and necessary.

Unfortunately, just as with the monolithic setup, while co-observability is effectively decidable [RudWil95] it is not preserved by language union and in general a supremal (unique maximally permissive) solution to the decentralized SCOP fails to exist; nor is it obvious how any acceptable solution might be found, granting that one existed at all. In mitigation as before, co-observability may be relaxed to its counterpart *co-normality* [DL14], or more subtly to *relative co-observability* [CZW15a] provided the

decision fusion rule is (severely) restricted to ensure the property of closure under union.

If the system fails to be co-observable, it is tempting to adjoin the stronger feature that agents be allowed to exchange information in accordance with an appropriate 'topology' of communication. Unfortunately again, design of the relevant protocols has turned out to be extremely difficult, owing to the interaction of agents' decisions due to the intertwining of communication and control. A fallback led to the simpler *state disambiguation* problem [RLL03], though bringing with it the technical obstruction of non-monotonicity, namely that enhanced observation need not imply improved disambiguation [WLL08]. In any case, state-dependent (or 'dynamic') observation with either or both observation and communication costs suggested the *sensor activation* problem [TT07] of optimizing the relevant tradeoffs; related goals could include optimal communication strategies to minimize network bandwidth or preserve network security [SR16].

A successful blend of architectural and observability concepts has been brought to bear in 'heterarchical' supervision, which exploits both decentralized and hierarchical control based on suitable system abstractions. First the given (large) system is split into smaller-scale subsystems, for which decentralized supervisors and coordinators (which enforce nonblocking) can be efficiently synthesized. Abstracted models of the resulting controlled subsystems are then computed by natural projections. These must be chosen to have the technical properties of being *natural observers* [WW96] and in some sense *control consistent* [FW08], [SB11]. The result is a hierarchical array of decentralized supervisors and coordinators that achieves global optimality with nonblocking. This approach has been demonstrated with the benchmark Production Cell, of state size 10^8 [FCW09].

Another effective approach to distributed control has been introduced called *supervisor localization* [CW10a]. The latter envisages a plant composed (using *synchronous product*) of several modular components and a specification composed of several individual component specifications. By contrast with decentralized control, which usually means the allocation of separate specialized controls to separate component specifications, distributed control allocates separate controls to distinct plant components. This allocation (or *localization*) is achieved by decomposition of a given monolithic supervisor (or more generally each member of a given family of decentralized supervisors) by means of constructing suitable *control congruences* [SW04] (equivalence relations that respect both dynamics and control actions) on the relevant supervisor state sets. The result is to convert each plant component into a 'smart agent'. In general each agent communicates with an optimistically small number of (logical) 'neighbors' for exchange of information on event occurrence that is essential for control. This pattern of agent intercommunication is not assigned *a priori* but emerges as part of the problem solution. It is proved that the resulting distributed control behavior is identical with the monolithic or decentralized behavior adopted at the start, so if the latter is optimal then so is the localized result. There is thus no question of the latter's existence or in principle its feasible computation. In case the monolithic controller is too large to compute, localization may possibly be combined with the heterarchical approach described above [CW10b].

Finally, as with any distributed control architecture, it has been deemed important to investigate its 'robustness' when inter-agent communication is subject to channel delay. The general problem of SCOP with several agents, no *a priori* architectural restrictions, and bounded or unbounded communication delay, has been proved to be undecidable [Tri04]. Nevertheless, more narrowly defined problems of this type have yielded useful insights [Lin14], [ZCG+16].

6. EXTENSIONS TO SCDES WITH BROADER FUNCTIONALITY

For completeness' sake we note several extensions of SCDES that have been proposed for broader functionality and richer specifications. One of the earliest was based on temporal logic [Ram89], allowing (among other things) the expression of 'eventuality', namely the occurrence or otherwise of some event 'in the long run' without regard to an *a priori* bound in time. For this the more technical setting is needed of *ω-languages* [TW94], which include infinite strings (as distinct from the regular languages of SCDES, whose strings are always finite albeit may be of unbounded length).

Stimulated by earlier work [OW85] on temporal logic, a timed version of DES, or TDES, was introduced [BW94], allowing the incorporation of event delays and deadlines as measured by a global digital clock, and including a notion of *forcible event* thought of as preempting the clock's tick. An alternative approach via the notion of *timed automaton* (properly a generic term, but here with a specific definition) was proposed [AD90]; this more technical setting admits multiple local clocks, analogous to the event 'timers' of TDES, but possibly measuring the 'real time' of physics.

Timed versions of some of the other DES representations noted above are an active area of current research.

7. INDUSTRIAL APPLICATIONS

To conclude this historical overview we report that realistic industrial applications of SCDES are as yet few in number. This situation is due in part to a lack of experience among control engineers with modeling and specification in the framework of automata, but (more seriously) to the lack of software of industrial strength adapted to engineering design. While numerous applications have been proposed in the literature, relatively few have been demonstrated on actual hardware in a commercial environment. In any case, the widespread industrial technologies of *programmable logic controllers* (PLCs) and *sequential function charts* (SFCs) were utilized in experimental SCDES controllers at an early stage [LW95], [HFL01]; and there is now convincing evidence that such a bridge between theory and practice is feasible. One of the first such applications was the testbed assembly process of the Atelier Interétablissement Productique (AIP) in Grenoble, France [BC94].

Another application with commercial implications has been the design of a telephone directory assistance call center [Sei06].

Finally, the power of SCDES in the control synthesis (as opposed to cut-and-try design) of a complex DES with over 6 billion states, namely the patient support system for a magnetic resonance image (MRI) scanner, has been impressively demonstrated in [TPS+13].

REFERENCES

[AD90] R. Alur and D. Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages and Programming. Lecture Notes on Computer Science (LNCS) No. 443, Springer*, pages 322–335, 1990.

[AF66] M. Athans and P. Falb. *Optimal Control: An Introduction to the Theory and its Applications*. McGraw-Hill, 1966.

[AN80] A. Arnold and M. Nivat. Controlling behaviours of systems: some basic concepts and some applications. *Theoretical Foundations of Computer Science, Lecture Notes on Computer Science (LNCS) No. 88, Springer-Verlag*, pages 113–122, 1980.

[Arb65] M. A. Arbib. A common framework for automata theory and control theory. *J. Society of Industrial and Applied Mathematics (SIAM) Ser. A, Control*, 3(2):206–222, 1965.

[Ave74] R.L. Aveyard. A boolean model for a class of discrete event systems. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-4(3):249–258, 1974.

[Bae04] J.C.M. Baeten. A brief history of process algebra. Technical report, Rpt. CSR 04-02, Vakgroep Informatica, Technical University of Eindhoven, The Netherlands, 2004.

[BAri82] M. Ben-Ari. *Principles of Concurrent Programming*. Prentice-Hall International, 1982.

[BC94] B. Brandin and F. Charbonnier. The supervisory control of the automated manufacturing system of the AIP. In *Proc. Fourth International Conf. on Computer Integrated Manufacturing and Automation Technology*, pages 319–324, Troy, NY, USA, 1994.

[BM91] G. Basile and G. Marro. *Controlled and Conditioned Invariants in Linear System Theory*. U. of Bologna, Italy, 1991.

[BN80] J. Beauquier and M. Nivat. Application of formal language theory to problems of security and synchronization. *In: R.V. Book (Ed.), Formal Language Theory - Perspective and Open Problems, Academic Press*, pages 407–454, 1980.

[Bry86] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, C-35(8):677–691, 1986.

[BW94] B. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. Automatic Control*, 39(2):329–342, 1994.

[CDFV88] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Automatic Control*, 33(3):249–260, 1988.

[CL00] Y.-L. Chen and F. Lin. Modeling of discrete event systems using finite state machines with parameters. In *Proc. 2000 IEEE Intnl. Conf. on Control Applications*, pages 941–946, Anchorage, AL, USA, 2000.

[CL13] Y. Chen and Z. Li. *Optimal Supervisory Control of Automated Manufacturing Systems*. CRC Press, 2013.

[CW10a] K. Cai and W. M. Wonham. Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Trans. Automatic Control*, 55(3):605–618, 2010.

[CW10b] K. Cai and W. M. Wonham. Supervisor localization for large discrete-event systems – case study Production Cell. *Intnl. J. Advanced Manufacturing Technology*, 50(9-12):1189–1202, 2010.

[CZW15a] K. Cai, R. Zhang, and W. M. Wonham. On relative coobservability of discrete-event systems. In *Proc. American Control Conf.*, pages 371–376, Chicago, IL, USA, 2015.

[CZW15b] K. Cai, R. Zhang, and W. M. Wonham. Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Trans. Automatic Control*, 60(3):659–670, 2015.

[Dij65] E.W. Dijkstra. *Cooperating Sequential Processes*. Technological University, Eindhoven, The Netherlands, 1965.

[DL14] J. Dai and H. Lin. A learning-based synthesis approach to decentralized supervisory control of discrete event systems with unknown plants. *Control Theory and Technology*, 12(3):218–233, 2014.

[EVW80] A. Ephremides, P. Varaiya, and J. Walrand. A simple dynamic routing problem. *IEEE Trans. Automatic Control*, 25(4):690–693, 1980.

[FCW09] L. Feng, K. Cai, and W. M. Wonham. A structural approach to the nonblocking supervisory control of discrete-event systems. *Intnl. J. Advanced Manufacturing Technology*, 41(11):1152–1167, 2009.

[Fis78] G.S. Fishman. *Principles of Discrete Event Simulation*. Wiley, 1978.

[FW08] L. Feng and W. M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Trans. Automatic Control*, 53(6):1449–1461, 2008.

[Gol89] T. Gold. New ideas in science. *J. of Scientific Exploration*, 3(2):103–112, 1989.

[Gun97] J. Gunnarson. *On Modeling of Discrete Event Dynamic Systems, Using Symbolic Algebraic Methods*. Thesis No. 502, Division of Automatic Control, Lund University, 1997.

[Har87] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[HFL01] A. Hellgren, M. Fabian, and B. Lennartson. Modular implementation of discrete event

systems as sequential function charts applied to an assembly cell. In *Proc. 2001 IEEE International Conf. on Control Applications*, pages 453–458, Mexico City, Mexico, 2001.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[How60] R.A. Howard. *Dynamic Programming and Markov Processes*. The M.I.T. Press, 1960.

[Kro87] B.H. Krogh. Controlled Petri nets and maximally permissive feedback logic. In *Proc. 25th Annual Allerton Conf. on Communication, Control, and Computing*, pages 317–326, Allerton, IL, USA, 1987.

[Lin14] F. Lin. Control of networked discrete event systems: dealing with communication delays and losses. *SIAM J. Control and Optimization*, 52(2):1276–1298, 2014.

[LW88] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(2):173–198, 1988.

[LW94] Y. Li and W. M. Wonham. Control of vector discrete-event systems, I-the base model; II-controller synthesis. *IEEE Trans. on Automatic Control*, 38(8), 1214-1227, 1993; 39(3), 512-531, 1994.

[LW95] R.J. Leduc and W. M. Wonham. Discrete event systems modeling and control of a manufacturing testbed. In *Proc. 1995 Canadian Conf. on Electrical and Computer Engineering*, pages 793–796, Montreal, QC, Canada, 1995.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[MW05] C. Ma and W. M. Wonham. *Nonblocking Supervisory Control of State Tree Structures*. Lecture Notes in Control and Information Sciences (LNCIS) No. 317, Springer, 2005.

[OW85] J.S. Ostroff and W. M. Wonham. A temporal logic approach to real time control. In *Proc. 24th IEEE Conf. on Decision and Control*, pages 656–657, New York, NY, USA, 1985.

[Pet81] J.L. Peterson. *Petri Net Theory and Modeling of Systems*. Prentice-Hall, 1981.

[Ram89] P. J. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Trans. on Automatic Control*, 34(1):10–19, 1989.

[RLL03] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. *IEEE Trans. Automatic Control*, 48(6):957–975, 2003.

[RudW92] K. Rudie and W. M. Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Trans. Automatic Control*, 37(11):1692–1708, 1992.

[RudWil95] K. Rudie and J.C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Automatic Control*, 40(7):1313–1319, 1995.

[RW82] P. J. Ramadge and W. M. Wonham. Supervisory control of discrete event processes. *Joint Workshop on Feedback & Synthesis of Linear & Nonlinear Systems. Istituto di Automatica, Univ. di Roma, June 1981.* In *D. Hinrichsen, A. Isidori (Eds.), Feedback Control of Linear and Nonlinear Systems. Lecture Notes on Control and Information Sciences (LNCIS), No. 39, Springer-Verlag*, pages 202–214, 1982.

[RW87a] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.

[RW87b] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 25(5):1202–1218, 1987.

[SAF07] M. Skoldstam, K. Akesson, and M. Fabian. Modeling of discrete event systems using finite automata with variables. In *Proc. 46th IEEE Conf. Decision and Control*, pages 3387–3392, New Orleans, LA, 2007.

[SB11] K. Schmidt and C. Breindl. Maximally permissive hierarchical control of decentralized discrete event systems. *IEEE Trans. Automatic Control*, 56(4):723–737, 2011.

[Sei06] M. Seidl. Systematic controller design to drive high-load call centers. *IEEE Trans. Control Systems Technology*, 14(2):216–223, 2006.

[Shi79] M.W. Shields. Cosy train journeys. Technical report, Rpt. ASM/67, Computing Laboratory, Univ. of Newcastle-upon-Tyne, 1979.

[SR16] D. Sears and K. Rudie. Minimal sensor activation and minimal communication in discrete-event systems. *Discrete Event Dynamic Systems*, 26(2):295–349, 2016.

[SW04] R. Su and W. M. Wonham. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):31–53, 2004.

[TPS+13] R.J.M. Theunissen, M. Petreczky, R.R.H. Schiffelers, D.A. van Beek, and J.E. Rooda. Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner. *IEEE Trans. Automation Science and Engineering*, 11(1):20–32, 2013.

[Tri04] S. Tripakis. Decentralized control of discrete-event systems with bounded or unbounded delay communication. *IEEE Trans. Automatic Control*, 49(9):1489–1501, 2004.

[Tsi89] J. N. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems*, 2(2):95–107, 1989.

[TT07] D. Thorsley and D. Teneketzis. Active acquisition of information for diagnosis and supervisory control of discrete event systems. *Discrete Event Dynamic Systems*, 17(4):531–583, 2007.

[TW94] J.G. Thistle and W. M. Wonham. Control of infinite behavior of finite automata. *SIAM J. Control and Optimization*, 32(4):1075–1097, 1994.

[WLL08] W. Wang, S. Lafortune, and F. Lin. On the minimization of communication in networked systems with a central station. *Discrete Event Dynamic Systems*, 18:415–443, 2008.

[Won85] W. M. Wonham. *Linear Multivariable Control: a Geometric Approach*. 3rd ed., Springer, 1985.

[WW96] K. C. Wong and W. M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, 6(3):241–273, 1996.

[YG05] Y. Yang and P. Gohari. Embedded supervisory control of discrete-event systems. In *Proc. 2005 IEEE Intnl. Conf. on Automation and Engineering*, pages 410–415, 2005.

[YL02] T. S. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems*, 12(3):335–377, 2002.

[ZCG+16] R. Zhang, K. Cai, Y. Gan, Z. Wang, and W. M. Wonham. Distributed supervisory control of discrete-event systems with communication delay. *Discrete Event Dynamic Systems*, 26(2):263–293, 2016.

[ZW90] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. Automatic Control*, 35(10):1125–1134, 1990.