Sensor Activation and Communication Problems in Discrete-Event Systems

Technical Report 2013-601

David Sears[*]and Karen Rudie[†]

January 23, 2013

Abstract

This paper is an overview of the current research on problems involving dynamic sensor activation and communication in discrete-event systems. In problems of sensor activation agents observing a discrete-event system can turn their sensors responsible for detecting observable event occurrences on / off dynamically. In problems of communication agents communicate information such as observed event occurrences, state or string estimates of a discrete-event system to each other when necessary. Sensor activation and communication have been investigated in problems of control, diagnosis and opacity of discrete-event systems. This paper attempts to provide a coherent categorization and presentation for works on these problems. Their strengths, weaknesses and avenues for future research are highlighted.

1 Introduction

A Discrete-Event System (DES) is a system with a (usually) discrete state space and event-driven dynamics. Events occur sequentially and asynchronously in such systems. As an example, an elevator can be modeled as a DES. The events would be the opening and closing of doors, the pressing of buttons, the arrival at a floor, etc. The states of the system would be defined by information such as the current floor of the elevator, the set of floors in which the elevator is to travel, the direction it is moving, etc. As events occur sequentially in the model, the simultaneous pressing of two buttons is not modeled (but, in reality, the low-level event processing unit ensures the sequencing of events). As events occur asynchronously, the time between two event occurrences is not captured in the model. Even though an elevator is a much more complicated system, in reality, the electronic control of elevators uses a model similar to the one described. Other examples include computer and

[*]E-Mail: sears@cs.queensu.ca; Address: School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

[†]E-Mail: karen.rudie@queensu.ca; Address: Department of Electrical & Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada

communication networks, automated manufacturing systems, air traffic control systems, advanced monitoring and control systems in automobiles and distributed software systems.

The monitoring or controlling of discrete-event systems by agents external to the system is of potential practical interest. Left uncontrolled, such systems may execute in a manner that results in undesirable behaviour given some specification of desirable behaviour. The control of DES was initiated by Ramadge and Wonham in their influential works [61, 62, 111] where agents referred to as *supervisors* observe the events of a DES and, depending on the sequence of events that they observe, disable certain events from occurring in the system to satisfy a specification of desirable behaviour.

When the control of a DES by agents is overly-restrictive or cannot be implemented it may instead suffice to monitor the operation of the DES to determine when faults or violations of desirable behaviour occur. The monitoring and diagnosing of faults in DES by external agents was initially considered in [42] and [75].

Also, recently a new topic of opacity which is related to studies in computer security has been gaining attention in the DES community. Opacity is used to study the effectiveness of a system to keep secret a predicate of the system from a (potentially malicious) external observer. A predicate is opaque if an observer of the system will never be able to determine the truth of that predicate.

In problems of control, diagnosis and opacity agents may not have complete observation of the system. Some events generable by the system may be observed by an agent while others may not. In such cases we say that the agent's observation of the DES is *static*. To further complicate matters, individual occurrences of a given observable event may be observed while others may not. In such cases we say that the agent's observation of the DES is *dynamic*. When sensor activation and communication is concerned typically an agent's observation is dynamic but in some cases it is static. We consider both cases in this report.

In problems of sensor activation an agent has associated with it sensors responsible for detecting the occurrence of events observable to the agent. Furthermore, the agent is capable of activating / deactivating these sensors following the observation of individual event occurrences. When the sensor for observable event e is active (respectively, inactive) any occurrence of e will (resp., will not) be observed by the agent. The criteria for determining when an event sensor should be activated (resp., deactivated) is dependent on the problem at hand and an agent's state or string estimate of a DES. When sensor activation is concerned, usually cost is associated with the activation of sensors or with active sensors and the problems considered require that sensor activation maps be computed which minimize this cost. Studies in minimizing the activation of sensors and duration of active sensors are motivated by increasing the life span and availability of sensors, for conserving power when only limited battery power is available or for security purposes.

In problems of communication in decentralized and distributed DES, information about the DES required for satisfying a given objective (e.g., control or diagnosis of DES) is distributed among agents and typically agents individually do not have enough information for satisfying the given objective. Then communication of some of the distributed information amongst the agents is required. Typically the information communicated consists of observed event occurrences or estimates of the state of the DES or of the string generated by the DES. Communicated messages may or may not incur delay. Also the topology of the communication network may be variable. In all problems considered here communicated messages are not corrupted or dropped by the channels that they travel through and message delivery is in-order. Similar to problems of sensor activation, problems of communication usually require that communication among agents be minimized in some way. The motivation for this is also similar to that for sensor activation problems.

We proceed in Section 3 by first considering some early works on communication for purposes of control in DES. These works include [88, 109, 98, 2]. In [88] the fundamental questions whose answers affect the modeling of problems involving communication are highlighted. In [109, 98, 2] various problems of minimal communication are proposed. In [2] an approach is provided for solving one of these problems. It is also demonstrated in [2] that, interestingly, for a specific class of control problems and communication maps, the set of control problems solvable by agents which maintain and communicate state estimates and anticipate future communications strictly contains the set of control problems solvable by agents which maintain and communicate string estimates. These works and other early works (e.g., [69, 65]) also identify that in order that agents be able to communicate to solve control objectives the condition of observability ([46]) of a specification language needs to be satisfied with respect to the DES and all collective controllable and observable events. This condition is used in most subsequent works on communication to guarantee existence of communication solutions.

Subsequently, we review the decidability of some control problems involving communication where communication does (resp., does not) incur delay in Section 4. The works covered include [94, 26, 96]. In [94, 26] the authors prove that two different control problems where agents communicate all observed event occurrences to one another with unbounded delay are undecidable. When bounded delay communication is considered the resulting problems become decidable. Furthermore, in [94] for bounded delay communication problems the author proves that an infinite hierarchy of strict containments of nonblocking distributed control problems is induced as the maximum delay that a message incurs in a communication channel decreases. The authors also prove that if a control solution is nonblocking when communication incurs a delay of up to $k + 1$ event occurrences then the control solution is also nonblocking when communication incurs a delay of up to k event occurrences. Also, another case in [26] which is proven to be decidable is when any cycle in G contains an event observable by all agents. In [96] the author explores the relationship between two decentralized observation conditions: unbounded-memory joint observability (JO) and finite-memory joint observability (FJO). In both cases decision of membership in a given regular specification language depends on the events observed by all agents collectively. As a result, an implementation would require communication among agents or that each agent send their observations to a coordinator for determining membership in the regular specification language. The authors find that JO is strictly weaker than FJO. Under certain conditions FJO is proven to be equivalent to JO. Decidability and undecidability results are proven for the verification of JO.

In Section 5 we review work conducted on communication between agents for refining state and string estimates of distributed DES. The works covered include [83, 82, 57, 112, 31]. In [83] the authors present a distributed protocol where agents iteratively exchange projections of their string estimates for refining the string estimate of the components that they monitor in a modular DES which can only execute strings of bounded length. This work is continued in [82] where algorithms are provided for computing string estimates which satisfy conditions called local consistency and global consistency. Here the DES and initial string estimates of agents may be of arbitrary expressivity, however termination when computing estimates that satisfy local consistency is not guaranteed in some cases. In [57] a complex distributed protocol for the exchange among agents of state-transition estimates of a given DES is provided where communication incurs delay. A distributed protocol where observed event occurrences are exchanged with delay is provided in [112] for refining state estimates of a given DES. This protocol does not suffer the same shortcomings as the protocol of [57]. Lastly, [31] considers the task where agents estimate the states of a set of interacting processes modeled by communicating finite state machines where state estimates and vector clocks are communicated with arbitrary delay between agents. The authors present a method using abstract interpretation techniques for computing state estimates.

In Section 6 we review communication in diagnosis and prognosis problems of DES. The works covered include [5, 18, 78, 59, 58, 87]. A diagnosis problem involving two agents is considered in [5] where one agent approximates the state estimate of the other agent and communicates its own state estimate to the other agent under some circumstances to ensure that occurrences of a failure event can be diagnosed by the other agent. Diagnosis of faults by a coordinator is considered in [18]. In this work two agents maintain state estimates of the DES and send their state estimates (along with some auxiliary information for synchronization of state estimates) to the centralized coordinator following observed event occurrences. However, the coordinator observes a totally ordered sequence of message receptions and so cannot determine the exact order in which messages were sent if messages were timestamped with a global clock. Consequently, the authors consider messages received from a specific agent are received in order but may be one step out of order with respect to messages received from the other agent. The authors present an approach for computing state estimates at the coordinator from which diagnoses can be performed and characterize when this procedure satisfies diagnosability. In [78] the authors present "decentralized diagnosability", a type of joint diagnosability condition, and prove its verification to be undecidable for some general cases. A stronger formulation of joint diagnosability, called joint$_\infty$-diagnosability, is considered in [59] which allows for communication to incur unbounded delay. Verification of this stronger formulation is shown to be decidable. Motivated by this result, a joint diagnosability condition where communication incurs a bounded delay of up to $k \geq 0$, called joint$_k$-diagnosability, is considered in [58]. The authors provide an algorithm for verifying this condition, describe how diagnoser automata may be synthesized when the condition holds, and demonstrate an infinite hierarchy of joint diagnosability conditions where joint$_\infty$-diagnosability is the weakest and joint$_0$-diagnosability the strongest. Analogously, a problem of prognosing failure occurrences some number of events before their occurrence when communication of observed event occurrences between agents is bounded is considered in [87]. The authors introduce the condition of joint^k-prognosability which they prove to be necessary and sufficient for solving a distributed failure diagnosis problem where all failures are prognosed before their occurrence and no incorrect prognoses (i.e., false alarms) are made. An algorithm is presented for verifying joint^k-prognosability.

The control of DES where the decision to communicate is made on the knowledge of an agent and inferences made from it will be reviewed in Section 7. The works covered include [66, 67, 35, 23, 33, 52]. In [66] two-agent decentralized DES control problems are cast in the knowledge theory of [25]. Here the authors formulate a knowledge-based condition called *Kripke-observability*, which is analogous to controllability and coobservability. In [67] the authors present a generalization of Kripke-observability called *inference-observability* which, when satisfied, allows conditional disablement and "do not know" actions to be used effectively for control purposes. When inference-observability is satisfied and such agent control actions are employed, communication of individual agent control decisions is required to compute final control decisions. An impressive work which subsumes [67] is [35]. In [67] only one level of inferencing over the knowledge of other agents to determine individual control decisions is conducted. However, in this work agents perform multiple levels of inferencing over the knowledge of other agents. The authors introduce N-inference-observability of a specification language which generalizes inference-observability. This condition is necessary and sufficient to be able to synthesize controllers which conduct up to N levels of inferencing and whose control on the system exactly equals a given specification language. An algorithm for verifying this condition is provided. A problem where the execution of transitions of a DES automaton must satisfy a safety specification and priority specification are considered in [23]. Here transitions of the system are controlled by agents in a disjunctive-antipermissive manner. Agents have a partial view of the state of the system and use this to determine whether or not a safe, maximal priority transition whose occurrence it controls can be executed or if some other agent knows that a safe, maximal priority transition that it control can be executed. A model-checking algorithm is described for determining when these knowledge propositions hold and, when neither proposition holds, for determining with which other agents an agent needs to exchange its partial view for refining its state estimate of the system until one of the propositions holds. This work is continued in [33] where instead a 1-safe Petri Net is used to represent the DES and only a safety specification is considered which is defined over state-transition pairs of the net's reachability graph. Associated with each controller is a set of transitions of the net which they both observe and control. Controllers observe those input and output places associated with transitions that it observes. Given this partial view of the state / marking of the net, knowledge properties similar to those considered in [23] are used for dictating the actions of a controller. When the controller does not know what action to take, it can coordinate with other controllers given in a prespecified set. The authors present two techniques for reducing the number of controllers involved in a coordination for refining state estimates of the DES. This line of work is continued in [52] where uncontrollable events are considered and controllers maintain a history of observed transition executions (i.e., knowledge of perfect recall) in the net which is used to refine their state estimates and, correspondingly, knowledge of the DES.

In Section 8 we briefly review approaches to synthesize controllers and a communication protocol between them from and which realizes the control implemented by a centralized, monolithic controller. Such approaches are useful in distributed settings where a monolithic controller cannot be feasibly implemented. The works covered include [47, 15, 16]. The novelty of these works is that the nature of the information to be communicated between the synthesized agents does not have to be specified beforehand. The information to be communicated is an output of the synthesis procedures.

In Section 9 we consider problems of minimal and minimum communication, which arise out of practical interest of minimizing bandwidth usage and for security purposes. The works covered include [70, 45, 104, 103, 65, 41, 64, 73, 74]. In [70] a two-agent communication problem is considered. An automaton is associated with each agent. Transitions in an agent's automaton may be defined over events not observable by the agent but observable by the other agent. The problem considered is to compute communication maps for each agent from strings generated in the synchronous product automaton to locally observed events which: (1) yield observations for each agent that allow them to unambiguously determine the state reached in their respective automata by any string generated in the synchronous product automaton; (2) can be defined over the transitions of the synchronous product of the agent automata; (3) allow each agent to unambiguously determine what should be communicated by it and when; (4) is minimal with respect to the previous three conditions. The authors propose an algorithm for computing such communication maps which requires few iterations between updating one agent's communication map and the other agent's communication map. The problem considered is generalized in [45] where, instead of requiring each agent to precisely determine the current state of its automaton representation, it is required that certain transitions in the agents' automata representations be observed. An exhaustive algorithm is provided which computes all sets of minimal communication maps for the two agents which solve this generalized problem. The work of [104] considers a problem of state disambiguation where an arbitrary number of agents which communicate with a central station and an automaton representation of the DES which is absent of cycles besides self-loops on states is considered. Individual agents are required to distinguish certain pairs of states in the DES automaton representation while the central station must always know the exact state of the DES automaton representation. Communication maps are defined over the transitions of DES automaton and must satisfy similar requirements as those considered in [70] and [45]. The authors provide an algorithm polynomial in all of its inputs for computing minimal communication maps for each agent and the central station which satisfy these requirements. This work is generalized in [103] to strongly connected communication topologies that do not require the presence of a central station. A general algorithm is provided for computing minimal communication maps for each agent. In [65] a two-agent control problem is considered where agents may communicate their state estimates of an automaton representation of a DES to each other from states in the representation to resolve control conflicts. The authors consider that communication occur as early as possible. An algorithm is provided which takes as input a set of states from which communication occurs from one agent to the other and computes a minimal subset of these states where communication allows for the other agent's control conflicts to be resolved and where the communicating agent unambiguously knows what to communicate based on its observation of the DES. In [41] the control of reactive DES are considered with and without communication. It is required that interactions between the DES and an unpredictable environment exactly satisfy a given regular language specification. Controllers are associated with components of the DES to achieve this. The authors introduce necessary and sufficient conditions for solving the control problem analogous to controllability and coobservability. An algorithm is provided for computing minimal communication maps where communication of event occurrences occurs as early as possible between agents. In [64] a method for computing communication maps between agents which resolve violations of coobservability and which minimizes communication for strings of infinite length is presented. This is in contrast to previous works on minimal communication which consider a different notion of minimality and where communication maps considered are based on structural properties of the DES automaton model. In [73] a previously published algorithm for computing a Nash equilibrium is applied for computing communication maps for a set of controllers which have minimum cost, satisfy coobservability and where agent communication decisions are unambiguous following any string generable by the DES. In [74] a problem of computing minimum-cost control and communication maps which generate behaviour in a given regular specification where cost is associated with event enablement, disablement and communication is considered. The authors describe the application of an evolutionary algorithm for computing minimal cost solutions.

In Section 10 we study important conditions necessary for solving control and diagnosis problems when an agent's observation of an event is more specifically based on the transitions between states of a given DES. The works covered include [29, 100, 51]. In [29] the authors introduce and study a transition-based version of coobservability. An algorithm is described for its verification given observation maps for two agents defined over the transitions of the automaton representation of a DES. Also, an algorithm is described for computing additional transitions to be observed to satisfy transition-based coobservability for a given input set of transitions observed and events controllable by each controller. In [100] a reduction from transition-based coobservability to transition-based codiagnosability is proven. An algorithm for verifying transition-based codiagnosability is provided which can also be applied for verifying transition-based coobservability for an arbitrary number of agents. In [51] a reduction from transition-based coobservability to an observational condition called decentralized no-opacity is provided.

In Section 11 we study problems of minimum and minimal sensor activation in problems of control and diagnosis. The works covered include [92, 12, 106, 102, 101, 105, 30, 14, 79]. In [92] the authors present dynamic programming algorithms for computing minimum-cost sensor activation maps for a single controller to satisfy observability. In [12] several contributions are provided for computing and verifying sensor activation maps which satisfy diagnosability. An algorithm for computing all sensor activation maps which allow a centralized diagnoser to diagnose faults following k event occurrences in a DES is provided. Furthermore, an algorithm is provided for computing a minimum-cost sensor activation map when cost is associated with active event sensors. An algorithm for verifying if a given sensor activation map satisfies k-diagnosability is also provided. In [106] the authors consider the computation of minimal sensor activation maps which can be used to disambiguate user-specified pairs of states of a DES automaton representation. A polynomial-time algorithm is provided for this purpose. This algorithm is also applied in the computation of minimal sensor activation maps for a set of agents to satisfy transition-based coobservability. In [102] an algorithm is provided for computing minimal sensor activation maps for satisfying diagnosability and codiagnosability using partitions of the DES language instead of a state-transition representation. In [101] the computation of sensor activation maps for satisfying diagnosability of a DES which are minimal with respect to the DES language, not merely with respect to any specific representation of the DES (e.g., state-transition structure or language partitions), is considered. In [105] an approach to computing sensor activation decisions online following any event observation using one-step lookahead is provided. The resulting sensor activation map allows user-specified pairs of states of an automaton representation of a DES to be disambiguated following any string generated by the DES and furthermore is a minimal sensor activation map. In [30] the authors propose an online algorithm for computing sensor activation decisions following $k \geq 1$ event occurrences, which generalizes the work of [105]. In [14] an alternative algorithm is provided for computing the most permissive observer automaton of [12] which is used for defining all sensor activation maps which satisfy k-diagnosability of a given DES automaton representation. This algorithm is more efficient by applying an optimization for reducing the state set of the most permissive observer as it is constructed. In [79] the authors study detectability of DES with dynamic observation maps. They present an algorithm for computing a minimal sensor activation policy for satisfying one of their detectability conditions. The approach used is similar to that used in [101].

In Section 12 we study problems of opacity where an agent's observation is dynamic. The works covered include [8, 10, 43, 3, 4, 51]. In [8] opacity of DES modeled by labelled transition systems is studied. The authors demonstrate that verification of security properties such as anonymity and non-interference is reducible to verifying various forms of opacity. They also prove that verifying opacity for labelled transition system and even Petri Net representations of DES is undecidable. In [10] the authors consider the use of dynamic observers introduced in [12] rather than intrusive means of control for enforcing opacity. They prove that verifying opacity of a given secret for a given dynamic observer and automaton representation of a DES is PSPACE-complete. They then present an approach for computing all dynamic observers which satisfy opacity of a given secret and for computing a minimum cost dynamic observer in this set. In [43] the author introduces strong and weak opacity of a given secret regular specification language with respect to another non-secret regular language, both of which are contained in the language recognized by an automaton representation of a DES. Observation maps are considered to be defined over the transitions of the DES automaton representation. The author introduces the notions of strong, weak and no-opacity of the given secret with respect to another non-secret language. Algorithms are provided for their verification. The author demonstrates that anonymity and secrecy, two properties in security and privacy, can be studied as special cases of opacity. The modification of secret and non-secret languages for satisfying strong (resp., weak, no)-opacity is studied in [3]. The authors prove a number of closure properties for the various opacity conditions and demonstrate how super-languages and sub-languages of secret and non-secret languages may be computed which satisfy opacity conditions. In [4] the authors provide algorithms for computation of the supremal controllable, normal sublanguage of a regular language modeling a DES for satisfying strong (resp., no) opacity when the controller's observation of the DES is static (i.e., given using a natural projection) and the external observer's observation of the DES is string-based. Problems of decentralized opacity are investigated in [51]. Decentralized variants of strong (resp., weak, no)-opacity are introduced where each agent views the DES through its own string-based observation map and no coordination is involved amongst the agents. Variants where coordination is incorporated among agents are also introduced. The authors prove that for a specific setting where agents can coordinate, verifying if a secret is strongly (resp., weakly, not)-coopaque (can be discovered by the coordinating agents) is reducible to verifying if the secret is strongly (resp., weakly, not)-opaque with respect to a specific observation map.

We next provide a list of features distinguishing works in this report. Concepts and notation necessary for reading the body of the paper are provided in Section 2. Following the survey we conclude in Section 13 with some perspective on the research field, general avenues for future research and open problems concerning specific works surveyed.

The following is a list of the key features that distinguish one work from another in this survey:

- The model used to represent the system. Typically we will consider a regular language / automaton representation but in some instances we consider Petri Nets, labelled transition systems or a formal language representation of greater expressivity.

- The particular problem to be solved. We consider problems of control, diagnosis, opacity (a recent notion being explored in software security) and disambiguation of certain pairs of states in state-based models of systems.

- The models used to represent a specification of desirable behaviour (for problems of control), the occurrence of faults / failures (for problems of diagnosis) and a secret to be protected from malicious observers / attackers in a system (for problems of opacity). For example, typically we will consider a regular language / automaton representations for these entities.

- For problems of control, how individual agent control decisions are computed and how final control decisions are computed from them.

- How an agent's observation of a system is defined. For instance, an agent's observations may be defined over event occurrences labelling the transitions of state-transition representations of systems.

- The information provided to and maintained by each agent about the system and other agents. For example, an agent typically requires the model of the system and the set of events that it (or other agents) observes and controls. The information it typically maintains is an estimate of the state of the system or an estimate of the sequence of events that the system could have generated accounting for the observations that the agent has made.

- For problems involving communication between agents:

  - The number of communicating agents.

  - The type of information communicated (e.g., occurrence of system events, estimates of system state).
  - The constraints dictating which agents can interact with each other (i.e., communication topology).
  - Whether or not communication is point-to-point or broadcast.
  - Whether or not communicated messages incur delay.

- For problems of optimal observation, the definition of optimality considered.
- The computational complexity of the problems considered as well as the proposed algorithms for solving them.

2 Preliminaries

Here we attempt to cover the basic requirements for understanding the content of this report. We refer the reader to the text [9] and monograph [110] for a more comprehensive and detailed coverage of concepts used in DES.

2.1 Formal Languages & Automata

Given a set, S, we denote the cardinality of S by $|S|$. An *alphabet* is a finite set of distinct symbols (also called *events*, for our purposes). Let Σ represent an alphabet. Let Σ^+ denote the set of all finite sequences of symbols in Σ of the form $\sigma_1\sigma_2\ldots\sigma_k$ where $k \geq 1$ is arbitrary and $\sigma_i \in \Sigma$ for all $i \in \{1,\ldots,k\}$. We refer to a sequence of length zero (i.e., consisting of no symbols) as the empty sequence, denoted by $\varepsilon \notin \Sigma$. The *Kleene-closure* of Σ, denoted by Σ^*, is defined as $\Sigma^* = \{\varepsilon\} \cup \Sigma^+$. An element of Σ^* is a *string* over the alphabet Σ. We also refer to ε as the *empty string*.

For $s \in \Sigma^*$ we say $t \in \Sigma^*$ is a *prefix* of s, denoted by $t \leq s$, if $s = tu$ for some $u \in \Sigma^*$. Thus $\varepsilon \leq s$ and $s \leq s$ for all $s \in \Sigma^*$. The set of all prefixes of string s is denoted by $\overline{s}$. If $L \subseteq \Sigma^*$ then the *prefix-closure* of L is the language $\overline{L}$ consisting of all prefixes of strings of L: $\overline{L} = \{t \in \Sigma^* \mid t \leq s \text{ for some } s \in L\}$. Clearly $L \subseteq \overline{L}$. A language L is *prefix-closed* if $L = \overline{L}$.

A *nondeterministic finite automaton* (NFA), A, is defined as a tuple $A = (Q, \Sigma, \delta, q_0, Q_m)$ where Q is the nonempty set of states, Σ is the alphabet, q_0 is the initial state, $Q_m \subseteq Q$ the set of accepting / *m*arked states and $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is the (state-)transition function, a partial function which determines the transitions from state to state on occurrence of symbols in $\Sigma \cup \{\varepsilon\}$. We extend δ to a function $\delta : Q \times \Sigma^* \to 2^Q$ by induction on length of strings. The alphabet of A, Σ, is also denoted by $alph(A)$. Automaton A is said to be a *deterministic finite automaton* (DFA) if δ is more specifically of the form $\delta : Q \times \Sigma \to Q$. We say that A is *live* if there is a transition defined from every state in A. We associate two languages with A: $\mathscr{L}_m(A) = \{s \in \Sigma^* \mid \delta(q_0, s) \cap Q_m \neq \varnothing\}$; $\mathscr{L}(A) = \{s \in \Sigma^* \mid \delta(q_0, s) \text{ is defined}\}$. Given A, a state-transition graph can be constructed ([27]). The language $\mathscr{L}_m(A)$ (resp., $\mathscr{L}(A)$) consists of the set of strings recognized / marked (resp., generated) by A. Automaton A is said to be a recognizer for $\mathscr{L}_m(A)$. When A

models a DES, $\mathscr{L}_m(A)$ (resp., $\mathscr{L}(A)$) is intended to represent completed behaviour (resp., uncompleted behaviour) of the DES.

When A is an NFA an algorithm known as the *subset construction* can be used for constructing a DFA $DET(A)$ where $\mathscr{L}_m(DET(A)) = \mathscr{L}_m(A)$ and $\mathscr{L}(DET(A)) = \mathscr{L}(A)$ ([27]). The state space of $DET(A)$ is a subset of the power set of the state space of A. A state in $DET(A)$ is an accepting / marked state if the subset of Q it represents contains a state in Q_m. For space considerations, we do not outline the operation of the subset construction and instead refer the reader to [27].

Given two DFAs, A_1, A_2 with transition functions δ_1, δ_2 where $\mathscr{L}(A_1) \subseteq \mathscr{L}(A_2)$, we say that A_1 is a *subautomaton* of A_2, denoted by $A_1 \sqsubseteq A_2$, if the state-transition graph of δ_1 is isomorphic to a subgraph of the state-transition graph of δ_2 from the initial states of A_1, A_2.

Given two DFAs, $A_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1}, Q_{m,1})$, $A_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2}, Q_{m,2})$, the synchronous product of A_1 and A_2, denoted by DFA $A_1 || A_2$, recognizes the shuffle of strings recognized by A_1 or A_2 where synchronization occurs on the occurrence of events in $\Sigma_1 \cap \Sigma_2$. Specifically, it is defined as $A_1 || A_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2})$ where $\delta : (Q_1 \times Q_2) \times (\Sigma_1 \cup \Sigma_2) \to (Q_1 \times Q_2)$ is defined as follows:

$$\delta((q_1, q_2), \sigma) \begin{cases} (\delta_1(q_1, \sigma), q_2) & \text{if } \sigma \in \Sigma_1 \setminus \Sigma_2 \text{ and } \delta_1(q_1, \sigma) \text{ is defined} \\ (q_1, \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_2 \setminus \Sigma_1 \text{ and } \delta_2(q_2, \sigma) \text{ is defined} \\ (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and both } \delta_1(q_1, \sigma) \text{ and} \\ & \delta_2(q_2, \sigma) \text{ are defined} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Unless otherwise specified, we assume that the plant / system under consideration in work to be surveyed is modeled by DFA $G = (X, \Sigma, \delta, x_0, X_m)$. In some cases a Petri Net (PN) representation of the plant is assumed. For an introductory reference to the PN model we refer the reader to [53]. Knowledge of the basic operation of a PN in terms of firing of transitions and execution semantics / reachability graph of PNs is required. We refer to *1-safe* Petri Nets as the class of Petri Nets where the number of tokens in any place for any marking of the net's reachability graph is at most one.

2.2 Control & Observation of DES

We regard the plant, G, as a generator of events in its alphabet Σ. For control problems, we wish to model controllers / supervisors which interact with G in a feedback manner. Controllers control (enable / disable) the execution of events in Σ which are generated / executed by G. While some events in Σ can be controlled, others cannot. This induces a partition on Σ into the set of events controllable by controllers, $\Sigma_c \subseteq \Sigma$, and otherwise uncontrollable events $\Sigma_{uc} \subseteq \Sigma$ (i.e., $\Sigma_{uc} = \Sigma \setminus \Sigma_c$). Formally, a controller, S, is specified as a function from $\mathscr{L}(G)$ to the power set of Σ: $S : \mathscr{L}(G) \to 2^{\Sigma}$. For $s \in \mathscr{L}(G)$, $S(s) \supseteq \Sigma_{uc}$ represents the set of events whose generation / execution by G is enabled (i.e., not prevented / disabled) by controller S following s. Given G and a single controller S, the resulting closed-loop system is denoted by S/G (read as "S controlling G"). We can characterize the generated and marked languages of S/G, denoted by $\mathscr{L}(S/G)$ and $\mathscr{L}_m(S/G)$. The *language generated* by S/G is defined recursively as follows:

1. $\varepsilon \in \mathscr{L}(S/G)$
2. $[(s \in \mathscr{L}(S/G)) \wedge (s\sigma \in \mathscr{L}(G)) \wedge (\sigma \in S(s))] \Leftrightarrow s\sigma \in \mathscr{L}(S/G)$.

The *language marked* by S/G is defined as follows: $\mathscr{L}_m(S/G) = \mathscr{L}(S/G) \cap \mathscr{L}_m(G)$.

Closed-loop behaviour S/G is *blocking* if $\mathscr{L}(S/G) \neq \overline{\mathscr{L}_m(S/G)}$. Otherwise, S/G is *non-blocking*. Blocking occurs when there exists behaviour in $\mathscr{L}(S/G)$ from which completed behaviour in $\mathscr{L}_m(S/G)$ cannot be reached. Deadlock and livelock are instances of blocking behaviour.

Given two alphabets, Σ_1, Σ_2, the *(natural) projection* from Σ_1 to Σ_2 is a function which takes as input a string $s \in \Sigma_1^*$ and erases all symbols from s that are not in Σ_2. A controller may not be able to "see" / "observe" all events in Σ generated by G. This induces a partition of Σ into events observable to the controller, $\Sigma_o \subseteq \Sigma$, and events unobservable to the controller, $\Sigma_{uo} \subseteq \Sigma$ (i.e., $\Sigma_{uo} = \Sigma \setminus \Sigma_o$). When $\Sigma_{uo} \neq \varnothing$ the observation to S of strings generated by G is modeled by a natural projection $P : \Sigma^* \to \Sigma_o^*$. For $s \in \mathscr{L}(G)$, S observes $P(s)$. Note that for $s \in \Sigma_o^*$, $P^{-1}(s)$ denotes the set of all strings in Σ^* whose projection onto Σ_o equals s. It is required that, for $s_1, s_2 \in \mathscr{L}(G)$, if $P(s_1) = P(s_2)$ then S must make the same control decision following s_1 and s_2. To capture this fact, S is respecified: $S : P(\mathscr{L}(G)) \to 2^{\Sigma}$. Here the control decision of S can only change after the occurrence of events observed in Σ_o. We require that, for $s \in \mathscr{L}(G)$, $S(P(s)) \supseteq \Sigma_{uc}$. When $\Sigma_{uo} \neq \varnothing$, the languages $\mathscr{L}(S/G)$ and $\mathscr{L}_m(S/G)$ are defined analogous to the previous definition but where control decisions are based on $P(s)$ ($\sigma \in S(P(s))$ in the third clause of 2) rather than s.

The controller S is introduced because the behaviour of G is assumed to be unsatisfactory and must be restricted to within a sublanguage or range of sublanguages of $\mathscr{L}(G)$ / $\mathscr{L}_m(G)$. For the control problems surveyed, usually a single prefix-closed specification language $\mathscr{L}(K) \subseteq \mathscr{L}_m(G)$ is considered and it is required that we construct S which has partial observation of G such that $\mathscr{L}_m(S/G) = \mathscr{L}(K)$. If multiple controllers are considered then we require that each controller be constructed such that the marked closed-loop behaviour by the controllers acting together on G must equal $\mathscr{L}(K)$. When the closed-loop behaviour is equal to $\mathscr{L}(K)$ we say that the behaviour satisfies $\mathscr{L}(K)$. Unless otherwise specified, we assume that $\mathscr{L}(K)$ is a regular language and can be modeled by a DFA K.

In order that a controller(s) exist that can control G such that $\mathscr{L}(S/G) = \overline{\mathscr{L}(K)}$ a condition of *controllability* is required ([60]): $\overline{\mathscr{L}(K)}\Sigma_{uc} \cap \mathscr{L}(G) \subseteq \overline{\mathscr{L}(K)}$. Informally, controllability states that the occurrence of any uncontrollable event in G from a behaviour in $\overline{\mathscr{L}(K)}$ will not lead to behaviour in $\mathscr{L}(G)$ outside of $\overline{\mathscr{L}(K)}$. If controllability does not hold for nonempty Σ_{uc} then no controller(s) exists whose closed-loop behaviour with G satisfies $\mathscr{L}(K)$. If it is further required that $\mathscr{L}_m(S/G) = \mathscr{L}(K)$, that is, that closed-loop behaviour be nonblocking, then a condition of $\mathscr{L}_m(G)$-closure of $\mathscr{L}(K)$ needs to be satisfied: $\mathscr{L}(K) = \overline{\mathscr{L}(K)} \cap \mathscr{L}_m(G)$.

When a controller observes G through a natural projection P and Σ_{uo} is nonempty then, in addition to controllability, the observational condition that characterizes when closed-loop behaviour using the controller which bases its control decisions on $P(s)$ for $s \in \mathscr{L}(G)$ satisfies $\mathscr{L}(K)$ is *observability*. Informally, observability characterizes that if an event following a string s must be disabled then it should be disabled following all strings that appear the same / are indistinguishable with s according to the controller's observation function, P. Formally, $\mathscr{L}(K)$ is said to be *observable* with respect to $\mathscr{L}(G)$, Σ_o and Σ_c if for all $s \in \overline{\mathscr{L}(K)}$ and for all $\sigma \in \Sigma_c$,

$$s\sigma \notin \overline{\mathscr{L}(K)} \wedge s\sigma \in \mathscr{L}(G) \Rightarrow P^{-1}(P(s))\sigma \cap \overline{\mathscr{L}(K)} = \varnothing.$$

Given $\mathscr{L}(K)$, $\mathscr{L}(G)$, when Σ_{uc} and Σ_{uo} are both nonempty the necessary and sufficient conditions for the existence of a centralized controller S which controls events in Σ_c and observes events in Σ_o such that $\mathscr{L}(S/G) = \overline{\mathscr{L}(K)}$ is controllability and observability ([46]). Furthermore, if $\mathscr{L}_m(G)$ is nonempty and nonblockingness ($\mathscr{L}_m(S/G) = \mathscr{L}(K)$) is required then $\mathscr{L}(K)$ needs to be $\mathscr{L}_m(G)$-closed.

When we are dealing with $|I| > 1$ controllers, it is assumed that each controller $i \in I$ controls (resp., observes) events in a given set $\Sigma_{i,c}$ (resp., $\Sigma_{i,o}$). Each controller observes G using its natural projection $P : \Sigma^* \to \Sigma_{i,o}^*$. Given an event $\sigma \in \Sigma_c$, the set of controllers which control σ is denoted by $I_c(\sigma)$.

Whether an event occurrence of $\sigma \in \Sigma_c$ generable by G should be enabled or not depends on the individual control decisions of controllers in $I_c(\sigma)$ as well as how those individual control decisions are used to define the final control decision. There are two common methods by which individual control decisions are *fused* into a final control decision. One method is to enable σ if its occurrence is enabled by all controllers in $I_c(\sigma)$. This is known as the *conjunctive architecture* and is the most commonly studied method. The other common method is to enable σ if it is enabled by some controller in $I_c(\sigma)$. This is known as the *disjunctive architecture*.

If a controller $i \in I_c(\sigma)$ is in doubt as to whether or not to enable or disable σ (i observes string s and there exist two strings $s_1, s_2 \in \mathscr{L}(K)$ where $P_{\Sigma_{i,o}}(s_1) = P_{\Sigma_{i,o}}(s_2) = s$, $s_1\sigma \in \mathscr{L}(K)$, $s_2\sigma \in \mathscr{L}(G)$ and $s_2\sigma \notin \mathscr{L}(K)$; we refer to such situations as *control conflicts*) then it may choose to issue an enable or disable action (in some works "do not know" decisions are also considered). When an enable action is issued we say that the controller is being *permissive* with respect to its control conflicts. Otherwise, it is being *antipermissive* with respect to its control conflicts.

When the conjunctive architecture is used and controllers are permissive with respect to control conflicts we say that the architecture used is more specifically a conjunctive-permissive (CP) architecture. This is the most commonly studied combination of decision fusion architecture and individual decision policy for control conflicts. When the disjunctive architecture is used and controllers are antipermissive with respect to control conflicts we say that the architecture used is more specifically a disjunctive-antipermissive (DA) architecture. Unless otherwise stated, the control architecture used in most works to be surveyed is CP.

When the CP architecture is used, the observational condition that characterizes when closed-loop behaviour using the controllers satisfies $\mathscr{L}(K)$ is *CP-coobservability*, which is a natural generalization of observability: $\mathscr{L}(K)$ is said to be *CP-coobservable* with respect to $\mathscr{L}(G)$, $\Sigma_{i,o}$, $\Sigma_{i,c}$, $i \in I$ if for all $s \in \overline{\mathscr{L}(K)}$ and for all $\sigma \in \Sigma_c = \bigcup_{i \in I} \Sigma_{i,c}$,

$$s\sigma \notin \overline{\mathscr{L}(K)} \wedge s\sigma \in \mathscr{L}(G) \Rightarrow \exists j \in I_c(\sigma) \text{ such that } P^{-1}_{\Sigma_{j,o}}(P_{\Sigma_{j,o}}(s))\sigma \cap \overline{\mathscr{L}(K)} = \varnothing.$$

Given $\mathscr{L}(K)$, $\mathscr{L}(G)$, controller set I, $\Sigma_{i,o}$ and $\Sigma_{i,c}$ for all $i \in I$ where $\bigcup_{i \in I} \Sigma_{i,c} = \Sigma_c$, $\bigcup_{i \in I} \Sigma_{i,o} = \Sigma_o$ and the CP architecture is used, the necessary and sufficient conditions for the existence of control maps for each $i \in I$ to satisfy $\mathscr{L}(K)$ is controllability and CP-coobservability ([72]). Furthermore, if $\mathscr{L}_m(G)$ is nonempty and nonblockingness (marked closed-loop behaviour equals $\mathscr{L}(K)$) of closed-loop behaviour is required then $\mathscr{L}(K)$ needs to be $\mathscr{L}_m(G)$-closed.

Given G, $\Sigma_{i,o}$ and if i observes $\mathscr{L}(G)$ using its natural projection, $P_{\Sigma_{i,o}}$, then i's unobservable reach of a state $x \in X$ is defined as $\{y \in X \mid \exists s \in \Sigma \smallsetminus \Sigma^*_{i,o}, \delta(x,s) = y\}$. The unobservable reach is a generalization of ε-reach defined in [27].

Given DFA G, $\Sigma_{i,o}$, that i observes $\mathscr{L}(G)$ using its natural projection $P_{\Sigma_{i,o}}$ and that i knows / has provided to it G, one can construct i's "observer" DFA which recognizes $P_{\Sigma_{i,o}}(\mathscr{L}_m(G))$ and generates $P_{\Sigma_{i,o}}(\mathscr{L}(G))$. For any string $s \in \mathscr{L}(G)$, the state of i's observer DFA reached by $P_{\Sigma_{i,o}}(s)$ represents the subset of states in X that are reached by all strings in $P^{-1}_{\Sigma_{i,o}}(P_{\Sigma_{i,o}}(s))$. That is, the state provides an estimate of the state that G is in given i's observation. The observer DFA is constructed by replacing events in $\Sigma \smallsetminus \Sigma_{i,o}$ labeling transitions in G by ε then applying the subset construction.

Given its observation of strings generated by G, an agent i can maintain different estimates of the actual string generated by G. If i only has bounded memory then it can maintain a "state estimate" of G, provided by states reached by strings it observes in its observer DFA. If i has unbounded memory then it can store the entire sequence of events it observes which are generated by G and can instead compute the set of all strings $s \in \mathscr{L}(G)$ where $P_{\Sigma_{i,o}}(s)$ is equivalent to the sequence i observes. Such a set is referred to as a "string estimate". String estimates provide a more precise estimate of the system behaviour than state estimates. Indeed, the set of all states reached by strings in a string estimate is equivalent to a state estimate returned by i's observer DFA. However, string estimates require substantially more memory to maintain. Both state and string estimates will be considered in the surveyed works.

2.3 Diagnosis of DES

Besides problems of control, other work in DES has considered problems of fault / failure diagnosis. In such problems agents merely observe the behaviour generated by G and do not interact with G. In these problems there exists a set of fault / failure events Σ_f which are not observed by any agent. The basic problem considered is that if an occurrence of a fault event $\sigma_f \in \Sigma_f$ is generated by G then the agents should be able to individually or collectively determine that a fault was generated by G a bounded number of event occurrences following the occurrence of σ_f. More difficult problems consider that the specific fault event be diagnosed (not just that an event in Σ_f occurred) or that every fault event occurrence be uniquely diagnosed.

For solving the basic diagnosis problem in the centralized case, a "diagnoser" DFA may be constructed. The construction of the diagnoser DFA is similar to the construction of the observer DFA except that it is performed from a modified version of G where the state-transition function of G is refined and where states are labelled by an "N" if no fault event occurs in all strings leading to the state and "Y" if a fault event occurs in all strings leading to the state. This modified version of G can be computed by the synchronous product of G and the "label automaton" depicted in Fig. 1.

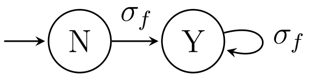

Figure 1: Label automaton for building diagnoser.

We denote the diagnoser DFA for σ_f by $Diag_{\sigma_f}(G)$. If $|\Sigma_f| > 1$ and we need to uniquely diagnose $\sigma_{f,i} \in \Sigma_f$ then a diagnoser DFA can be computed for each $\sigma_{f,i}$ from G synchronously composed with a label automaton whose transitions are labelled with $\sigma_{f,i}$.

On-the-fly diagnosis of σ_f by a centralized diagnoser can be performed by tracking and examining the diagnoser DFA's current state in response to the observable events generated in G (from [9]):

- If all the states of G in the current state of $Diag_{\sigma_f}(G)$ have label "Y" (resp., "N") then event σ_f has (resp., has not) occurred at some point in the past.
- If the current state of $Diag_{\sigma_f}(G)$ contains at least one state of G with label "N" and at least one state of G with label "Y" then the event σ_f may or may not have occurred in the past. We call such a state an *uncertain* state.

For solving the basic diagnosis problem, the diagnoser DFA can be used for diagnosing the occurrence of a fault event if and only if a condition known as *diagnosability* is satisfied ([75]). Informally, unobservable event σ_f is not diagnosable in live G if there exist two strings $s_N, s_Y \in \mathscr{L}(G)$ that satisfy the following conditions: (i) s_Y contains σ_f and s_N does not; (ii) s_Y is of arbitrarily long length after σ_f; and (iii) $P(s_N) = P(s_Y)$. When no such pair of strings exist σ_f is said to be diagnosable in $\mathscr{L}(G)$.

Diagnosability can be verified by determining whether or not there exist certain cycles of uncertain states in $Diag_{\sigma_f}(G)$. For a more comprehensive and illustrative introduction to problems of failure diagnosis in DES, the reader is referred to Section 2.5.3 of [9].

2.4 Notes on Observation in DES

In some works to be surveyed instead of using a natural projection to observe G, a mask is used instead. A mask M is a function $M : \Sigma \cup \{\varepsilon\} \to \Sigma_{output} \cup \{\varepsilon\}$ where $\Sigma \cap \Sigma_{output} = \varnothing$. The set Σ_{output} contains the actual observed symbols. Events in $\Sigma_{uo} \cup \{\varepsilon\}$ are mapped to ε, events in Σ_o are mapped not necessarily injectively to symbols in Σ_{output}. This differs for natural projections where events in Σ_o are mapped injectively to their observed symbols.

Even more general than natural projections and masks, observation maps defined over the transitions of G or over the strings of $\mathscr{L}(G)$ may be considered. Using such observation maps, the observation of any two occurrences of an event $\sigma \in \Sigma_o$ may differ. That is, following some string $s \in \mathscr{L}(G)$ an occurrence of σ may be observed but following some other string $s' \in \mathscr{L}(G)$ an occurrence of σ may not be observed. This contrasts with natural projections and masks where all occurrences of an event $\sigma \in \Sigma$ are observed (resp., not observed). For a DES control (resp., diagnosis) problem, when such observation maps are considered we say that we are dealing with a problem of "dynamic observation". Problems where only natural projections or masks are considered are problems of "static observation". As different types of observation maps will be considered in this paper, when we say that two strings / states appear identical / indistinguishable to an agent, this should be interpreted in terms of the problem setting of the work being discussed.

3 Early Works on Communication For Control

To our knowledge, one of the first papers to consider inter-agent communication in problems of DES is [88]. Here the author identifies some of the fundamental issues for incorporating inter-agent communication into decentralized control problems. The answers to these questions affect the model design and the subsequent verification and synthesis of control and communication solutions:

- Why should communication be introduced?
- Who should communicate with whom and when?
- What information should be communicated?
- Who should know what and when?

The initial proposals where these questions are addressed focus on agents communicating to maintain string estimates ([109, 98]) or state estimates ([2, 65]).

The CP control of DFA G by two given controllers is considered in [109]. Here controllers maintain string estimates which are updated based on the observation of events in Σ using their local natural projection, $P_{\Sigma_{i,o}} : \Sigma^* \to \Sigma_{i,o}^*$. Additionally, a one-way communication channel from controller 2 to controller 1 is considered. Controller 2 communicates its observed event occurrences to controller 1. If controller 2 communicates an event to controller 1 then it does so immediately upon its observation of the event. Communication of events is assumed to be reliable and delayless. Communication and computation of control decisions is performed between generation of events by G. The authors consider the problem of deciding, following any string generated by G, which event occurrences that controller 2 observes should be communicated to controller 1 in order for controller 1 to refine its string estimate in such a way that the CP control of both controllers on G generates exactly a prefix-closed, controllable $\mathscr{L}(K) \subseteq \mathscr{L}(G)$. It is assumed that controller 2 does not require any information from controller 1 to solve this problem. The authors assert that a communication protocol from controller 2 to controller 1 exists if, and only if, $\mathscr{L}(K)$ is observable with respect to $\mathscr{L}(G)$, $\Sigma_{1,o} \cup \Sigma_{2,o}$ and $\Sigma_{1,c} \cup \Sigma_{2,c}$. That is, if a centralized controller having the observational capabilities of both controllers does not know enough to control G in a manner that only strings in $\mathscr{L}(K)$ are generated then no communication protocol exists. This is not difficult to see. However, it is a necessary requirement for the existence of communication protocols that is used by subsequent works on communication.

The authors also consider minimal communication policies. Minimality is defined in the sense that, for a minimal communication policy, if controller 2 communicates any fewer occurrences then the control problem cannot be solved. This notion of minimality is a logical one which is considered in several subsequent works. The authors assert the existence of minimal communication protocols.

The work of [98] is a generalization of [109]. It considers an arbitrary number of controllers. Of note, it proposes three communication problems where it is the objective to compute communication and control maps for agents which result in the same control decisions for each agent as given control maps which require observation of all events in Σ by each controller. One of the problems considers that controllers request information from other controllers. Solutions to this problem result in dynamic observation of event occurrences for controllers. However, a solution approach remains open.

To our knowledge, [2] forms the first major study of communication for control in DES. This work was conducted in parallel with [98]. The primary differences are: (i) Broadcast communication is considered rather than point-to-point communication; (ii) The controllers collectively control and observe all events in Σ, which is not a requirement in [98]. There are also some similarities: (i) Communication is reliable and delayless; (ii) Communication occurs between generation of events by the plant; (iii) Communication occurs in two phases: once following the execution of an event in the plant, then once more modeling replies to the initial communication.

Controllers maintain state (resp., string) estimates of G. When a controller initiates a two-phase communication it broadcasts its estimate to all other controllers. In response, these controllers broadcast their estimates to every other controller. Afterwards, each controller's estimate is refined by computing the intersection of its estimate with the estimates received from all other controllers.

A controller's decision to initiate a two-phase communication is based on a change in its observation. Strings of $\mathscr{L}(G)$ are extended to incorporate two-phase communications between event occurrences. These strings are referred to here as extended strings. Given control and communication maps based on local observations of extended strings for the $|I|$ controllers, the CP closed-loop control of the controllers acting on G is defined using extended strings. The general control problem to be solved is the computation of communication and control maps for each controller such that the closed-loop control equals a specification language $\mathscr{L}(K)$. This is different from [98] where the objective is to compute communication and control maps which exert the same control decisions as given control maps where it is assumed that controllers observe all events in Σ.

The authors first consider and compare two cases where estimates maintained and communicated between controllers are: (1) based on the states of G or (2) based on the maximum cardinality set of extended strings whose projection onto Σ is contained in $\mathscr{L}(G)$.

The authors prove that, when state estimates are maintained and communicated, the problem is solvable if, and only if, $\mathscr{L}(K)$ is controllable with respect to $\mathscr{L}(G)$, Σ_{uc} and observable with respect to $\mathscr{L}(G)$, Σ_o, Σ_c (i.e., can be solved by a centralized controller). In the construction used in the proof, though each controller may not be able to unambiguously predict what state estimates will be communicated to it, it does have knowledge of the communication policy used by itself and all other controllers and can use this to refine its state estimate following any local observation of an extended string. In this case, it is said that each controller can "anticipate future communications".

Interestingly, the authors demonstrate that the set of problems solved by controllers maintaining and communicating string estimates that do not anticipate future communications (i.e., are myopic) is strictly contained in the set of problems solved by controllers maintaining and communicating state estimates that do anticipate future communications.

However, the authors argue that computing controllers which anticipate future communications is difficult due to dependencies between controller communication decisions and controller inference maps for providing future estimates. So they consider computation of myopic controllers and their communication maps.

The authors then consider problems of minimal communication for myopic controllers. Three different notions of minimality are studied. The first is described below:

(C1) A communication policy θ is minimal if there exists a control policy such that the control problem is solved under the communication-control policy pair and there does not exist a communication-control policy pair which does solve the control problem where the communication policy is obtained from θ by removing two-phase broadcast communications from θ following any set of strings in $\mathscr{L}(G)$.

The second and third, denoted by (C2) and (C3), are successive refinements of (C1) where communication must occur as late as possible. An approach is provided for computing communication policies for a specific class of control policy which are minimal in the sense of (C1). Computing communication-control pairs that are minimal in the sense of (C2)–(C3) remain open.

In the approach provided controllers maintain and communicate state estimates. Given G, K, $\Sigma_{i,o}$, $\Sigma_{i,c}$, a "finite estimator" automaton is defined. Control and communication policies are defined over the state set of this automaton. Each controller's state estimate is encoded into the states of the estimator which are used to define its control map.There may exist reachable states in the estimator where an event should be disabled but none of the controllers which control the event can unambiguously decide from their state estimates whether or not to disable the event from occurring. Such states are referred to as *conflict states*. It is the objective that control-communication policies used avoid conflict states.

For a specific control policy, the authors define a general communication rule R1* for avoiding conflict states which may be realized by multiple different communication policies, all of which are proven to be minimal in the sense of (C1).

The authors prove that computing the estimator structure is in $O(|\Sigma| \cdot |X| \cdot 2^{|I| \cdot |X|})$ where X is the state set of G. It follows that verifying whether or not a control-communication policy will avoid conflict states using the estimator suffers the same complexity. However, the authors conjecture that more efficient verification algorithms exist.

4 Decidability of Control Problems Involving Communication

In this section we survey some of the known decidability results for control and communication problems.

One of the first important works on considering delays in communication between controllers for distributed control problems is [94]. It establishes a hierarchy of control problems with decreasing delay in communication between controllers as well as decidability results for some bounded and unbounded delay communication problems.

In this work the plant and controllers are modelled as finite automata. Only two controllers are considered. The controllers communicate all local observations of event occurrences to one another. Communication is reliable but can incur delay. A message sent from one controller to another incurs delay d if, during transmission of the message, the plant generates d events. Communication channels between the controllers are first in, first out (FIFO). Communicated messages incur a delay of up to some bound, $k \geq 0$.

A restricted type of regular language specification referred to as a "responsiveness property" is considered. A responsiveness property is a causality condition characterizing that any occurrence of event e_1 must be eventually followed by an occurrence of an event e_2. Though responsiveness properties can model many types of practically-relevant specifications, they are very restricted in comparison to the expressivity of the family of regular languages. The class of control problems considered are those where communication between the two controllers incurs bounded or unbounded delay and any marked string generated in the closed-loop control should satisfy a given set of responsiveness properties. Also, it is required that the closed-loop control be nonblocking.

The author proves that an infinite hierarchy of strict containments of such nonblocking distributed control problems is induced as the maximum delay that a message incurs in a communication channel decreases. That is, the set of control problems where communicated messages experience at most delay $k \geq 0$ strictly contains the set of control problems where communicated messages experience at most delay $k + 1$. Also, the author proves that if communication can incur unbounded delay or if no communication occurs then determining if there exists two communicating controllers that solve the control problem is undecidable. The author also proves that the class of problems where no communication occurs are strictly contained in the class of problems where communication can incur unbounded delay.

Also, the author proves that if two controllers can be used to solve a given $k+1$-bounded delay communication problem then they can also be used to solve the same problem but where communication is k-bounded. Furthermore, if these controllers yield nonblocking behaviour in the closed-loop control on the plant when communication is $k + 1$-bounded then they also yield nonblocking behaviour when communication is k-bounded. However, this is only proven when responsiveness properties are considered for the specifications. The author provides an example demonstrating that this property does not hold for regular language specifications in general.

Towards proving that, in general, control problems involving two controllers communicating with bounded delay are decidable, the author proves the decidability of a specific decentralized observation problem with bounded delay communication.The author introduces the notion of k-bounded-delay joint observability, which characterizes that if a string t is generated by G in $\mathscr{L}(K)$ and both observers have completed communicating their local observations along t to one another then it is possible to use the observed sequence of local and received communicated events by either observer to determine that t is in $\mathscr{L}(K)$. The author proves that verifying k-bounded-delay joint observability is decidable and provides a decision procedure exponential in k for solving it.

While this result is an important step towards proving that control problems with bounded delay communication are decidable, the joint observability condition is incomplete: it only considers strings where communication between the controllers has completed and does not consider when messages still exist in communication channels.

The work of [26] follows and is directly comparable to [94]. The main differences are listed:

- Considers an arbitrary number of controllers.
- Communication topologies are not necessarily complete.
- Considers arbitrary regular language specifications contained in $\mathscr{L}(G)$ rather than responsiveness properties.
- Considers that the automaton modeling CP closed-loop control, denoted by S/G_{com}, should be bisimilar ([49]) with a given specification automaton, K, rather than requiring that $P_\Sigma(\mathscr{L}(S/G_{com})) \subseteq P_\Sigma(\mathscr{L}(K))$.

Bisimilarity between two automata is a stronger notion than equivalence of the projections from the languages recognized by two automata to the intersection of their alphabets. That is, if S/G_{com} is Σ-bisimilar with K for $\Sigma \subseteq alph(G) \cap alph(K)$ then $P_\Sigma(\mathscr{L}(S/G_{com})) = P_\Sigma(\mathscr{L}(K))$. The converse does not necessarily hold.

The author argues that, when the observations that a controller receives along a string $t \in \mathscr{L}(G)$ are nondeterministic due to nondeterministic delays in receiving communicated events, language equivalence between $P_\Sigma(\mathscr{L}(S/G_{com}))$ and $P_\Sigma(\mathscr{L}(K))$ is "insufficient" and instead Σ-bisimilarity between S/G_{com} and K should be used for determining when K is satisfied. Language equivalence is insufficient in that, for strings $s, t, s' \in \mathscr{L}(S/G_{com})$, $k \in \mathscr{L}(K)$, $P_\Sigma(s) = P_\Sigma(t)$, $s \in \overline{s'}$ and $P_\Sigma(s') = P_\Sigma(k)$ but there may not exist any string $t' \in \mathscr{L}(S/G_{com})$ where $t \in \overline{t'}$ and $P_\Sigma(t') = P_\Sigma(k)$. That is, when language equivalence is satisfied it is not necessarily the case that any string generated in $\mathscr{L}(S/G_{com})$ can be completed to a string whose projection onto Σ is in $\mathscr{L}(K)$ when another observationally equivalent string can. When Σ-bisimilarity is satisfied this must be satisfied.

Communication is modeled by events which are controllable and observable by agents. Though these communication events are not generable from the initial plant model itself (G), FIFO communication channels between agents are modeled by automata transitioning on these events, and these are synchronously composed with the plant to obtain the plant model with communication between agents (G_{com}). In the problem considered, there is a bijection between the set of observable plant events and the set of communication events. Basically, when an agent observes an event, it communicates its occurrence to the other agents by generating a corresponding communication event. Under this setting, the author shows that, given G_{com}, K, $\Sigma_{1,o} \dots \Sigma_{n,o}$, $\Sigma_{1,c} \dots \Sigma_{n,c}$, computing a map for each agent from their observations to control decisions such that S/G_{com} is Σ-bisimilar with K and where delay in communication of events can be unbounded is undecidable. This result is stronger than the undecidability result for unbounded-delay communication in [94].

The author also considers the two following cases where, under either case, it is proven that computing control maps for controllers which satisfy the control objective is decidable: (1) k-bounded delay communication; (2) any cycle in G contains an event observable by all controllers. Case (1) and the decidability results of [94] strongly suggest that control problems with bounded delay communication are decidable.

In [96] the author considers the relationships between observation conditions where agents may have bounded or unbounded memory and communicate with one another or send all local observations to a coordinator. In contrast to [94] communication is delayless. Four different conditions of decentralized observability are defined. The first two conditions are types of joint observability (an implementation would require communication among agents) and the last two decentralized observability conditions where decisions are made locally (no communication among agents).

The first and most general condition explored is that of unbounded-memory joint observability (JO). A regular specification $\mathscr{L}(K)$ is JO with respect to $\mathscr{L}(G)$ and $(\Sigma_{1,o},\ldots,\Sigma_{n,o})$ if, and only if, there exists a total function $f : \Sigma_{1,o}^* \times \ldots \times \Sigma_{n,o}^* \to \{0,1\}$ such that for all $p \in \mathscr{L}(G)$, $p \in \mathscr{L}(K)$ if, and only if, $f(P_{\Sigma_{1,o}}(p),\ldots,P_{\Sigma_{n,o}}(p)) = 1$. That is, JO holds if, and only if, there exists some Boolean function that takes as input all local observations of the string p generated by the plant and outputs 1. Obviously there must be some coordination involved amongst the agents to decide when a string is in $\mathscr{L}(K)$ or the agents must send their local observations of p to a coordinator which determines when p is in $\mathscr{L}(K)$. This version of decentralized observability is referred to as "unbounded-memory" since p could be of arbitrary (unbounded) length and each agent must remember its observation of p (which could also be unbounded).

The second definition of decentralized observability is referred to as finite-memory joint observability (FJO) and is defined as above but, instead of basing decisions of membership of $p \in \mathscr{L}(K)$ on $P_{\Sigma_{i,o}}(p)$, which is potentially unbounded, decisions are based on the states reached by $P_{\Sigma_{i,o}}(p)$ in given DFA representations of agents.

The last two definitions of unbounded-memory local observability (LO) and finite-memory local observability (FLO) are defined analogously to the previous two but differ in that the decision of membership of a string $p \in \mathscr{L}(K)$ is based on an individual agent's local observation.

The author proves that JO is strictly weaker than FJO, LO. Finite-memory joint observability is strictly weaker than FLO. Unbounded-memory local observability is weaker than FLO.

The author considers a specific subclass of regular languages for the plant known as trace languages. They prove that if $\mathscr{L}(G)$ is a trace language and all $\Sigma_{i,o}$ are pairwise disjoint then $\mathscr{L}(K) \subseteq \mathscr{L}(G)$ is JO with respect to $\mathscr{L}(G)$ and $(\Sigma_{1,o},\ldots,\Sigma_{n,o})$ if, and only if, $\mathscr{L}(G)$ is FJO with respect to $\mathscr{L}(G)$ and $(\Sigma_{1,o},\ldots,\Sigma_{n,o})$. That is, in this particular case, it suffices to base decisions on membership in $\mathscr{L}(K)$ using the states of the given agents' automata.

They also prove that, independent of the relationships among $\Sigma_{i,o}$, if $\mathscr{L}(G)$ and $\mathscr{L}(K)$ are trace languages then verifying JO of $\mathscr{L}(K)$ with respect to $\mathscr{L}(G)$ and $(\Sigma_{1,o},\ldots,\Sigma_{n,o})$ is decidable. In contrast, for more general cases, verifying JO of $\mathscr{L}(K)$ with respect to $\mathscr{L}(G)$ and $(\Sigma_{1,o},\ldots,\Sigma_{n,o})$ is undecidable under the following conditions ([93]): (i) $n \geq 2$

agents and $\mathscr{L}(G)$ is not prefix-closed; (ii) $n \geq 3$ agents and $\mathscr{L}(G)$ is prefix-closed. These two general cases also render a problem of verifying "decentralized diagnosability" in [78] to be undecidable.

5 Distributed Estimation Involving Communication

The work of [83] proposes a communication protocol between agents in a modular DES problem setting for refining agent string estimates. Though the application is for diagnosis of DES the protocol may be applied in other domains.

Here G is modeled as the synchronous product of a set of I DFA components G_i, $i \in I$. The languages recognized by each G_i and consequently by G are assumed finite. Each G_i is capable of generating unobserved fault events. A diagnoser D_i is associated with each G_i which observes events generated by G_i. Diagnoser D_i maintains string estimates E_i of G_i based on the observation of event sequences generated by G_i. Diagnosis decisions are based on E_i, though no specific diagnosis problem is formulated. The authors consider that D_i and D_j can communicate projections of their string estimates, E_i and E_j, with each other to refine E_i and E_j if $\Sigma_i \cap \Sigma_j \neq \varnothing$ where Σ_i (resp., Σ_j) denote the alphabets of G_i (resp., G_j). To this end they specify a communication protocol between D_i and D_j that can be used to compute a refinement of E_i and E_j. The authors do not specify how the protocol should be implemented in terms of low-level message passing.

Computation and communication of estimates between diagnosers occurs at certain pre-specified times during execution of G. It is up to the designer / practitioner to define these times. Events generated by G_i observable by D_i are buffered by each D_i between consecutive specified times. At each specified time, D_i updates E_i using its buffered observed event sequence s_i in the following manner: $E_i = \{ss' \in \mathscr{L}(G_i) \mid s \in E_i, P_{\Sigma_{i,o}}(s') = s_i\}$. Following this D_i communicates with all other D_j where $\Sigma_i \cap \Sigma_j \neq \varnothing$ until E_i cannot be refined further using the string estimates $P_{\Sigma_{i,o}}(E_j)$ received from D_j. The precise communication procedure is provided in [83].

The authors assert that this procedure terminates after a finite number of iterations. No bound on the number of iterations is claimed. It is useful for D_i to apply such a procedure when there exists another agent D_j that can observe events in $\Sigma_i \cap \Sigma_j$ that D_i cannot observe. The authors define the notion of a supremal estimate for all I diagnosers and assert that the collection of estimates $(E_1, \dots, E_n)$ computed using this procedure is the supremal estimate. In a sense, each E_i in the supremal estimate is the best string estimate of G_i observationally consistent with strings in E_j ($P_{\Sigma_{j,o}}(E_i) = P_{\Sigma_{i,o}}(E_j)$) where $\Sigma_i \cap \Sigma_j \neq \varnothing$ given the previous estimates of agents at the last specified time and observations that they have buffered since. That is, knowing the estimate E_j of some other D_j will not refine E_i when $\Sigma_i \cap \Sigma_j \neq \varnothing$. The authors assert that the procedure computes the supremal estimate for each D_i at a given specified time.

Using the supremal estimates computed at specified times, diagnoses can be issued at the local component level of G_i directly or at the global plant level of G by computing global estimates from the supremal estimates. However, the authors assert that computing the set of global estimates for a given set of agent supremal estimates ξ is NP-hard.

The authors assert that, for a given set of sequences s_i of events generated by G_i, if

D_i updates E_i using the outlined procedure an arbitrary number of specified times during the generation of s_i followed by a final update after all s_i have been generated then the E_i obtained is equivalent to only updating E_i once after all s_i have been generated. This is a useful result which indicates that for more specific problems one can compute a minimum cardinality set of prespecified times for updating E_i, depending on the observational condition to be satisfied (e.g., k-diagnosability, coobservability with respect to a global specification, observability with respect to individual component specifications, etc.). Computation of minimum cardinality of specified times for these specific problems is a possible direction for future work.

The work of [83] is continued in [82]. Here plant component languages can be of any language expressivity, not just regular. In [83] the notion of observational consistency of string estimates ($P_{\Sigma_{j,o}}(E_i) = P_{\Sigma_{i,o}}(E_j)$), a property that the supremal estimates of [83] satisfy, is referred to as *local consistency* here. Recall that this property, when satisfied, characterizes that knowing the estimate E_j of D_j does not refine E_i when $\Sigma_i \cap \Sigma_j \neq \varnothing$. A set of estimates $(E_1, \ldots, E_n)$ which satisfy local consistency is referred to as a *local support.* The supremal estimate of [83] is the supremal local support.

The authors introduce a strictly stronger consistency condition called *global consistency.* This condition characterizes that knowing all other estimates E_j will not refine E_i, independent of the relation between Σ_i and Σ_j. A set of estimates $(E_1, \ldots, E_n)$ satisfying global consistency is referred to as a *global support.* Given preliminary local estimates $(M_1, \ldots, M_n)$ computed using each agent's local observations and without communicating with other agents, the supremal global support of $(M_1, \ldots, M_n)$ provides the best possible string estimate for D_i of G_i.

The authors prove that if the communication topology used by agents is acyclic then $(E_1, \ldots, E_n)$ is a global support if, and only if, it is a local support. As a corollary, the procedure of [83] for computing the supremal local support of preliminary local estimates $(M_1, \ldots, M_n)$ can be applied to compute the supremal global support.

The authors provide centralized algorithms for computing individual E_i of the supremal local and global supports. The algorithm for computing the supremal global support has complexity of the same order as computing the synchronous product of all local preliminary estimates. However, in practice the complexity may be improved by considering specific orderings of I. The authors present a heuristic ordering algorithm for computing such orderings. The algorithm for computing the supremal local support is recursive and not guaranteed to terminate, even when the plant component languages are regular languages ([91]). However, the authors prove that termination is guaranteed when component languages are finite (as is the case in [83]) or when the communication topology used by agents is acyclic. More general than the second case, it has been proven that if the communication topology contains a special spanning tree called a skeleton then termination is guaranteed ([90]).

Though a distributed algorithm for computing supremal local supports is provided in [83], the authors do not provide a distributed algorithm for computing supremal global supports. This is a possible future research direction.

A communication protocol for refining agent state-transition (equivalently, string) estimates of G in the setting of [58] is presented in [57]. The proposed application is in the distributed diagnosis setting of [58] though modifications to the protocol may be useful in other distributed observation problems.

Here it is required that there are no cycles of events unobserved by any agent in G. String estimates are communicated rather than observed event occurrences in [58]. Estimates are broadcast-communicated from an agent following every locally-observed event occurrence. An agent's estimate is updated based on local event observations and is truncated using the estimates communicated to it by other agents over k-bounded delay channels.

This work is comparable to [83, 82]. In both, string estimates are maintained and communicated, and communication occurs until some notion of observational consistency is satisfied between the estimates of agents which communicate with one another. The problem domain in this work differs in that communication incurs delay. The approach differs in that agents communicate their entire string estimate rather than a projected version of their estimate to other agents.

The authors present a complicated three-phase protocol for updating and communicating string estimates which is initiated every time an agent observes an event occurrence. They assert with proof in [55] that $\mathscr{L}(G)$ is diagnosable with respect to $\mathscr{L}(K)$ and the three-phase communication protocol when communication is k-bounded if, and only if, $\mathscr{L}(G)$ is joint$_k$-diagnosable with respect to $\mathscr{L}(K)$ and the mask-based local observation maps of each agent. Joint$_k$-diagnosability is an observational condition introduced in [58]. A verification procedure is provided for diagnosability under the three-phase communication protocol whose complexity is linear in the number of diagnosers but exponential in the number of states of G, K and Σ.

Though the distributed communication protocol of [57] may be used when joint$_k$-diagnosability of G is satisfied, there are three problems with this protocol: 1) Each site / agent communicates its entire state-transition estimate of G to all other sites, which can be expensive; 2) It requires a complex distributed synchronization strategy for the truncation of unnecessary history of local state-transition estimates; 3) It requires that there are no cycles of events unobserved by any agent in G. The authors of [112] propose a protocol that does not require all agents to communicate their entire string estimates, is conceptually much simpler than the protocol of [57] in that no synchronization with other sites is necessary for truncating history of estimates and does not require the absence of cycles of unobservable events in G.

State estimates of G are maintained by agents. Agents observe events through masks. Every observed event occurrence is communicated by an agent to every other agent. Communication incurs a delay of up to some bound. No specific observational condition such as observability, diagnosability, etc. is considered. It is assumed that each agent knows *a priori* what events are locally observed by every other agent.

A directed state estimate graph is maintained by each agent i for refining its state estimate of G using local event observations and received communicated events which incur delay. The states of this graph are copies of individual states in G where a transition exists between two states in the graph if the transition exists between the corresponding pair of states in G. Additionally, states are labelled by agent identifiers. Those states of a graph labelled by agent i are in i's state estimate. States of i's state estimate graph are also labelled by $j \neq i$ defining a subset of those states in j's state estimate at the time of j

sending its last communication which has subsequently been received by i.

States and transitions are added to i's state estimate graph on the occurrence of i's local observations. Those states labelled by i are updated on the occurrence of i's local observations. Those states labelled by j are updated on the occurrence of events communicated and received at i from j. After states (resp., transitions) of i's state estimate graph have been updated and labeling of states has been performed on event observation / reception, those states and transitions in the graph which are not reachable from any state labeled by an agent identifier $j \neq i$ are pruned from the graph. We omit the precise details of how an agent's state estimate graph is updated.

The authors assert that the set of paths formed in the state estimate graph of an agent after generation of any string in G forms a regular set, so the state estimate graph is finite. They also assert that the true state of G is contained in an agent's state estimate provided by its graph. This property is referred to by the authors as "soundness". Given a string $t \in \mathscr{L}(G)$, t may appear differently to an agent due to nondeterminism in communication delays. Denote the set of strings that i may observe on occurrence of t by $O_i(t)$. Two strings $t, t' \in \mathscr{L}(G)$ are indistinguishable to i if $O_i(t) \cap O_i(t') \neq \varnothing$. The authors assert that, for any string s observed by i, the state estimate maintained by agent i is contained in the union of all states reached by strings $t \in \mathscr{L}(G)$ where $s \in O_i(t)$. That is, the state estimate only contains those states reached by strings whose observation to i is what i has observed. This property is referred to by the authors as "completeness".

The last work to exclusively consider distributed state estimation in DES is [31]. Here the plant is modular and modeled by the asynchronous product of communicating finite state machine (CFSM) representations of distributed processes. Processes communicate with each other by the use of reliable, unbounded FIFO queues. Messages communicated between processes can incur delay. Events considered in the distributed system which change the state of a process' CFSM are enqueue and dequeue message events associated with its inter-process communication queues. A local state estimator is associated with each distributed process. Estimators observe all events of its process' CFSM. Each estimator is presumed to know *a priori* the CFSM of each process. Each estimator maintains an estimate of the global state of the plant (consisting of the states that each process CFSM could be in and contents of each inter-process queue in the system) using observations of events which change the CFSM state of its process as well as information piggybacked onto messages sent from processes to its process by other estimators. An estimator piggybacks information onto all messages sent from its process to another process. The information piggybacked consists of an estimator's state estimate and associated vector clock (introduced in [40]). The vector clock is a vector of natural numbers of size n where n is the number of processes in the system. The value at index i of j's vector clock represents the minimum number of event occurrences estimator j knows have occurred in process i's CFSM.

The authors present an algorithm for maintaining the state estimate of an estimator. When an output / enqueue event occurs in process i, estimator i updates its state estimate to the unobservable reach of its current state estimate following the occurrence of the event. When an input / dequeue event occurs at process i from process j, estimator i compares values of its vector clock with the vector clock from estimator j piggybacked on the message received at process i to determine how its state estimate should be updated using its estimate and the estimate piggybacked from estimator j.

Specifically, computing a state estimate in the algorithm assumes that a reachability operator on the global system states is provided. However, computation of the reachability set for a distributed system modeled using CFSMs is known to be undecidable ([6]). So the authors propose a method using abstract interpretation techniques (introduced in [13]) where the contents of a queue in the system are over-approximated using regular languages. It is argued in [21] that a good abstract domain for approximating FIFO queue contents are regular languages.

Soundness and completeness of state estimates are defined in a similar manner to [112]. The authors assert (with proof provided in an associated technical report) that if a given reachability operator computes overapproximations of the global system state then an agent's state estimate is complete. If, on the other hand, a given reachability operator computes underapproximations then an agent's state estimate is sound. Soundness and completeness can not be both satisfied. This is in contrast to the problem domain and distributed state estimation algorithm of [112] where the authors assert that soundness and completeness are both satisfied by their distributed state estimation algorithm.

The authors assert (with proof in an associated technical report) that the state estimate maintained by an estimator using the presented algorithm and their defined overapproximation of the reachability operator is sound.

There are differences between this work and [82, 112]. Here the plant is modeled as the product of CFSMs rather than automata. Furthermore, communication channels between plant components may be unbounded, resulting in the plant being infinite-state. Estimators are assumed to have complete observation of their associated processes, which was not assumed in the previous two works. Communication differs in that state estimates and vector clocks are communicated rather than string estimates or event observations. In contrast to [82], agents maintain estimates of the global system (CFSM state estimates and FIFO queue estimates) rather than local process estimates. In contrast to [112], the plant is considered to consist of a modular set of processes rather than a single, monolithic process and communication channels are modeled directly in the state of the plant.

This state estimate work has since been applied in [32] to solve a control problem for state avoidance in the same setting.

6 Distributed Diagnosis & Prognosis Problems Involving Communication

One of the first works to consider communication in problems of diagnosis is [5]. The problem setting is the same as [109] where two agents are considered and communication is unidirectional from agent 2 to agent 1. The primary differences are that the problem is of diagnosis, not control, and that state estimates are communicated from 2 to 1, not event occurrences. Also, in this work agent 2 maintains a string estimate for computing an overapproximation of agent 1's state estimate. The decision to communicate is made based on its state estimate and the overapproximation of agent 1's state estimate. Agent 2 communicates its state estimate when it cannot be determined from its overapproximation of agent 1's state estimate whether or not a failure has occurred and it is not the case that agent 2 can determine from its state estimate that a failure has not occurred. Only one unobserved failure event σ_f is assumed to occur in G. The problem considered only requires that if the failure event occurs then agent 1 will eventually be able to unambiguously determine that a failure has occurred.

A different distributed diagnosis problem and approach is considered in [18]. Similar to the previous work, a coordinator responsible for making diagnosis decisions receives state estimates from individual, decentralized agents which only communicate to the coordinator, do not receive messages from the coordinator and do not communicate with each other. From this information diagnosis is performed by the coordinator. The coordinator observes a totally ordered sequence of messages comprised of all messages sent by agents. However, due to delay in communication, the order in which the messages were received does not correspond with the order in which they were sent if sending of each message were to be timestamped with a global clock. So the coordinator can infer that, in addition to the total ordering of messages received, the true total ordering indicating when messages were sent exists in a set of multiple possible total orderings of the messages. However, the coordinator cannot infer directly which is the correct total ordering.

Each agent maintains a specific type of state estimate of G. Following each observed event occurrence an agent communicates its state estimate, a specific type of unobservable reach from this state estimate and status bits indicating whether or not the observed event is also observable by other agents. This latter information allows for proper synchronization of state estimates at the coordinator. As in most work on distributed state estimation, it is presumed that each agent knows the events observable by other agents.

The objective is to determine conditions under which any failure event occurrence can be diagnosed by the coordinator within bounded delay. However, different from many works on diagnosis, the coordinator's diagnostic procedure does not take as input the state-transition structure of G. Only the information communicated to it by agents are input.

The authors consider only two agents in this work. They consider that reception of messages at the coordinator from a given agent is in-order. They also only consider that messages received at the coordinator incur a delay between transmission and reception of up to one received event occurrence (abbreviated hereafter by delay up to one). Under these restrictions it is easy to compute the set of possible sequences explaining the observed sequence.

The authors describe a procedure used by the coordinator for updating its state estimate which buffers up to $n + 1$ received messages when messages incur a delay of up to n. From buffered messages all possible orders in which the messages could have been transmitted are computed. For each order a state estimate is computed from the coordinator's latest state estimate before receiving the buffered messages. If all computed state estimates contain only states in G where it is certain a failure event has occurred then a diagnosis can be reported.

Under the delay of up to one assumption on messages where all agents communicate their state estimates to the coordinator after an observed event occurrence the authors prove that the diagnostic procedure used by the coordinator can be used to diagnose all failures in the system in the same manner as the centralized diagnoser of [75] which observes all observable event occurrences if, and only if, there are no non-failure strings of arbitrarily long length in $\mathscr{L}(G)$ that look the same to each agent as another string which contains a failure occurrence. Under the delay of up to one assumption on messages the authors prove that, in general, it will take longer for the coordinator to diagnose a failure occurrence than a centralized diagnoser.

A similar result is obtained when regarding another communication protocol between agents and the coordinator. Instead, state estimates are communicated from agents only on commonly observed event occurrences in this protocol. This savings on communication which still allows the same diagnoses as a centralized diagnoser is surprising.

The results of this paper only consider that messages are received with delay of up to one. The authors conjecture that the results of this paper hold for the more general delay of up to $m \geq 1$. They also assert that the results of the paper are generalizable to $|I| \geq 2$ decentralized agents as the delay of up to n assumption is not affected by the number of decentralized agents from which messages originate.

Though [18] is one of the first works to investigate bounded delay communication in problems of diagnosis in DES subsequent works have investigated the decidability of general diagnosis problems with bounded / unbounded-delay communication.

In [78] the authors present "decentralized diagnosability", a type of joint diagnosability condition, and prove, analogous to the results of [93] for verifying joint-observability, that verifying if $\mathscr{L}(G)$ is decentralized diagnosable with respect to a set of unobservable fault events and sets of diagnoser observable events is undecidable for the following conditions: (i) $n \geq 2$ agents and $\mathscr{L}(G)$ is not prefix-closed; (ii) $n \geq 3$ agents and $\mathscr{L}(G)$ is prefix-closed. Specifically, a regular language $\mathscr{L}(G)$ is decentralized diagnosable with respect to a set of fault events Σ_f and sets of diagnoser observable events if $\exists m \in \mathbb{N}$ such that for all $t \in \mathscr{L}(G)$, if t contains a failure event in Σ_f followed by at least m event occurrences then for all $t' \in \mathscr{L}(G)$, if t is indistinguishable with t' by all diagnosers then the failure event must have occurred in t'. When decentralized diagnosability is satisfied diagnosis decisions must be made jointly by all diagnosers by communicating their observations of t to one another or to a coordinator. That verifying decentralized diagnosability is undecidable immediately implies that verifying delay-based generalizations of the condition are also undecidable. This corollary is one of the primary contributions of [78].

Not content with this decentralized diagnosability formulation when communications incur delay, the authors of [59] propose a different formulation of decentralized diagnosability. From [56] a diagnosability condition is recalled which generalizes the usual diagnosability condition ([75]) in the following respects: faults are characterized by strings rather than individual event occurrences; an agent's view is determined by a mask, not a natural projection.

The authors recall the codiagnosability condition of [56] which generalizes diagnosability in the same manner that coobservability generalises observability: following n event occurrences after execution of a string in the fault language there exists a diagnoser that can unambiguously determine that a fault string occurred.

The authors then introduce the condition of joint$_\infty$-diagnosability, which characterizes plant languages where fault strings in a given fault language can be diagnosed some number of event occurrences after their occurrence by a diagnoser when diagnosers can communicate some of their event observations to one another and communication incurs up to an infinite / unbounded delay. The condition does not assume any specific communication policy.

The authors assume that a DFA representation of $\mathscr{L}(G)$ is deadlock-free. They then prove that their codiagnosability and joint$_\infty$-diagnosability conditions are equivalent. As verifying their codiagnosability condition is decidable [56], verifying joint$_\infty$-diagnosability for a given $\mathscr{L}(G)$ and regular fault language is also decidable.

The authors also formulate a generalization of the decentralized diagnosability condition of [78] in terms of fault strings and observation masks.

The authors prove that joint$_\infty$-diagnosability is strictly stronger than the generalization of decentralized diagnosability condition of [78]. To satisfy either condition communication is required. What is surprising is that communication can incur unbounded delay to satisfy the former condition whose verification is decidable while communication does not incur any delay in satisfying the latter condition whose verification is undecidable.

Motivated by the decidability of joint$_\infty$-diagnosability, the authors of [59] investigate verification and synthesis algorithms for problems of diagnosability where communication between diagnosers incurs a delay of up to some bound in [58].

The problem setting is refined to one where a diagnoser communicates locally observed events to another agent that are not locally observed by the other agent over unidirectional FIFO channels and G can execute up to $k \in \mathbb{N}$ events between transmission and reception of a message. The authors consider two diagnosers but the approach presented can be generalized to more where care must be taken so that an agent only receives one message corresponding to a given event occurrence that it does not locally observe.

There exists a unique FIFO communication channel from a diagnoser to every other diagnoser. A k-bounded delay FIFO communication channel from diagnoser i to diagnoser j is modeled by a DFA $C_{i,j}$. Automata G and K can be extended to incorporate the k-bounded communication between diagnosers by synchronously composing them with all $C_{i,j}$. Denote the resulting automata by G^k, K^k.

Due to nondeterminism of when communicated event observations will arrive at a diagnoser i from $C_{j,i}$, a string $t \in \mathscr{L}(G)$ may appear differently to diagnoser i. A map O_i is defined which maps strings $t \in \mathscr{L}(G)$ to the set of possible observed sequences of t over events locally observed and communicated to diagnoser i with delay up to k. Two strings $t, t' \in \mathscr{L}(G)$ are indistinguishable to i if $O_i(t) \cap O_i(t') \neq \varnothing$.

The joint$_k$-diagnosability condition is introduced. It is defined in the same manner as the authors' codiagnosability condition in [56] but where diagnoser i's view of t is provided by $O_i(t)$ and indistinguishability of strings is defined as above.

The authors prove that $\mathscr{L}(G)$ is joint$_k$-diagnosable with respect to $\mathscr{L}(K)$, $O_1, \dots, O_m$ if, and only if, $\mathscr{L}(G^k)$ is codiagnosable with respect to $\mathscr{L}(K^k)$ and observation masks M_i for each diagnoser $i \in \{1, \dots, m\}$ which remove events not locally observed or not communicated to i. So the verification procedure for codiagnosability of [56] can be used to verify joint$_k$-diagnosability, reducing the problem involving communication to one without communication. However, the authors present a more efficient algorithm for verifying joint$_k$-diagnosability.

The authors demonstrate that verifying joint$_k$-diagnosability is polynomial in the number of plant and specification states but exponential in the communication delay bound (k) and the number of local diagnosers (m). To circumvent exponential complexity in the number of diagnosers, the authors provide an alternative verification procedure in [57] whose complexity is linear in the number of diagnosers but exponential in the number of states of G, K and in Σ. So, depending on the problem at hand, one verification algorithm may be preferable over the other.

The authors describe how diagnoser automata may be synthesized when joint$_k$-diagnosability is satisfied.

The authors also prove that there exists a hierarchy of joint diagnosability conditions, with joint$_\infty$-diagnosability being the weakest which is strictly weaker than joint$_{k+1}$-diagnosability which is strictly weaker than joint$_k$-diagnosability which is strictly weaker than joint$_0$-diagnosability which is equivalent to the diagnosability condition of [56]. This hierarchy is analogous to the hierarchy of control problems established in [94].

Following [58], a similar problem regarding failure prognosis is considered in [87]. The problem setting and solution approach used are very similar to that of [58]. In the problem of failure prognosis it is required that agents observing a plant can collectively predict when a given regular specification will be violated a number of event occurrences before the violation for every violation of the specification. That is, there are no “missed failure detections”. It is also required that no incorrect prognostic decisions are issued. That is, there are no “false alarms”.

Generally speaking, the observational conditions which characterize when agents can prognose a system (prognosability and its variants) are stronger than the conditions required for diagnosing specification violations in a system and weaker than conditions required for controlling a system in a nonblocking manner to prevent such violations from occurring.

The notion of coprognosability (introduced in [37]) is recalled which is known to be a necessary and sufficient condition for the existence of non-communicating decentralized agents which ensure that there are no missed failure detections or false alarms.

As in [58], all event observations are communicated by an agent to all other agents. Communicated messages incur a delay of up to k event occurrences before their reception.

To solve the distributed failure prognosis problem where communication must be used, the authors introduce the notion of joint^k-prognosability. This is defined relative to coprognosability as joint$_k$-diagnosability (introduced in [58]) is defined relative to the codiagnosability condition of [56].

The authors demonstrate that joint^k-prognosability is a necessary and sufficient condition for solving the distributed failure prognosis problem.

They also demonstrate that verifying joint^k-prognosability cannot be reduced to verifying coprognosability using extended versions of the plant and specification automata synchronously composed with communication channel automata. By contrast, as stated before, verifying joint$_k$-diagnosability can be reduced to verifying codiagnosability using extended versions of the plant and specification automata ([58]). However, the authors present an algorithm for verifying joint^k-prognosability which relies on searching the state set of an automaton constructed from extended versions of the plant and specification automata. The complexity of the algorithm is polynomial in the cardinalities of the state set of the plant and specification automata, the event set of the plant and the set of symbols communicated among agents but is exponential in the delay bound (k) and the number of agents (m).

7 Knowledge and Inference-Based Approaches to Solving Problems Involving Communication

Recently, there has been growing interest and activity in casting decentralized / distributed observation problems in the knowledge / modal logic domain. Here agents have knowledge ascribed to them and are capable of inferring from their knowledge whether or not a given predicate holds. When agents do not have sufficient knowledge to determine if a predicate holds they may communicate with one another to increase their knowledge from which inferences are conducted.

The first work to model decentralized control problems in this domain is [66]. This work recasts two-agent decentralized DES control problems in the knowledge theory of [25]. The motivation for this is to allow formal reasoning about what agents need to know to solve control problems.

Associated with each agent i is a tuple consisting of a DFA, S_i, and a map from states of S_i to control decisions (set of enabled events). The CP closed-loop control of G under controllers can be defined.

In the knowledge model considered agents believe the system could be in a set of possible worlds. A world is modeled as a triplet containing a state of G and a state from each S_i. Associated with each world are atomic propositions defined over the events of G. Given a world (x, x_1, x_2), σ_G (resp., $\neg\sigma_G$) is a proposition stating that σ can (resp., cannot) be generated from state x in G. Symbol σ_K (resp., $\neg\sigma_K$) is a proposition stating that σ does (resp., does not) occur from state x in the specification DFA $K \sqsubseteq G$. In addition to standard propositional logic, modal operators are introduced. Symbol $K_i\phi$ means that agent i knows ϕ to be true. Modal operators may be used to express complex statements that could guide decision-making in reaching a control solution such as "agent 1 knows that agent 2 does not know to disable event σ". However, statements considered here are restricted to the form of "agent 1 knows to disable event σ".

Kripke structures are introduced and used for formalizing the semantics of the knowledge model. A Kripke structure is a tuple consisting of a set of worlds, an interpretation function that maps atomic propositions to truth values for each world and a possibility relation, one for each agent, which is a binary relation defining those pairs of worlds that are indistinguishable to each agent. Given a world, w, agent i knows the truth of a proposition if that proposition is true at every world in the Kripke structure which is associated to world w in i's possibility relation (is indistinguishable from world w according to agent i).

Given G and S_i for each agent i, the set of worlds and consequently the interpretation function and possibility relation for each agent can be computed.

The control objective considered is that the CP closed-loop control of both agents on

G should equal $\mathscr{L}(K)$. The authors introduce the condition of Kripke-observability, which characterizes that if an event occurrence violates the specification then at least one agent that can control the occurrence knows that it should be disabled. This condition is analogous to coobservability and controllability, but is expressed in the knowledge model discussed thus far. When Kripke-observability fails agents may be able to pool their knowledge (intersect their possible worlds) by communication to increase their respective knowledge to resolve control conflicts. The authors introduce the notion of distributed observability which characterizes that if an event occurrence violates the specification then, by intersecting their possible worlds, the agents can collectively know that the event should be disabled.

In a subsequent work ([67]) the authors provide an algorithm for verifying Kripke-observability. They prove that this algorithm is in $O(|\Sigma_c| \cdot |W|^2 \cdot |I|)$ where W is the set of possible worlds of the Kripke structure. No efficient algorithm is provided for verifying distributed observability though one can exhaustively analyze the set of worlds of the Kripke structure using the possibility relations of agents to determine if it holds.

The authors of [66] continue their work in [67]. The setting is the same. However, they also consider that the actions of an agent may not simply be to disable or enable an event. An agent could also issue a "do not know" action. They also consider that the control actions of an agent could be conditional on the actions of the other agent. Specifically, they consider a conditional action known as conditional disablement ("disable if the other agent does not enable"), which is derived from [115].

They present a generalization of Kripke-observability called inference-observability. When satisfied, it allows controllers to use conditional disablement and "do not know" actions effectively for control purposes. Informally, this condition states that, for every controllable event σ, for every possible world, there exist two agents (not necessarily unique) that can control σ where one of the following holds: (i) σ is not generable from G; (ii) σ is legal; (iii) one of the agents knows that σ should be disabled or that the other agent knows to enable σ.

Agents issue individual control actions based on their knowledge. When an agent knows that an event is legal or is not generable by G then it issues the enable action. When it knows that the event should be disabled it issues the disable action. If it does not know unambiguously that an event should be disabled in all possible worlds indistinguishable from the true world but conjunct (iii) of inference-observability holds (for all possible worlds the event should be disabled or the other agent knows to enable the event) then a conditional disablement action is issued. If even this conjunct does not hold then a "do not know" action is issued.

A table mapping agent 1 and 2 decisions to one final control decision is provided. To implement the control policy of the table individual agent control decisions must be communicated amongst the agents or sent to a coordinating agent where they are mapped to the final control action.

The authors prove that if agent decisions are enable, disable, conditional disable and "do not know", and are fused according to the table provided then inference-observability is the necessary and sufficient condition for there to exist knowledge-based controllers whose closed-loop control on G generates exactly $\mathscr{L}(K)$.

There are generalizations and alternatives to inference-observability that could be explored. One such generalization is where multiple levels of agent inferencing are conducted. An impressive, subsequent work which explores this generalization is [35]. The results of this paper subsume those of [67] as well as other works using control decisions besides just event disablement / enablement and other control decision fusion rules besides just the CP fusion rule (e.g., [113, 115]). While not cast directly in the knowledge theory domain of [25], it effectively considers that an agent's control decision of an event, when ambiguous to a certain degree, can depend recursively on the knowledge of other agents whose knowledge of whether or not the event should be disabled is less ambiguous.

The following is a list of the extensions this work provides over the previous works:

- An agent's control decision can be based on a bounded number of levels of agent inferences.
- Considers conditional enablement decisions in addition to conditional disablement decisions
- The specification is not assumed to be represented as a subautomaton of G.

This work differs from the previous works in that string estimates are maintained by agents rather than state estimates. Also, agents observe events using masks instead of projections.

Following any string $s \in \mathscr{L}(G)$, for each event σ controllable by a given agent i where $s\sigma \in \mathscr{L}(G)$, an ambiguity level can be attributed to the disablement and enablement of σ following s by i. The ambiguity level of a decision is based on whether or not the agent itself is ambiguous about the decision as well as ambiguity levels other agents. The ambiguity levels of a decision for all agents can be greater than 0. An agent can compute its ambiguity level for enablement and disablement decisions for σ by recursive application of its observation mask and the observation masks of other agents which are assumed provided to it at the outset. That is, no communication is necessary for computing its ambiguity level. However, for control decision fusion, communication of agents' ambiguity levels for decisions is required to determine the final control decision.

Following any string $s \in \mathscr{L}(G)$, the decision to enable / disable an event σ where $s\sigma \in \mathscr{L}(G)$ is based on the decisions of agents in $I_c(\sigma)$ whose ambiguity level for σ is a minimum. If the ambiguity level of an agent's decision to enable (resp., disable) an event is less than the ambiguity level of its decision to disable (resp., enable) the event then its control decision is to enable (resp., disable) with its corresponding ambiguity level. If ambiguity levels are equal for disable and control decisions then the agent issues a "do not know" decision. If the agents with minimum ambiguity levels all make the same enablement or disablement control decision then this final control decision is executed. Otherwise, a "do not know" decision is executed. If the minimum ambiguity level of a control decision by agents is N then here we say that N levels of agent inferencing are performed.

The notion of N-inference observability of a specification language $\mathscr{L}(K)$ with respect to $\mathscr{L}(G)$ is introduced, which is a generalization of the inference observability condition considered in [67]: for all strings $s \in \mathscr{L}(G)$, for all controllable events $\sigma \in \Sigma_c$, by performing up to N levels of agent inferencing a final control decision (of the type enable / disable) for σ following s can be issued.

The authors introduce N-inferring decentralized supervisors, where control actions of agents are based on at most N levels of agent inferencing. The closed-loop control of G under such supervisors is defined. The authors prove that N-inference observability is the necessary and sufficient condition for there to exist N-inferring decentralized supervisors that ensure the closed-loop control of G equals $\mathscr{L}(K)$.

The authors describe a procedure for verifying N-inference observability of individual events, $\sigma \in \Sigma_c$. The complexity of the procedure is expensive though no formal complexity analysis is performed.

The authors prove that $\mathscr{L}(K)$ is CP-(resp., DA) coobservable if, and only if, $\mathscr{L}(K)$ is 0-inference observable for disablement (resp., enablement) actions on events. Conditional coobservabilities which are used when agents' actions are conditional on actions of others (e.g., conditional disablement) are also characterized in terms of 1-inference observable languages.

The authors continue this stream of work in [84]. There they demonstrate that N-inference observable specification languages are not closed under union. They also provide an algorithm for computing decentralized controllers which synthesize an N-inference observable sublanguage of a specification language or the specification language itself if it is N-inference observable. Also, in [85] the authors consider synthesizing decentralized controllers which synthesize an N-inference observable superlanguage of a specification language that is not N-inference observable.

In [36] diagnosis is considered. They prove that their work subsumes the work of [56, 107, 108]. Also, they consider failure prognosis similarly in [86].

In a different direction, the knowledge-based approach of [66] is applied in [23] for the synthesis of coordination protocols among controllers. This work differs from the previous surveyed knowledge and inference-based approaches in that:

- A controller's observation is based on its partial view of the state of the system.
- Specifications are transition-based, not language-based.
- Both safety and priority specifications on transitions are considered.
- Considers completely distributed coordination among controllers, not fusing control decisions using a coordinator.

Here a system is considered to be modeled by a DFA, G. The states of G are defined over a set of finite variables, V, each ranging over a finite domain. Transitions in G change the values of subsets of these variables. Each controller observes a subset of the variables in V and can control a subset of the transitions in G. Control is disjunctive-antipermissive.

The closed-loop control of G under a set of controllers is defined. Associated with G is a safety specification, K, on its transitions. Also a priority specification, P, on the transitions is considered. Priorities on transitions are partially-ordered. The control objective to be satisfied is that the controllers must disjunctive-antipermissively control G such that only transitions in the safety specification, K, are executed and, furthermore, the priority of each transition executed is maximal in P among transitions from the state of G.

The authors consider that if any two states in G have the same values for the variables that a controller observes then these two states are indistinguishable by the controller. Under this possible worlds model knowledge can be ascribed to controllers in the same way as considered in the possible worlds model of [66]: the controller knows a proposition is true if it holds in all states indistinguishable with the true state of G. Propositions are defined over maximal priority transitions in G and over the knowledge of other controllers. Specifically, the knowledge properties ascribed to a controller are the following, which are very similar to the properties used in [67] for characterizing inference-observability:

(a) The controller knows that a maximal priority transition whose occurrence it can control is executable by G;

(b) (a) is false and the controller knows that there exists another controller j which knows that a maximal priority transition whose occurrence it can control is executable by G;

(c) (a) is false and (b) is false.

When (a) is true the controller can enable the execution of a maximal priority transition. Otherwise, when (b) is true, the controller does nothing as it knows that another controller will execute a maximal priority transition. Otherwise (when (c) is true), the controller must coordinate with other controllers to refine its set of indistinguishable states (correspondingly, refine its knowledge) until (b) is true or (a) is true. During coordination controllers send to one another the values of the variables in V that they observe. The additional variables are used by a controller to refine the set of states of G it finds indistinguishable.

The coordination problem considered is, for every controller i, determine which other controllers i should coordinate with such that following coordination (b) is true or (a) is true. When this has been determined a control map for each controller can be computed which results in satisfaction of the control objective.

To solve this problem, the authors propose a model checking algorithm. For all controllers, the first iteration determines in which states of G (a) holds. In the second iteration those remaining states of G in which (b) hold are determined. In the third iteration coordination is determined among pairs of controllers for states s where (a) or (b) do not hold for either controller at s but combining their observed variables in V refines the set of indistinguishable states in such a way that (a) holds. That is, by combining their views (a) holds for at least one controller and (b) for the remaining controllers. In subsequent iterations coordination is determined among incrementally increasing sets of controllers. This iteration is performed until, in all states s of G, (a) or (b) holds for every controller after potentially performing coordination with other controllers.

Though the proposed algorithm solves the coordination problem it does not necessarily compute a minimal set of coordinations among controllers. The authors then consider a minimization problem which they reduce to another problem known to be NP-complete ([23]).

The authors also prove that determining the existence of decentralized controllers which do not communicate and can achieve the control objective is undecidable. Before this it was already known that this problem is undecidable for safety specifications ([89, 95]). However, this problem is even more specific as it also regards a priority specification on transitions.

Though controllers can observe transitions in G, reachability analysis using the observations of transitions and knowledge of the transition structure of G could be used to refine the set of states of G a controller finds indistinguishable. However, this is not considered here but is in subsequent works.

The work of [23] is continued in [33]. However, here a 1-safe Petri Net (PN) instead of a DFA is used for modeling the plant and the specification is defined over the reachability graph of the net. The control objective considered is that the controlled execution of the system only executes sequences of state-transition pairs of the reachability graph which exist in the specification and furthermore is deadlock-free.

Controllers observe and control a subset of transitions in the PN. These sets are not necessarily pairwise disjoint. A controller has two local views of the current state (i.e., PN marking) of the system: (1) the assignment of tokens in the state to input / output places of the transitions it observes / controls (this set of places is denoted as the “neighborhood” of the controller), termed *weak knowledge*; (2) the assignment of tokens to a subset of such places that cannot be affected by the execution of transitions outside of those transitions the controller controls / observes (and this set of places is said to be “owned” by the controller), termed *strong knowledge*.

Given a controller’s view of a state s, those states which are indistinguishable with s from the controller’s observation are those states in the reachability graph where the controller’s view is the same. Under this possible worlds model knowledge can be ascribed to controllers as considered before. Specifically, the knowledge properties ascribed to a controller are the same as properties (a)–(c) outlined previously in the context of [23] but where the set of possible worlds is characterized by a controller’s weak knowledge. The actions taken by each controller for each case are also the same.

Specifically, during coordination, controllers send to one another their strong knowledge / assignment of tokens to places that they own. This additional information is used by a controller to refine the set of states in the reachability graph it finds indistinguishable.

In [23] coordination was considered among arbitrary subsets of controllers. Here the authors assume that a partition on the set of controllers is provided. A controller may only coordinate with other controllers in the set to which it has been assigned in the partition. The set is referred to as the controller’s “supervisor”.

The coordination problem considered is similar to [23]. The problem considered is, for every controller i, determine when the controller should coordinate with other controllers in its supervisor set and also which other controllers it should coordinate with such that following coordination (b) is true or (a) is true.

To determine if coordination will work model checking can be conducted to test if coordination with all other controllers in a controller’s supervisor set can refine the set of indistinguishable states in such a manner that (a) or (b) holds in all states where a controller’s weak knowledge is insufficient ((a) or (b) is not satisfied).

In cases where coordinating with all other controllers works, one can consider reducing the number of coordinations and controllers involved in coordinations. The authors propose two different techniques for reducing coordinations. The authors do not claim whether or not these techniques compute coordination policies which are minimal in some respect.

The first technique considers a partial ordering on controllers. Instead of coordinating when a controller's weak knowledge is insufficient to satisfy (a) or (b), the controller first determines if it knows that the combined strong knowledge of all controllers which have order greater than it (according to the partial ordering) in some supervisor set is sufficient to support the execution of one of their transitions in the PN. If so, it does not coordinate with other controllers in its supervisor set. Otherwise, it "hangs" on its supervisor. At any given state in the reachability graph coordination is conducted among controllers in a supervisor (set) for controllers that hang on their supervisor. The weak knowledge of a controller which hangs on its supervisor may change due to the execution of transitions enabled by other controllers which change the assignment of tokens to places in its neighborhood. In such situations the controller may drop out of the coordination if its weak knowledge becomes sufficient to satisfy (a) or (b). The authors assert that under this modification the control objective is satisfied.

In the interest of fairness of participating in coordinations, a problem with the previous technique is that, for any two controllers i and j where $i \prec j$, j will participate in coordinations more often than i. To resolve this, the authors consider that under certain conditions a controller randomly determines whether or not it will hang on its supervisor. Specifically, instead of hanging on its supervisor when a controller does not know that the combined strong knowledge of all controllers which have order greater than it in some supervisor set is sufficient to support the execution of one of their transitions in the PN, it determines if it knows the converse. In this case the controller must hang on its supervisor. When this condition does not hold, the controller tosses a coin to determine whether or not it will hang on its supervisor. The authors assert that this modified controller decision policy also satisfies the control objective.

Though the combined view of strong knowledge of coordinating controllers will not change as transitions are enabled by other controllers and subsequently executed, sets of places in the neighborhoods of coordinating controllers that are not owned by any of the coordinating controllers may only be marked by a strict subset of all possible markings in the reachability graph of the PN. The authors refer to such sets of places as stable properties of the PN. Traps and siphons ([53]) defined over a set of places are particular cases of stable properties. The authors propose a procedure for computing traps and siphons in the neighborhood of coordinating controllers hanging on a supervisor which coordinate their strong knowledge and do not change their strong knowledge. Once such traps and siphons have been identified by the controllers (by some means of additional communication amongst the controllers or amongst the controllers and an entity modeling the supervisor) then the set of possible markings that the traps and siphons can occupy can be included in the coordination allowing one to further refine the knowledge of coordinators than if just the strong knowledge of each coordinator was used.

The authors of [33] continue their work in [52] where they handle the presence of automatically-enabled uncontrollable transitions which was lacking before. In addition to using weak knowledge and strong knowledge introduced in [33], the history of a controller's observations of executions of its transitions in the PN is also used to refine the set of system states that it finds indistinguishable. The set of possible states that a process believes the system could be in (and from which its knowledge is based) upon observing a particular history of transition executions can be computed by performing a type of subset construction on the reachability graph of the controlled PN where transitions not observed by an agent are replaced by the empty string.

Given a forbidden state-transition specification defined over the reachability graph, an extended, state-based version of the specification can be computed. In the controlled execution of the PN a state in F must not be entered for otherwise it is possible that an execution of a sequence of transitions may result in the state-transition specification being violated. The controlled PN can be defined as a subgraph of the uncontrolled reachability graph which does not have transitions to / from states in the state-based specification. The authors assert that computation of the controlled PN is exponential in the number of places of the plant PN and furthermore is EXPTIME-complete.

Once the controlled PN has been computed, model checking of knowledge properties (same as (a)–(c) in [23]) is done to determine when processes know enough based on their view of the system that a transition that they control can be supported or if they know that one of the other processes knows that it can support one of its transitions. The authors assert that model checking of such properties is in EXPTIME.

8 Approaches to Computing Distributed Communicating Controllers from a Monolithic Controller

An alternative means for constructing distributed communicating agents is to first compute a monolithic, centralized solution to an observational problem then to construct distributed agents and a communication protocol amongst the agents which effectively implements the centralized solution. This is the approach taken in the works of [47, 15, 16].

In [47] a regular plant, G, defined over alphabet Σ and prefix-closed, controllable and observable specification language, $\mathscr{L}(K) \subseteq \mathscr{L}(G)$, with respect to $\Sigma_o \subseteq \Sigma$ and $\Sigma_c \subseteq \Sigma$ are given. Since $\mathscr{L}(K)$ is controllable and observable a centralized controller S may be computed such that $\mathscr{L}(S/G) = \mathscr{L}(K)$. The authors consider the problem of distributing S over a set of communicating agents $I = \{1, ..., n\}$. For each agent $i \in I$ a set of controllable events $\Sigma_{i,c} \subseteq \Sigma_c$ and observable events $\Sigma_{i,o} \subseteq \Sigma_o$ are assumed given.

In most works events, state estimates or string estimates are communicated between agents. This work introduces a novel means of communication. Here Boolean variables encode the states of a centralized controller which are used for defining guards on when events should be enabled. Boolean variables X_i are owned and updated locally by an agent $i \in I$ according to event observations in $\Sigma_{i,o}$ and are communicated asynchronously to other agents $j \in I$ if j needs to know the value of variables in X_i to determine whether or not an event $\sigma \in \Sigma_{j,c}$ should be disabled (i.e., if the guard condition on σ is not satisfied).

To distribute the centralized controller over the set of I agents, an alternative representation of the centralized controller, denoted by automaton S', is computed from S which, in the worst case, has state cardinality twice the size of the state cardinality of S. Afterwards, the states of S' are labelled by vectors of n natural numbers. The natural number stored at the i^{th} position of a vector v labelling a state of S', denoted by $v(i)$, is the i^{th} agent's label. The set of label schemes mapping states of S' to vectors of agents labels considered are referred to as Agent-Wise Label Maps (ALMs).

The authors demonstrate how a specific type of ALM can be computed for S'. Given an ALM, the number of different values for $v(i)$ can be represented by a set of Boolean variables, X_i, which are owned by agent i and can be used instead to label states in S'. Guard conditions over the Boolean variables owned by all agents can then be defined for characterizing when events should be enabled as well as when a Boolean variable owned by an agent should be updated on event occurrences observed by the agent. Guard conditions can be defined over a minimal set of variables by applying "don't care" conditions ([48]). Agents will be required to know the most up-to-date values of Boolean variables in the set of variables which guard conditions are defined over to determine if an event should be enabled or if it needs to update one of its Boolean variables. In the former / latter case we say that communication is required for i's control / observation.

Though communication may be required for an agent to make a control decision or update its local Boolean variables, the authors only consider that at the level of message passing communication occurs asynchronously. In some cases if agents poll other agents for the values of their Boolean variables then communication may be reduced.

The authors then study the class of ALMs where no communication is required for an agent's observation. That is, updating an agent's local Boolean variables does not depend on the values of Boolean variables owned by other agents. However, for such ALMs, communication may be required for an agent's control. A condition stronger than observability but weaker than unbounded-memory joint observability of [96] referred to as modified joint observability is introduced. The authors prove that if there exists an ALM for S' where communication is not required for observation then $\mathscr{L}(K)$ is necessarily modified joint observable.

The authors then provide an example where they demonstrate that only two Boolean variables are communicated using their approach whereas two states must be communicated using the estimator approach of [2] where communication is state-based. Their approach results in fewer bits being communicated in such a case. However, they do not provide an exhaustive comparison of their work with [2]. It is difficult to determine how the approach of this work compares in general with the approaches of other works on communication in DES.

The works [15] and [16] present an approach for constructing a monolithic PN controller S such that, given G, regular specification sublanguage $\mathscr{L}(K)$:

(O1) $\mathscr{L}(S/G)$ is equal to the supremal controllable and observable sublanguage of $\mathscr{L}(K)$ when such a language exists;

(O2) in the case where the supremal controllable and observable sublanguage of $\mathscr{L}(K)$ does not exist or cannot be enforced by a monolithic PN controller (the class of which is strictly weaker than finite state controllers) $\mathscr{L}(A) \subseteq \mathscr{L}(S/G) \subseteq \mathscr{L}(K)$ for some given minimal acceptable behaviour $\mathscr{L}(A)$.

A method for distributing the monolithic controller over n agents modeled by message-passing automata is presented when the controller is a bounded PN for strings generated by G and a condition referred to as *distributability* is satisfied.

Petri Net controllers are defined. For every event generable by G there exists only one transition in the net modeling the event occurrence. This is extremely restrictive. It is required that (C1) unobservable event occurrences (transitions) do not change the marking of the net, and (C2) for any sequence of transitions in the PN that corresponds to a string in $\mathscr{L}(G)$, the number of tokens in any place of the net is bounded. Customarily, the closed-loop control of a net controller S on G, denoted by S/G, consists of those strings generable by both the net S and G.

The author describes a general synthesis algorithm for computing net controllers which satisfy (O1) or (O2). The algorithm is an iterative one which proceeds until a bounded net representation is obtained where unbounded net representations are computed at intermediate iterations. In the worst case the algorithm executes $2 \cdot |\Sigma| + 1$ iterations before it terminates. No complexity analysis is provided.

The computed net controller can be distributed into a set of n individual PNs whose reachability graphs are asynchronous message passing automata (AMPA) if it satisfies the condition of distributability: every event in the net is associated with one distributed location / agent and every place in the net only provides input to transitions associated with the same location.

A merit of this work is that bounded, distributable nets considered belong to a class of PNs known as the free-labeled and bounded PNs and that such PNs can be synthesized efficiently and produce compact representations of controllers ([22]).

There are many similarities between this approach for distributing a centralized controller with the approach of [47]:

- Communication is asynchronous.
- The encoding of the information to be exchanged between agents is automated. That is, in both works, it is not defined *a priori* what type of information will be communicated among agents.

However, the approach does have many limitations compared with [47]:

- An event can only label a single transition in the PN representation of the net controller from which AMPA are derived.
- For distributization purposes, it is assumed that any two agents' set of controllable (resp., observable) events are pairwise disjoint. This is not required in [47].
- If an event is associated with a given location, then the input places of its corresponding transition are also exclusively associated with the location.
- The approach is only applicable when bounded, distributable net controllers exist for satisfying (O1) or (O2). Such class of PNs are a strict subclass of finite automata. So it is possible that a finite-state controller may exist for satisfying (O1) or (O2) but no bounded, distributable PN exists for satisfying one of the objectives. By contrast, the approach of [47] is applicable to any finite-state controller.

- The author themself states that "the naïve algorithm we used to expand a distributable PN into a [expanded] PN results in high cost of communications".
- The author remarks in [16] that this approach cannot be applied when the control objective is avoiding deadlocks (more generally, nonblockingness must be satisfied). Extending the approach so that nonblockingness is satisfied is future work. The approach of [47] can be applied to compute distributed versions of finite-state nonblocking controllers when they exist.

9 Approaches to Computing Minimal Communication Policies

In this section we focus on approaches for computing inter-agent communication protocols. We concentrate on when observations of the plant and communications made between controllers are based on event occurrences labelling transitions of a DFA representation of the plant. We also concentrate on approaches for the computation of communication protocols that satisfy a set-theoretic notion of minimality.

One of the first works on transition-based communication in DES is [70]. This work considers two agents, associated with which are DFA modeling the actions of the agent. The DFA model could be a supervisory controller, diagnoser, or other application-dependent DFA agent model, depending on the specific problem at hand. A plant for generating events is assumed to exist but is not modeled explicitly. Each agent's DFA model can have transitions between different states labelled by events not directly observed by the agent. The objective of introducing communication between the agents is that they are each able to determine precisely which state their respective DFA representation is in after any string generated by the plant. Observed event occurrences are communicated between the agents reliably, without delay and immediately upon their observation. The assumption is that without communication each agent cannot unambiguously determine the state of its DFA representation and that if all observed event occurrences are communicated between agents then they can.

In addition to computing a communication protocol between the agents that allows them to determine the states of their respective DFA representations, it is also required that the communication protocol be minimal. Minimality is defined in the following sense for a communication protocol: after any string generated by the plant, if any fewer events are communicated from one agent to another then either: (a) one of the agents cannot properly distinguish which state its respective DFA model is in; (b) one of the agents cannot unambiguously determine whether or not it should communicate an event to the other agent; (c) a structural requirement on the communication protocol is not satisfied. In the setting of this paper two communication protocols may be incomparable. This arises from the fact that following any string generated by the plant the set of events to be communicated from one agent to another, upon their occurrence, may be incomparable. Thus no minimum communication protocol exists between the two agents and so only minimal communication protocols can be considered.

To compute minimal communication protocols, a structural requirement is considered which restricts attention to a specific subset of communication protocols. Communication protocols are to be derived from the (synchronous) product of the two agents' DFAs, denoted here by R. The subset of communication protocols considered must satisfy the following structural requirement: if any two strings lead to the same state in R then the decision to communicate an event from one agent to another must be the same following both strings. Communication protocols satisfying this requirement are said to be *implementable* with respect to R. If a communication protocol satisfies the implementability requirement then it can be characterized by a subset of transitions of R.

In addition to the structural implementability requirement another requirement is introduced: if any two strings generated by the plant appear identical to an agent due to the observation of event occurrences in its observable event set and due to event occurrences which are communicated to it by the other agent, then the decision to communicate an event to the other agent must be the same following both strings. Informally, this requirement states that, after the observation of any string of events, an agent should unambiguously know exactly what events it will communicate to the other agent upon their next occurrence. Communication protocols satisfying this requirement are said to be *feasible*.

Apart from the procedure introduced for computing minimal communication protocols, the modeling requirements of implementability and feasibility of communication protocols are arguably the primary contributions of this paper and are used for modeling purposes in subsequent works on transition-based communication and sensor activation in DES.

There is an apparent difficulty in this communication problem (in fact, in most communication problems) due to the mutual dependency of observation and communication: what agent 1 communicates to agent 2 affects agent 2's observation which, in turn, affects what agent 2 communicates to agent 1 which affects agent 1's observation which, in turn, affects what agent 1 communicates to agent 2, etc. Given this mutual dependency, a solution approach is to alternate between adding communications from agent 1 to agent 2's set of essential communications (communications that agent 2 requires from agent 1 for precisely identifying its state) and from agent 2 to agent 1's set of essential communications until feasibility of the resulting sets of communications is satisfied. Alternation may occur in multiple successive iterations, bounded by the transition cardinality of R. However, interestingly enough, the algorithm presented by the authors is asymmetric and only requires one iteration. Unfortunately, there are some issues with the algorithm. It is exponential in the state and transition cardinalities of R. Also, the algorithm provided cannot be used to compute all possible minimal communication protocols satisfying the implementability and feasibility requirements.

Another algorithmic approach one may conjecture could be used is to begin with each agent communicating all event occurrences to the other then iteratively remove communications from one agent to another until removing any communication (besides the essential communications) violates feasibility. Unfortunately, such an approach will not work in general. By further removing communications from sets of communications computed by such a procedure feasibility may become satisfied! This is due to an interesting property of transition-based communications: in some instances, by communicating less any two strings which were previously indistinguishable by the other agent can become distinguishable. This counterintuitive phenomenon is referred to as “lack of monotonicity” in [99] where a more detailed examination and example illustrating it is provided.

Due to the computational expense of the algorithm considered, the restriction that each agent’s DFA have total transition functions, that communication of events be reliable and instantaneous, that only two agents are considered, and that only a strict subset of the minimal communication protocols can be computed, the applicability of the approach of [70] is limited.

A generalization of [70] is provided in [45]. The problem considered is the same as in [70], but instead of introducing communication for purposes of each agent distinguishing exactly which state its DFA is in following any string generated by an underlying plant, a more general goal needs to be satisfied where each agent is required to have communicated to it any transition which exists in some set of essential transitions labelled by events observable exclusively by the other agent and defined over the set of transitions of its DFA. It is still required that communication protocols considered are feasible and implementable with respect to the synchronous product of the agents’ DFAs (R).

The algorithm considered for computing a minimal solution is a naïve and exhaustive one. The advantage of the algorithm provided is that it is capable of computing all minimal, feasible, implementable communication protocols where all of the essential transitions of an agent’s DFA are communicated to it. However, the complexity of the algorithm is in the worst-case exponential in the state and transition cardinalities of each agent’s DFA representations.

The setting of [70] and [45] are used to study a problem of minimal communication in [104]. This work considers $|I| \geq 2$ agents and an explicit DFA representation of the plant, G, is provided. Agents observe the transitions of G and are required to communicate observed event occurrences to one another to disambiguate certain pairs of states in G. However, there are several restrictions on the problem setting: the agents collectively can observe all event occurrences generated by G; the agents can only communicate bidirectionally with a coordinator; the coordinator observes all event occurrences generated by G; G is acyclic besides self-loops on its states.

Under the above restrictions, the authors provide an algorithm for computing feasible, minimal communication policies between each agent and the coordinator which is of polynomial complexity in all parameters: number of states of G, Σ, and I.

The solution procedure starts by fixing a minimal set of communications from each agent to the coordinator which guarantees that the coordinator observes all event occurrences. This asymmetry in the solution procedure is similar to the solution procedure of [70]. It remains to compute feasible, minimal sets of communications from the coordinator to each agent that allows each agent to satisfy its state-based specification.

Unfortunately, the lack of monotonicity phenomenon also occurs here. In spite of this difficulty, the authors provide an efficient algorithm for computing minimal sets of transitions to be communicated from the coordinator to each agent. The algorithm relies on computing a partition of the set of strings defined from states of G processed recursively from states with only self-loops on events to the initial state. The authors prove that the algorithm is in $O(|I| \cdot |\Sigma|^3 \cdot |X|^3)$ where X is the state set of G.

Besides the severe restrictions on the problem setting, one drawback of this work is that, for a given assignment of communications from the agents to the coordinator, the computation of a minimal communication policy from the coordinator to any agent is uniquely determined. It is not possible to compute other minimal communication policies from the coordinator to agents.

The previous work is generalized in [103] for the computation of completely distributed communication protocols where agents can communicate with each other without the presence of a coordinator. Here it is assumed that communication topologies are strongly connected. Like the previous work, the plant, G, is assumed to be acyclic except self-loops on states. Events are communicated between agents without delay, even if communication must pass indirectly (due to topology limitations) through a set of intermediate agents.

To solve the minimal communication problem the authors present a generalized version of the algorithm used in [104]. Due to the communication topology being a general strongly connected one where the communications of any agent can affect the observations of any other agent, this problem does not enjoy the same optimization based on a language partition as that considered in [104]. We have determined that the complexity of the algorithm is in $O(|X| \cdot 2^{|\Sigma| \cdot |I| \cdot (|I|-1)} \cdot (|X|^2 \cdot |\Sigma|))$ (but this bound may not necessarily be tight). As a result we can see that, even in the context of acyclic systems with general strongly connected communication topologies, the problem of computing minimal communication policies is difficult.

Minimal state-based communication is studied in [65] following the preliminary versions of [2] and [70]. It studies asynchronous communication between two agents for controlling G to satisfy exactly $\mathscr{L}(K)$.

Here it is assumed that $K \sqsubseteq G$. As in [2] agents locally observe strings generated by G using a projection and maintain state estimates of G. Control decisions are based on an agent's state estimate. Control conflicts exist for an event σ that agent $i \in \{1, 2\}$ controls when i's state estimate contains a state of K from which σ labels a transition to another state in K and i's state estimate also contains a state of K where σ labels a transition to a state in G outside of K. When control conflicts exist agents may communicate state estimates to one another to refine estimates and resolve such conflicts. This is the problem considered in this work.

Analogous to previous works, the authors prove that communication of state estimates can eliminate control conflicts if $\mathscr{L}(K)$ is observable with respect to $\mathscr{L}(G)$, Σ_o, Σ_c. Specifically, the authors prove that if $\mathscr{L}(K)$ is observable then, whenever agent i cannot make the correct control decision about $\sigma \in \Sigma_{i,c}$ following the observation of a string in G, agent j ($j \in \{1, 2\}, j \neq i$) can communicate its state estimate at prefixes of the string which will allow agent i's control conflict to be eliminated. The authors identify such prefixes for pairs of strings causing control conflicts for an agent. Communication of state estimates from states in G reached by such prefixes can be used to resolve control conflicts. Communicating from all such states is in some cases not necessary and so minimal communication is considered.

It is required that agent communications be feasible ([70]). Given a set of communication states of an agent for resolving control conflicts, the set of states which are indistinguishable with these states by the agent can be computed from which communication must also occur to satisfy feasibility.

The notion of minimal communication considered is that if any single communication of a state estimate from one agent to another is removed then either there exist control conflicts that are not resolved or feasibility of communications is not satisfied. An algorithm is proposed which takes as input a set of communication states of G from agent i to j for resolving control conflicts as well as the set of additional communication states from which communication is required to satisfy feasibility according to i's observations. The algorithm computes a subset of communication states from the input set which the authors claim satisfy minimality, feasibility and resolve i's control conflicts.

However, there are some drawbacks to [65]. The specification is required to be expressed as a subautomaton of the plant automaton. Also, the proposed algorithm for computing a minimal set of communication states only considers that communication occur as early as possible. This restricts the set of possible solutions considered. Alternative solutions are where communication occurs as late as possible (as considered in [2]) and everywhere in between communicating as early and as late as possible. An initial proposal for exploring alternative solutions in this range of communication solutions is [63].

In [41] reactive DES are considered. These are systems where the plant can be decomposed into a set of component DFA which interact with each other and an unpredictable environment. Interactions between the components and the environment / other components and the operation of components are modeled by events. Interactions can be point-to-point or broadcast. The overall plant is modeled as a type of synchronous product where synchronization occurs on events modeling interaction between the components.

It is required that plant components be controlled in such a manner that any interactions between the components and the environment satisfies exactly a regular language specification defined over the set of input-output events modeling interactions between the components and the environment. Controllers are associated with components to achieve this. A controller controls and observes events generated and used internally by its associated component, events modeling initiation of an interaction by its associated component, and observes events modeling initiation of an interaction with its component. It is assumed that controllable event sets of components are pairwise disjoint.

The authors prove that the usual controllability and coobservability conditions on the specification language are not the necessary and sufficient conditions for there to exist controllers associated with each component such that the specification language is synthesized in the closed-loop control with G. The authors introduce conditions of P_{ex}-controllability, P_{ex}-observability and P_{ex}-coobservability on the specification language. They demonstrate that P_{ex}-controllability, P_{ex}-observability are instead the necessary and sufficient conditions.

When G is P_{ex}-observable but not P_{ex}-coobservable then communication between controllers can be incorporated for solving the control problem. The authors consider the "communicate-as-early-as-possible" strategy of [65]. Communication of an event occurrence from one controller to another is handled differently depending on the event (e.g., internal, input, or output event) which, when communicated, will remove a violation of P_{ex}-coobservability.

The authors then consider minimal communication in the same sense as [65]: a communication policy is minimal if it is feasible, the control problem can be solved and removing the communication of any single event occurrence violates feasibility or violations of P_{ex}-coobservability (i.e., control conflicts) are introduced (i.e., the control problem cannot be solved). They present a greedy algorithm similar to that of [65] for computing minimal communication policies. This algorithm assumes that all violations of P_{ex}-coobservability when controllers do not communicate are provided as input and that a map from event occurrences labelling transitions in component DFA to violations of P_{ex}-coobservability that are removed upon communication of the event occurrence to the appropriate controller is provided as input.

The work of [64] considers computation of asymptotically minimal inter-agent communication policies. Specifically, it is the objective to minimize the number of event communications along infinite strings of G. For strings of bounded length, the procedure used to compute communication policies may not generate communication policies which satisfy any of the notions of minimality considered previously in this section. However, the author asserts that, for strings of infinite length, the policies generated do satisfy a notion of optimality in terms of asymptotic throughput.

The problem of finding minimal communications is reduced to an optimization problem for a set of Markov chains. The solution approach proceeds first by considering $K \sqsubseteq G$. The author illustrates how to compute a finite state-transition structure which recognizes violations of coobservability (i.e., control conflicts). Denote this structure by $\mathcal{U}$.

From $\mathcal{U}$ sets of transitions which, when forced to be communicated among controllers, avoid violations of coobservability can be defined. Using these sets of transitions "admissible" DFA can be defined. The authors demonstrate how to compute a unique, feasible communication policy satisfying coobservability from a given admissible DFA.

However, such a communication policy may not necessarily be minimal with respect to strings of infinite length. The authors propose an approach to evaluate the mean asymptotic throughput for policies which are uniquely associated with admissible DFA. The approach consists of reducing each admissible DFA to a Markov chain induced on a Markov model, evaluating the asymptotic throughput over an infinite horizon, then selecting the admissible DFA which corresponds to the chain with lowest cumulative cost.

The work of [73] lies in the same setting as and follows [64]. It applies game theory to compute minimal communication protocols. Depending on the nature of G, different automata may be used in computing a solution. If G is acyclic then it is used. Otherwise, an automaton isomorphic to the $\mathcal{U}$ structure of [64] is used.

For computing a minimal communication policy, the authors use an algorithm for computing a Nash equilibrium (introduced in [50]) of communications among an arbitrary number of agents. They formulate the communication problem as a normal-form game and define a Nash equilibrium for this game. The Nash equilibrium is essentially an inter-agent communication protocol which has logically minimal cost, is coherent (analogous to feasibility) and produces observations in such a manner that the control problem is solved. Nash's result asserts that "every normal-form game has at least one Nash equilibrium" ([50]). Hence there exists a Nash equilibrium which represents a solution for this minimal communication problem.

The authors then leverage an algorithm referred to as the "Support Enumeration

Method" published in [54] for computing a Nash equilibrium in the context of the normal-form game posed. The authors assert that the algorithm described is in $O(2^T)$ where T represents the number of transitions in the plant representation (G or the $\mathcal{U}$ structure, depending on the cyclicity of G). The authors assert that for acyclic G, using uniform cost for all communications is appropriate. However, they assert that more appropriate cost models need to be used in the cyclic case.

Finally, the application of evolutionary algorithms to compute approximately optimal solutions for a quantitative cost control & communication minimization problem involving any number of agents where cost is associated with event enablement, event disablement and with event communication is considered in [74]. Evolutionary algorithms are known for being able to compute approximate optimal solutions for solving problems where exhaustive search is computationally prohibitive and multiple, conflicting constraints need to be optimized. Control and communication are two such constraints: to minimize the number of event occurrences disabled controllers must typically (presuming the more a controller sees the better its distinguishing power) communicate more and, on the other hand, to minimize communication controllers must typically disable more event occurrences.

Communication is reliable, instantaneous and point-to-point. No specific communication topology is assumed. Controllers maintain string estimates of G. An agent's control and communication maps are assumed to be string-based. It is assumed that G is acyclic. When considering cyclic G, the cost functions considered should calculate the mean cost over an infinite horizon. When this is the case the approach of [64] can be used for computing the cost of an inter-agent communication policy.

The association of cost with event enablement, disablement and communication is inspired by the centralized control cost function of [77]. Any cost function from an agent's actions on individual event occurrences following a string in $\mathscr{L}(G)$ to the real numbers can be considered. It is preferred that control & communication maps used by controllers are defined such that the closed-loop control on G synthesizes exactly a regular specification, $\mathscr{L}(K)$. However, the minimization problem considered permits minimum-cost control / communication maps which, when used, results in a nonempty strict subset of $\mathscr{L}(K)$ being synthesized. To discourage over-restricting plant behaviour, a user-defined penalty can be applied to increase the cost of those disablement actions where the string estimate maintained by the controller disabling a particular event occurrence, σ, contains a string t where $t\sigma \in \mathscr{L}(K)$.

Assuming $\mathscr{L}(G)$ is finite, an agent's control cost is simply the sum of the costs of its event enablement and disablement decisions following all strings in $\mathscr{L}(G)$. The communication cost is defined similarly. Though communication is point-to-point if more than one controller requires observing an event occurrence communicated by controller i then i only incurs the cost of communicating the event occurrence to a single recipient.

In addition to computing control & communication maps which synthesize behaviour in $\mathscr{L}(K)$, it is also required that communication policies be feasible. The problem considered is, given some function f from control and communication actions of all controllers to the real numbers, compute control and communication maps which satisfy these aforementioned criteria and, furthermore, is minimum with respect to f.

A modified version of the Non-Dominated Sorting Genetic Algorithm - II (NSGA-II) (introduced in [17]) is applied for computing approximate optimal solutions to this problem. Though other evolutionary algorithms could be used (Strength Pareto Evolutionary Algorithm of [118], Pareto-Archived Evolution Strategy of [34]), the authors assert that they use NSGA-II as good solutions from previous generations computed by the algorithm are preserved (i.e., all children of generation k compete for membership in generation $k+1$) and because the algorithm features a strong fitness assignment procedure.

However, the authors do not propose what properties the initial generation or individual candidates within the initial generation should have. They also do not propose mutation or crossover operators and do not detail any fitness function.

10 Transition-Based Observation Conditions

The work in the previous section focused primarily on transition-based observations. In [29] the authors study a transition-based version of coobservability. Its study is continued [100] where additionally a relation with a transition-based version of codiagnosability is established.

A decentralized control problem involving two controllers whose observation of G depends on the transitions of G is studied in [29]. Transition-based observation maps for each controller are assumed given. That they are provided as input to the problem is in contrast with the previous surveyed works on transition-based observations and communication where the objective is to compute transition-based observation maps.

The authors introduce the notion of transition-based coobservability of a specification language $\mathscr{L}(K)$ with respect to G and observation maps $P_{1,G}, P_{2,G}$ defined over the transitions of G. This notion is simply a generalization of coobservability where observation maps defined over the transitions of G are used to capture the observations of each controller rather than natural projections.

Controllers are assumed to be modeled using DFA with a map from states to control decisions (enable / disable) on events. The CP control of G by two controllers ($S_1 \wedge S_2/G$) is defined as usual. Given G, $K \sqsubseteq G$, $\Sigma_{1,c}$, $\Sigma_{2,c}$ and transition-based observation maps $P_{1,G}, P_{2,G}$, the authors assert (with proof provided in [28]) that transition-based coobservability and controllability is necessary and sufficient for there to exist two controllers S_1, S_2 which control events in $\Sigma_{1,c}, \Sigma_{2,c}$ and observe G using $P_{1,G}, P_{2,G}$ such that $\mathscr{L}(S_1 \wedge S_2/G) = \mathscr{L}(K)$.

The authors then describe an algorithm for verifying transition-based coobservability. The algorithm proceeds by construction of a modification of a so-called "M-machine", an NFA which was originally introduced in [71] for verifying coobservability. We have determined that the construction of the M-machine is in $O(|X_K|^3 \cdot |X_G| \cdot |\Sigma|)$ (but this bound may not necessarily be tight) where X_K (resp., X_G) is the state set of K (resp., G) (note that the authors assume $K \sqsubseteq G$).

The authors then describe an iterative algorithm for making transitions of G observable to controllers to satisfy transition-based coobservability for given input transition-based observation maps and controllable events of each controller. No complexity analysis is provided for the algorithm in this paper or in the thesis [28] where it was originally defined, though we have determined that an upper bound is $O(|X_K|^5 \cdot |X_G| \cdot |\Sigma|)$ (but this bound may not necessarily be tight), which is polynomial in X_K, linear in X_G and Σ.

The results of [100] subsume the verification procedure based on M-machines of [29] when an arbitrary number of agents is considered. This work also considers decentralized diagnosis problems where observations are transition-based. The authors provide a polynomial-time reduction in the state and event cardinalities of the plant DFA G from the problem of verifying coobservability (resp., observability) to the problem of verifying codiagnosability (resp., diagnosability). This result leverages the approaches of many works where sensor activation and communication protocols are computed for purposes of satisfying (co)diagnosability to similar problems where (co)observability is to be satisfied.

Specifically, the reduction presumes that $K \sqsubseteq G$ and observation maps for controllers are defined over the transitions of K (not G). The authors prove that the algorithm for transforming problems of coobservability to problems of codiagnosability is in $O(|X| \cdot |\Sigma|^2)$ where X (resp., Σ) is the state (resp., event) set of G.

The authors demonstrate that, for a problem of minimizing transition-based sensor activations (a topic to be covered in the next section) in satisfying coobservability in cases where communication is not required between controllers, if the aforementioned transformation from a problem of coobservability to a problem of codiagnosability is applied then if observation maps (more specifically, sensor activation maps) for agents are optimized for satisfying the codiagnosability problem then the same observation maps are optimal for satisfying the coobservability problem. This result leverages the sensor activation optimization algorithms of [12] or [102] for problems of (co)diagnosability to problems of (co)observability.

In addition to providing a reduction from coobservability to codiagnosability, the authors also provide a method of verifying transition-based codiagnosability. For a given G with state set X, set of agents I with respective transition-based observation maps and set of fault event types F, the authors prove that the complexity of the verification procedure is in $O(|F| \cdot |X|^{|I|+1})$.

A previous algorithm for verifying diagnosability (not codiagnosability) with dynamic observations was provided in [12]. However, this algorithm does not require that observation maps be restricted to the transitions of G. Instead, a deterministic, finite transducer representation of the dynamic observation map is required as input. Using the complexity results of this work, we have determined that the complexity of this algorithm is in $O(|F| \cdot (|X| \cdot |Y|)^2)$ where Y is the state set of the transducer. So in cases where observations are transition-based, the algorithm of [100] is preferable in the worst-case.

From the algorithm of [100], the complexity of verifying coobservability in the case of two agents with $C \subseteq \Sigma$ controllable events is the cost of the transformation and the cost of the verification procedure provided for codiagnosability: $O(|X| \cdot |\Sigma|^2 + |C| \cdot |X|^3)$. Until tighter complexity bounds on the verification procedure of [29] are established, it is not clear when the verification procedure of [29] is preferable over the verification procedure of [100] and vice versa. However, from the strict upper bound provided for [29], it is likely that this depends on the cardinality of Σ.

The last known approach for verifying transition-based coobservability is provided in [51]. Here the authors reduce the verification of transition-based coobservability to verification of an observational condition called *decentralized no-opacity*. When $\mathscr{L}(K)$ is regular, the complexity of the reduction and verification using their approach is in the worst-case linear in the number of agents, which contrasts with the reduction and verification algorithm of [100] which is exponential in the number of agents. In the two-agent case the choice of using the transformation and verification procedure of [51] or the verification procedure of [29] for transition-based coobservability depends on K, $\mathscr{L}(G)$, Σ and Σ_c. Also, here it is not assumed that the specification language $\mathscr{L}(K)$ is regular whereas in [100] and [117] $\mathscr{L}(K)$ is taken to be regular.

11 Approaches to Solving Sensor Activation Problems

Up until now we have focused on communication as the only means under which an agent's observation of a plant may change dynamically. However, recent work has considered sensor activation problems. Here an agent has sensors associated with events that it can observe locally. When a sensor is active occurrences of the event to which it is associated are observed by the agent. Otherwise, they are not observed. In such problems it is assumed that agents have control over their own sensors and not over the sensors of other agents. Problems of sensor activation are, for analytical and implementation purposes, simpler than problems of communication. Analytically they are simpler as agents have complete control over their own observations. They are simpler for implementation purposes as no communication channels between agents are required.

Problems of sensor activation are different from problems of sensor selection which have already been considered in the literature (e.g., [24, 114, 116, 68]). Sensor selection concerns computing a set of observable events having minimum cost / cardinality where observational properties such as observability or diagnosability are satisfied. Sensor selection problems for diagnosability, observability and normality (a property that is stronger than observability) have been proven to be NP-complete ([114]).

When event occurrences have non-uniform cost computing minimum-cost sensor activation policies is also expensive ([92, 12]). However, when event occurrences have uniform cost there exist efficient algorithms for computing solutions to sensor activation problems ([106] and related works).

The first work to consider sensor activation in the domain of DES is [92]. This work considers centralized control (resp., diagnosis) of systems modeled by automata, timed automata and stochastic automata. We restrict attention to logical / untimed automata. A centralized agent is considered which maintains a string estimate of $\mathscr{L}(G)$. The determination of which sensors to activate is based on the string estimate. A cost function for activating event sensors is given. The problem considered is to compute a minimum-cost sensor activation map which satisfies observability (resp., diagnosability).

The authors consider the cases of acyclic and cyclic automata separately. For both cases dynamic programming algorithms are provided for computing a solution. However, these algorithms are expensive due to the fact that string estimates are maintained. In the acyclic case, the dynamic programming solution is implemented over a state set which is in the worst case doubly-exponential with respect to the maximum length of strings recognized by the system DFA. In the cyclic case, the solution is in the worst case exponential in the state set of the system DFA.

The authors also consider computing suboptimal solutions using similar dynamic programming solutions where a limited lookahead window is used to prune the state sets. The algorithms are in the worst case doubly-exponential with respect to the lookahead window size. No results or claims are provided for quantifying the cost of the computed suboptimal sensor activation maps or how the cost relates to the cost of minimum solutions.

Algorithms to compute sensor activation maps for a centralized diagnoser to diagnose faults following k event occurrences in a system modeled by an NFA, G, when G is k-diagnosable are presented in [12]. Cases where both cost is / is not associated with active event sensors are considered. When cost is considered, an algorithm to compute a minimum-cost diagnoser is presented. This problem is similar to the one considered in [92]. However, it differs in three ways:

1. k-diagnosability is addressed, not just diagnosability.

2. Sensor activation decisions are modeled by a finite state transducer (FST), which has only bounded memory for recording observations. As such, only state estimates rather than string estimates can be maintained by an agent.

3. Cost is associated with the activation of an event sensor as well as the duration (number of event occurrences) for which the sensor is active, not just activation of the sensor.

The algorithm for computing a minimum cost sensor activation map that enables a diagnoser to diagnose all faults after k event occurrences is proven by the authors to be in $O(|\Sigma| \cdot |T| \cdot 2^{|X|^2+2^{|\Sigma|}+k})$ where X is the state set, T the transition set and Σ the event set of G. In the case of cyclic G, this approach is more expensive than [12] for computing a minimum cost sensor activation map. The approach relies on computing all sensor activation maps that enable diagnoses of all faults after k event occurrences. The algorithm presented to do this is proven by the authors to be in $O(2^{|X|^2+2^{|\Sigma|}+k})$.

Also, given an FST representation of a sensor activation map, denoted here by D, an algorithm for deciding if G is diagnosable with respect to D is provided. One can verify that the algorithm is in $O(|F| \cdot (|X| \cdot |Y|)^2)$ where F is the number of fault events and Y the state set of D. An algorithm for computing the minimum k such that G is k-diagnosable with respect to D is also provided. As the authors prove that this algorithm is of the same complexity as verifying if G is diagnosable with respect to D, one can verify that it is in $O(|F| \cdot (|X| \cdot |Y|)^2)$.

After having considered problems of communication which inherited features from [70] the authors of [104, 103] have since turned their attention towards problems of sensor activation in DES in [106]. Here the specific objective of control is considered.

The problem and solution approach in [106] primarily differs from [92] and [12] in that

1. The notion of an optimal sensor activation map is logical, not numerical.

2. Sensor activation maps can be computed in polynomial time in all input parameters.

It is assumed that $K \sqsubseteq G$. The authors first consider a centralized control problem. It is assumed that if the centralized controller observes all of its observable event occurrences then, for any string s generated in K, the set of strings which yield the same observation as the observation of s according to the controller's observation map are all followed by the same control decision for any event controllable by the controller. State-based (equivalent to transition-based) observation maps for the centralized controller are introduced which map from states of G to a subset of the controller's observable events which label outgoing transitions from that state. These maps are referred to as sensor activation maps. For a given state, if an event exists in the set mapped to by the state-based observation map for the state then we say the sensor is activated for that event at the state. If the sensor is activated for an event then the controller will observe the event upon its occurrence.

From K a state-based specification condition is derived which characterizes observability. Specifically, it is a set containing all pairs of states where there exists a control conflict with respect to the controller's set of controllable events. To satisfy the specification, the controller needs to activate sensors dynamically in such a manner that no two states in any state pair of the specification condition be indistinguishable.

As was the case with communication, sensor activation maps are required to be feasible. That is, if any two strings of events appear identical according to a map then the sensor activation decisions following both strings need to be the same.

The problem considered then is to compute a sensor activation map from states to transitions which is feasible, satisfies the specification condition and is logically minimal. As in previous problems of communication for logical DES, there is no minimum solution when set inclusion is used as the optimality criterion.

The authors demonstrate that, contrary to problems of communication, the "lack of monotonicity" phenomenon is not encountered for feasible sensor activation maps. That is, for feasible sensor activation maps: the more that sensors are activated the fewer strings / states become indistinguishable. Effectively, the more an agent sees, the more it is able to distinguish.

Using this constructive result, the authors prove that the union of any two feasible sensor activation maps is also a feasible sensor activation map. A consequence of this is that, for any given state-based sensor activation map ω (not necessarily feasible), there exists a unique maximum feasible state-based sensor activation map ω' such that, $\forall s \in \mathscr{L}(G)$, $\omega'(s) \subseteq \omega(s)$. The authors then present an algorithm for computing the maximum feasible sensor activation map and associated pairs of indistinguishable states of G for a given input map. They prove that this algorithm is in $O(|X|^2 \cdot |\Sigma|)$ where Σ is the event set and X the state set of G. Using this an algorithm is provided for computing a feasible, minimal sensor activation policy for the controller which satisfies its specification condition. The authors prove that this algorithm is in $O(|X|^3 \cdot |\Sigma|^2)$. The authors prove that any feasible, minimal sensor activation map defined over the transitions of G which satisfies the specification condition can be computed by a choice of the input permutation of the transitions labeled by events observed by the centralized controller provided to the algorithm.

The authors then consider the case of decentralized control in the absence of communication. It is the objective to compute minimal sensor activation policies for all agents such that transition-based coobservability is satisfied. It is assumed that some procedure for verifying coobservability is given, such as the algorithms of [29, 100] or [51]. Using a simple generalization of the algorithm for calculating a centralized controller's minimal sensor activation policy, the authors provide an algorithm for calculating a solution to the decentralized problem. Similarly, all solutions can be computed by an appropriate permutation of the input arguments to the algorithm. The algorithm is proven to be of polynomial complexity in all parameters of the system.

In conjunction with the previous work for minimizing sensor activations for purpose of control, the same authors also investigated minimizing sensor activations for purposes of diagnosis in [102]. This work presents a general algorithm for computing minimal sensor activation maps which guarantee (co)diagnosability holds where no fixed state-transition structure of the plant is provided. Instead, a fixed partition of the language of the plant is assumed to be given over which sensor activation policies are defined. By considering more refined language partitions one can obtain more refined minimal sensor activation maps. This is analogous to a refinement in sensor activation maps derived from more refined state-transition structures of the plant.

This work differs from the previous works on sensor activation for event diagnosis (e.g., [92, 12]) for the same reasons listed in [106]. It also differs in that no specific state-transition representation of the plant is considered. However, when the plant is modeled by a DFA, the computation of minimal sensor activation maps is in the worst-case polynomial in the state set of the plant, as opposed to the previous works, although it is exponential in the size of a so-called language window size (to be introduced later). In the decentralized case it is also exponential in the number of agents. When the plant is modeled by a DFA the sensor activation maps considered can be modeled by FSTs. In the general case no specific model is assumed for the sensor activation maps.

The general algorithm proceeds in a manner very similar to the algorithm in [106] where active sensors for events following all strings in a partition are attempted to be deactivated, one partition at a time in a prespecified order, until a minimal, feasible sensor activation map is computed. The complexity of the general algorithm cannot be established precisely. It depends on the complexity of algorithms for verifying diagnosability and for calculating the maximum feasible sub-policy of a given sensor activation policy, which are dependent on the class of languages to which the language recognized by the plant belongs.

When regular language representations of the plants are assumed diagnosability can be verified using the algorithm of [100]. Additionally, when the language partition is based on the last n event occurrences leading to a state of an automaton representation of the plant (we say such a partition has window size n), the maximum feasible sub-policy of a given sensor activation policy can be computed. We have determined that the complexity of calculating the maximum feasible sub-policy is in $O(|X|^2 \cdot |\Sigma|^{2n+3})$. This is worse than the algorithm in [106] for computing maximum feasible sub-policies over transitions of an automaton representation (rather than partitions). Consequently the algorithm for computing minimal sensor activation maps is exponential in the window size (specifically, $O(|X|^3 \cdot |\Sigma|^{3n+4})$).

The authors also consider the case of minimal sensor activation in decentralized diagnosis. Their generalization to the decentralized case is exactly the same as the generalization to the decentralized case considered in [106]. However, when automaton representations of the plant and window partitions are considered, the authors prove that the algorithm is in $O(|P|^{I+2} \cdot |\Sigma|)$ where $|P| \le |X| \cdot |\Sigma|^{n+1}$. So the decentralized algorithm is in the worst-case exponential in I and n. The authors do not indicate whether or not all minimal sensor activation maps can be computed from a choice of language partition.

By reduction from the problem of deciding transition-based (co)observability to transition-based (co)diagnosability and Theorem 4 of [100] the approach of this work may also be applied for solving the same problems considered in [106]. However, the algorithms of [106] are significantly more efficient than converting the problem to a diagnosability one and applying the algorithm of this work when considering a fixed automaton representation of the plant.

The merits of this work are that it provides algorithms for computing minimal sensor activation policies for satisfying diagnosability and codiagnosability where language partitions of the plant language are provided instead of a state-transition structure representation and, when automata representations of the plant are assumed, for fixed computational resources it allows one to compute the best possible refinement of minimal sensor activation maps by refining parameterized window partitions and their window sizes.

However, in both cases of [106, 102] the solution space is considered to be fixed, either by a fixed state set representation or fixed language partition. The solutions computed are minimal with respect to the state set or given language partition. By refining the state set or language partition of a plant further, one can compute sensor activation maps where strictly fewer sensors need to be activated following some strings generated by the plant. In [101], the authors provide an algorithm for computing sensor activation maps which are minimal not merely with respect to a state set or language partition of a plant, but minimal with respect to the plant language. That is, for such sensor activation maps, by refining the state set of the plant to any degree, it is not possible to compute a sensor activation map where fewer sensors are active without violating correctness criteria of diagnosability and feasibility of the sensor activation map.

Only the centralized diagnosis problem is considered in [101]. The approach is not explained very well and the algorithm is extremely complicated. No correctness proofs or complexity analysis are provided. The approach can only be used to compute a subset of the class of sensor activation maps which are minimal with respect to the plant language as the algorithm utilizes the refinement of a given plant state-transition structure in a specific manner.

The previous approaches of [106, 102, 101] all compute minimal sensor activation maps for centralized (resp., decentralized agents) offline to be used for control (resp., diagnosis) purposes. In [105], the authors consider computing minimal sensor activation maps online using one-step lookahead of event occurrences that the plant is capable of generating. Here a fixed automaton representation of the plant, G, and a state-based specification condition over pairs of states in G are assumed to be given. The problem is to compute a sensor activation map which is implementable with respect to G, feasible, prevents any pair of states in the specification from being indistinguishable and is minimal.

An online approach to sensor activation has been considered previously in DES in [92]. However, in [92], variable cost may be incurred when sensors are activated for different events. In [105] the cost incurred for activating a sensor is the same for all events. Also, only suboptimal solutions may be computed using the online method of this previous work and the complexity of the procedure is expensive relative to this work due to the fact that string estimates are maintained instead of just state estimates.

The authors prove that, for a given state estimate of G after the observation of any string in G, if no pair of states formed from the estimate are contained in the specification then upon observing an event corresponding to any activated sensor a state estimate will be produced such that none of the pairs from the resulting estimate are contained in the specification. However, if a sensor for an arbitrary observable event is deactivated then this property is not guaranteed. This result suggests that only one-step lookahead of sensor activation decisions is required for solving the problem at hand online.

The online algorithm suggested by this result is an iterative one which considers deactivating sensors in a prespecified total order until a certain set of sensors must remain active for otherwise state pairs from the specification may become indistinguishable, etc. In this manner it is similar to the algorithmic approach of [106]. The algorithm can be applied in scenarios where the time to update the state estimate following an observation and for computing a sensor activation decision from this estimate online is less than the time between any two consecutive event occurrences. The complexity of the procedure for computing the current sensor activation decision using the one-step lookahead is polynomial in the state set of G. It is proven that the sensor activation map computed by the algorithm satisfies feasibility, implementability, satisfaction of the specification and minimality. By considering different orders on deactivating sensors, different sensor activation decisions can be made online.

In [30] the authors attempt to generalize the online sensor activation algorithm for state disambiguation provided in [105]. Here, instead of making sensor activation changes after every observed event occurrence, the authors consider only making sensor activation changes after $k \geq 1$ observed event occurrences. The authors claim that the proposed online algorithm is polynomial in all plant parameters (state / event sets of G). Though the work is a generalization in terms of the number of observed event occurrences allowed before a change in active sensors, it relies on assumptions not made in [105]: (1) G is deadlock-free, i.e., for any state of G, at least one event labels a transition from that state; (2) there are no cycles of unobservable events in G.

The work of [14] concerns computing sensor activation maps for a centralized diagnoser to diagnose faults following k event occurrences in a plant modeled by a DFA, G, when G is k-diagnosable. This work follows from [12]. It presents an alternative way to compute the most permissive observer (MPO), an NFA which encodes all sensor activation maps that enable diagnoses of all faults after k event occurrences in a plant. The MPO was first defined in [12] and forms the basis for computing minimum-cost sensor activation maps when cost is associated with activating event sensors and the duration over which they are active. A cost function for active sensors and an algorithm for computing a minimum-cost sensor activation map from the MPO was provided in [12].

For a given fault event, every occurrence of the fault can be tracked using the constructed MPO. We omit the details on how the MPO is defined in this work, which is a different procedure for defining the MPO than that used in [12]. However, the authors do introduce and incorporate an optimization for reducing the state set of the MPO which can be used during construction of the MPO.

The authors propose a depth-first search algorithm for computing the MPO. The authors prove that the algorithm is in $O([(2(k+2))^{|X|}][2^{|\Sigma_s|}][|X|^2])$ where Σ_s is the set of event sensors that can be activated / deactivated by the diagnoser. This outperforms the algorithm of [12] for computing the MPO which is in $O(2^{|X|^2+2^{|\Sigma|}+k})$.

The authors assert that if multiple faults can be generated by G then an MPO should be created for each fault type instead of a single MPO maintaining a separate count for each fault type.

The last work to be surveyed on sensor activation is [79]. This work investigates a recently introduced notion of detectability in DES. Detectability refers to the ability of an agent to determine the current and subsequent states of a DFA G based on event observation. In prior work ([81]) the authors assumed that event observation was static. In this work they consider that observation is based on the transitions of G.

Automaton G is considered to have multiple possible initial states. The actual initial state of G is not known to an agent. As usual, G's events can be partitioned into events unobservable and observable by the observer. Some assumptions regarding G are made: (1) G is deadlock-free, i.e., for any state of G there exists at least one outgoing transition labeled by an event; (2) there are no cycles of unobservable events in G.

Four different type of detectability conditions are introduced, given G and observations made using observation map M defined over the transitions of G:

- Detectability (resp., Strong Detectability): G is detectable (resp., strongly detectable) with respect to M if it can be determined, after a finite number of event occurrences, the current state and subsequent states of G for some (resp., all) infinite strings of G.
- Periodic Detectability (resp., Strong Periodic Detectability): G is periodically (resp., strongly periodically) detectable with respect to M if the current state of the system can be determined at a period of event occurrences for some (resp., all) infinite strings of G.

The authors demonstrate that detectabilities can be verified using the observer automaton computed by G using the subset construction. However, the authors also introduce nondeterministic detector automata, which are similar to the product automata of [97], for verifying strong and strong periodic detectabilities. Detector automata can be constructed in $O(|X|^2 \cdot |\Sigma|)$ ([80]) whereas the subset construction is in the worst case exponential in $|X|$. Verification can be performed online during construction of the detector automata, thus incurring the same complexity. No mention is made of whether or not there exist efficient approaches for verifying detectability and periodic detectability.

The authors then consider the problem of computing a minimal sensor activation policy for satisfying a given detectability condition. The setup is very similar to the centralized minimal sensor activation problem of [106]. The notions of implementability and feasibility of sensor activation policies are recalled. Generalizing the results of [106], the authors prove monotonicity of feasible sensor activation policies and provide an algorithm for computing the maximum feasible sub-policy of a given sensor activation policy. These results are a generalization in that the initial state of G is uncertain by the agent, whereas in [106] the initial state is certain.

The authors then present an algorithm for computing a minimal, feasible sensor activation policy satisfying a given detectability condition. The algorithm is similar to the one used in [106]. Analogous to the algorithm of [106], all minimal, feasible sensor activation policies can be computed using the algorithm by a choice of the input permutation provided on the transitions of G labelled by observable events.

12 Approaches to Solving Problems of Opacity Involving Dynamic Observation

Though control and diagnosis are the most commonly studied topics of DES when dynamic observation is concerned, recently a new topic of opacity which is related to studies in computer security has been gaining attention. Opacity is used to study the effectiveness of a system to keep secret a predicate of the system from a (potentially malicious) external observer. A predicate is opaque if an observer of the system will never be able to determine the truth of that predicate.

One of the first works in opacity of DES is [8]. Here systems are considered to be modeled using deterministic labelled transition systems (LTS) which may be infinite-state and whose initial state may be uncertain to an external observer. An external observer is assumed to have knowledge of the transition-structure of LTS and has an observation map. Predicates are defined over strings generated by the LTS. A predicate is opaque with respect to the LTS and an observation map used by an external observer if any string in the predicate is indistinguishable, according to the observation map, from a string generable by the LTS that is not in the predicate. Additionally, state-based opacity can be defined in terms of the states reached by prefixes of strings in the predicate. This can be used to model requirements in which the initial (resp., intermediate, final) states of strings in a predicate must be kept secret.

The authors demonstrate that verification of security properties such as anonymity ([76]) and non-interference ([20]) can be reduced to verifying various forms of opacity for LTS. The authors then prove that checking opacity (string-based / state-based) for LTS is undecidable even when static observation maps are considered. This is done by showing that the reachability problem for Turing Machines is reducible to this problem. So the authors then investigate PN representations of systems, as reachability is decidable for this class of systems. The authors prove that even checking initial-state opacity for reachability graphs of PNs is undecidable for dynamic observation maps. This suggests that system models weaker than PNs should be employed when decidability of verifying opacity is required.

The work of [10] applies a similar approach, used in [12] for solving a sensor activation problem of k-bounded diagnosis, to solving a problem of opacity. As in [12], the authors assume the plant is to be modeled by an NFA. The secret of the plant to be protected from an external attacker / observer is defined as a subset of the states of G. It is said that the secret is opaque if, for the observation of any string in G, the state estimate following the observation is not included in the set of secret states. This definition of opacity is different from the string-based version considered in [8].

Dynamic observers were introduced in [12] as FSTs for mapping strings generated by G to their observation according to some sensor activation map. They are recalled in this work for ensuring opacity of a secret. However, their application is different. In [12] the external observer is assumed to have control over the sensors for events it can observe. In this work, this is not the case. Instead, some entity which wants to protect the secret of G from the external observer has control over sensors for events observable by the external observer.

Using dynamic observers is a non-intrusive approach to opacity-enforcement. It differs from previous works (e.g., [1, 19]) where observation masks are static and, instead, control is imparted on the plant in an intrusive manner to guarantee that the secret of the plant remained opaque.

The authors consider the problem of verifying if a secret is opaque with respect to G and a dynamic observer D. They prove that this problem is PSPACE-complete. So, even for automata models, checking opacity is expensive. This is in stark contrast to a similar verification problem concerning k-diagnosability which is proven to be in PTIME in [12].

The authors then consider computation of the most permissive observer for opacity, a structure which encodes all dynamic observers that ensure opacity of the secret in G. They describe its computation and prove the complexity of the computation to be in $O(2^{|X|} \cdot 2^{|\Sigma|})$. In a subsequent work ([11]), the authors prove that computation of the most permissive observer for opacity is in EXPTIME and this is a lower bound.

When cost is associated with sensor activations and the duration for which sensors are active a minimum-cost dynamic observer can be computed. The authors consider this problem next. They use the same approach and tools of [12] to solve this problem and demonstrate that it is in $O(|X| \cdot (2^{|X|} \cdot 2^{|\Sigma|}))$. They prove that computing the cost of a given dynamic observer with respect to G is in PTIME.

The work of [43] follows from [10]. However, here G is a DFA, dynamic observation maps are defined over the transitions of G and the secret of G is string-based.

Let the secret be modeled by a regular language $\mathscr{L}(K) \subseteq \mathscr{L}(G)$. In this work, opacity of $\mathscr{L}(K)$ is considered with respect to another regular language $L \subseteq \mathscr{L}(G)$, not merely $\mathscr{L}(G) \setminus \mathscr{L}(K)$. This is in contrast to the previous surveyed works.

The author considers two types of opacity: strong and weak opacity. Specification $\mathscr{L}(K)$ is strongly opaque with respect to L and observation map θ if $\theta(\mathscr{L}(K)) \subseteq \theta(L)$. Specification $\mathscr{L}(K)$ is weakly opaque with respect to L and θ if $\theta(\mathscr{L}(K)) \cap \theta(L) \neq \varnothing$. Otherwise, $\mathscr{L}(K)$ is not opaque with respect to L and θ if $\theta(\mathscr{L}(K)) \cap \theta(L) = \varnothing$. In previous works only strong opacity has been considered.

The author provides an algorithm for checking strong opacity. This algorithm requires determinization of NFA and so is in EXPTIME. It is not difficult to see that the notion of opacity here is a generalization of that considered in [10]. From this fact, the author asserts that exponentiality in verifying their generalized version of opacity is "unavoidable" since verifying opacity of [10] is PSPACE-complete ([10]).

Though the author initially thought verifying weak opacity was also in EXPTIME ([43]), he and colleagues have since published a work ([117]) illustrating that verifying weak opacity can be done in PTIME by using the nondeterministic detector automaton of [79].

Similar to the work of [8], the author demonstrates that security properties of anonymity and secrecy can be formulated in terms of their generalized opacities.

Following from [43, 3] focuses on modification of secret and non-secret regular languages to satisfy opacity when transition-based observation maps are considered. This work is the first step in an alternative direction for opacity synthesis to previous works where either control is imposed on the system or observation maps are modified to satisfy opacity.

The authors prove a number of closure properties for the various opacities under union and intersection of secret / non-secret languages. Counterexamples are also provided for demonstrating cases where opacity is not closed under intersection.

The authors then use these closure properties to define super-languages and sub-languages of $\mathscr{L}(K)$, L which satisfy strong (resp., weak, no)-opacity if $\mathscr{L}(K)$ is not strongly (resp., weakly, no)-opaque with respect to L.

For satisfying strong opacity, one can either (1) shrink $\mathscr{L}(K)$ or (2) enlarge L. Due to closure under union, it is possible to compute the unique supremal sublanguage of $\mathscr{L}(K)$ that is strongly opaque with respect to L. However, due to one of the counterexamples, the unique infimal superlanguage of L such that $\mathscr{L}(K)$ is strongly opaque with respect to this language may not exist. The authors provide an example demonstrating that minimal, incomparable superlanguages of L exist such that $\mathscr{L}(K)$ is strongly opaque with respect to them. However, no algorithm is suggested for computing any minimal superlanguage.

For satisfying weak opacity, one can either (1) enlarge $\mathscr{L}(K)$ or (2) enlarge L. By one of the counterexamples, there may not exist a smallest superlanguage of $\mathscr{L}(K)$ which is weakly opaque with respect to L. As before, they demonstrate an example where minimal, incomparable superlanguages of $\mathscr{L}(K)$ are found such that these languages are strongly opaque with respect to L. However, no algorithm is suggested for computing such languages. The case for enlarging L is symmetric to enlarging $\mathscr{L}(K)$.

For satisfying no opacity, one can either (1) shrink $\mathscr{L}(K)$ or (2) shrink L. By a previous union closure property, it follows that a supremal sublanguage of $\mathscr{L}(K)$ that is not opaque with respect to L can be found. The case is symmetric for shrinking L.

The authors of [3] then proceed to consider an intrusive approach where control is imposed on G to ensure that strong (resp., weak, no) opacity of a secret language $\mathscr{L}(K) \subseteq \mathscr{L}(G)$ with respect to language $L \subseteq \mathscr{L}(G)$ is satisfied in [4].

In the problem of this paper the set of events observed by the controller controlling G is assumed to contain the set of events observed by an external observer. The controller observes the system through the static natural projection defined over its set of observable events. The external observer, however, views the system through a string-based observation map. It is also assumed that the controller observes all events it controls.

For a given $\mathscr{L}(K)$ and L, a desirable control solution for enforcing strong (resp., no) opacity between $\mathscr{L}(K)$ and L is the largest controllable, observable sublanguage of $\mathscr{L}(G)$, denoted by M, such that $\mathscr{L}(K) \cap M$ is strongly (resp., not) opaque with respect to $L \cap M$. It is obvious that by restricting the behaviour of G weak opacity cannot be satisfied if $\mathscr{L}(K)$ is not weakly opaque with respect to L.

The authors define an operator, which applied to a sublanguage $H \subseteq \mathscr{L}(G)$, generates the largest sublanguage of H that yields strong opacity. Since the controller observes events it controls, an operator can be defined for computing the supremal controllable and normal (stronger than observable) sublanguage ([7]). An algorithm is then proposed for computing a control solution which simply iterates between the two operators until a sublanguage of $\mathscr{L}(G)$ is computed which yields strong opacity.

The authors prove that if $\mathscr{L}(K)$, L and $\mathscr{L}(G)$ are regular and the external observer's observation map θ preserves regularity[1] then the algorithm terminates.

The authors then consider the no opacity control problem. The approach taken here is similar to the previous one. They first consider computation of the largest sublanguage of $\mathscr{L}(G)$, denoted by M, such that $\mathscr{L}(K) \cap M$ is not opaque with respect to $L \cap M$. The authors prove that the supremal controllable, normal sublanguage of M is a solution to the no opacity control problem. They do not have to formalize a no-opacity preserving operator and iterative algorithm for computing it, which contrasts with the computation of the largest controllable, normal, strongly-opaque sublanguage of $\mathscr{L}(G)$.

Decentralized opacity problems following the work on centralized opacity in [43] are investigated in [51]. The authors introduce decentralized versions of strong, weak and no-opacity considered in [43].

The authors take $\mathscr{L}(G)$ to be regular. The authors assert that verifying transition-based coobservability (introduced in [29]) of a prefix-closed language $H \subseteq \mathscr{L}(G)$ is reducible to verifying decentralized no-opacity of multiple secret languages with respect to other non-secret languages using the same agent transition-based observation maps.

The authors then consider opacity of a secret from the perspective of a coordinator that receives communications from agents observing G. Here agents compute inferences from their observations of G then send them to the coordinator where a final inference is made. This is intended to model situations in which multiple external agents may coordinate their efforts to infer whether or not the system executed a behaviour in the secret language, $\mathscr{L}(K)$.

The following assumptions are made on the observation map of each agent, the inference map of each agent, and the inference map of the coordinator:

1. The observation map of each agent is based on the transitions of G.
2. The domain of an agent's inference map is $\mathscr{L}(G)$. The codomain of the map can be any set of symbols / strings. For example, an agent's string (resp., state) estimate of $\mathscr{L}(G)$ (resp., G).
3. The inference map of an agent is monotonic: if an agent sees more then its inference will be more precise.
4. The coordinator utilizes all the inferences provided to it by all agents after generation of a string in $\mathscr{L}(G)$ and only these inferences in computing its final inference. Also, the final inference made is a refinement over any inference communicated to it by any individual agent.

[1]For regular language J, $\theta(J)$ and $\theta^{-1}(J)$ are regular.

Strong, weak and no coopacity are introduced in terms of agent inference maps. Informally, $\mathscr{L}(K)$ is strongly coopaque with respect to L if all strings in $\mathscr{L}(K)$ are indistinguishable with strings in L even with the help of the coordinator. Weak and no coopacity are defined similarly.

The authors prove that under the aforementioned set of assumptions the most useful class of symbols that could be communicated from agents to the coordinator for computing its inference are the event occurrences observed locally by agents. This is a useful result. Let θ denote the transition-based observation map formed by the union of the transition-based observation maps of all agents. A corollary of the previous result is that if $\mathscr{L}(K)$ is strongly (resp., weakly) opaque with respect to L and θ then $\mathscr{L}(K)$ is strongly (resp., weakly) coopaque with respect to L under any agent and coordinator inference maps where the same observation maps for agents are used. Under the set of assumptions, this result implies an easy test for verifying whether or not a secret remains opaque / not opaque from the perspective of coordinating agents regardless of the specific inference maps used by the agents. However, defining algorithms for verifying strong (resp., weak) coopacity remain open.

13 Conclusions, Future Work & Open Problems

In this paper an overview of the current research on sensor activation and communication in discrete-event systems was presented. Topics covered included the early works on communication between agents in control of DES, decidability of control problems involving communication, distributed estimation involving communication, communication in problems of diagnosis and prognosis, works where knowledge of agents is explicitly treated and used to determine when agents should communicate, works where distributed communicating controllers are derived from monolithic controllers, works in communication and sensor activation where observations are transition-based as well as dynamic observation in problems of opacity.

There are a few general criticisms that we have on research conducted on sensor activation and communication in DES. We have observed that many of the algorithmic solutions to problems tend to be naïve and exhaustive. For many problems we believe that there exist better algorithmic solutions (in time and space). Complexity analysis of the algorithmic solutions in most works is more often than not conducted in a lazy manner with no asymptotically tight lower or upper bounds provided. Also, most complexity analyses provided are directly related to the algorithms proposed by authors and little to no discussion is made on the complexity of the problems themselves.

Future directions in the research field are summarized next followed by open problems of specific works.

An assumption which is made by every work surveyed is that, following a change in its observation, an agent is capable of updating its decision (e.g., sensor activation, communication) before the next event to be generated by the plant. This is restrictive. For practical purposes, it may be useful to investigate problems of dynamic observation where this assumption is relaxed.

For most problems regarding dynamic observation, the dynamic observation protocols are restricted to the state-transition structure of the system model. By considering a state set refinement of the system one can broaden the set of dynamic observation protocols. One could consider problems where the language of the system is given and the goal is to determine a suitable state-transition model and state set refinement that produces dynamic observation protocols which satisfy specific properties. Work to this end has already been conducted in [101] for problems of sensor activation in centralized diagnosis of DES. However, solutions to other similar centralized and decentralized observation problems remain open.

An agent's implementation of most solutions to dynamic observation problems usually requires determinizing NFA with ϵ-transitions. This is known to be exponential in the state set cardinality of the NFA in the worst-case. However, in practice, rarely is it the case that these exponential corner-cases are encountered (from [9]): "Practical experience has shown that real systems possess sufficient structure so that the worst-case bounds rarely occur". So, a useful research direction is to classify which dynamic observation problems yield NFA with ϵ-transitions in their solution procedure where determinization is not in the worst-case exponential in the state set cardinality of the NFA.

Casting control and communication problems in the knowledge domain is promising. The information provided to an agent at the outset of the execution of the plant as well as the knowledge properties it stores and maintains during execution of the plant can be used for solving different classes of observational problems. In the works considered agents are provided at the outset the plant model and information about other agents such as their sets of observable and controllable events. From this information and an agent's observations inferences about the plant, the agent itself and other agents can be computed from which application-specific (e.g., control) decisions can be made. By providing agents with more information at the outset and storing more information accumulated during execution of the plant such as previous queries by / interactions with other agents, more precise inferences can be computed which can be used to refine agent communications and expand the set of observational problems the agents can collectively solve.

Some of the works on communication surveyed consider only two communicating agents (e.g., [65, 70, 45, 66, 67]). It remains to generalize these works to the case of more than two agents.

Most of the works on communication surveyed consider restrictions on which agents may communicate with other agents (the communication topology). In most of these works the communication topology is a complete graph or is acyclic and strongly connected. To our knowledge no work has been done where communication topologies are generally just strongly connected besides [47].

In most of the works surveyed on communication it is assumed that communication of messages between agents is reliable, error-free and instantaneous. Little work has been done on communication problems where observation is dynamic and these assumptions are relaxed. Work has been done when all observations are communicated between agents and communication incurs delay (e.g., [94, 26, 18, 78, 59, 58, 112, 31, 87]). Similar problems related to dynamic sensor activation where there exists channels between an event generated by the plant and its observation by an agent have yet to be explored. Work on such channels when event sensors are always / never active has been explored in [44, 39]. Of note, [39] and subsequent works introduce an approach for computation of a state estimate of modular G where observation channels are unreliable, error-prone, result in delay and where event observations are also only partially ordered, not totally ordered as in the works considered in this survey.

In the works [47] and [16] the nature of the information to be communicated between agents is not specified *a priori*. In all other works either event occurrences, state estimates or string estimates are communicated. It would be interesting to investigate, for future work, other problems where the information to be communicated is not specified *a priori*.

All investigated problems of minimal communication / sensor activation where logical notions of minimality are considered could be extended to problems of minimum communication / sensor activation when quantitative cost is associated with communications / active sensors. Works where logical notions of minimality are considered include [70, 45, 104, 103, 106, 102, 101, 79]. Works surveyed where minimum cost observation maps are computed includes [74, 92, 12].

Research on opacity has considered centralized or decentralized problems where no communication occurs among agents. It would be interesting to investigate problems where system administrators guarding a secret or external attackers which aim to discover the secret do communicate with one another.

Open problems of specific works are summarized next.

From [2] computing agent minimal communication and control policies where communication is made as late as possible following any string or in general remains open.

In [38] the author mentions that "at this point, it remains to understand more clearly the sources of the lack of monotonicity and to identify other structural assumptions for the development of efficient algorithms for the synthesis of communication protocols. Then one can seek to exploit monotonicity properties for the development of efficient algorithms for the synthesis of communication policies." We agree that it would be worthwhile to investigate the sources of lack of monotonicity for this reason. In [104] the authors identify a structural restriction of the plant model (absence of cycles besides self-loops on system states) and communication topology (agents only communicate with a central agent / station) which allows for an efficient algorithm for computing communication policies. Finding other structural restrictions which allow for efficient algorithms is an important research direction as described in [38].

In [100] a reduction from (transition-based) coobservability to (transition-based) codiagnosability is provided. It remains open to determine whether or not codiagnosability can be reduced to coobservability.

While computation of the most permissive observer in [10] is in EXPTIME, EXPTIME-hardness is left open. Determining the exact complexity of computation of the most permissive observer used in [12] and [14] is also open.

Most works on sensor activation consider that agents activate sensors to satisfy observational conditions such as observability, diagnosability or CP-coobservability. In [106] the authors mention that it would be possible to apply the approach that they take for minimal sensor activation for purposes of satisfying CP-coobservability to computing minimal sensor activation policies for satisfying alternative versions of coobservability investigated in [113, 115, 35]. This is assuming that verification procedures exist for transition or state-based versions of these coobservability conditions. So defining state / transition-based characterizations of DA-coobservability, CP&DA-coobservability and other coobservability variants is an important research direction.

From [79] it is mentioned that the problem of efficiently verifying detectability and periodic detectability remains open. To date, verification is based on computation of the observer automaton which is in the worst case exponential in the state set of G.

In [100] the problem of verifying coobservability is reduced to verifying codiagnosability in the context of transition-based dynamic observations. In [51] the problem of verifying transition-based coobservability is reduced to verifying decentralized weak opacity in the same observational context. So it is possible that codiagnosability and decentralized weak opacity are comparable. Investigating the relationship between the two would be useful. It is already known that, in the case of static observation, verifying diagnosability is reducible to verifying weak opacity [43].

In [43] verification of strong / weak opacity is considered when the secret language is regular. It would be useful to investigate algorithms for verification of strong / weak opacity of languages which are not regular as in works following [43] (e.g., [3, 4, 51]) the secret and non-secret languages considered are not assumed to belong to any specific language class but no opacity verification algorithms for these languages are provided.

In [51] strong / weak co-opacity is defined but no algorithms for their verification are provided.

References

[1] E. Badouel, M. A. Bednarczyk, A. M. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4):425–446, 2007.

[2] G. Barrett and S. Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, 45(9):1620–1638, September 2000.

[3] M. Ben-Kalefa and F. Lin. Opaque superlanguages and sublanguages in discrete event systems. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC/CCC 2009*, pages 199–204. IEEE, 2009.

[4] M. Ben-Kalefa and F. Lin. Supervisory control for opacity of discrete event systems. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing*, pages 1113–1119, Illinois, USA, Sept. 2011.

[5] R. K. Boel and J. H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In *WODES '02: Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02)*, pages 175–181, Los Alamitos, CA, USA, 2002.

[6] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[7] R. Brandt, V. Garg, R. Kumar, F. Lin, S. Marcus, and W. M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems & Control Letters*, 15(2):111–117, Feb. 1990.

[8] J. Bryans, M. Koutny, L. Mazaré, and P. Y. A. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.

[9] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., 2nd edition, 2008.

[10] F. Cassez, J. Dubreil, and H. Marchand. Dynamic observers for the synthesis of opaque systems. In Z. Liu and A. P. Ravn, editors, *ATVA*, volume 5799 of *Lecture Notes in Computer Science*, pages 352–367. Springer, 2009.

[11] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1):88–115, 2012.

[12] F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88(4):497–540, 2008.

[13] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th POPL, Los Angeles, CA*, pages 238–252, Jan. 1977.

[14] E. Dallal and S. Lafortune. Efficient computation of most permissive observers in dynamic sensor activation problems. In *Proceedings of the 2nd International Workshop on Logical Aspects of Fault-Tolerance (LAFT 2011)*, Toronto, ON, Canada, June 2011.

[15] P. Darondeau. Distributed implementations of Ramadge-Wonham supervisory control with Petri nets. In *44th IEEE Conference on Decision and Control and European Control Conference*, pages 2107–2112, Sevilla, Spain, December 2005.

[16] P. Darondeau and S. L. Ricker. Towards distributed control of discrete-event systems. In *Proceedings of the Workshop on Applications of Region Theory*, pages 63 – 78, Newcastle upon Tyne, UK, 2011.

[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H.-P. Schwefel, editors, *PPSN*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858. Springer, 2000.

[18] R. Debouk, S. Lafortune, and D. Teneketzis. On the effect of communication delays in failure diagnosis of decentralized discrete event systems. *Discrete Event Dynamic Systems*, 13(3):263–289, 2003.

[19] J. Dubreil, P. Darondeau, and H. Marchand. Opacity enforcing control synthesis. In *Proceedings of the 9th International Workshop on Discrete Event Systems (WODES 2008)*, pages 28–35, Göteborg, Sweden, May 2008.

[20] R. Focardi and R. Gorrieri. A taxonomy of trace-based security properties for CCS. In *Proceedings of the Computer Security Foundations Workshop VII (CSFW '94)*, pages 126–137, Washington - Brussels - Tokyo, June 1994. IEEE.

[21] T. L. Gall, B. Jeannet, and T. Jéron. Verification of communication protocols using abstract interpretation of fifo queues. In M. Johnson and V. Vene, editors, *AMAST*, volume 4019 of *Lecture Notes in Computer Science*, pages 204–219. Springer, 2006.

[22] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transactions on Robotics*, 19(1):137–141, 2003.

[23] S. Graf, D. Peled, and S. Quinton. Achieving distributed control through model checking. In T. Touili, B. Cook, and P. Jackson, editors, *CAV*, volume 6174 of *Lecture Notes in Computer Science*, pages 396–409. Springer, 2010.

[24] A. Haji-Valizadeh and K. A. Loparo. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions on Automatic Control*, 41(11):1579–1593, Nov. 1996.

[25] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990.

[26] K. Hiraishi. On solvability of a decentralized supervisory control problem with communication. *IEEE Transactions on Automatic Control*, 54(3):468–480, 2009.

[27] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, Reading, Massachusetts, USA, 1979.

[28] Y. Huang. Transition-based co-observability in distributed discrete-event systems. Master's thesis, Department of Electrical and Computer Engineering, Queen's University, 2005.

[29] Y. Huang, K. Rudie, and F. Lin. Decentralized control of discrete-event systems when supervisors observe particular event occurrences. *IEEE Transactions on Automatic Control*, 53(1):384–388, Feb. 2008.

[30] X. Juqin, J. Yan, and S. Shaolong. Minimal k-step event observation policy for on-line observability of discrete event systems. In *Proceedings of the 29th Chinese Control Conference*, pages 1476–1482, Beijing, China, July 2010.

[31] G. Kalyon, T. L. Gall, H. Marchand, and T. Massart. Global state estimates for distributed systems. In R. Bruni and J. Dingel, editors, *FMOODS/FORTE*, volume 6722 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2011.

[32] G. Kalyon, T. L. Gall, H. Marchand, and T. Massart. Synthesis of communicating controllers for distributed systems. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 1803–1810, December 2011.

[33] G. Katz, D. Peled, and S. Schewe. The buck stops here: Order, chance, and coordination in distributed control. In *Proceedings of the 9th international conference on Automated technology for verification and analysis*, ATVA'11, pages 422–431, Berlin, Heidelberg, Germany, 2011. Springer-Verlag.

[34] J. Knowles and D. Corne. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 98–105, New Jersey, 1999.

[35] R. Kumar and S. Takai. Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems. *IEEE Transactions on Automatic Control*, 52(10):1783–1794, 2007.

[36] R. Kumar and S. Takai. Inference-based ambiguity management in decentralized decision-making: Decentralized diagnosis of discrete event systems. *IEEE Transactions on Automation Science and Engineering*, 6(3):479–491, 2009.

[37] R. Kumar and S. Takai. Decentralized prognosis of failures in discrete event systems. *IEEE Transactions on Automatic Control*, 55(1):48–59, 2010.

[38] S. Lafortune. On decentralized and distributed control of partially-observed discrete event systems. *Lecture Notes in Control and Information Sciences*, 53:171 – 184, 2007.

[39] G. Lamperti and M. Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artif. Intell.*, 137(1-2):91–163, 2002.

[40] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[41] D. Liang, S. L. Ricker, and P. Gohari. Decentralized supervisory control and communication for reactive discrete-event systems. In *Proceedings of the 2006 American Control Conference*, pages 6045–6050, Minneapolis, MN, USA, June 2006.

[42] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):197–212, May 1994.

[43] F. Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[44] F. Lin. Control of networked discrete-event systems. In *Proceedings of the 2012 24th Chinese Control and Decision Conference*, pages 51 – 56, 2012.

[45] F. Lin, K. Rudie, and S. Lafortune. Minimal communication for essential transitions in a distributed discrete-event system. *IEEE Transactions on Automatic Control*, 52(8):1495–1502, Aug. 2007.

[46] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[47] A. Mannani and P. Gohari. Decentralized supervisory control of discrete-event systems over communication networks. *IEEE Trans. Automat. Contr.*, 53(2):547–559, 2008.

[48] M. M. Mano. *Digital Design.* Prentice-Hall, Englewood Cliffs, NJ, 3 edition, 2002.

[49] R. Milner. *Communication and Concurrency.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[50] J. F. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences of the United States of America*, 1950.

[51] A. Paoli and F. Lin. Decentralized opacity of discrete event systems. In *Proceedings of the 2012 American Control Conference*, pages 6083–6088, Montreal, Quebec, Canada, June 2012.

[52] D. Peled and S. Schewe. Practical distributed control synthesis. In F. Yu and C. Wang, editors, *INFINITY*, volume 73 of *EPTCS*, pages 2–17, 2011.

[53] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice–Hall Inc., Englewood Cliffs, 1981.

[54] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, 2008.

[55] W. Qiu. *Decentralized / distributed failure diagnosis and supervisory control of discrete event systems.* PhD thesis, Iowa State University, 2005.

[56] W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 36(2):384–395, Mar. 2006.

[57] W. Qiu and R. Kumar. A new protocol for distributed diagnosis. In *Proceedings of the 2006 American Control Conference*, pages 6063–6068, Minneapolis, MN, USA, June 2006.

[58] W. Qiu and R. Kumar. Distributed diagnosis under bounded-delay communication of immediately forwarded local observations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 38(3):628–643, 2008.

[59] W. Qiu, R. Kumar, and S. Jiang. On decidability of distributed diagnosis under unbounded-delay communication. *IEEE Transactions on Automatic Control*, 52(1):114–116, Jan. 2007.

[60] P. J. Ramadge and W. M. Wonham. Supervisory control of discrete event processes. In *Feedback Control of Linear and Nonlinear Systems*, pages 202–214. Berlin, West Germany: Springer–Verlag, 1982.

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

[62] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, Jan. 1989.

[63] L. Ricker and B. Caillaud. Mind the gap: Expanding communication options in decentralized discrete-event control. *Automatica*, 47(11):2364–2372, 2011.

[64] S. L. Ricker. Asymptotic minimal communication for decentralized discrete-event control. In *Proceedings of the 9th International Workshop on Discrete Event Systems*, pages 486–491, May 2008.

[65] S. L. Ricker and K. Rudie. Incorporating communication and knowledge into decentralized discrete-event systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 1326–1332, 1999.

[66] S. L. Ricker and K. Rudie. Know means no: Incorporating knowledge into discrete-event control systems. *IEEE Transactions on Automatic Control*, 45(9):1656–1668, September 2000.

[67] S. L. Ricker and K. Rudie. Knowledge is a terrible thing to waste: Using inference in discrete-event control problems. *IEEE Transactions on Automatic Control*, 52(3):428–441, 2007.

[68] K. Rohloff, S. Khuller, and G. Kortsarz. Approximating the minimal sensor selection for supervisory control. *Discrete Event Dynamic Systems*, 16(1):143–170, 2006.

[69] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event control system. In *Proceedings of the 1999 American Control Conference*, pages 1965–1970, San Diego, California, USA, June 1999.

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

[71] K. Rudie and J. C. Willems. The computational complexity of decentralized discrete-event control problems. In *Proc. of 1993 European Control Conf.*, Groningen, The Netherlands, 1993.

[72] K. Rudie and W. M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, Nov. 1992.

[73] W. H. Sadid, S. L. Ricker, and S. Hashtrudi-Zad. Nash equilibrium for communication protocols in decentralized discrete-event systems. In *2010 American Control Conference*, pages 3384–3389, Baltimore, MD, USA, June / July 2010.

[74] W. H. Sadid, S. L. Ricker, and S. Hashtrudi-Zad. Multiobjective optimization in control with communication for decentralized discrete-event systems. In *CDC-ECE*, pages 372–377. IEEE, 2011.

[75] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, Sept. 1995.

[76] S. Schneider and A. Sidiropoulos. CSP and anonymity. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *ESORICS*, volume 1146 of *Lecture Notes in Computer Science*, pages 198–218. Springer, 1996.

[77] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM J. Control Optim.*, 36(2):488–541, Mar. 1998.

[78] R. Sengupta and S. Tripakis. Decentralized diagnosability of regular languages is undecidable. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002*, volume 1, pages 423–428, Dec. 2002.

[79] S. Shu and F. Lin. Detectability of discrete event systems with dynamic event observation. *Systems & Control Letters*, 59(1):9–17, Jan 2010.

[80] S. Shu and F. Lin. Generalized detectability for discrete event systems. *Systems & Control Letters*, 60(5):310–317, 2011.

[81] S. Shu, F. Lin, and H. Ying. Detectability of discrete event systems. *IEEE Trans. Automat. Contr.*, 52(12):2356–2359, 2007.

[82] R. Su and W. M. Wonham. Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE Transactions on Automatic Control*, 50(12):1923–1935, Dec. 2005.

[83] R. Su, W. M. Wonham, J. Kurien, and X. Koutsoukos. Distributed diagnosis for qualitative systems. In *Proceedings of the 6th International Workshop on Discrete Event Systems (WODES'02)*, pages 169–174, Zaragoza, Spain, Oct. 2002.

[84] S. Takai and R. Kumar. Synthesis of inference-based decentralized control for discrete event systems. *IEEE Transactions on Automatic Control*, 53(2):522–534, 2008.

[85] S. Takai and R. Kumar. Synthesis of over-approximating inference-based decentralized supervisors for discrete event systems. *IEEE Transactions on Automatic Control*, 55(8):1881–1887, 2010.

[86] S. Takai and R. Kumar. Inference-based decentralized prognosis in discrete event systems. *IEEE Transactions on Automatic Control*, 56(1):165–171, 2011.

[87] S. Takai and R. Kumar. Distributed failure prognosis of discrete event systems with bounded-delay communications. *IEEE Transactions on Automatic Control*, 57(5):1259–1265, 2012.

[88] D. Teneketzis. On information structures and nonsequential stochastic control. *CWI Quarterly*, 10(2):179–199, 1997.

[89] J. G. Thistle. Undecidability in decentralized supervision. *Systems & Control Letters*, 54(5):503–509, 2005.

[90] J. G. Thistle and R. Su. Uncomputability of supremal local supports and effective consistent distributed diagnosis. Technical report, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, April 2005.

[91] J. G. Thistle and R. Su. Uncomputability of supremal local supports in distributed diagnosis. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 6317–6322, Seville, Spain, Dec. 2005.

[92] 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, December 2007.

[93] S. Tripakis. Undecidable problems of decentralized observation and control. In *Proceedings of the 40th IEEE Conference on Decision and Control*, volume 5, pages 4104–4109, Orlando, FL, USA, 2001.

[94] S. Tripakis. Decentralized control of discrete event systems with bounded or unbounded delay communication. In *WODES '02: Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02)*, page 18, Washington, DC, USA, 2002.

[95] S. Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Inf. Process. Lett.*, 90(1):21–28, 2004.

[96] S. Tripakis. Decentralized observation problems. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pages 6–11, Seville, Spain, Dec. 2005.

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

[98] J. H. van Schuppen. Decentralized supervisory control with information structures. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 36–41, Cagliari, Italy, Aug. 1998.

[99] W. Wang. *Optimization of Communication and Coverage in Classes of Distributed Systems.* PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan, 2007.

[100] W. Wang, A. R. Girard, S. Lafortune, and F. Lin. On codiagnosability and coobservability with dynamic observations. *IEEE Transactions on Automatic Control*, 56(7):1551–1566, 2011.

[101] W. Wang, C. Gong, and A. R. Girard. Language-based minimization of sensor activation for event diagnosis. In *49th IEEE Conference on Decision and Control*, pages 6734–6739. IEEE, Dec. 2010.

[102] W. Wang, S. Lafortune, A. R. Girard, and F. Lin. Optimal sensor activation for diagnosing discrete event systems. *Automatica*, 46(7):1165–1175, 2010.

[103] W. Wang, S. Lafortune, and F. Lin. Minimization of communication of event occurrences in acyclic discrete event systems. *IEEE Transactions on Automatic Control*, 53(9):2197–2202, Oct. 2008.

[104] W. Wang, S. Lafortune, and F. Lin. On the minimization of communication in networked systems with a central station. *Discrete Event Dynamic Systems: Theory and Applications*, 18(3):415–443, Sept. 2008.

[105] W. Wang, S. Lafortune, F. Lin, and A. R. Girard. An online algorithm for minimal sensor activation in discrete event systems. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC/CCC 2009*, pages 2242–2247, December 2009.

[106] W. Wang, S. Lafortune, F. Lin, and A. R. Girard. Minimization of dynamic sensor activation in discrete event systems for the purpose of control. *IEEE Transactions on Automatic Control*, 55(11):2447–2461, Nov. 2010.

[107] Y. Wang, T.-S. Yoo, and S. Lafortune. New results on decentralized diagnosis of discrete-event systems. In *Proceedings of the 2004 Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, USA, September 2004.

[108] Y. Wang, T.-S. Yoo, and S. Lafortune. Decentralized diagnosis of discrete event systems using unconditional and conditional decisions. In *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05.*, pages 6298–6304, Dec. 2005.

[109] K. C. Wong and J. H. van Schuppen. Decentralized supervisory control of discrete-event systems with communication. In *Proceedings of the International Workshop on Discrete Event Systems (WODES96)*, pages 284–289, Edinburgh, U. K., Aug. 1996.

[110] W. M. Wonham. Supervisory control of discrete-event systems. Systems Control Group, Dept. of ECE, Univ. of Toronto, 2012 [Online]. Available: URL: http://www.control.toronto.edu/people/profs/wonham/wonham.html.

[111] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, 1987.

[112] S. Xu and R. Kumar. Distributed state estimation in discrete event systems. In *Proceedings of the 2009 American Control Conference*, pages 4735–4740, June 2009.

[113] 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.

[114] T.-S. Yoo and S. Lafortune. NP-completeness of sensor selection problems arising in partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1495–1499, 2002.

[115] T.-S. Yoo and S. Lafortune. Decentralized supervisory control with conditional decisions: Supervisor existence. *IEEE Transactions on Automatic Control*, 49(11):1886–1904, Nov. 2004.

[116] S. D. Young and V. K. Garg. Optimal sensor and actuator choices for discrete event systems, July 1994.

[117] B. Zhang, S. Shu, and F. Lin. Polynomial algorithms to check opacity in discrete event systems. In *Proceedings of the 24th Chinese Control and Decision Conference*, pages 763–769, 2012.

[118] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.