On Tractable Instances of Modular Supervisory Control

Ramakrishna Gummadi, *Student Member, IEEE*, Nikhil Singh, *Student Member, IEEE*, and Ramavarapu S. Sreenivas, *Senior Member, IEEE*

***Abstract*—An instance of a *modular supervisory control problem* involves a *plant* automaton, described either as a monolithic, finite-state automaton (SUP1M), or as the *synchronous product* of several finite-state automata (SUPMM), along with a set of finite state, *specification automata* on a common alphabet. The marked language of the synchronous product of these automata represents the desired specification. A supervisory policy that solves the instance selectively disables certain events, based on the past history of event-occurrences, such that the marked behavior of the supervised system is a non-empty subset of the desired specification.**

Testing the existence of a supervisory policy for a variety of instances of modular supervisory control is $PSPACE$-complete [1]. This problem remains intractable even when the plant is a monolithic finite state automaton and the specification automata are restricted to have only two states with a specific structure [2]. We refer to this intractable class as SUP1Ω in this paper. After introducing *complement sets* for events in a plant automaton, we identify a subclass of SUP1Ω that can be solved in polynomial time. Using this class as the base, inspired by a family of subclasses of SAT (cf. section 4.2, [3]) that can be solved in polynomial time [4], we develop a family of subclasses of SUP1Ω that can be solved in polynomial time. The results of this paper are also used to identify a polynomial time hierarchy for certain intractable subclasses of SUPMM identified in this paper.

***Index Terms*—Discrete event systems, supervisory control.**

I. Introduction

TRADITIONAL system theory is replete with models of systems described by differential or difference equations. The prevalence of differential equations can be traced to Classical Mechanics [5], while difference equations arise out of sampling a continuous-time system. This "physics-based" representation of systems loses much of its relevance when one considers "person-made" systems. Differential or difference equations are inadequate to represent and/or analyze the operational level control of a manufacturing system, the contention-resolution of shared resources in computer networks, or the operations-management of multi-component organizations with event driven dynamics like shipyards and airports, etc. These systems are representatives of the family of *Discrete-Event/Discrete-State* (DEDS) systems. They must be characterized by *state variables* that have a logical or symbolic, as opposed to an algebraic, interpretation. The dynamics of DEDS systems arise due to the occurrence of specific events, and the state-variables of such systems are piecewise constant with discontinuities at the occurrence-instants of these events.

Air-traffic control systems; automated manufacturing systems; computer networks; integrated command, control, communication and information (C^3I) systems; operations-management of multi-component organizations with event-driven dynamics like shipyards, airports, hospitals, etc. are examples of DEDS systems. The common-theme in each of these examples is that the system that is to be controlled, the *plant*, is usually modeled using finite-state automata, labeled Petri nets, or some form of a restricted Grammar. In the absence of any supervision, the plant generates a set of event-strings, referred to as the *plant-language*. There might be strings in this language that are not acceptable, in that, they violate some desired specification, that is usually represented as another language. If there is at least one string that is common to the plant- and the specification-language, and if every event in the plant can be prevented by an external agent, we can find a *supervisory policy* that prevents events in the plant in such a way that a desired string is generated under supervision. The computational effort exerted by any procedure that decides the existence of a supervisory policy is therefore no smaller than that of a procedure that decides the existence of a common string in the plant- and specification-language. This observation has been useful in identifying families of intractable instances in the literature. Additionally, there will be plant-events in typical DEDS systems that are external to the system, and therefore cannot be prevented from occurring by supervision. The presence of these unpreventable events does not reduce the computation effort in solving an instance, and the families of problem instances deemed intractable when each event is preventable, remain intractable when there are events that cannot be prevented from occurring by supervision.

Variations to the above problem include—supervision in the presence of partial observations; cooperative-supervision, where a collection of supervisors, that have partial-observation of the events generated by the plant, cooperate to enforce the desired specification; supervisory policies that enforce deadlock/livelock freedom, fairness, etc. Even though the theory behind the synthesis of supervisory policies is mature and well-understood, it has found limited use despite its twenty-year

Manuscript received September 15, 2009; revised January 22, 2010; accepted October 05, 2010. Date of publication October 25, 2010; date of current version July 07, 2011. This work was supported in part by the National Science Foundation under Grant CNS-0437415. Recommended by Associate Editor S. Haar.

R. Gummadi and N. Singh are with the Coordinated Science Laboratory and Electrical and Computer Engineering, University of Illinois, Urbana, IL 61801 USA (e-mail: gummadi2@illinois.edu; nsingh1@illinois.edu).

R.S. Sreenivas is with the Coordinated Science Laboratory and the Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign,Urbana, IL 61801 USA (e-mail: rsree@illinois.edu).

Digital Object Identifier 10.1109/TAC.2010.2089563

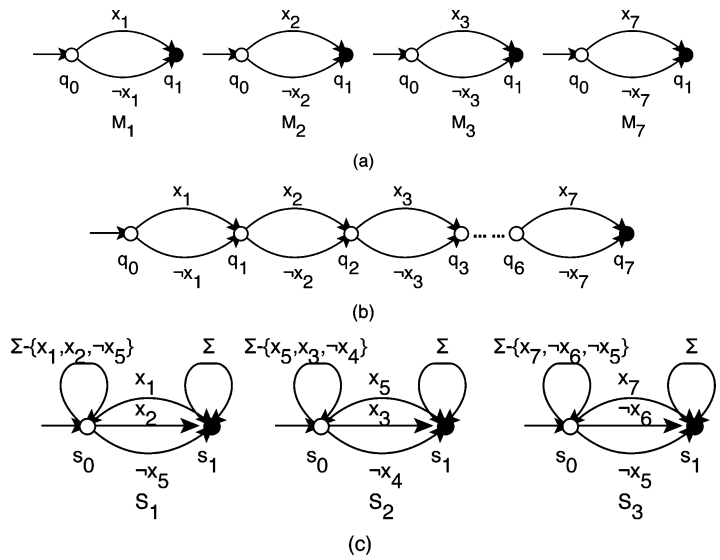

Fig. 1. (a) Automata $\{M_1, M_2, \ldots, M_7\}$. (b) Automaton M, (c) Automata $\{S_1, S_2, S_3\}$.

history. It is surmised that this is due to the difficulty of presenting a single, all-encompassing, monolithic-prescription of the specification- and plant-language. Researchers in the supervisory control of DEDS systems have proposed the use of *modularity* to combat this impediment in practice. A *modular-specification* of the desired behavior essentially lists each desirable feature independent of others. The goal of supervisory control is to ensure that each of these desirable features are retained in the controlled-behavior of the system. A supervisory control problem where the plant-language has a monolithic presentation, but the specification-language is presented in a modular fashion is an instance of a family of supervisory control problems denoted as SUP1M in the literature [2]. A natural extension to this would be the case where the plant- and specification-language have a modular description. This would be an instance of the family of problems denoted as SUPMM [2].

Using a reduction to the *3SAT-problem* [6], it can be shown that testing the existence of a supervisory policy that solves an arbitrary instance of SUP1M or SUPMM is intractable. Consider a discrete-state plant that is the synchronous product of the deterministic finite-state automata $\{M_1, M_2, \ldots, M_7\}$ shown in Fig. 1(a), or, the deterministic finite-state automaton shown in Fig. 1(b). The state that has an arrowhead attached to it denotes the *initial-state*, and the states represented by filled-circles identity the *final-states* of the automata in Fig. 1. The desired specification is represented as the *meet* (cf. Section II for a formal definition) of the automata $\{S_1, S_2, S_3\}$ shown in Fig. 1(c). Let us suppose all events in the set $\Sigma = \{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_7, \neg x_7\}$ are controllable. There is a supervisory policy that enforces the desired specification if and only if the $3SAT$-formula $(x_1 \vee x_2 \vee \neg x_5) \wedge (x_5 \vee x_3 \vee \neg x_4) \wedge (x_7 \vee \neg x_6 \vee \neg x_5)$ is satisfiable. Using a generalization of this observation, and the intractability of $3SAT$, Gohari and Wonham [2] establish the intractability of a generalized, and probably more realistic version of the supervisory control problem that does not involve monolithic descriptions of the plant and specification. Rohloff and Lafortune [1] consider several variations of the modular supervisory control paradigm and note that deciding the existence of a supervisor is $PSPACE$-complete if the specification automata are arbitrary deterministic finite-state automata.

These results, together with the axiom that ease of modular supervision is essential to the popularity of the theory of supervisory control, can be taken as a call for investigation into the additional structure within these colossal, intractable families of problems with eye towards providing normative guidelines for the identification of tractable instances of modular supervision. A family of problems deemed intractable when viewed as a single indivisible entity is oftentimes composed of strata that are defined according to the computational effort required to resolve an instance within each stratum. In the case of the *polynomial hierarchy* (cf. section 17.2, [3]) of the $PSPACE$ colossus, only the base level corresponds to the set of problems that can be solved in polynomial time, and as such this hierarchy provides very little normative guidance that we seek. On the other hand, there is a stratification of the NP colossus that is more encouraging. Gallo and Scuttela [4] identify strata $\Gamma_0 \subseteq \Gamma_1 \subseteq \cdots \subseteq \Gamma_k \subseteq \cdots$, where $\lim_{k\to\infty} \Gamma_k$ is the set of all possible instances of SAT. Any instance in Γ_k can be solved in $O(n^* n^k)$ time, where n is the number of *boolean variables*, and $n^* = O(nm)$, where m is the number of *clauses* in the instance (cf. Section II for formal definitions of these terms). This paper is about similar stratification for various intractable subclasses of SUP1M and SUPMM, with an eye towards providing normative guidelines for spotting tractable instances in the practice.

Of particular note, SUP1M and SUPMM remain intractable when the specification automata are restricted to a family, Ω, of automata that contain no more than two states like those shown in Fig. 1(c), and the plant automaton is either a single, monolithic automaton, or is the synchronous product of automata that have no common symbols like those shown in Fig. 1(a). The (intractable) family denoted as SUP1$\Omega \subset$ SUP1M represents problem instances where the specification automata belong to

the family Ω introduced earlier. We present a hierarchy of subclasses of SUP1Ω that yield tractable solutions. We also identify a hierarchy of tractable instances of SUPMΩ where the automata that define the plant do not share common events, which as the example in Fig. 1(a) and (c) shows, forms an intractable class.

The rest of this paper is organized as follows. Section II introduces *complement sets* of events and their properties, which are used to develop a polynomial time algorithm that solves a class of modular supervisory control problems in Section IV, following the identification of a necessary and sufficient condition for the existence of a solution to instances of SUP1M and SUPMM in Section III. Using the results in Section IV, and [4], we develop a family of subclasses of the intractable families SUP1Ω and SUPMΩ that can be solved in polynomial time. Section VI contains a discussion about the implications of these results along with a discussion about directions for future research.

II. Complement Sets and Their Properties

A nondeterministic, finite-state, automaton (NDFA) M is represented by a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is the finite-set of *states*, Σ is the finite-set of *symbols*, $q_0 \in Q$ is the *initial-state*, $F \subseteq Q$ denotes the set of *final-states*, and $\delta : Q \times \Sigma \rightarrow 2^Q$ denotes the (possibly partial) state-transition function, where $\delta(q, \sigma) \subseteq Q$, when it is defined, identifies the set of states that can be reached from $q \in Q$ following the occurrence of event $\sigma \in \Sigma$. There is a directed edge, (q_1, q_2), from $q_1 \in Q$ to $q_2 \in Q$ with label $\sigma \in \Sigma$, in the graphical description of M iff $q_2 \in \delta(q_1, \sigma)$. The function $\delta : Q \times \Sigma \rightarrow 2^Q$ can be naturally extended to a function $\delta^* : Q \times \Sigma^* \rightarrow 2^Q$ as follows—for any $q_1 \in Q$, $\omega \in \Sigma^*$, and $\sigma \in \Sigma$, $\delta^*(q_1, \omega\sigma) = \{q_2 \in Q \mid \exists q_3 \in \delta^*(q_1, \omega)$, and $q_2 \in \delta(q_3, \sigma)\}$, where $\delta^*(q, \epsilon) = \{q\}$, and $\epsilon \in \Sigma^*$ is the empty-string. If $q_2 \in \delta^*(q_1, \omega)$ for some $\omega \in \Sigma^*$, then there is a path from q_1 to q_2 with an associated label of ω, in the graphical description of M.

The *score* of an event $\sigma \in \Sigma$ in a string $\omega \in \Sigma^*$ is denoted by $\#(\sigma, \omega)$, and is equal to the number of occurrences of σ in ω. The *language* marked by M is denoted by $L_m(M) = \{\omega \in \Sigma^* \mid \delta^*(q_0, \omega) \in F\}$. On occasion, we might find it necessary to define the *behavior* of M, denoted by $L(M)$, where $L(M) = \{\omega \in \Sigma^* \mid \delta^*(q_0, \omega) \in Q\}$. The *prefix-closure* of $L_m(M)$ is denoted by $\overline{L_m}(M)$. That is, $\overline{L_m}(M) = \{pr(\omega) \mid \omega \in L(M)\}$, where $pr(\bullet)$ denotes the prefix-set of the string argument. Since $F \subseteq Q$, it follows that $L_m(M) \subseteq L(M)$ and $\overline{L_m}(M) \subseteq L(M)$. On the other hand, $\overline{L}(M) = L(M)$.

If the (possibly partial) state transition function $\delta(\bullet, \bullet)$ is single-valued, that is, $\delta : Q \times \Sigma \rightarrow Q$, the automaton M is said the to be a *deterministic, finite-state, automaton* (DFA). Each vertex in the graphical description of a DFA has at most one outgoing edge labeled by any member of Σ. Consequently, there is a one-to-one correspondence between paths in the graphical description of M that start from $q_0 \in Q$ to members in $F \subseteq Q$ and members of $L_m(M)$ if M is a DFA. Every NDFA M has an equivalent DFA $\widehat{M}$ such that $L_m(M) = L_m(\widehat{M})$. In the worst case, $\widehat{M}$ could have an exponentially-large set of states as compared to M.

The *synchronous product* of k NDFAs $\{M_i\}_{i=1}^k$, where $M_i = (Q_i, \Sigma_i, \delta_i, q_0^i, F_i)$ is an NDFA denoted by $\prod_{i=1}^k M_i = (Q, \Sigma, \delta, q_0, F)$, where $Q = \prod_{i=1}^k Q_i$, $\Sigma = \bigcup_{i=1}^k \Sigma_i$, $q_0 = (q_0^1, q_0^2, \ldots, q_0^k)$, $F = \prod_{i=1}^k F_i$, and $(\widehat{q}^1, \ldots, \widehat{q}^k) \in \delta((q^1, \ldots, q^k), \sigma) \Leftrightarrow \forall\ i \in \{1, 2, \ldots, k\}\ \{(\widehat{q}^i \in \delta(q^i, \sigma) \wedge \sigma \in \Sigma_i) \vee (\widehat{q}^i = q^i \wedge \sigma \notin \Sigma_i)\}$

When $\Sigma = \Sigma_k$ for all k, the above construction yields the *meet* of the constituent automata, we will denote it by $\bigwedge_{i=1}^k M_i$. By construction it follows that $L_m\left(\bigwedge_{i=1}^k M_i\right) = \bigcap_{i=1}^k L_m(M_i)$. If $P_i : \Sigma^* \rightarrow \Sigma_i^*$ denotes the natural projection of strings in Σ to strings in $\Sigma_i \subseteq \Sigma$, then $L_m(M) = \{\omega \in \Sigma^* \mid \forall\ i \in \{1, \ldots, k\}, P_i(\omega) \in L_m(M_i)\} = \bigcap_{i=1}^k P_i^{-1}(L_m(M_i))$.

For $\sigma \in \Sigma$ and language $L \subseteq \Sigma^*$, depending on the nature of the specifications that are to be enforced there are several notions of the *complement* of an event $\sigma \in \Sigma$. For a specification that is the *meet* of a set of automata that involve the occurrences of atomic events (cf. Fig. 1(c), for example), we can define the complement of $\sigma \in \Sigma$ with respect to a language $L \subseteq \Sigma^*$ as the set of events, $comp(\sigma, L) \subseteq \Sigma$, such that the occurrence of any one of them in a string $\omega \in L \subseteq \Sigma^*$ is sufficient to conclude that there can be no occurrences of σ in ω. That is, for $\sigma_1 \in \Sigma, comp(\sigma_1, L) := \{\sigma_2 \in \Sigma \mid \forall\ \omega \in L, (\#(\sigma_2, \omega) \neq 0) \Rightarrow (\#(\sigma_1, \omega) = 0)\}$.

Equivalently, $(\sigma_2 \notin comp(\sigma_1, L)) \Leftrightarrow \exists \omega \in L, ((\#(\sigma_1, \omega) \neq 0) \wedge (\#(\sigma_2, \omega) \neq 0)) \Leftrightarrow (\sigma_1 \notin comp(\sigma_2, L))$. Or, $(\sigma_1 \in comp(\sigma_2, L)) \Leftrightarrow (\sigma_2 \in comp(\sigma_1, L))$. Also, $(\forall\ \sigma \in \Sigma, \sigma \in comp(\sigma, L)) \Leftrightarrow (L = \emptyset)$. That is, $L = \emptyset$ iff each member of Σ belongs to its own complement set. This follows directly from the definition of the complement set. The following result shows that there is a $O(card(\Sigma)^2 (card(Q))^3)$ algorithm that computes $comp(\sigma, L_m(M))$ for all $\sigma \in \Sigma$, for an NDFA $M = (Q, \Sigma, \delta, q^0, F)$.

Lemma 2.1: For an NDFA $M = (Q, \Sigma, \delta, q_0, F)$ the set $\{comp(\sigma, L_m(M))\}_{\sigma \in \Sigma}$ can be computed in $O(card(\Sigma)^2 (card(Q))^2)$ time.

Proof: A language accepted by an NDFA on n states is nonempty if and only if there is a path from the initial-state to one of the final-states, that is no longer than n. This can be decided by a depth-first search on the graphical description of the NDFA, which is an $O(n^2)$ operation.

If $\sigma_1 \neq \sigma_2$, we note $(\sigma_1 \in comp(\sigma_2, L_m(M))) \Leftrightarrow (L_m(M) \cap L_m(\widehat{M}) = \emptyset)$, where $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0, \widehat{F})$, where $\widehat{Q} = \{\widehat{q}_0, \widehat{q}_1, \widehat{q}_2, \widehat{q}_3\}3$, $F = \{\widehat{q}_3\}$, and the state-transition function $\widehat{\delta} : \widehat{Q} \times \Sigma \rightarrow 2^{\widehat{Q}}$ can be inferred from the graphical description of $\widehat{M}$ shown in Fig. 2(a). There is an NDFA on $4 \times card(Q)$ states that accepts $L_m(M) \cap L_m(\widehat{M})$, therefore the emptiness of $L(M) \cap L_m(\widehat{M})$ can be decided in $O(card(Q)^2)$ time, which in turn implies membership of σ_1 in $comp(\sigma_2, L_m(M))$ can be decided in $O(card(Q)^2)$ time.

If $\sigma_1 = \sigma_2$, the process described above is done with the NDFA, $\widetilde{M} = (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F})$, where $\widetilde{Q} = \{\widetilde{q}_0, \widetilde{q}_1\}$, $F = \{\widetilde{q}_1\}$, and $\widetilde{\delta} : \widetilde{Q} \times \Sigma \rightarrow \widetilde{Q}$, is as defined in the graphical description

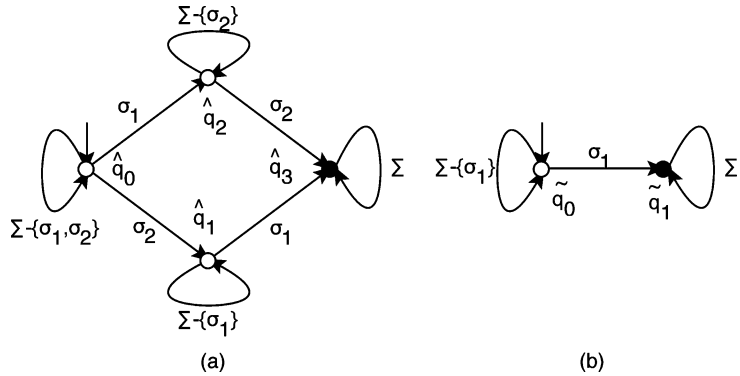

Fig. 2. (a) NDFA $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta}, \widehat{q}_0, \widehat{F})$, and (b) NDFA $\widetilde{M} = (\widetilde{Q}, \Sigma, \widetilde{\delta}, \widetilde{q}_0, \widetilde{F})$, used in lemma 2.1.

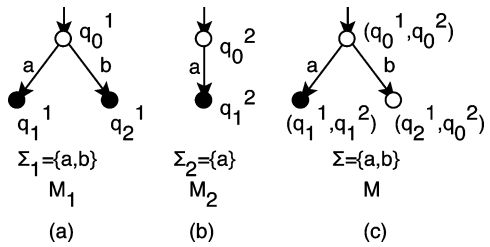

Fig. 3. (a) NDFA M_1. (b) NDFA M_2. (c) NDFA $M = \prod_{i=1}^{2} M_i$. Note, $comp(b, L_m(M_1))(= \{a\}) \subset comp(b, L_m(M))(= \{a, b\})$.

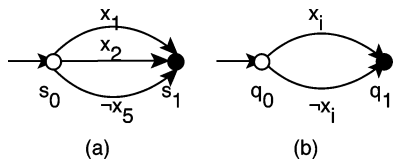

Fig. 4. (a) Illustration of how a clause in an instance of $3SAT$ (for example, $(x_1 \vee x_2 \vee \neg x_5)$ is converted into a NDFA. (b) NDFA that ensures the execution of exactly one event in the set $\{x_i, \neg x_i\}$. These are used in the proof of lemma 2.2.

of Fig. 2(b). The membership of σ_1 in $comp(\sigma_1, L_m(M))$ can also be decided in $O(card(Q)^2)$ time.

Consequently, the set $\{comp(\sigma, L_m(M))\}_{\sigma \in \Sigma}$ can be computed in $O(card(\Sigma)^2 card(Q)^2)$ time. ■

If an NDFA M is converted into an equivalent DFA $\widehat{M}$, it follows that $\{comp(\sigma, L_m(M)\}_{\sigma \in \Sigma} = \{comp(\sigma, L_m(\widehat{M})\}_{\sigma \in \Sigma}$ as $L_m(M) = L_m(\widehat{M})$. We now turn our attention to the issue of computing the set $\{comp(\sigma, L_m(M))\}_{\sigma \in \Sigma}$ when the NDFA $M = (Q, \Sigma, \delta, q_0, F)$ is the synchronous product of k NDFA's. That is, $M = \prod_{i=1}^{k} M_i$, where $M_i = \left(Q_i, \Sigma_i, \delta_i, q_0^i, F_i\right)$. We note that $(\sigma_1 \in comp(\sigma_2, L_m(M_i)) \Rightarrow (\sigma_1 \in comp(\sigma_2, L_m(M))$. However, the reverse implication of this observation is not always true. As an illustration, consider the DFA M_1 (M_2) shown in Fig. 3(a) and (b). The DFA $M = \prod_{i=1}^{2} M_i$ is shown in Fig. 3(c). We have, $comp(a, L_m(M_2)) = \emptyset$, $comp(a, L_m(M_1)) = comp(a, L_m(M)) = \{b\}$, $comp(b, L_m(M_1)) = \{a\}$, and $comp(b, L_m(M)) = \{a, b\}$. Lemma 2.1, and the fact that $card(Q)$ could be exponentially related to k in the worst-case [7], might suggest that the computation of $\{comp(\sigma, L_m(M))\}_{\sigma \in \Sigma}$ could be intractable. This is confirmed in lemma 2.2 below using a reduction to the *3SAT-problem*, which is reviewed below.

A boolean formula $\phi(x_1, x_2, \ldots, x_n)$ is in the *conjunctive normal form* (CNF) if it can be written as

$$\phi(x_1, x_2, \ldots, x_n) = \bigwedge_{k=1}^{m} \left(\bigvee_{i \in P_k} x_i \bigvee_{i \in N_k} \neg x_i \right)$$

where $x_1, x_2, \ldots, x_n$ are boolean variables, $\neg x_1, \neg x_2, \ldots, \neg x_n$ are their *complements*, and $P_1, \ldots, P_m$, $N_1, \ldots, N_m$ are subsets of $\{1, 2, \ldots, n\}$ satisfying $P_k \cap N_k = \emptyset$, $\forall\ k \in \{1, 2, \ldots, m\}$. For $k \in \{1, 2, \ldots, m\}$ the term $C_k = (\vee_{i \in P_k} x_i \vee_{i \in N_k} \neg x_i)$ is called a *clause* of $\phi(\bullet)$ and $card(P_k \cup N_k)$ is called the *degree* of C_k. The *degree* of $\phi(\bullet)$ is the maximum degree of its constituent clauses. The *size* of $\phi(\bullet)$, denoted by $\mid \phi(\bullet) \mid$, is the sum of the degrees of its clauses.

The *Satisfiability Problem* (SAT-problem) requires us to check if $\phi(x_1, x_2, \ldots, x_n)$ is *satisfiable*. That is, to check if there is an binary-assignment $(a_1, a_2, \ldots, a_n) \in \{0, 1\}^n$ such that $\phi(a_1, a_2, \ldots, a_n) = 1$. The SAT-problem is NP-complete even if the constituent clauses are only of degree 3. In this case, $\phi(\bullet)$ is said to be an instance of $3SAT$. However, there is a linear time solution, in terms of $\mid \phi(\bullet) \mid$ when

1) the degree of $\phi(\bullet)$ is 2 (i.e., $\phi(\bullet)$ is an instance of $2SAT$), or
2) $\forall\ k \in \{1, 2, \ldots, m\}$, $card(P_k) \leq 1$ (i.e., $\phi(\bullet)$ is a *Horn* formula). The subclass of SAT that involves testing the satisfiability of a Horn formula is called $HORNSAT$.

Lemma 2.2: $3SAT$ is reducible to the computation of the set $\{comp(\sigma, L_m(M))\}_{\sigma \in \Sigma}$ for an NDFA $M = \prod_{i=1}^{k} M_i$, where $M_i = \left(Q_i, \Sigma_i, \delta_i, q_i^0, F_i\right)$.

Proof: A $3SAT$ formula Φ with m clauses on boolean variables $\{x_1, x_2, \ldots, x_n\}$ can be effectively converted into $(m + n)$ NDFAs on the symbol set $\Sigma = \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$. The process by which each clause in the formula is converted into a NDFA can be inferred from the process by which the clause $(x_1 \vee x_2 \vee \neg x_5)$ is converted to the specification S_1 in Fig. 4(a). In addition, there is an NDFA for each boolean variable as shown in Fig. 4(b).

If $M = \prod_{i=1}^{m+n} M_i$, then $((\Phi \text{ is satisfiable}) \Leftrightarrow (L_m(M) \neq \emptyset)$. Since, $(\forall\ \sigma \in \Sigma, \sigma \in comp(\sigma, L_m(M))) \Leftrightarrow (L_m(M) = \emptyset)$, any procedure that computes the set

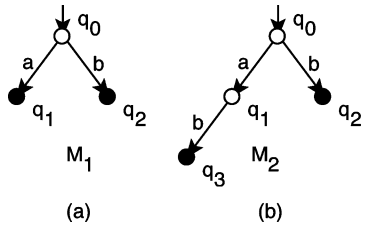

Fig. 5. (a) NDFA M_1 that is persistent. (b) NDFA M_2 that is not *persistent*. Note, $\Gamma(q_0, M_2) = \{a, b\}$, $\Gamma(q_2, M_2) = \emptyset$, and $comp(b, L_m(M_2)) = \emptyset$.

$\{comp(\sigma, L_m(M))\}_{\sigma\in\Sigma}$ can be effectively used to check the satisfiability of Φ. ∎

As a counterpoint to lemma 2.2, we note the existence of special cases where the computation of $\{comp(\sigma, L_m(M))\}_{\sigma\in\Sigma}$ for an NDFA $M = \prod_{i=1}^{k} M_i$, where $M_i = (Q_i, \Sigma_i, \delta_i, q_i^0, F_i)$ is tractable in Section VI. These special cases cover the automata of Fig. 1(a). However, as we will see later, the observations of intractability of modular supervision, identified in the discussion in the introductory section that accompanied Fig. 1, remain unchanged for these special cases.

The set of *live* events at a state $q \in Q$ of an NDFA $M = (Q, \Sigma, \delta, q_0, F)$ is denoted as $\Gamma(q, M) \subseteq \Sigma$, where $\Gamma(q, M) = \{\sigma \in \Sigma \mid \exists \omega \in L_m(\widehat{M})$, where $\widehat{M} = (Q, \Sigma, \delta, q, F)$, and $\#(\sigma, \omega) \neq 0\}$. That is, $\Gamma(q, M) \subseteq \Sigma$ denotes the set of events in Σ that occur (as edge labels) in some path from q to one of the members in F in the graphical representation of M. It follows that $\forall\, q_2 \in \delta^*(q_1, \omega), \omega \in \Sigma^*, \Gamma(q_2, M) \subseteq \Gamma(q_1, M)$. Additionally, if $q_2 \in \delta(q_1, \sigma)$, for $\sigma \in \Sigma$, then

$$\Gamma(q_2, M) - \{\sigma\} \subseteq \Gamma(q_1, M) - \{\sigma\} - comp(\sigma, L_m(M))\}.$$

This is because $\Gamma(q_2, M) \subseteq \Gamma(q_1, M)$ and $(\widehat{\sigma} \in \Gamma(q_2, M) - \{\sigma\}) \Rightarrow (\widehat{\sigma} \notin comp(\sigma, L(M))$. The NDFA M is said to be *persistent* if $\forall\, q_2 \in \delta(q_1, \sigma)$,

$$\Gamma(q_2, M) - \{\sigma\} = \Gamma(q_1, M) - \{\sigma\} - comp(\sigma, L_m(M)). \quad (1)$$

In words, if an event σ_1 was live at q_1 but is no longer live at q_2, and if M is persistent we can conclude that the event σ (that caused the state transition from q_1 to q_2) must be a member of the complement set of σ_1. Alternately, the only way to make a live event in a persistent automaton, non-live at some subsequent state, is to execute an event from its complement set. For example, the automaton M_1 shown in Fig. 5(a) is persistent. The NDFA M_2 shown in Fig. 5(b) is not persistent, as

$$\underbrace{\Gamma(q_2, M_2)}_{=\emptyset} - \{b\} \neq \underbrace{\Gamma(q_0, M_2)}_{=\{a,b\}} - \{b\} - \underbrace{comp(b, L_m(M_2))}_{=\emptyset}.$$

From lemma 2.1 we note $\{comp(\sigma, L_m(M))\}_{\sigma\in\Sigma}$ can be computed in $O(card(\Sigma)^2 card(Q)^2)$ time. The set of states in Q that are connected to some member of $F \subseteq Q$ can be computed in $O(card(Q)^3)$ time. This can be done by adding a single, heretofore unused state, along with a directed edge from every member of $F \subseteq Q$ to this state. A state in Q is connected to a member of F if and only if it is connected to this new state in the modified graphical representation. Connectivity of a pair of vertices in a directed graph on n states can be checked in $O(n^2)$ time (cf. section 23.3, [8]), which means the set of all states in Q that are connected to some member of $F \subseteq Q$ can be tested in $O(card(Q)^3)$ time. For each $q \in Q$, the set $\Gamma(q, M)$ is the set of events (which appear as edge-labels) in a depth-first-search of the graphical representation of the subgraph of M involving just the members of Q that are connected to some member of $F \subseteq Q$, with q as the root vertex. This is an $O(card(Q)^2)$ operation. The set $\{\Gamma(q, M)\}_{q\in Q}$ can be computed in $O(card(Q)^3)$ time. This implies that the persistence of M can be tested in $O(\max\{card(\Sigma)^2 card(Q)^2, card(Q)^3\})$ time. The following lemma is about live events in a persistent NDFA.

Lemma 2.3: If $M = (Q, \Sigma, \delta, q_0, F)$ is a persistent NDFA, then $\forall\, q_1, q_2 \in \delta^*(q_0, \omega), \omega \in \Sigma^*, \Gamma(q_1, M) - \{\sigma \mid \#(\sigma, \omega) \neq 0\} = \Gamma(q_2, M) - \{\sigma \mid \#(\sigma, \omega) \neq 0\}$.

Proof: This is established by induction on $\mid \omega \mid$, the length of the string ω. The base case is established by letting ω equal the null-string ϵ. As the induction hypothesis, we assume the observation holds for any ω where $\mid \omega \mid = n$. For the induction step we consider $q_1, q_2 \in \delta^*(q_0, \omega\sigma)$, where $\mid \omega \mid = n$. Suppose $\widehat{q_1} \in \delta(q_0, \omega)$ ($\widehat{q_2} \in \delta(q_0, \omega)$) and $q_1 \in \delta(\widehat{q_1}, \sigma)$ ($q_2 \in \delta(\widehat{q_2}, \sigma)$). By the induction hypothesis $\Gamma(\widehat{q_1}, M) - \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega) \neq 0\} = \Gamma(\widehat{q_2}, M) - \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega) \neq 0\}$. From the persistence of M we have: $\Gamma(\widehat{q_1}, M) - \{\sigma\} - comp(\sigma, L_m(M)) = \Gamma(q_1, M) - \{\sigma\}$ and $\Gamma(\widehat{q_2}, M) - \{\sigma\} - comp(\sigma, L_m(M)) = \Gamma(q_2, M) - \{\sigma\}$. From the induction hypothesis and these two facts we have $\Gamma(q_1, M) - \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega\sigma) \neq 0\} = \Gamma(q_2, M) - \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega\sigma) \neq 0\}$. ∎

The import of lemma 2.3 is that the nondeterministic choice in the state-transition function does not pose a hindrance in deciding the set of live events when it comes to persistent NDFAs. The following lemma characterizes the set of live events at any state that is reachable from the initial state in a persistent NDFA.

Lemma 2.4: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a persistent NDFA. For any $q \in \delta^*(q_0, \omega)$, $\omega \in \Sigma^*$, $\Gamma(q, M) - \{\sigma \mid \#(\sigma, \omega) \neq 0\} = \Gamma(q_0, M) - \{\sigma \mid \#(\sigma, \omega) \neq 0\} - \bigcup_{\widehat{\sigma}\in\{\sigma \mid \#(\sigma,\omega)\neq 0\}} comp(\widehat{\sigma}, L(M))$.

Proof: This claim is established by an induction argument over $\mid \omega \mid$, the length of ω. The base case of induction is established by letting $\omega = \epsilon$, the empty string. For the induction hypothesis, we suppose the above claim is true for all strings of length n.

For the induction step, let $q_1 \in \delta^*(q_0, \omega)$, where $\mid \omega \mid = n$, and $q_2 \in \delta(q_1, \widetilde{\sigma})$. From the persistence of M we infer that $\Gamma(q_1, M) - \{\widetilde{\sigma}\} - comp(\widetilde{\sigma}, L_m(M)) = \Gamma(q_2, M) - \{\widetilde{\sigma}\}$. From the induction hypothesis we have: $\Gamma(q_1, M) - \{\sigma \mid \#(\sigma, \omega) \neq 1\} = \Gamma(q_0, M) - \{\sigma \mid \#(\sigma, \omega) \neq 0\} - \bigcup_{\widehat{\sigma}\in\{\sigma \mid \#(\sigma,\omega)\neq 0\}} comp(\widehat{\sigma}, L_m(M))$. In turn, we have $\Gamma(q_2, M) - \{\sigma \mid \#(\sigma, \omega\widetilde{\sigma}) \neq 1\} = \Gamma(q_0, M) - \{\sigma \mid \#(\sigma, \omega\widetilde{\sigma}) \neq 0\} - \bigcup_{\widehat{\sigma}\in\{\sigma \mid \#(\sigma,\omega\widetilde{\sigma})\neq 0\}} comp(\widehat{\sigma})$, ∎

The following lemma is about a requirement of complementary sets of events with respect to a persistent NDFA that marks a nonempty language.

Lemma 2.5: Let $M = (Q.\Sigma, \delta, q_0, F)$ be a persistent NDFA, where $L_m(M) \neq \emptyset$, and let $q_1 \in \delta^*(q_0, \omega_1) \cap \delta^*(q_0, \omega_2)$ for some $\omega_1, \omega_2 \in \Sigma^*$, then

$$\left(\bigcup_{\sigma \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_1) \neq 0\}} comp(\sigma, L_m(M)) \right) - \{\sigma \mid \#(\sigma, \omega_2) \neq 0\} = \left(\bigcup_{\sigma \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_2) \neq 0\}} comp(\sigma, L_m(M)) \right) - \{\sigma \mid \#(\sigma, \omega_1) \neq 0\}.$$

Proof: Since $L_m(M) \neq \emptyset$, it follows that $\forall\, \sigma \in \Sigma, \sigma \notin comp(\sigma, L_m(M))$. If $\widetilde{\sigma} \in comp(\sigma, L_m(M))$ for some $\sigma \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_1) \neq 0\}$, then $\widetilde{\sigma} \notin \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_1) \neq 0\}$, and if $\widetilde{\sigma} \notin \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_2) \neq 0\}$, it follows from lemma 2.4 that $\widetilde{\sigma} \notin \Gamma(q_1, M)$, which in turn implies that $\widehat{\sigma} \in comp(\sigma, L_m(M)$ for some $\widetilde{\sigma} \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_2) \neq 0\}$, or,

$$\left(\bigcup_{\sigma \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_1) \neq 0\}} comp(\sigma, L_m(M)) \right) - \{\sigma \mid \#(\sigma, \omega_2) \neq 0\} \subseteq \left(\bigcup_{\sigma \in \{\widehat{\sigma} \mid \#(\widehat{\sigma}, \omega_2) \neq 0\}} comp(\sigma, L_m(M)) \right) - \{\sigma \mid \#(\sigma, \omega_1) \neq 0\}.$$

Containment in the opposite direction can be shown by reversing the above argument. ∎

A binary-assignment on Σ, $\Upsilon : \Sigma \to \{0, 1\}$, is said to be *consistent* with respect to $L_m(M)$ if $\forall\, \sigma_1 \in \Sigma$, $((\Upsilon(\sigma_1) = 1) \Rightarrow (\forall\, \sigma_2 \in comp(\sigma_1, L_m(M)), \Upsilon(\sigma_2) = 0))$. For any binary-assignment $\Upsilon : \Sigma \to \{0, 1\}$ that is consistent with respect to $L_m(M)$, where M is persistent and $\Gamma(q_0, M) = \Sigma$, there is a $\omega \in L_m(M)$ such that $\forall\, \sigma \in \Sigma, \#(\sigma, \omega) \geq \Upsilon(\sigma)$. This is formally stated in lemma 2.6, which is a direct consequence of lemma 2.4.

Lemma 2.6: If $M = (Q, \Sigma, \delta, q_0, F)$ is a persistent NDFA such that $\Gamma(q_0, M) = \Sigma$, then for any consistent $\Upsilon : \Sigma \to \{0, 1\}$, there is an $\omega \in L_m(M)$ such that $\forall\, \sigma \in \Sigma, \#(\sigma, \omega) \geq \Upsilon(\sigma)$.

In Section III we present results that are relevant to SUP1M and SUPMM. Since these problems are stated in the deterministic-setting of supervisory control, the automata in Section III are deterministic finite-state automata (DFA), as opposed to the more general nondeterministic finite-state automata (NDFA) of this section.

III. Characterizing the Solution of SUP1M and SUPMM Instances

A discrete-state *plant* is represented as a DFA $G = (Q, \Sigma, \delta, q_0, F)$. The set of strings generated by the plant is denoted by $\mathcal{L}(G)(= L(G))$, and $\mathcal{L}_m(G)(= L_m(G))$ denotes the set of strings marked by the plant. The set Σ is partitioned into two subsets Σ_c and Σ_u, where the events in the set Σ_c (Σ_u) can (cannot) be prevented from occurring in G by the *supervisor*, represented as a function $\Psi : \Sigma^* \to \Xi$, where $\Xi = \{\alpha \in 2^{\Sigma} \mid \alpha \cap \Sigma_u = \Sigma_u\}$ is the set of control-inputs. $\Psi(\omega)$ is the set of events in Σ that are permitted to occur after the occurrence of $\omega \in \mathcal{L}(G)$ in G. The supervised behavior, denoted by $\mathcal{L}(G, \Psi)$, is defined as the smallest set that satisfies

1) $\epsilon \in \mathcal{L}(G, \Psi)$, where $\epsilon \in \Sigma^*$ denotes the empty string, and
2) $(\omega\sigma \in \mathcal{L}(G, \Psi)) \Leftrightarrow ((\omega \in \mathcal{L}(G, \Psi)) \wedge (\sigma \in \Psi(\omega)) \wedge (\omega\sigma \in \mathcal{L}(G)))$

The *marked* supervised behavior is represented by $\mathcal{L}_m(G, \Psi)$, where $\mathcal{L}_m(G, \Psi) = \mathcal{L}(G, \Psi) \cap \mathcal{L}_m(G)$.

The supervisory problem SUP11 [2] involves deciding if there is a $\Psi : \Sigma^* \to \Xi$, such that $\emptyset \subsetneq \mathcal{L}_m(G, \Psi) \subseteq L_m(S)$, where G and S are represented as DFAs. If all events in this plant are controllable, the existence of a supervisor Ψ reduces to the non emptiness of $L_m(G) \cap L_m(S)$. This observation forms the foundation of the proof of intractability of a generalization of SUP11, called SUP1M, described in the following paragraph, following a brief motivation.

There have been significant contributions to the theory of supervisory control since the appearance of the original papers on supervisory control [9]–[11], these can be found in the survey article [12], or, in texts on supervisory control [13], [14]. One of the obstacles to the incorporation of these theories into practice has been the requirement of a monolithic description of the plant G and specification S. As noted in [2], the specifications are rarely presented as a single automaton in applications. It is more likely that the specifications are presented as the meet of a collection of automata $\{S_i\}_{i=1}^m$. This scenario results in a modification where the input is a DFA G that represents the plant, and a set of DFAs $\{S_i\}_{i=1}^m$ that represents the specifications. In this case, the decision problem (SUP1M in [2]) involves the existence of a $\Psi : \Sigma^* \to \Xi$, such that $\emptyset \subsetneq \mathcal{L}_m(G, \Psi) \subseteq L_m(S)$, where $S = \bigwedge_{i=1}^m S_i$.

Using a plant G where $\Sigma_u = \emptyset$, and the observation that the existence of a supervisor Ψ that solves this instance of SUP1M is equivalent to the requirement that

$$L_m(G) \cap \underbrace{\left(\bigcap_{i=1}^{m} L_m(S_i) \right)}_{=L_m(S)} \neq \emptyset, \tag{2}$$

SUP1M is shown to be NP-hard in theorem 2 of [2], even when the automata in $\{S_i\}_{i=1}^m$ have only two states, as for example, the automata in Fig. 1(c). The presence of uncontrollable events does not improve the tractability of these problems. Motivated by (2), and the fact that the non emptiness of the intersection of the languages represented by an arbitrary number of DFAs is $PSPACE$-complete (cf. Lemma 3.2.3, [15]), Rohloff and Lafortune show that the decision problems associated with several variations of the modular supervisory control paradigm are $PSPACE$-complete [1]. The following lemma identifies a necessary and sufficient condition for the existence of a solution to an arbitrary instance of SUP1M.

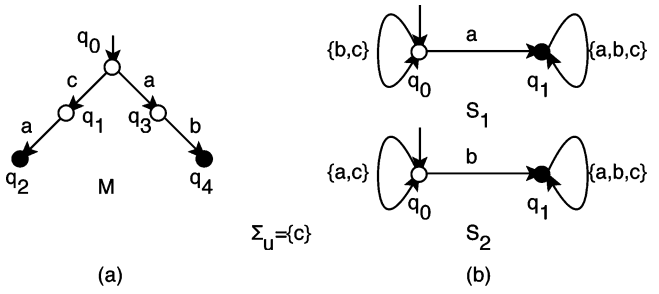

Fig. 6. (a) DFA G. (b) Specification DFAs S_1 and S_2 used in the illustration of lemma 3.1.

Lemma 3.1: Let G be a plant DFA, with a set of specification DFAs $\{S_i\}_{i=1}^m$. There is a supervisor $\Psi : \Sigma^* \to \Xi$, such that $\emptyset \neq \mathcal{L}_m(G,\Psi) \subseteq \bigcap_{i=1}^m L_m(S_i)$ if and only if

$$\exists A(\neq \emptyset) \subseteq L_m(G) \cap \left(\bigcap_{i=1}^m L_m(S_i)\right),$$

such that $\overline{A}\Sigma_u^* \cap L_m(G) \subseteq A$.

Proof: (*Only if*) If there is a supervisor $\Psi : \Sigma^* \to \Xi$ such that $\emptyset \neq \mathcal{L}_m(G,\Psi) \subseteq \bigcap_{i=1}^m L_m(S_i)$, we let $A = \mathcal{L}_m(G,\Psi)(\Rightarrow \overline{A} \subseteq \mathcal{L}(G,\Psi))$. It follows that $A \neq \emptyset$, $\overline{A}\Sigma_u^* \cap L_m(G) \subseteq A$, and $A \subseteq L_m(G) \cap (\bigcap_{i=1}^m L_m(S_i))$.

(*If*) Let $A(\neq \emptyset) \subseteq L_m(G) \cap (\bigcap_{i=1}^m L_m(S_i))$ satisfy the requirement $\overline{A}\Sigma_u^* \cap L_m(G) \subseteq A$. We use a supervisor $\Psi : \Sigma^* \to \Xi$, where for any $\sigma_c \in \Sigma_c$, for any $\omega \in \Sigma^*$, $\sigma_c \in \Psi(\omega)$ iff $\omega\sigma_c \in \overline{A}$. Using an induction argument over the length of ω, it can be shown that $\omega \in \mathcal{L}(G,\Psi) \Leftrightarrow \omega \in \overline{A}\Sigma_u^* \cap L(G)$. From this, we infer $\emptyset \neq (\mathcal{L}_m(G,\Psi) = A) \subseteq \bigcap_{i=1}^m L_m(S_i)$. ∎

As an illustration of lemma 3.1 consider the plant DFA $G = (Q,\Sigma,\delta,q_0,F)$, shown in Fig. 6(a), where members of $F \subseteq Q$ are identified by filled circles, $\Sigma_c = \{a,b\}$ and $\Sigma_u = \{c\}$. The specification automata $\{S_i\}_{i=1}^2$ are shown in Fig. 6(b). We have $A = L_m(S_1) \cap L_m(S_2) \cap L_m(G) = \{ab\}$, and $\overline{A}\Sigma_u^* \cap L_m(G) = \{ab\}(= A)$. A supervisor $\Theta : \Sigma^* \to \Xi$ that stops the occurrence of a after the first occurrence of c results in $\emptyset \neq \mathcal{L}_m(G,\Theta) \subseteq L_m(S_1) \cap L_m(S_2)$. On the other hand, if $\Sigma_u = \{a,c\}$, there can be no supervisory policy that will solve the instance of SUP1M in this figure.

To draw the distinction between lemma 3.1 and other properties in the literature, we note that $K_1 = L_m(S_1) \cap L_m(G)(= \{ca,ab\})$ satisfies the requirement $\overline{K_1}\Sigma_u \cap L(G) \subseteq \overline{K_1}$, while for $K_2 = L_m(S_2) \cap L_m(G)(= \{ab\})$, $\Sigma_u = \{c\}$, we have $\overline{K_2}\Sigma_u \cap L(G) \not\subseteq \overline{K_2}$. Additionally, $\overline{K_1 \cap K_2} = \overline{K_1} \cap \overline{K_2} = \overline{K_2}$, $\overline{K_1 \cap K_2} \cap L_m(G) = \{ab\} = K_1 \cap K_2$, and $(K_1 \cap K_2)^{\uparrow C} = \emptyset$ (cf. section 3.4, [13], for a formal definition of $(\bullet)^{\uparrow C}$).

Borrowing from the standard practice (cf. section 3.4, [13]; Section IV, [12], for example), from lemma 3.1, we conclude that there is a supervisor $\Psi : \Sigma^* \to \Xi$ such that $\emptyset \neq \mathcal{L}_m(G,\Psi) \subseteq \bigcap_{i=1}^m L_m(S_i)$ iff $(\bigcap_{i=1}^m K_i)^{\uparrow G} \neq \emptyset$, where

$$\left(\bigcap_{i=1}^m K_i\right)^{\uparrow G} := \bigcup_{\left\{A \subseteq \bigcap_{i=1}^m K_i \mid \overline{A}\Sigma_u^* \subseteq A\right\}} A,$$

and $K_i = L_m(S_i) \cap L_m(G)$, $i \in \{1,2,\ldots,m\}$. If each $K_i = L_m(S_i) \cap L_m(G)$ satisfies the requirement $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $i \in \{1,2,\ldots,m\}$, then

$$\left(\bigcap_{i=1}^m K_i\right)^{\uparrow G} = \left(\bigcap_{i=1}^m K_i\right).$$

To show this we observe that if $\omega \in \overline{(\bigcap_{i=1}^m K_i)}\Sigma_u^* \cap L_m(G)$, then $\omega = \omega_1\omega_2$, where $\omega_2 \in \Sigma_u^*$ and $\omega_1 \in \overline{(\bigcap_{i=1}^m K_i)} \Rightarrow \omega_1 \in \left(\bigcap_{i=1}^m \overline{K_i}\right)$ (as $\overline{(\bigcap_{i=1}^m K_i)} \subseteq \left(\bigcap_{i=1}^m \overline{K_i}\right)$). This, together with the fact that $\omega_1\omega_2 \in L_m(G)$, where $\omega_2 \in \Sigma_u^*$, and the fact that each $K_i = L_m(S_i) \cap L_m(G)$ satisfies the requirement that $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $i \in \{1,2,\ldots,m\}$, results in the $\omega_1\omega_2 \in \bigcap_{i=1}^m K_i$. This leads us to the following result.

Lemma 3.2: Let $G = (Q,\Sigma,\delta,q_0,F)$ and $\{S_i\}_{i=1}^m$ denote the plant and specification automata for an instance of SUP1M. If each $K_i = L_m(S_i) \cap L_m(G)$ satisfies the requirement $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $i \in \{1,2,\ldots,m\}$, then there exists a supervisor $\Psi : \Sigma^* \to \Xi$ such that $\emptyset \neq \mathcal{L}_m(G,\Psi) \subseteq \bigcap L_m(S_i)$ iff $\bigcap_{i=1}^m K_i \neq \emptyset$.

There is an $O(card(\widehat{Q_i})^3 card(Q)^3)$ algorithm that can test $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $S_i = \left(\widehat{Q_i},\Sigma,\widehat{\delta_i},\widehat{q_0^i},\widehat{F_i}\right)$ and $K_i = L_m(S_i) \cap L_m(G)$. The first step in this process involves checking if $K_i = \emptyset$, which can be done in $O(card(Q)^2 card(\widehat{Q_i})^2)$ time. If $K_i = \emptyset$, then $\overline{K_i}\Sigma_u^* = \emptyset$, and $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$. If $K_i \neq \emptyset$, an automaton that recognizes $\overline{K_i}\Sigma_u^*$ can be constructed from an automaton that recognizes K_i as follows.

1) Each state of the automaton that recognizes K_i that is connected to a member of F is declared as a final-state. This is a $O(card(Q)^3 card(\widehat{Q_i})^3)$ operation, as noted in an earlier construction,
2) A (single) new state $\widetilde{q}$ is added to this set of final states, and the state-transition function for the automaton that recognizes K_i is augmented in a way that its graphical description results in a self-loop around $\widetilde{q}$ for any member of Σ_u, and
3) If the graphical description of the automaton has a state that is connected to a member of F with no outgoing arc with a label that corresponds to some $\sigma_u \in \Sigma_u$, then the state transition function is augmented so as to yield an outgoing arc from that state to $\widetilde{q}$ with label σ_u.

This results in a recognizer for $\overline{K_i}\Sigma_u^*$ that has $O(card(\widehat{Q_i}) \times card(Q))$ states, which can be constructed in $O(card(\widehat{Q_i}) \times card(Q))$ time. An automaton that recognizes $\overline{K_i}\Sigma_u^* \cap L_m(G)$ can be constructed from this automaton in $O(card(\widehat{Q_i}) \times card(Q)^2))$ time. Since an automaton that recognizes $\Sigma^* - K_i$ can be constructed in $O(card(\widehat{Q_i}) \times card(Q))$ time, the emptiness of $\left(\overline{K_i}\Sigma_u^* \cap L_m(G)\right) \cap (\Sigma^* - K_i)$ can be tested in $O(card(\widehat{Q_i})^2 \times card(Q)^3)$ time. The steps described above can be completed in $O(card(\widehat{Q_i})^3 card(Q)^3)$ time. If there is a fixed upper bound on the number of states in the specification automata (as we will see in subsequent text), it follows that the implicant of lemma 3.2 can be tested in $O(m \times card(Q)^3)$ time.

A variation on SUP1M could be when the plant G is the synchronous product of a set of automata $\{G_i\}_{i=1}^k$. That is, $G =$

$\prod_{i=1}^{k} G_i$, and $S = \bigwedge_{i=1}^{m} S_i$. The decision problem (SUPMM in [2]) is defined analogously, where the input are the sets $\{G_i\}_{i=1}^{k}$ and $\{S_i\}_{i=1}^{m}$. This problem is $PSPACE$-complete in general [1], and it remains intractable (cf. theorem 1, [2]) even when G_i and S_i belong to a collection of 2-state DFAs of the types shown in Fig. 1(a) and (c) respectively. Of particular note, the intractability remains even when there are no events that are common between the automata in the set $\{G_i\}_{i=1}^{k}$.

The results of lemma 3.2 can be applied to instances of SUPMM after the automata in $\{G_i\}_{i=1}^{k}$ are appropriately modified to have a common alphabet. Let $G_i = (Q_i, \Sigma_i, \delta_i, F_i)$ for each G_i in $\{G_i\}_{i=1}^{k}$. The structure of G_i can be augmented to include symbols in $\left(\bigcup_{i=1}^{k} \Sigma_i\right) - \Sigma_i$ with the use of self-loops as follows – $\forall\, \sigma \in \left(\bigcup_{i=1}^{k} \Sigma_i\right) - \Sigma_i$, $\forall\, q_i \in Q_i$, $\delta_i(q_i, \sigma) = q_i$. Suppose the resulting automaton is denoted by $\widehat{G}_i$, then $L_m(G) = \bigcap_{i=1}^{k} L_m(\widehat{G}_i)$. For $i \in \{1, 2, \ldots m\}$ and $j \in \{1, 2, \ldots, k\}$, let $K_{i,j} = L_m(S_i) \cap L_m(\widehat{G}_j)$, and $\overline{K}_{i,j}\Sigma_u^* \cap L_m(\widehat{G}_i) \subseteq K_{i,j}$. Then there exists a supervisor $\Psi : \Sigma^* \to \Xi$ such that $\emptyset \neq \mathcal{L}_m(G, \Psi) \subseteq \bigcap L_m(S_i)$ iff $\bigcap_{i=1}^{m} \bigcap_{j=1}^{k} K_{i,j} \neq \emptyset$. Using the same argument as before, there is a $O\left(card(\widehat{Q}_i)^2 card(Q_j)^3\right)$ algorithm that can test $\overline{K}_{i,j}\Sigma_u^* \cap L_m(\widehat{G}_i) \subseteq K_{i,j}$. If the number of states in each automaton in the sets $\{G_i\}_{i=1}^{k}$ and $\{S_i\}_{i=1}^{m}$ are bounded (for example, like those in Fig. 1(a) and (c), they contain only two states), there is a $O(mk)$ test for

$$\forall\, i \in \{1, 2, \ldots, m\}$$
$$\forall\, j \in \{1, 2, \ldots, k\}, \overline{K}_{i,j}\Sigma_u^* \cap L_m(\widehat{G}_i) \subseteq K_{i,j}.$$

The tractability of testing the existence of a supervisory policy that solves this subclass of SUPMM is determined by the tractability of testing $\bigcap_{i=1}^{m} \bigcap_{j=1}^{k} K_{i,j} \neq \emptyset$. In Section V we show there is a tractable test for the existence of a supervisory policy for a subclass of SUPMM where the automata G_i share no common events and the automata S_i belong to a family of automata Ω defined as follows—if $S_i \in \Omega$, then $S_i = (\{q_0, q_1\}, \Sigma, \delta_i, q_0, \{q_1\})$, where $\forall\, \sigma \in \Sigma, \delta_i(q_1, \sigma) = q_1$, and $\forall\, \widehat{\sigma} \notin \{\sigma \in \Sigma \mid \delta_i(q_0, \sigma) = q_1\}$, $\delta(q_0, \widehat{\sigma}) = q_0$. That is, each DFA $S_i \in \Omega$ requires the occurrence of at least one event in the set $Event(S_i) = \{\sigma \in \Sigma \mid \delta_i(q_0, \sigma) = q_1\}$.

First, we concern ourselves with a subclass of SUP1M, which we denote as SUP1Ω, where the plant $G = (Q, \Sigma, \delta, q^0, F)$ is a persistent DFA, and $\Gamma(q_0, M) = \Sigma$. The specification automata $\{S_i\}_{i=1}^{m}$ belong to the class Ω identified above. Additionally, we assume each $K_i = L_m(S_i) \cap L_m(G)$, $i \in \{1, 2, \ldots, m\}$, satisfies the requirement $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$. As shown earlier, these prerequisites can be checked in polynomial time, and the existence of a supervisor $\Psi : \Sigma^* \to \Xi$ that solves this sub-class of SUP1M reduces to testing the non emptiness of $\bigcap_{i=1}^{m} K_i = L_m(G) \cap (\bigcap_{i=1}^{m} L_m(S_i))$. The construction that establishes the intractability of SUP1M in [2], satisfies the above requirements, and the testing the existence of a supervisory for an arbitrary instance of SUP1Ω remains intractable. As a consequence the search for tractable instances would require us to further constrain the class of SUP1Ω instances in addition to restrictions described above. To this end we identify a tractable subclass of SUP1Ω in Section IV where the specification automata $\{S_i\}_{i=1}^{m}$ belong to a subclass $\Omega_{2SAT} \subset \Omega$. The subclass of SUPMM where the specification automata are from Ω is denoted by SUPMΩ in subsequent test. We conclude this section by presenting a few observations that relate the existence of consistent binary-assignments and the non emptiness of $\bigcap_{i=1}^{m} K_i$ for the subclass SUP1Ω identified above.

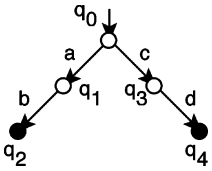

Fig. 7. Persistent DFA $G = (Q, \Sigma, \delta, q^0, \{q_2, q_4\})$, where $\Gamma(q_0, G) = \{a, b, c, d\}$, and $comp(a, L_m(G)) = comp(b, L_m(G)) = \{c, d\}$ and $comp(c, L_m(G)) = comp(d, L_m(G)) = \{a, b\}$.

A consistent, binary-assignment $\Upsilon : \Sigma \to \{0, 1\}$ is said to *satisfy* the set of specifications $\{S_i\}_{i=1}^{m}$ if and only if for each $S_i \in \{S_i\}_{i=1}^{m}$, $\exists \sigma \in Event(S_i)$ such that $\Upsilon(\sigma) = 1$.

We now highlight points of divergence between literals in a combinatorial formula, and the complement set of events introduced in Section II. In the context of combinatorial formulae, the complement-set of each literal is a singleton, and the intersection of the complement-set of two distinct literals is always empty. In contrast, $comp(a, L_m(G)) = comp(b, L_m(G)) = \{c, d\}$ and $comp(c, L_m(G)) = comp(d, L_m(G)) = \{a, b\}$, where $G = (Q, \Sigma, \delta, q^0, \{q_2, q_4\})$ is the persistent DFA shown in Fig. 7. Note that $\Gamma(q_0, G) = \Sigma = \{a, b, c, d\}$. As a consequence, observations that might be routine in the context of combinatorial logic require further elaboration in the context of supervisory control.

Lemma 3.3: If $G = (Q, \Sigma, \delta, q_0, F)$ is a persistent DFA such that $\Gamma(q_0, G) = \Sigma$, then $\bigcap_{i=1}^{m} K_i = L_m(G) \cap (\bigcap_{i=1}^{m} L_m(S_i)) \neq \emptyset$ if and only if there is a consistent binary-assignment that satisfies the set of specifications $\{S_i\}_{i=1}^{m}$.

Proof: (*Only if*) Suppose $\omega \in L_m(G) \cap (\bigcap_{i=1}^{m} L_m(S_i))$, the consistency of the binary assignment

$$\Upsilon(\sigma) = \begin{cases} 0, & \text{if } \#(\sigma, \omega) = 0 \\ 1, & \text{otherwise} \end{cases}$$

follows from the fact that for any $\omega \in L_m(G)$, if $\sigma \in \{\widehat{\sigma} \in \Sigma \mid \#(\sigma, \omega) \neq 0\}$, then no member of $comp(\sigma, L_m(G))$ can occur in ω. Since $\omega \in \bigcap_{i=1}^{m} L_m(S_i)$, it follows that for every $S_i \in \{S_i\}_{i=1}^{m}$, $\exists \sigma \in Event(S_i)$ such that $\#(\sigma, \omega) \neq 0$, it follows that the binary-assignment shown above satisfies $\{S_i\}_{i=1}^{m}$.

(*If*) Let $\Upsilon : \Sigma \to \{0, 1\}$ be a consistent binary assignment that satisfies $\{S_i\}_{i=1}^{m}$. From lemma 2.6 we know $\exists \omega \in L_m(G)$ such that $\forall\, \sigma \in \Sigma$, $\#(\sigma, \omega) \geq \Upsilon(\sigma)$. Additionally, since Υ satisfies $\{S_i\}_{i=1}^{m}, \omega \in L_m(S_i), \forall\, i \in \{1, 2, \ldots, m\}$. Therefore, $\bigcap_{i=1}^{m} K_i = L_m(G) \cap (\bigcap_{i=1}^{m} L_m(S_i)) \neq \emptyset$. ■

Lemma 3.4: For the subclass SUP1Ω, if $\bigcap_{i=1}^{m} K_i \neq \emptyset$ and G is the plant automaton, then there is a consistent binary assignment $\Upsilon : \Sigma \to \{0, 1\}$ that satisfies $\{S_i\}_{i=1}^{m}$, and $\forall\, \sigma \in \Sigma$, $\exists \widehat{\sigma} \in \{\sigma\} \cup comp(\sigma, L_m(G))$ such that $\Upsilon(\widehat{\sigma}) = 1$.

Proof: If $\widehat{\Upsilon} : \Sigma \rightarrow \{0,1\}$ is a consistent binary assignment that satisfies $\{S_i\}_{i=1}^m$, and $\widehat{\Upsilon}(\widehat{\sigma}) = 0$, $\forall\ \widehat{\sigma} \in \{\sigma\} \cup comp(\sigma, L_m(G))$. It must be that

$$\left(\bigcup_{i=1}^{m} Event(S_i)\right) \cap \underbrace{(\{\sigma\} \cup comp(\sigma, L_m(G)))}_{=B} = \emptyset.$$

Additionally,

$$\underbrace{B - \{\widehat{\sigma} \in B \mid \exists \widetilde{\sigma} \in comp(\widehat{\sigma}, L_m(G), \text{ such that } \widehat{\Upsilon}(\widetilde{\sigma}) = 1\}}_{=C} \neq \emptyset,$$

as $\sigma \in C$. Any refinement of $\widehat{\Upsilon}$ where a member of C is assigned a non-zero value, would be a consistent assignment that satisfies $\{S_i\}_{i=1}^m$. This process can be repeated as often as necessary till the conditions of lemma 3.4 are met. ∎

IV. A Tractable Sub-Class of SUP1Ω Inspired by the $2SAT$ Problem

In this section we consider specifications that are expressed as the meet of automata from the set $\Omega_{2SAT} \subset \Omega$, where

$$\Omega_{2SAT} = \{S_i \in \Omega \mid card(Event(S_i)) \leq 2\}.$$

Each $S_i \in \Omega_{2SAT}$ must have a structure similar to one of the DFAs shown in Fig. 8. If S_j is similar to the DFA in Fig. 8(b) then it is imperative that the event σ_3 occurs in G, this in turn would mean that none of the events in $comp(\sigma_3, L_m(G))$ can occur. Any specification automaton $S_j \in \{S_i\}_{i=1}^m$ where $\sigma_3 \in Event(S_j)$, is removed from the set $\{S_i\}_{i=1}^m$. Additionally, modifications are made to the state-transition functions of members in $\{S_i\}_{i=1}^m$ that results in the removal of edges labeled with any member of $comp(\sigma_3, L_m(G))$ from their graphical descriptions. This process is repeated as often as necessary until all members of the set $\{S_i\}_{i=1}^m$ have a structure similar to the automaton shown in Fig. 8(a) or Fig. 8(c). If the set $\{S_i\}_{i=1}^m$ has any automaton that is similar in structure to the one shown in Fig. 8(c), we conclude that the original set of specifications are un-enforceable. We only need to address the case when all members of $\{S_i\}_{i=1}^m$ have a structure similar to the automaton shown in Fig. 8(a) at the conclusion of this process. This is a parallel to *unit-resolution*, where the literal associated with an unit-clause in a CNF is posited, and appropriate modifications made to the other clauses (cf. chapter 2, [16]). If this process results in an empty clause, the original CNF formula is deemed unsatisfiable. Any CNF whose satisfiability is to be tested is therefore assumed to be free of unit-clauses. Just as with unit-resolution, the process of eliminating automata of the type shown in Fig. 8(b) and (c) can be done in $O(m)$ time. For the remainder of this section we assume we are given a set of specification automata $\{S_i\}_{i=1}^m$, where each member of the set has a structure that is similar to the one shown in Fig. 8(a).

The *implication graph*, $IG(G, \{S_i\}_{i=1}^m) = (V, A)$, is a directed graph with a vertex set V, where each vertex $v \in V$ is associated with a member of Σ via a labeling-bijection $\beta : V \rightarrow \Sigma$. The edge set $A \subseteq V \times V$ is constructed as follows—if $Event(S_i) = \{\sigma_1, \sigma_2\}$, then $\forall\ \sigma_3 \in comp(\sigma_1, L_m(G))$ ($\forall\ \sigma_4 \in comp(\sigma_2, L_m(G))$) there is an arc from $\beta^{-1}(\sigma_3)$ to $\beta^{-1}(\sigma_2)$ ($\beta^{-1}(\sigma_4)$ to $\beta^{-1}(\sigma_1)$).

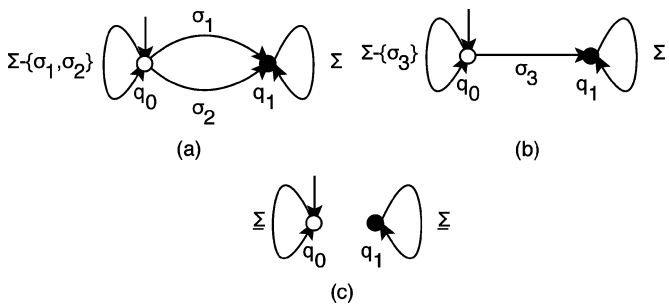

Fig. 8. Representatives of the specification class Ω_{2SAT}.

The import of the implication graph is this—if there is a directed path from vertex v_1 vertex v_2, then any consistent binary assignment that satisfies $\{S_i\}_{i=1}^m$ that assigns a value of 1 (0) to $v_1(v_2)$ must assign a value of 1 (0) to $v_2(v_1)$. The semantics of binary-assignment is that an event is assigned a value of 1 (0) if it occurs (does not occur). If $\sigma_3 \in comp(\sigma_1, L_m(G))$ is assigned a value of 1, then the event σ_1 cannot occur. Therefore, if S_i is to be satisfied, then it is imperative that σ_2 be assigned a value of 1, as $Event(S_i) = \{\sigma_1, \sigma_2\}$. A similar argument establishes the fact that if σ_2 is assigned a value of 0, then it is imperative that none of the members in $comp(\sigma_1, L_m(G))$ be assigned a value of 1, as this would prevent the occurrence of σ_1 and consequently S_i would remain unsatisfied. Any consistent binary assignment that satisfies $\{S_i\}_{i=1}^m$ must satisfy the implication graph.

This construction is often used in establishing the tractability of $2SAT$ in combinatorial logic. Since each literal has only one complement, the structure of the implication graph in the context of combinatorial logic has properties that can be exploited in the tractable resolution of any instance of $2SAT$ (cf. section2.4.1, [16]). Since these properties are not necessarily true for the discourse in this paper (cf. the discussion in Section III pertaining to Fig. 7), we resort to alternate approaches to tractably resolve the subclass of SUP1Ω of this section.

Any consistent binary assignment that satisfies $\{S_i\}_{i=1}^m$ must be a refinement of $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$ where $\widetilde{\Upsilon}(\sigma) = 0$ if there are directed paths from $\beta^{-1}(\sigma)$ to $\beta^{-1}(\sigma_1)$, and $\beta^{-1}(\sigma)$ to $\beta^{-1}(\sigma_2)$ in $G(G, \{S_i\}_{i=1}^m)$, where $\sigma_1 \in comp(\sigma_2, L_m(G))$. These assignments can be done in $O(card(\Sigma)^3)$ time, as the connectivity of any two vertices in a graph on n vertices can be tested in $O(n^2)$ time (cf. section 23.3, [8]). The binary assignment $\widetilde{\Upsilon}$ is used in the main result of this section, which is stated below.

Theorem 4.1: Let $G = (Q, \Sigma, \delta, q^0, F)$ be a persistent plant DFA where $\Gamma(q_0, G) = \Sigma$, and each specification automaton from the set $\{S_i\}_{i=1}^m$ is from Ω_{2SAT}. If $K_i = L_m(S_i) \cap L_m(G)$, $i \in \{1, 2, \ldots, m\}$, then $\bigcap_{i=1}^m K_i = \emptyset$ iff $\exists \sigma \in \Sigma$ such that $\forall\ \widehat{\sigma} \in \{\sigma\} \cup comp(\sigma, L_m(G))$, $\widetilde{\Upsilon}(\widehat{\sigma}) = 0$.

Proof: (*If*) As noted in the definition, $\widetilde{\Upsilon}(\sigma) = 0$, if $\beta^{-1}(\sigma)$ is connected to $\beta^{-1}(\sigma_1)$ and $\beta^{-1}(\sigma_2)$, where $\sigma_1 \in comp(\sigma_2, L_m(G))$. The alternative would be the assignment of 1 to σ, which would require that σ_1 and σ_2 be assigned a value of 1 too, which would be inconsistent, and there cannot

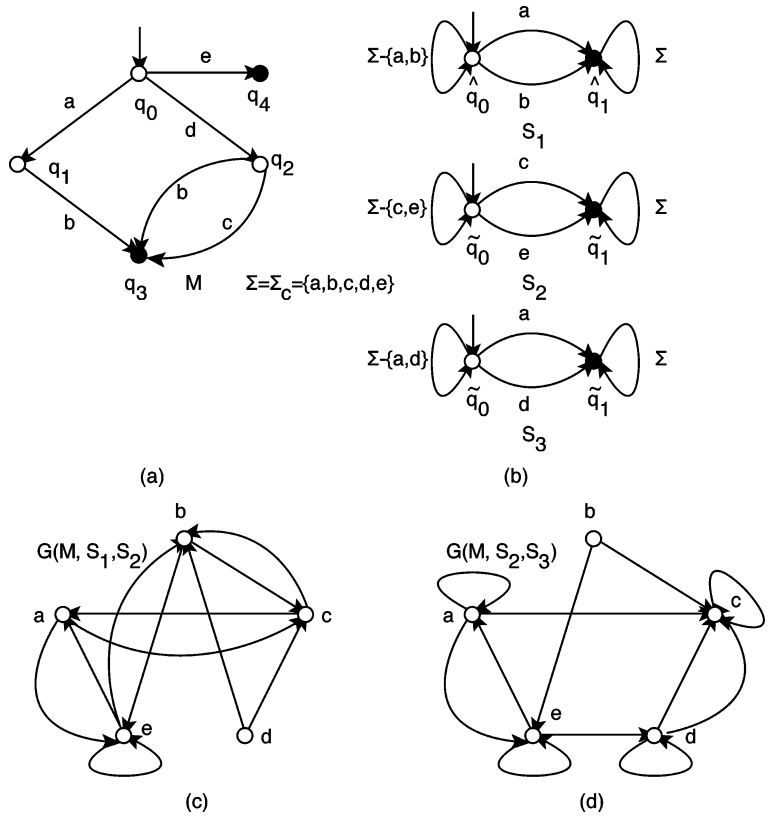

Fig. 9. (a) Plant $G = (Q, \Sigma, \delta, q_0, F)$ that is persistent and $\Gamma(q_0, G) = \Sigma(= \{a, b, c, d, e\})$. (b) Three specification automata S_1, S_2 and S_3 that belong to Ω_{2SAT}. (c) Implication graph $IG(G, S_1, S_2)$. (d) Implication graph $IG(G, S_2, S_3)$.

be an $\omega \in L_m(G)$ that contains σ_1 and σ_2. The implication follows directly from lemma 3.3 and the contrapositive assertion of lemma 3.4.

(*Only if*) We show that there is a consistent refinement of $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$ that satisfies the implication graph $IG(G, \{S_i\}_{i=1}^m)$. Consequently, this consistent refinement satisfies $\{S_i\}_{i=1}^m$, and the implication follows from lemma 3.3.

If $\exists \sigma_1 \in \Sigma$ such that $\widetilde{\Upsilon}(\sigma_1)$ is undefined, then $\forall\ \sigma_2 \in \Sigma$ such that there is a directed path from $\beta^{-1}(\sigma_1)$ to $\beta^{-1}(\sigma_2)$ in $G(G, \{S_i\}_{i=1}^m)$, $\widetilde{\Upsilon}(\sigma_2)$ is either 1 or undefined (i.e., it cannot be 0). In addition, it must be that $\forall\ \sigma_3 \in comp(\sigma_2, L_m(G))$, either $\widetilde{\Upsilon}(\sigma_3)$ is either 0 or is undefined (i.e., it cannot be 1), as $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$ is a consistent binary assignment.

If $\exists \sigma_1 \in \Sigma$ such that $\widetilde{\Upsilon}(\sigma_1)$ is undefined, then $\forall\ \sigma_4 \in \Sigma$ such that there is a directed path from $\beta^{-1}(\sigma_1)$ to $\beta^{-1}(\sigma_4)$ in $G(G, \{S_i\}_{i=1}^m$, and $\widetilde{\Upsilon}(\sigma_4)$ is undefined, we refine $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$ to include $\widetilde{\Upsilon}(\sigma_4) = 1$. This is followed by ensuring $\widetilde{\Upsilon}(\sigma_5) = 0$, for each $\sigma_5 \in comp(\sigma_4, L_m(G))$. As noted above these steps are well-defined, and $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$ remains consistent after refinement. This process is repeated as often as necessary till every member of Σ is assigned a value of 0 or 1 by $\widetilde{\Upsilon}$. The claim follows from lemma 3.3. ■

As noted in Section II, the set $\{comp(\sigma, L_m(G)\}_{\sigma \in \Sigma}$ can be computed in $O(card(\Sigma)^2 card(Q)^2)$ time, and there is an $O(card(\Sigma)^3)$ algorithm that assigns the initial binary assignment for $\widetilde{\Upsilon} : \Sigma \rightarrow \{0,1\}$. A $O(card(\Sigma)^3)$ search can ascertain if there are any candidates for $\sigma_1 \in \Sigma$, where $\sigma_2 \in \{\sigma_1\} \cup comp(\sigma_1, L_m(G))$, $\widetilde{\Upsilon}(\sigma_2) = 0$. We conclude there is a $O(\max\{card(\Sigma)^3, card(\Sigma)^2 card(Q)^2\})$ algorithm that decides if $\bigcap_{i=1}^m K_i = \emptyset$. This leads us to the following observation about the tractability of the subclass of SUP1Ω introduced in this section.

Lemma 4.2: There is a polynomial-time algorithm that decides if there is a solution to an instance of a subclass of SUP1Ω that (i) involves a persistent plant DFA $G = (Q, \Sigma, \delta, q_0, F)$, where $\Gamma(q_0, G) = \Sigma$, (ii) each member in the set of specification DFAs $\{S_i\}_{i=1}^m$ is from Ω_{2SAT}, and (iii) if $\forall\ i \in \{1, 2, \ldots, m\}$, $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $K_i = L_m(S_i) \cap L_m(G)$.

As an illustration consider the plant G of Fig. 9(a) and the specification automata S_1, $S_2 \in \Omega_{2SAT}$ of Fig. 9(b). We have $comp(a, L_m(G)) = \{c, d, e\}$, $comp(b, L_m(G)) = \{c, e\}$, $comp(c, L_m(G)) = \{a, b, e\}$, $comp(d, L_m(G)) = \{a, e\}$ and $comp(e, L_m(G)) = \{a, b, c, d\}$. Let us suppose $\Sigma_u = \emptyset$, then $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $K_i = L_m(S_i) \cap L_m(G)$ for $i = 1, 2$. Testing the existence of a solution for this instance of SUP1Ω would require us to construct the implication graph $IG(G, S_1, S_2)$ shown in Fig. 9(c). For each $\sigma \in \{a, b, c, d, e\}$, there is a path from σ to a member of $comp(\sigma, L_m(G))$ in $G(G, S_1, S_2)$, any consistent binary assignment that is intended to satisfy S_1, S_2, would assign a value of 0 to all events. From theorem 4.1 we conclude there can be no solution to this instance of SUP1Ω.

As a variation of the above problem, the specification automaton S_1 is replaced by S_3 of Fig. 9(b). Let us suppose $\Sigma_u = \{c, d\}$. It can be shown that $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $K_i = L_m(S_i) \cap L_m(G)$ for $i = 2, 3$. The implication graph

$IG(G, S_2, S_3)$ is shown in Fig. 9(d). Since a, b and e are connected to some member in their complement set, any consistent binary assignment that satisfies S_1 and S_2 will assign a value of 0 to them. Elements c and d can be assigned a value of 1 without violating any consistency requirements. So, from theorem 4.1 we infer there is a solution to this instance of SUP1Ω. A supervisory policy Ψ that permanently disables all members of $\Sigma_c = \{a, b, e\}$ will solve the problem at hand.

In Section V we develop a polynomial time hierarchy of instances of SUP1Ω that uses the class of problems of this section as the base-class. These results can also be used to identify a polynomial time hierarchy for intractable subclasses of SUPMΩ that contain the instance where the plant is described by the synchronous product of the automata of the kind shown in Fig. 1(a), and the specification automata are like the automata shown in Fig. 1(c).

V. A Family of Polynomially Solvable Instances of SUP1Ω and SUPMΩ

Gallo and Scutella [4] identify a family of subclasses of the SAT-problems: $\Gamma_0 \subseteq \Gamma_1 \subseteq \Gamma_k \subseteq \cdots$, where the *base class* $\Gamma_0 = HORNSAT$, and $\lim_{k\to\infty} \Gamma_k$ is the set of all possible SAT-instances. The problem instances in $\Gamma_k, k \in \{1, 2, \ldots\}$ are solvable in $O(n^* n^k)$ time, where n is the number of boolean variables, m is the number of clauses, and $n^* = O(mn)$ is the size of the formula $\phi(\bullet)$. This result was generalized by Pretolani [17], who has shown that under appropriate conditions, a similar polynomial-time hierarchy can be developed using base classes that are not necessarily $HORNSAT$.

The remainder of this section is about a similar hierarchy for SUPΩ where the plant DFA $G = (Q, \Sigma, \delta, q_0, F)$ is persistent, and $\Gamma(q_0, G) = \Sigma$. The specification automata $\{S_i\}_{i=1}^m$ are from the set Ω defined in Section III. We also require $\overline{K_i} \cap \Sigma_u^* \subseteq K_i$, $i \in \{1, 2, \ldots, m\}$, where $K_i = L_m(S_i) \cap L_m(G)$.

Borrowing heavily from Gallo and Scutella [4], we present a family of subclasses $\Pi_0 \subseteq \Pi_1 \subseteq \cdots, \Pi_k \subseteq \cdots$, such that the base-class Π_0 corresponds to the class of problems solved in Section IV. For an instance that belongs to Π_k, we show there is an $O(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^{k+3})\})$ algorithm that decides the existence of a supervisory policy that solves the problem at hand.

Let $S = \{Event(S_i)\}_{i=1}^m$, and for $\sigma \in \Sigma$, we define

$$S_{\{\sigma\}} := S - \{X \in S \mid \sigma \in X\}$$
$$S\Theta\{\sigma\} := \{X - \{\sigma\} \mid X \in S\}.$$

Note that if we deleted all members of S that contained $\sigma \in \Sigma$, we get the set $S_{\{\sigma\}}$. In the context of this paper, this would correspond to the process of removing all members of the specification set $\{S_i\}$ that are satisfied when the event σ occurs in the plant. In contrast, when an event in $comp(\sigma, L_m(G))$ occurs in the plant, the event σ cannot occur. This will require the removal of arcs labeled σ in the graphical description of the automata in $\{S_i\}_{i=1}^m$. If the resulting set of automata is denoted by $\{\widehat{S}_i\}_{i=1}^m$, then $S\Theta\{\sigma\}$ is the set $Event\left(\{\widehat{S}_i\}_{i=1}^m\right)$.

We define the classes $\Pi_0, \Pi_1, \ldots, \Pi_k, \ldots$ recursively as follows:

$$S \in \Pi_0 \Leftrightarrow X \in S \Rightarrow card(X) \leq 2;$$
$$S \in \Pi_k \Leftrightarrow \exists \sigma \in \Sigma, \text{ such that } S_{\{\sigma\}} \in \Pi_{k-1} \;\&\; S\Theta\{\sigma\} \in \Pi_k.$$

If the empty set belongs to all classes $\Pi_0, \Pi_2, \ldots$ by assumption, then we have $\Pi_{k-1} \subseteq \Pi_k$, $k \geq 1$. So, $\exists h$, such that $S \in \Pi_k$, $\forall\ k \geq h$. For instance, $S = \{\{x_1, x_2, \neg x_5\}, \{x_5, x_3, \neg x_4\}, \{x_7, \neg x_6, \neg x_5\}\}$ for the instance of SUP1M in Fig. 1 belongs to Π_2.

For $S = \{Event(S_i)\}_{i=1}^m$, an event $\sigma \in Event(S_j)$ $(1 \leq j \leq m)$, is a *k-candidate* for S iff $S_{\{\sigma\}} \in \Pi_{k-1}$, for $k \geq 1$. The following lemma from [4] is about the existence of k-candidates for any member in the hierarchy $\Pi_0 \subseteq \Pi_1 \subseteq \cdots \subseteq \Pi_k \subseteq \cdots$ described above.

Lemma 5.1: (*Lemma 2.1,* [4]) For $k \geq 0$, $S = \{Event(S_i)\}_{i=1}^m$, $S \in \Pi_k$ iff there is a sequence $(\sigma_1, \sigma_2, \ldots, \sigma_p)$ where $\sigma_j \in Event(S_k)$ $(1 \leq k \leq m, 1 \leq j \leq p)$, such that for $S^j = S^{j-1}\Theta\{\sigma_{j-1}\}$, with $S^1 = S$, (i) $S^j_{\{\sigma_j\}} \in \Pi_{k-1}$ (i.e., σ_j is a k-candidate for S^j), and (ii) $S^{p+1} = \emptyset$.

Lemma 5.2, which is also from [4], states that the removal of an arbitrary $\sigma \in \Sigma$ from any member of S that contains it (i.e., the operation $S\Theta\{\sigma\}$) will yield a set that is in the same level of the hierarchy that S originally belonged to. This is followed by an lemma about the existence of a polynomial time algorithm that decides if $S \in \Pi_k$.

Lemma 5.2: (*Lemma 2.2,* [4]) For any $k \geq 0$ and any $\sigma \in \Sigma$, $S \in \Pi_k \Rightarrow S\Theta\{\sigma\} \in \Pi_k$.

Lemma 5.3: There is a $O\left(\left(\sum_{i=1}^m card(S_i)\right) card(\Sigma)^k\right)$ algorithm that decides if $S \in \Pi_k$, for any S.

MEMBER(S, k) in [4] is a $O\left(\left(\sum_{i=1}^m card(S_i)\right) card(\Sigma)^{2k}\right)$ algorithm that decides the membership of $S \in \Pi_k$, while FASTMEMBER(S, k) in the same reference is a $O\left(\left(\sum_{i=1}^m card(S_i)\right) card(\Sigma)^k\right)$ algorithm for the same decision problem. Unlike that of [4], where the base class for $k = 0$ was $HORNSAT$, the base class in our case involves checking if the specification automata in $\{S_i\}_{i=1}^m$ are from Ω_{2SAT}. That is, if each member of $card(Event(S_i)) \leq 2, i \in \{1, 2, \ldots, m\}$, which can be accomplished in $O\left(\sum_{i=1}^m card(Event(S_i))\right)$ time. The rest of the analysis that leads to theorem 3.4, [4] remains unaltered, which leads to the above lemma. These procedures also produce the sequence $(\sigma_1, \sigma_2, \ldots, \sigma_p)$ of lemma 5.1 if $S (= \{Event(S_i)\}_{i=1}^m) \in \Pi_k$. We conclude this section with the following theorem on the polynomial-time solvability of any member of the subclass of SUP1M of this section when $S \in \Pi_k$.

Theorem 5.4: Consider an instance of SUP1Ω where $G = (Q, \Sigma, \delta, q_0, F)$ is a persistent plant DFA where $\Gamma(q_0, G) = \Sigma$. Suppose the specification automata $\{S_i\}_{i=1}^m$ are from the set Ω, and $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, $i \in \{1, 2, \ldots, m\}$, where $K_i = L_m(S_i) \cap L_m(G)$.

There is an $O\left(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^{k+3}\right)\})$ algorithm that solves this instance of SUP1Ω if $S (= \{Event(S_i)\}_{i=1}^m) \in \Pi_k$.

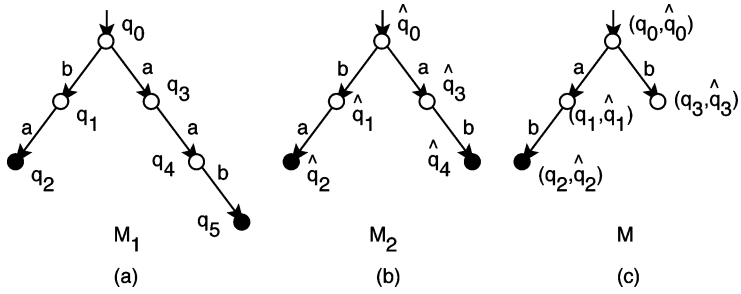

Fig. 10. (a) Persistent DFA G_1. (b) Persistent (N)DFA G_2. (c) DFA $G = \prod_{j=1}^{2} G_j$, which is not persistent. Note, $comp(a, L_m(G)) = comp(b, L_m(G)) = \emptyset$, $\Gamma((q_0, \widehat{q_0}), G) = \{a, b\}$, and $\Gamma((q_3, \widehat{q_3}), G) = \emptyset$.

Proof: The computation of the set $\{comp(\sigma, L_m(G))\}_{\sigma \in \Sigma}$ is a $O(card(\Sigma)^2 card(Q)^2)$ operation (cf. lemma 2.1). This accounts for the first argument in $\max\{\bullet, \bullet\}$. The proof of theorem 4.1 in [4], *mutatis mutandis*, yields a $O(card(\Sigma)^{k+3})$ algorithm that decides if there is a consistent binary assignment that satisfies $\{S_i\}_{i=1}^m$. This, together with lemma 3.3 accounts for the second argument in $\max\{\bullet, \bullet\}$ in the above observation. We now adopt the proof from [4] to the SUP1Ω problem.

The proof proceeds by induction on k. The base case for $k = 0$ is established by the results in Section IV, where it is shown that there is a $O(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^2)\})$ algorithm that decides the existence of a supervisory policy that solves an instance of SUP1Ω. The induction hypothesis assumes the above claim to be true for any $k \leq h - 1$.

For the induction step, the presence of a sequence $\{\sigma_1, \sigma_2, \ldots, \sigma_p\}$ that correspond to lemma 5.1 that is output by FASTMEMBER(S, k) of [4]. For $1 \leq j \leq p$, we perform steps (1) and (2) described below.

Step 1) Assume event σ_j is permitted to occur in the plant. This would require the deletion of all specification automata S_k where $\sigma_j \in Event(S_k)$. This will result in the set $S_{\{\sigma_j\}} \in \Pi_{h-1}$. From lemma 5.2, we have $S \leftarrow S_{\{\sigma_j\}} \Theta \{\sigma_l\} \in \Pi_{h-1}$, for $\sigma_l \in comp(\sigma_j, L_m(G))$. So, the process of deleting all edges with label σ_j in the remaining specification automata, would result in a (new) set S that is in Π_{h-1}. By the induction hypothesis, this instance of SUP1Ω can be solved in $O(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^{h+2}\})$ time. If the result is positive, we halt the process and declare the SUP1Ω instance at hand to have a solution. If the result is negative, we proceed with Step 2.

Step 2) We require that σ_j is not permitted in the plant (because if it were permitted, we know the instance of SUP1Ω has no solution). We delete all edges in the specification automata that are labeled σ_j. This will result in a (new) set $S \leftarrow S\Theta\{\sigma_j\}$, which belongs to Π_h, but with at least one less event than before. *Step 1* is repeated with the (new) set S.

At most $card(\Sigma)$ many subproblems, each taking $O(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^{h+2}\})$ time, will be executed in this process. This results in a $O(\max\{card(\Sigma)^2 card(Q)^2, card(\Sigma)^{h+3}\})$ procedure that solves the original problem, completing the induction step. ■

We turn our attention to the development of a similar hierarchy for SUPMΩ. First, we note that there are three prerequisites to the applicability of theorem 5.4 to any class of problems involving a plant $G = \prod_{j=1}^{k} G_j$ and specification automata $\{S_i\}_{i=1}^m$ from the set Ω. They are (i) the tractable computation of the complement sets, (ii) tractable tests for persistence in the plant, and (iii) a tractable test for $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, $i = 1, 2, \ldots, m$, where $K_i = L_m(S_i) \cap L_m(G)$.

Each of these prerequisites limit the applicability of the results in this paper to instances of SUPMΩ in the general setting. First, as noted in lemma 2.2, the computation of the complement sets $\{comp(\sigma, L_m(G)\}_{\sigma \in \Sigma}$ is intractable when the plant, $G = \prod_{j=1}^{k} G_j$, is the synchronous product of arbitrary automata $G_j = \left(Q_j, \Sigma_j, \delta_j, q_0^j, F_j\right)$, and $\Sigma = \bigcup_{j=1}^{k} \Sigma_j$. Second, the persistence of G is not guaranteed even if each G_i is persistent. For example, the DFAs G_1 and G_2 shown in Fig. 10(a) and (b) are persistent, but the DFA $G = \prod_{j=1}^{2} G_j$ shown in Fig. 10(c) is not. Finally, a procedure that tests the requirement $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $K_i = L_m(S_i) \cap L_m(G)$, can be converted into a test for the emptiness of $\bigcap_{j=1}^{k} L_m(G_j)$, a problem that is known to be intractable in the general setting [7], [15]. To see this, we consider the case when $\forall\ j \in \{1, 2, \ldots, k\}$, $\Sigma_j = \Sigma$. We use two, heretofore unused symbols $\{\widetilde{\sigma}_1, \widetilde{\sigma}_2\} \not\subseteq \Sigma$, and modify each DFA in $\{G_j\}_{j=1}^k$ in a way that its graphical description has, in addition to existing labeled edges, (i) a loop around each final-state with label $\widetilde{\sigma}_1$, and (ii) an edge, labeled $\widetilde{\sigma}_2$, from every final-state to a new, heretofore unused state, which is added to the set of final-states. $\{\widehat{G}_j\}_{j=1}^k$ denotes the set of resulting DFAs. For the specification automaton, we use a $\widehat{S}_i \in \Omega_{2SAT}$, with $Event(\widehat{S}_i) = \{\widetilde{\sigma}_1\}$. We choose a plant $\widehat{G} = \prod_{j=1}^{k} \widehat{G}_j = \bigwedge_{j=1}^{k} \widehat{G}_j$ and $K_i = L_m(\widehat{S}_i) \cap \left(\bigcap_{j=1}^{k} L_m(\widehat{G}_j\right)$. For this choice, $(K_i = \emptyset) \Leftrightarrow \left(\bigcap_{j=1}^{k} L_m(\widehat{G}_j\right) = \emptyset) \Leftrightarrow \left(\bigcap_{j=1}^{k} L_m(G_j) = \emptyset\right)$. If we let $\Sigma_u = \{\widetilde{\sigma_2}\}$, we have $\left(\overline{K_i}\Sigma_u^* \cap L_m(\widehat{G}) \subseteq K_i\right) \Leftrightarrow (K_i = \emptyset)$, which completes the necessary reduction.

However, there are subclasses of SUPMΩ that yield a tractable test for the three prerequisites mentioned earlier. For example, if $(\Sigma_i \cap \Sigma_j \neq \emptyset) \Rightarrow (i = j)$, then $comp(\sigma, L_m(G)) = comp(\sigma, L_m(G_i))$ where $\sigma \in \Sigma_i$. By lemma 2.1, there is a $O(\alpha^2 \beta^2)$ algorithm that can compute $\{comp(\sigma, L_m(G))\}_{\sigma \in \Sigma}$, where $\Sigma = \bigcup_{i=1}^{k} \Sigma_i$, $\alpha = \max\{card(\Sigma_i)\}_{i=1}^k$, and

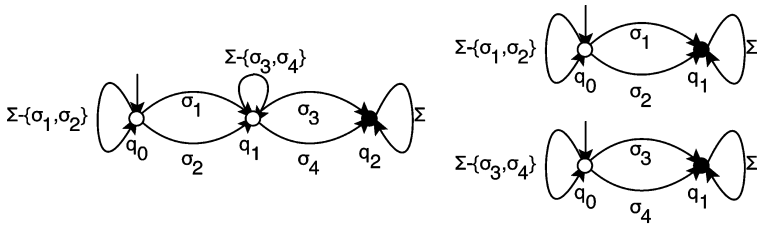

Fig. 11. (a) Specification DFA that is is not in Ω. (b) Equivalent set of DFAs that are in $\Omega_{SAT} \subseteq \Omega$, if $\{\sigma_3, \sigma_4\} \not\subseteq comp(\sigma_1, L_m(G)) \cup comp(\sigma_2, L_m(G))$, and the plant $G = (Q, \Sigma, \delta, q_0, F)$ is persistent and $\Gamma(q_0, G) = \Sigma$.

$\beta = \max\{card(Q_i)]\}_{i=1}^{k}$. Also, in this case if each DFA G_j is persistent then $G = \prod_{j=1}^{k} G_j$ is persistent. Additionally, for $P_j = L_m(S_i) \cap L_m(G_j)$ and $K = L_m(S_i) \cap L_m(G)$, $\big(\forall\, j \in \{1, \ldots, k\}, \overline{P_j}\Sigma_u^* \cap L_m(G_j \subseteq P_j) \Leftrightarrow \big(\overline{K}_i\Sigma_u^* \cap L_m(G) \subseteq K_i\big)$. The example of SUPMM in Fig. 1 satisfies these requirements, consequently this class of SUPMΩ is intractable. Theorem 5.4 can be used to develop a polynomial time hierarchy for the subclass of SUPMΩ where (i) the automata G_j that constitute the plant $G = \prod_{j=1}^{k} G_j$ share no events among themselves, and (ii) the specification automata $\{S_i\}_{i=1}^{m}$ are from Ω.

It is possible to extend this observation to the case when there is limited event sharing among the automata $\{G_j\}_{j=1}^{k}$. For instance, if the set $\{G_j\}_{j=1}^{k}$ can be partitioned into subsets $H_1, H_2, \ldots, H_l$, such that (i) $\exists\gamma$ such that $\forall\ i$, $card(G_i) \leq \gamma$, and (ii) if $G_1 = (Q_1, \Sigma_1, \delta_1, q_0^1, F_1) \in H_i$, $G_2 = (Q_2, \Sigma_2, \delta_2, q_0^2, F_2) \in H_j$, $i \neq j$, then $\Sigma_1 \cap \Sigma_2 = \emptyset$. In this case the plant G can be expressed as the product of l-many DFAs that do not share events among themselves, each having a size that is no more than $\big(\max\{card(Q_i)\}_{i=1}^{k}\big)^{\gamma}$. de Queiroz and Cury use a similar approach to modular supervisory control in [18] and obtain a computational complexity that is exponential in the number of components of the plant and the number of specification automata. This is in agreement with the above observation that SUPMM remains intractable even when there are no common events between the automata whose synchronous product defines the plant. However, the observations made in the previous paragraph apply in this case, which implies there is a polynomial time hierarchy for this subclass of SUPMΩ too.

In Section VI we present a discussion of some consequences of the results in this section and Section IV along with some suggestions for future research.

VI. Discussion

A specification $S_i \in \Omega$ essentially requires that one among a collection of events occurs in the plant. The role of supervisory control in these instances is to prevent events that might result, possibly via a string of uncontrollable event occurrences, in a marked-behavior that does not meet one of the specifications. [19] contains some specifications, that are not from Ω, that yield tractable decision procedures when used in conjunction with specific plant structures. There can be conditions, usually stated on the complement sets of events along specific paths in the specification automata, under which a specification automaton that does not belong to Ω can be expressed as a set of automata that are in Ω. For example, given a persistent plant $G = (Q, \Sigma, \delta, q_0, F)$, where $\Gamma(q_0, G) = \Sigma$ and $\{\sigma_3, \sigma_4\} \not\subseteq comp(\sigma_1, L_m(G)) \cup comp(\sigma_2, L_m(G))$, the specification automaton shown in Fig. 11(a) can be equivalently represented by the pair of automata from Ω_{2SAT} shown in Fig. 11(b). We suggest further investigations on generalizing the subclass of SUP1Ω in this paper as a topic for future research. Another direction for future research could involve investigations into the existence of alternate base-classes and its associated polynomial time hierarchy.

The hierarchy $\Pi_0 \subseteq \Pi_1 \subseteq \cdots \subseteq \Pi_k \subseteq \cdots$ naturally suggests a hardness parameter that is associated with any instance of the intractable class of SUP1Ω, which is the smallest index k such that the instance at hand belongs to Π_k. This parameter can be computed in polynomial time (cf. Lemma 5.3). Following the identification of a different base-family, it might be possible to identify a different hierarchy for SUP1Ω. An instance with a hardness parameter in one hierarchy, might have a lower hardness parameter in another. This presents an incentive to find other hierarchies for SUP1Ω.

Gallo and Scutella used $HORNSAT$ as the base class for the polynomial time hierarchy reported in [4]. This brings up the existence of a parallel to $HORNSAT$ in SUP1Ω. As noted in Section II, each clause in a *Horn* formula contains at most one posited literal. The satisfiability of any instance of $HORNSAT$ can be decided in linear time in terms of the number of literals in the Horn formula (cf. section 2.3.1, [16]). Additionally, unit resolution on the posited literals is sufficient to decide satisfiability of any instance of $HORNSAT$. In this process, each unit clause that contains a posited literal is set to true, while its complement is eliminated from the other clauses. This process is repeated till there are no more unit clauses with posited literals. If there are no empty clauses at the conclusion of this process, the instance of $HORNSAT$ is deemed satisfiable. This is because an instance of $HORNSAT$ that contains no posited literals is trivially satisfiable by setting all negated literals to true.

A subclass of SUP1Ω that is a parallel to $HORNSAT$ would require the same preamble as in the previous sections—we assume the plant $G = (Q, \Sigma, \delta, q_0, F)$ to be a persistent DFA, where $\Gamma(q_0, G) = \Sigma$. The specification automata $\{S_i\}_{i=1}^{m}$ are assumed to be from the set Ω, and $\overline{K_i}\Sigma_u^* \cap L_m(G) \subseteq K_i$, where $K_i = L_m(S_I) \cap L_m(G)$. In addition, we require: $\forall\, S_i \in \{S_i\}_{i=1}^{m}, \exists P(S_i), N(S_i) \subseteq \Sigma$ such that $Event(S_i) = P(S_i) \cup N(S_i), \forall\, i \in \{1, 2, \ldots, m\}$,

1) $card(P_i) \leq 1$, and
2) $\forall\ j \in \{1,2,\ldots,m\} - \{i\}, comp(N(S_i),\ L_m(G)) \cap N(S_j) = \emptyset$,

where $comp(N(S_i), L_m(M)) := \bigcup_{\sigma \in N(S_i)} comp(\sigma, L_m(M))$. The set $P(S_i)$ $(N(S_i))$ is analogous to the set of literals that are posited (negated) in an instance of $HORNSAT$.

Lemma 6.1: Any instance of the subclass of SUP1Ω identified above has a solution if $\forall\, i \in \{1,2,\ldots,m\}, N(S_i) \neq \emptyset$.

Proof: Consider a binary assignment $\Upsilon : \Sigma \rightarrow \{0,1\}$ that assigns a value of $\Upsilon(\sigma) = 1$ to an arbitrarily chosen $\sigma_i \in N(S_i)$, $i \in \{1,2,\ldots,m\}$. For any $i \in \{1,2,\ldots,m\}$, $j \in \{1,2,\ldots,m\} - \{i\}$, we have $comp(\sigma_i, L_m(G)) \cap N(S_j) = \emptyset$, which guarantees the consistency of the above assignment, which satisfies $\{S_i\}_{i=1}^m$ by design. The binary assignment $\Upsilon : \Sigma \rightarrow \{0,1\}$ can be refined to assign a value of 1 (0) to any $\sigma \in \Sigma$ $(comp(\sigma) \subseteq \Sigma)$, if none of the members of $\sigma \cup comp(\sigma, L_m(G)$ are assigned a value in course of the process described above. The claim follows from lemmata 3.3 and 3.4. ■

A consequence of lemma 6.1 is that unit-resolution on the set $\{S_i\}_{i=1}^m$ is sufficient to determine if an instance of the subclass of SUP1Ω identified above has a solution. This parallels the similar observation about $HORNSAT$ instances. Notwithstanding lemma 6.1, the main roadblock to its use as a base class is the tractable procedure that decides if each member $Event(S_i)$ in an arbitrary collection $\{Event(S_i)\}_{i=1}^m$ can be decomposed into its constituent components $P(S_i)$ and $N(S_i)$ that meets the requirements stated above. It is imperative that this decomposition be made without placing any restrictions on the nature of $\{comp(\sigma, L_m(G))\}_{\sigma \in \Sigma}$ if it is to serve as an alternate base class for a (different) polynomial time hierarchy.

We suggest the extension of the results in this paper to the modular supervisory control of nondeterministic systems [20]–[22] as another topic for future research. We also suggest investigations into polynomial time hierarchies non-blocking refinements to SUP1M and SUPMM, supervisory control under partial observations, and cooperative supervision, as topics for further investigation.

VII. Conclusions

Gohari and Wonham [2] introduced a class of modular supervisory control problems called SUP1M and SUPMM, and showed that there are no tractable procedures that decide the existence of a solution to an arbitrary instance of these problems. Rohloff and Lafortune [1] extended these observations and showed that in the general setting, modular supervisory control problems are $PSPACE$-complete. In this paper, we have taken a normative approach to modular supervisory control and identified a polynomial time hierarchy within intractable subclasses of these problems. Borrowing heavily from a family of polynomially solvable subclasses of SAT identified by Gallo and Scutella [4], we identified a polynomial time hierarchy for an intractable subclass of SUP1M, which we call SUP1Ω. The preface to the development of this hierarchy involved the introduction of *complement sets* for events of a finite-state automaton and their properties, along with the identification of a base class of the polynomial time hierarchy of SUP1Ω that is inspired by the $2SAT$ problem. A similar polynomial time hierarchy is identified for an intractable subclass of SUPMM, that we call SUPMΩ, where the automata that define the plant share no common events. The paper concludes with some directions for future research.

References

[1] K. Rohloff and S. Lafortune, "PSPACE-completeness of modular supervisory control problems," *Discrete-Event Dynamics Syst.*, vol. 15, no. 2, pp. 145–167, Jun. 2005.

[2] P. Gohari and W. Wonham, "On the complexity of supervisory control design in the RW framework," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 5, pp. 643–652, Oct. 2000.

[3] C. Papadimitriou, *Computational Complexity*. Norwell, MA: Addison Wesley, 1994.

[4] G. Gallo and M. Scutella, "Polynomially solvable satisfiability problems," *Inf. Process. Lett.*, vol. 29, pp. 221–227, Nov. 1988.

[5] L. Landau and E. Lifshitz, *Mechanics: Course of Theoretical Physics, Volume 1*. Oxford, U.K.: Pergamon Press, 1976.

[6] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[7] J. Birget, "Intersection of regular languages and state complexity," *SIGACT News*, vol. 22.2, pp. 49–49, 1991.

[8] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: The MIT Press and McGraw-Hill Book Company, 1990.

[9] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event systems," *SIAM J. Contr. Optimizat.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[10] P. Ramadge and W. Wonham, "Modular feedback logic for discrete event systems," *SIAM J. Contr. Optimizat.*, vol. 25, no. 5, pp. 1202–1218, May 1987.

[11] W. Wonham and P. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Optimizat.*, vol. 25, no. 3, pp. 637–659, May 1987.

[12] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81–98, Jan. 1989.

[13] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. New York: Springer, 2007.

[14] R. Kumar and V. Garg, *Modeling and Control of Logical Discrete Event Systems*. New York: Springer, 1994.

[15] D. Kozen, "Lower bounds for natural proof systems," in *Proc. 18th FOCS*, 1977, pp. 254–266.

[16] V. Chandru and J. Hooker, *Optimization Methods for Logical Inference*. New York: Wiley Interscience, 1999.

[17] D Pretolani, "Hierarchies of polynomially solvable satisfiability problems," in *Annal. Math. Artif. Intell.*, 1996, vol. 17, pp. 339–357.

[18] M. de Queiroz and J. Cury, "Modular supervisory control of large scale discrete event systems," in *Discrete Event Systems: Analysis and Control*, R. Boel and G. Stremersch, Eds. Norwell, MA: Kluwer, 2000, pp. 103–110.

[19] N. Paliwal, "Tractable Instances of Supervisory Control Theory," M.S., Electrical and Computer Engineering, Univ. Illinois at Urbana-Champaign, Urbana, IL, 2007.

[20] M. Heymann and F. Lin, "Discrete-event control of nondeterministic systems," *IEEE Trans. Autom. Contr.*, vol. 43, no. 1, pp. 3–17, Jan. 1998.

[21] R. Kumar and M. A. Shayman, "Non-blocking supervisory control of nondeterministic systems via prioritized synchronization," *IEEE Trans. Autom. Contr.*, vol. 41, no. 8, pp. 1160–1175, Aug. 1996.

[22] M. Fabian and B. Lennartson, "On non-deterministic supervisory control," in *Proc. 35th IEEE Conf. Decision Contr.*, Dec. 1996, pp. 2213–2218.

R. Gummadi (S'08) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 2005, and the M.S. degree in electrical and computer engineering from the University of Illinois at Urbana Champaign, in 2007, where he is currently pursuing the Ph.D. degree.

During his Ph.D., he has also been a research intern at Thomson Paris Research Laboratory, EPFL Switzerland, Alcatel Lucent Bell Labs, Murray Hill, NJ and Microsoft Research, Cambridge, U.K. His research interests include the study of discrete-event/discrete-state systems with an emphasis on network algorithms and coding.

Mr. Gummadi was the recipient of a Vodafone graduate fellowship in 2007–2008 and the Mavis Fellowship from the College of Engineering in 2009.

N. Singh (S'06) received the B.Tech. degree from Institute of Technology, Banaras Hindu University, India, in 2001, and the M.S. and Ph.D. degrees from University of Illinois at Urbana Champaign, in 2004 and 2009, respectively.

He is currently a Postdoctoral Researcher in the Department of Electrical and Computer Engineering at University of Illinois at Urbana Champaign. During his Ph.D., he has been a research intern at Cisco Systems, San Jose, CA, and Microsoft Research, Cambdrige, U.K. His research interests include the study of discrete-event/discrete-state systems with an emphasis on the analysis, design, control and optimization of communication networks.

R.S. Sreenivas (S'83–M'93–SM'02) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, India, in 1985, and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1987 and 1990, respectively.

He was a Post-Doctoral Fellow in Decision and Control at the Division of Applied Sciences, Harvard University, Cambridge, MA, before he joined the University of Illinois at Urbana-Champaign in September 1992. He is currently an Associate Professor of Industrial and Enterprise Systems Engineering, Research Associate Professor with the Coordinated Science Laboratory, Information and Trust Institute, and the Illinois Center for Transportation. His research interests include modeling, analysis, control and performance evaluation of discrete-event/discrete-state systems.