Decentralized Supervisory Control of Discrete Event Systems with Unknown Plants: A Learning-based Synthesis Approach

Jin Dai and Hai Lin

***Abstract*— In this paper, we consider automatic synthesis of decentralized supervisor synthesis for uncertain discrete event systems. In particular, we study the case when the uncontrolled plant is unknown *a priori*. To deal with the unknown plants, we first characterize the co-normality of prefix-closed regular languages and propose formulas for computing the supremal co-normal sublanguages; then sufficient conditions for the existence of decentralized supervisors are given in terms of language co-normality and a learning-based algorithm to synthesize the supervisor automatically is proposed. The correctness and convergence of the algorithms is proved, and its implementation and effectiveness are illustrated through examples.**

I. INTRODUCTION

The discrete event system (DES) supervisory control theory initiated by Ramadge and Wonham [14], [15] has been widely used to model and control large-scale systems, including multi-agent systems, traffic networks and manufacturing systems, see e.g., [2,8] and references therein.. Motivated by the fact that more and more complex systems built up nowadays are becoming physically distributed and networked, the decentralized control architecture of DESs has arisen in the study of supervisory control problems and has attracted many researchers' interest, see e.g. [17] [7] [13].

There have been a lot of studies of the decentralized supervisory control problems. In [11] a sufficient condition for the existence of decentralized supervisors that the controlled behavior of the system lies in a given range expressed by local specifications was proposed. Cieslak et al. [3] considered the decentralized control where the specification is given as a prefix-closed regular language, and the property of co-observability was introduced in place of observability and the result was then generalized to the case of non-prefix-closed specification by Rudie and Wonham [17]. In [19], Willner and Heymann studied the decentralized supervisory control of concurrent discrete event systems, and a necessary and sufficient condition for the existence of the decentralized supervisor was proposed based on separability of the specification. However, most of the existing supervisor synthesis algorithms require prior and complete knowledge of the uncontrolled plant. This requirement has been pointed out to be unreasonable [4] due to the uncertain nature of the plant.

There have been some studies of the uncertainty of the plant. Lin considered the plant as a set of possible plants and designed robust supervisor applicable for the whole range of plants [10]. In recent years, fault-tolerant control scheme has been proposed to deal with the faults occurring during the evolution of the system. Wen et al. [18] proposed a framework of fault-tolerant supervisory control of DESs, in which the supervisor was designed to ensure the recovery from fault. Liu and Lin [12] investigated the reliable decentralized supervisory control problem of DES under the general architecture, which seeked the minimal number of supervisors required for correct functionality of the supervised systems.

This paper differs from the aforementioned work in the sense that we focus on automatic decentralized supervisor synthesis for unknown plant instead of partially known or bounded cases. The contribution of this paper is as follows: first, we propose a sufficient condition for the existence of decentralized supervisor in terms of language controllability and co-normality; secondly, to deal with the uncertain or even unknown nature of the plant, we propose an L^* learning based synthesis algorithm where new dynamical membership queries are used instead of static ones in the original learning procedure, and the algorithm can synthesize a sub-optimal decentralized supervisors that are consistent with the supremal controllable and co-normal sublanguage of the given prefix-closed specification language; thirdly, the correctness and convergence of the proposed synthesis algorithm are proved, and its effectiveness is illustrated through examples.

The rest of this paper is organized as follows. Section II briefly reviews the result of decentralized supervisory control and L^* learning procedure. The discussion about language co-normality and the sufficient conditions for the existence of decentralized supervisor based on co-normality are given in Section III. The modified L^* learning algorithm for supervisor synthesis is provided in section IV. Section V presents a simple example to illustrate the effectiveness of the proposed algorithm. In conclusion we will discuss future work.

II. PRELIMINARIES

A. Decentralized Supervisory Control of Discrete-event Systems

An uncontrolled plant is modeled by a deterministic finite automaton $G = (Q, \Sigma, q_0, \delta, Q_m)$, where Σ is a set of events, Q is a set of states, $q_0 \in Q$ is the initial state, $Q_m \subseteq Q$ is the set of marked states, δ is the (partial) transition function. Let Σ^* denote the set of all finite strings over Σ plus the null string ϵ, then δ can be extended to $\delta : Q \times \Sigma^* \to Q$ in a natural way. The languages generated

This work was supported by the National Science Foundation (NSF-CNS-1239222 and NSF- EECS-1253488)

The authors are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. {jdai1, hlin1}@nd.edu.

by G is given by $L(G) = \{s \in \Sigma^* | \delta(q_0, s) \text{ is defined}\}$ and the language marked by G is given by $L_m(G) = \{s \in \Sigma^* | s \in L(G), \delta(q_0, s) \in Q_m\}$. The prefix closure $\overline{K}$ of a language $K \subseteq \Sigma^*$ is the set of all prefixes of strings in K. K is called prefix-closed if $\overline{K} = K$.

In supervisory control theory, the event set are divided into the set of controllable events and the set of uncontrollable events, i.e., $\Sigma = \Sigma_{uc} \dot{\cup} \Sigma_c$. Given a non-empty prefix-closed specification $K \subseteq L(G)$, a supervisor S exists such that $L(S||G) = K$ if and only if K is controllable, i.e., $\overline{K}\Sigma_{uc} \cup L(G) \subseteq \overline{K}$ [14]. If not, then a supervisor is synthesized for the supremal controllable (also prefix-closed) sublanguage of K, namely, $supC(K)$.

Moreover, the supervisor may also be constrained to observe only events in a specified set of observable events and the event set is divided into the subset of unobservable and observable events, i.e., $\Sigma = \Sigma_{uo} \dot{\cup} \Sigma_o$. The presence of partial observation can be captured by an the natural projection mapping $P : \Sigma \to \Sigma_o \cup \{\epsilon\}$ that erases the occurrence of all unobservable events. A prefix-closed language $K \subseteq L(G)$ is said to be observable if the conditions $s, t \in \overline{K}, \sigma \in \Sigma$, $P(s) = P(t)$, $s\sigma \in \overline{K}$, and $t\sigma \in L(G)$ together imply $t\sigma \in \overline{K}$ [9]. Given a non-empty prefix-closed specification $K \subseteq L(G)$, a supervisor S exists such that $L(S||G) = K$ if and only if K is controllable and observable [2].

In this paper, we study the general decentralized supervisory control problem, where the plant is controlled jointly by n local supervisors, each of which observes the locally observable events and controls the locally controllable events. Let $I = \{1, 2, \ldots, n\}$ denote the indices of the supervisors. For each $i \in I$, let Σ_i denote the i-th local event set, hence $\Sigma = \bigcap_{i \in I} \Sigma_i$. Σ_i is divided into the set of controllable events $\Sigma_{i,c}$ and the set of uncontrollable events $\Sigma_{i,uc}$. Let $\Sigma_{i,o}$ and $\Sigma_{i,uo}$ denote the sets of locally observable and unobservable events, respectively. For notational simplicity, the set of globally controllable events is denoted as $\Sigma_c = \bigcup_{i \in I} \Sigma_{i,c}$ and the globally observable event set is denoted as $\Sigma_o = \bigcup_{i \in I} \Sigma_{i,o}$. The sets $\Sigma_{uc} = \Sigma - \Sigma_c = \bigcap_{i \in I} \Sigma_{i,uc}$ and $\Sigma_{uo} = \Sigma - \Sigma_o = \bigcap_{i \in I} \Sigma_{i,uo}$ denote the uncontrollable and unobservable event sets, respectively, with the natural projection mapping $P_{\Sigma_i} : \Sigma^* \to \Sigma_{i,o}^*$. We use P_i to replace P_{Σ_i} and use P to denote the natural projection from Σ^* to Σ_o^* in the rest of this paper. The closed loop behavior of the DES under decentralized supervisors $\{S_i\}_{i \in I}$ is denoted as $L(\{S_i\}_{i \in I}, G)$.

Let $In(\sigma) = \{i \in I | \sigma \in \Sigma_{i,c}\}$ denote the index set. A language $K \subseteq L(G)$ is said to be

- *Normal* [3] [9] if $P^{-1}P(K) \cap L(G) = K$.
- $C\&P$ *co-observable* with respect to $A \subseteq \Sigma_c$ [7] if for any $s \in \overline{K}$ and any $\sigma \in A$ such that $s\sigma \in L(G) - \overline{K}$, then there exists $i \in In(\sigma)$ such that: for any $s' \in \overline{K}$, $[P_i(s) = P_i(s')] \wedge [s'\sigma \in L(G)] \Rightarrow [s'\sigma \notin \overline{K}]$.
- $D\&A$ *co-observable* with respect to $A \subseteq \Sigma_c$ [7] if for any $s \in \overline{K}$ and any $\sigma \in A$ such that $s\sigma \in \overline{K}$, then there exists $i \in In(\sigma)$ such that: for any $s' \in \overline{K}$, $[P_i(s) = P_i(s')] \wedge [s'\sigma \in L(G)] \Rightarrow [s'\sigma \in \overline{K}]$.

Remark 1: The term "$C\&P$" and "$D\&A$" in the definition of co-observability stands for "conjunctive architecture and permissive decision rule" and "disjunctive architecture and antipermissive decision rule", respectively. In general, we may use "co-observability" when the considered language is either $C\&P$ or $D\&A$ co-observable.

In this paper, we aim at solving the problem that given a non-empty prefix-closed specification $K \subseteq L(G)$, find the decentralized supervisors S_i such that $L(\{S_i\}_{i \in I}, G)$ with no prior knowledge of G, and we expect to apply a modified L^* learning method.

B. L^* learning

The L^* learning algorithm introduced by Angluin [1] and improved by Rivest and Schapire [16] learns an unknown regular language U over alphabet Σ and produces a minimal deterministic finite automaton (DFA) that accepts it. The algorithm infers the structure of the DFA by asking an *oracle* that answers two types of queries. The first type is a *membership query*, in which L^* asks whether a string $s \in \Sigma^*$ is included in U. The second type is a *conjecture*, in which L^* constructs a conjectured DFA M and asks whether M is such that $L(M) = U$. If $L(M) \neq U$ the oracle returns a counterexample, which is a string s in the symmetric difference of $L(M)$ and U. At any given time, L^* has, in order to construct the conjectured DFA M, information about a finite collection of strings over Σ, classified either as members or non-members of U. L^* creates an *observation table* to incrementally record and maintain the information whether strings in Σ^* belong to U. The observation table is a three-tuple (S, E, T) consisting of: a non-empty finite set S of prefix-closed languages, a non-empty finite set E of suffix-closed languages and a function $T : (S \cup S\Sigma)E \to \{0, 1\}$ which is often referred to as the *membership oracle*, i.e., the function T takes strings in $s \in (S \cup S\Sigma)$ onto 0 if they are not in K, otherwise the function T returns 1.

The ith observation table constructed by L^* will be denoted as T_i. Each table can be depicted as a 2-dimensional array whose rows are labeled by strings $s \in S \cup S\Sigma$ and whose columns are labeled by symbols $\sigma \in E$. The entries in the labeled rows and columns are given by the function value $T(s\sigma)$. The *row function* $row : (S \cup S\Sigma) \to \{0, 1\}^{|E|}$ denotes the table entries in the row labeled by string $s \in S \cup S\Sigma$.

An observation table is said to be

- *Closed* if for all $t \in S\Sigma$, there exists an $s \in S$ such that $row(t) = row(s)$.
- *Consistent* if there exist strings $s_1, s_2 \in S$ such that $row(s_1) = row(s_2)$, and for all $\sigma \in \Sigma$, $row(s_1\sigma) = row(s_2\sigma)$.
- *Complete* if it is closed and consistent.

Once the observation table is complete, a candidate DFA $M(S, E, T) = (Q, q_0, \delta, Q_m)$ over the alphabet Σ is con-

structed isomorphically by the following rules:

$$Q = \{row(s) : s \in S\},$$
$$q_0 = row(\epsilon),$$
$$Q_m = \{row(s) : (s \in S) \wedge (T(s) = 1)\},$$
$$\delta(row(s), \sigma) = row(s\sigma).$$

The DFA M is presented as a conjecture to the oracle. If the conjecture is correct, i.e., $L(M) = U$, then the oracle returns "True" with the current DFA M; otherwise, a counterexample $c \in (U - L(M)) \cup (L(M) - U)$ is generated by the oracle. The L^* algorithm analyzes the counterexample c and finds the longest prefix c_p of c that witnesses the difference between $L(M)$ and U. Adding c_p to S reflects the difference in next conjecture by splitting states in M. Once c_p is added to S, L^* iterates the entire process to update M with respect to c_p.

The L^* algorithm is guaranteed to construct a minimal DFA accepting the unknown regular language U using only $O(|\Sigma|n^2 + n\log m)$ membership queries and at most $n - 1$ equivalence queries, where n is the number of states in the final DFA and m is the length of the longest counterexample provided by the oracle when answering equivalence queries [1].

III. Decentralized Control Based on Co-normality

The sufficient condition for existence of decentralized supervisors is related to the language decomposability, which is first introduced by Rudie and Wonham [17], as follows.

Definition 1: A language $K \subseteq L(G)$ is said to be *decomposable* (with respect to G and $\{P_i\}_{i\in I}$) if $K = L(G) \cap (\bigcap_{i\in I} P_i^{-1}P_i(K))$.

In general, decomposability is stronger than co-observability, i.e., a language K is decomposable implies that K is also co-observable. However, under certain conditions of decentralized local control [17], decomposability and co-observability are equivalent..

Remark 2: If we extend $L(G) = \Sigma^*$, then the condition for language decomposability is reduced to $K = \bigcap_{i\in I} P_i^{-1}P_i(K)$, which is equivalent to the language separability discussed in [19].

The following theorem provides a necessary and sufficient condition for the existence of decentralized supervisors based on language decomposability.

Theorem 1: [6] Let G be a plant, Σ be the global event set, and $\Sigma_i \subseteq \Sigma, i \in I$ be the local event sets. Let P_i be the natural projection from Σ to Σ_i. Then for a given non-empty, prefix-closed specification language $K \subset L(G)$, local supervisors $\{S_i, i \in I\}$ exist such that $L(\{S_i\}_{i\in I}, G) = K$ if and only if K is Σ_{uc}-controllable and $\{P_i\}_{i\in I}$-decomposable.

When the given specification K is not controllable or decomposable, a supervisor synthesis problem arises that we need to find a controllable and decomposable sublanguage of K. However, it has been pointed out that decomposability is not closed under unions [13], hence it is therefore not guaranteed that this set contains a unique supremal element, which implies that no optimal solution (in the sense of maximal permissiveness) exists for the decentralized supervisory control problem. One suggestion to obtain a sub-optimal solution takes advantage of the following co-normality property.

Definition 2: A language $K \subseteq L(G)$ is said to be *co-normal* with respect to $\{P_i\}_{i\in I}$ if $K = L(G) \cap (\bigcup_{i\in I} P_i^{-1}P_i(K))$

Recall the definition of language normality and decomposability, it is proved that co-normality is preserved under arbitrary unions and is a stronger property than decomposability [17] [13]. The following theorem provide a sufficient condition for the existence of decentralized supervisors in terms of co-normality.

Theorem 2: Let G be a plant, Σ be the global event set, and $\Sigma_i \subseteq \Sigma, i \in I$ be the local event sets. Let P_i be the natural projection from Σ to Σ_i. Then for a given non-empty, prefix-closed specification language $K \subseteq L(G)$, if K is Σ_{uc}-controllable and $\{P_i\}_{i\in I}$-co-normal, then there exist decentralized supervisors $\{S_i, i \in I\}$ such that $L(\{S_i\}_{i\in I}, G) = K$.

Proof The theorem is a direct corollary of Theorem 1 since co-normality always implies decomposability. ■

Since controllability and prefix-closeness of regular languages are also preserved under union, the supremal prefix-closed, controllable and co-normal sublanguage of a given prefix-closed language K, denoted as $supCCN(K)$, exists. The following proposition illustrates that co-normality is a "decentralized version" of normality.

Proposition 1: If $K \subseteq L(G)$ is co-normal with respect to $\Sigma_i, i \in I$, then K is normal with respect to each Σ_i, respectively.

Proof The inclusion $K \subseteq P_i^{-1}P_i(K) \cap L(G)$ is always satisfied for any language $K \subseteq L(G)$. Therefore it is sufficient to prove that $K \supseteq P_i^{-1}P_i(K) \cap L(G)$. In fact, K is co-normal with respect to $\Sigma_i, i \in I$ implies that $K = L(G) \cap (\bigcup_{i\in I} P_i^{-1}P_i(K))$, thus

$$\begin{aligned} K &= L(G) \cap (\bigcup_{i\in I} P_i^{-1}P_i(K)) \\ &= \bigcup_{i\in I} [P_i^{-1}P_i(K)) \cap L(G)] \\ &\supseteq P_i^{-1}P_i(K)) \cap L(G), \forall i \in I \end{aligned}$$

Hence, $K = P_i^{-1}P_i(K) \cap L(G)$, which implies that K is normal with P_i, $i \in I$. ■

Intuitively, Proposition 1 states that, if a language $K \subseteq L(G)$ is co-normal, then it is normal with all the local observation mappings, therefore, if there exists $i \in I$ such that K is not normal with respect to P_i, then we can conclude that K is not co-normal. Based on this conclusion, we propose the following formula for the computation of supremal co-normal language of a given language K, denoted as $supCN(K)$.

Theorem 3: For a language $K \subseteq L(G)$, the supremal co-normal sublanguage of K is given by

$$supCN(K) = L(G) - [\bigcup_{i\in I} P_i^{-1}P_i(L(G) - K)]\Sigma^* \quad (1)$$

Proof The proof of Theorem 3 can follow immediately the proof for the computation of supremal normal sublanguage of the given language in [8], Chapter 4, by simply replacing P by $\bigcup_{i\in I} P_i^{-1}P_i$ in the proof. ■

IV. L^* LEARNING IN SYNTHESIS OF DECENTRALIZED SUPERVISORS

In the previous work by Yang et al. [20] [21], L^*-based algorithm for supervisor synthesis, and in their framework, the knowledge of the plant behaviors is confined to a limited lookahead window, which was first introduced by Chung and Lafortune [4]. However, this assumption is difficult in realization. Due to this drawback of the limited lookahead windows, in this section, we derive a modified L^* learning algorithm to synthesize the supervisor with an totally unknown plant model.

Recall that a language K is controllable if $\overline{K}\Sigma_{uc}\cup L(G) \subseteq \overline{K}$, therefore it is difficult to verify the controllability of the given specification language due to lack of knowledge of the plant, hence modification to the existing L^* learning procedure is required. We solve this difficulty by using dynamical membership queries discussed that is capable of learning the supremal controllable sublanguage of the given specification (note that since the specification language is prefix-closed, its supremal controllable sublanguage is also prefix-closed), and hence a supervisor is synthesized.

A. Learning for controllability

We start by modifying the L^* so that it can learn the supremal contrllable sublanguage $supC(K)$ of a given prefix-closed specification K. A string s generated by G is said to be *legal* with respect to K if $s \in K$, and s is said to be *illegal* if $s \notin K$.

In this paper, the dynamical membership queries presented to the oracle in the modified L^* is based on the observed illegal behaviors generated by the system along with the given specification language. A plant behavior $st \in \Sigma^*$ is said to be *uncontrollably illegal* if s is legal, $t \in \Sigma_u^*$ and $st \notin K$. Let C denote the collection of observed uncontrollably illegal behaviors. We define the operator $D_u(\cdot)$ as

$$D_u(C) = \{s \in L(G) : \exists t \in \Sigma_{uc}^* \text{ such that } st \in C\}$$

to represent the collection of the strings formed by discarding the uncontrollable suffixes of strings in C, and let C_i denote the set of uncontrollably illegal behaviors after the i-th iteration, then if a new uncontrollably illegal behavior s_i (a new counterexample) is generated by the oracle, we update C_i to $C_{i+1} = \{s_i\} \cup C_i$. Finally, we propose the following membership oracle T_i for $i \in \mathbb{N}$. For $t \in \Sigma^*$, let $T(t)$ denote the membership Boolean function, initially,

$$T_1(t) = \begin{cases} 0, & \text{if } t \notin K \\ 1. & \text{otherwise} \end{cases} \tag{2}$$

For $i > 1$

$$T_i(t) = \begin{cases} 0, & \text{if } T_{i-1}(t) = 0 \text{ or } t \in D_u(C_i)\Sigma^* \\ 1. & \text{otherwise} \end{cases} \tag{3}$$

Note that different from conventional L^* discussed in Section II, the dynamical membership queries used in (2) and (3) are dynamical.

The correctness and convergence of the modified L^* algorithm using membership queries (2) and (3)is summarized as the following theorem.

Theorem 4: Assume that $K \subseteq L(G)$ is prefix-closed, then the modified L^* learning procedure using membership queries (2) and (3) converges to a supervisor S, such that $L(S||G) = supC(K)$. Furthermore, this iteration procedure of synthesizing S will be done in a finite number of counterexample tests. [5]

B. Learning for Co-normality

1) Learning for Co-normality: We now consider using L^* to learn $supCCN(K)$. To compute $supCCN(K)$ using L^* learning procedure, we first consider how to deal with the co-normality. For $j \geq 1$, define recursively

$$K_1 = K, \tag{4}$$

$$C_{cn}(K_j) = \left\{ s \in \overline{K}_j : \exists s' \in L(G), \bigcup_{i\in I} P_i^{-1}P_i(s) = \bigcup_{i\in I} P_i^{-1}P_i(s'), s' \notin \overline{K}_j \right\}, \tag{5}$$

$$\tilde{K}_j = K_j - C_{cn}(K_j) \tag{6}$$

to denote the collection of indistinguishable (with respect to $\bigcup_{i\in I} P_i^{-1}P_i$ and K) behaviors. It follows immediately from Theorem 3 that by using the $C_{cn}(\cdot)$ operator, the above iteration will converge to the supremal co-normal sublanguage of the given prefix-closed specification $K \subseteq L(G)$ within a finite number of steps.

2) Modified membership queries: To capture the illegal behavior generated by the unknown plant under partial observations in the decentralized control structure, we modify C in previous section to be

$$\tilde{C} = \left\{ st \in \bigcup_{i\in I} P_i^{-1}P_i(L(G)) : s \in \bigcup_{i\in I} P_i^{-1}P_i(\tilde{K}), t \in \Sigma_{uc}^*, st \notin \bigcup_{i\in I} P_i^{-1}P_i(\tilde{K}) \right\} \tag{7}$$

Then, we define the following membership queries $\tilde{T}_j, j \in \mathbb{N}$ as follows:

$$\tilde{K}_1 = K - C_{cn}(K)$$

$$\tilde{T}_1(t) = \begin{cases} 0, & \text{if } t \notin \bigcup_{i\in I} P_i^{-1}P_i(\tilde{K}_1)) \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

$$K_j = L(M(T_{j-1})), j > 1.$$

If $t \in C_{cn}(K_j)$, remove t from K_j to obtain $\tilde{K}_j$, and,

$$\tilde{T}_j(t) = \begin{cases} 0, & \text{if } \tilde{T}_{j-1}(t) = 0 \text{ or } t \in D_u(\tilde{C}_j)\Sigma_o^* \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

The algorithm of supervisor synthesis is summarized as Algorithm 1.

Algorithm 1 L^* for learning $supCCN(K)$.

1: Set $S = \epsilon$ and $E = \epsilon$.
2: Use the membership oracle to form the initial observation table $\tilde{T}_j(S, E, \tilde{T})$ where $j = 1$
3: **while** $\tilde{T}_j(S, E, \tilde{T})$ is not completed **do**
4: **if** $\tilde{T}_j$ is not consistent **then**
5: find $s_1, s_2 \in S$, $\sigma \in \Sigma_o$ and $e \in E$ such that $row(s_1) = row(s_2)$ but $\tilde{T}(s_1\sigma e) \neq \tilde{T}(s_2\sigma e)$;
6: Add σe to E;
7: extend $\tilde{T}_j$ to $(S \cup S\Sigma)E$ using aforementioned membership queries.
8: **end if**
9: **if** $\tilde{T}_j$ is not closed **then**
10: find $s_1 \in S$, $\sigma \in \Sigma_o$ such that $row(s_1\sigma)$ is different from $row(s)$ for all $s \in S$;
11: Add $s_1\sigma e$ to S;
12: extend $\tilde{T}_j$ to $(S \cup S\Sigma)E$ using aforementioned membership queries.
13: **end if**
14: **end while**
15: Once $\tilde{T}_j$ is completed,let $\tilde{M}_j = M(\tilde{T}_j)$ as the acceptor; make the conjecture $\tilde{M}$
16: Set the current $\tilde{T}_j$ as the membership oracle. Use the acceptor $M(\tilde{T})$ as the supervisor, and let the closed loop system evolve.
17: **if** the counterexample oracle declares that the conjecture to be false and a counterexample (indistinguishable behavior) $t \in C_{cn}(\tilde{K}_j)$ is generated **then**
18: remove t from $\tilde{K}_j$.
19: **end if**
20: Obtain $\tilde{K}_{j+1}$.
21: **if** the counterexample oracle declares that the conjecture to be false and a counterexample(illegal behavior) $t \in \Sigma^*$ is generated **then**
22: Add $P(t)$ and all its prefixes into S;
23: update $\tilde{T}_j$ to $\tilde{T}_{j+1}$ by using the counterexample t;
24: **end if**
25: Set $j = j+1$ and return to **while** until that $\exists n \in \mathbb{N}$ such that $\tilde{M}_n = \tilde{M}_{n+1}$.
26: Let S_i over Σ_i such that $L(S_i) = P_i(M_n)$, then the obtained $S_i, i \in I$ are the decentralized supervisors.

3) Correctness and convergence: The following theorem states the convergence and correctness of the modified L^* by using membership queries (8) and (9).

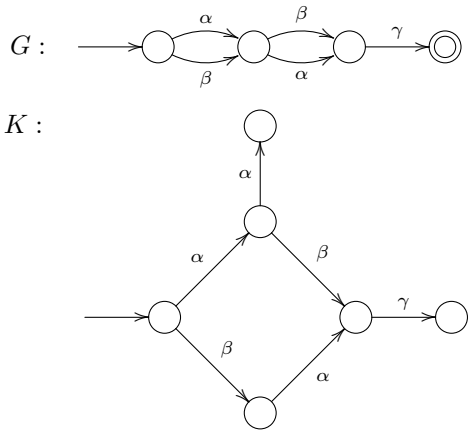
Theorem 5: Let $K \subseteq L(G)$ be a non-empty and prefix-closed specification, then L^* with dynamical membership queries (8) and (9) converges to decentralized supervisors $S_i, i \in I$, such that $L(\{S_i\}_{i\in I}, G) = supCCN(K)$. Furthermore, this iteration procedure of synthesizing S will be done in a finite number of counterexample tests.

Proof The convergence property of Algorithm 1 can be shown using the similar approach of Theorem 1 in [5] and is omitted here. We show that the obtained language from Algorithm 1 is $supCCN(K)$. First, we show that the obtained language $\tilde{K}$ is co-normal, which can be proved by contradiction. For the $i-$th step of iteration, assume that $\tilde{K}_i$ is not normal, then there exists a string $s \in \overline{\tilde{K}_i}$ and another string $t \in \overline{L(G)}$ such that $\bigcup_{i\in I} P_i^{-1}P_i(s) = \bigcup_{i\in I} P_i^{-1}P_i(t)$ but $t \notin \overline{\tilde{K}_i}$; then by the definition of $C_{cn}(K)$, it is clear to find that $s \in C_{cn}(K_i)$ and should be eliminated from $\tilde{K}_i$. Thus we get the contradiction and $\tilde{K}_i$ is a co-normal and so is $\tilde{K}$. Next we show that $\tilde{K} = supCCN(K)$. In fact, by comparing dynamical membership queries (8) and (9) with (2) and (3), respectively, we alternatively compute the supremal co-normal sublanguage and the supremal controllable sublanguage in each iteration, hence it is clear to show that the obtained language $\tilde{K} = supCCN(K)$. ■

V. Illustrative Example

In this section, the effectiveness of Algorithm 1 is illustrated through the following example.

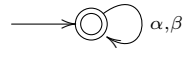
Consider the global event set $\Sigma = \{\alpha, \beta, \gamma\}$. The local controllable events and observable events are given by $\Sigma_{1,c} = \{\alpha\,\gamma\}$, $\Sigma_{2,c} = \{\beta\,\gamma\}$, $\Sigma_{1,o} = \{\alpha\}$ and $\Sigma_{2,o} = \{\beta\}$, respectively. The specification is given as $K = \overline{\alpha\alpha + (\alpha\beta + \beta\alpha)\gamma}$ and the language generated by the plant is $L(G) = \overline{(\alpha+\beta)(\alpha+\beta)\gamma}$. Note that $L(G)$ is not known to the supervisors. Both $L(G)$ and K are depicted as the following.



We start from the first observation table, and set $S = E = \{\epsilon\}$ for Algorithm 1. The first complete observation table with its corresponding acceptor is given as the following table (note that we only care about the rows whose first entries are 1's).

TABLE I
T_1 in Example

T_1		ϵ
S	ϵ	1
$S\Sigma - S$	α	1
	β	1

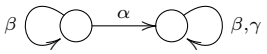
$M(T_1)$: 

We detect that the string $\alpha\beta\beta \in M(T_1) - P(K_1)$, hence it is a counterexample and we use Algorithm 1 to update and complete the observation table.

TABLE II
T_2 IN EXAMPLE

T_2		ϵ	α
S	ϵ	1	1
	α	1	0
	$\alpha\beta$	1	0
$S\Sigma - S$	$\alpha\delta$	1	0
	β	1	0
	$\alpha\beta\gamma$	1	0

$M(T_2)$:

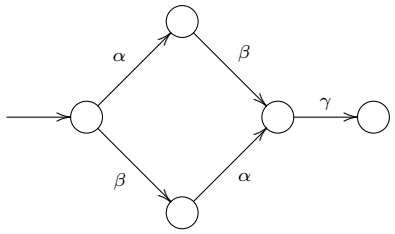


We use $M(T_2)$ as the supervisor to control the plant behaviors, the string $\beta\alpha\gamma\gamma \in M(T_2) - P(K_2)$ is a counterexample, then we add the prefixes of $\alpha\delta\delta\delta$ into S and update the observation table to T_3.

TABLE III
T_3 IN EXAMPLE

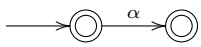
T_2		ϵ	α	β	γ
S	ϵ	1	1	1	0
	α	1	0	1	0
	$\alpha\beta$	1	0	0	1
	β	1	1	0	0
	$\beta\alpha$	1	0	0	1
	$\beta\alpha\gamma$	1	0	0	0
$S\Sigma - S$	$\alpha\beta\gamma$	1	0	0	0

$M(T_3)$:

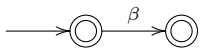


In this case no more counterexamples are detected, and we can conclude that $supCCN(K) = \overline{(\alpha\beta + \beta\alpha)\gamma}$, the local(decentralized) supervisors can then be obtained such that $S_i = P_i(supCCN(K)), i = 1, 2$, which are depicted respectively as follows.

S_1 :



S_2 :



VI. CONCLUSIONS

In this paper, the decentralized supervisory control and synthesis problem with no prior knowledge of the plant is investigated. By using the modified membership queries, the L^* can learn the supremal controllabel and co-normal sublanguage of a given prefix-closed specification language, and an illustrative example is also provided to show the effectiveness of the proposed algorithm. The future work will be focused on the modular synthesis of the decentralized supervisors.

REFERENCES

[1] D. Angluin,"Learning regular sets from queries and counterexamples," *Int. J. Inform. and Computation*, vol. 75, no. 1, pp. 87-106, 1987.

[2] C. G.Cassandras, S. Lafortune, *Introduction to Discrete Event Systems*, USA: Springer, 2008.

[3] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Valaiya, "Supervisory Control of Discrete-event Process with Partial Observation," *IEEE Trans. Autom. Control*, vol. 33, no. 3, pp. 249-260, 1988.

[4] S. Chung. S. Lafortune, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1921-1935, 1992.

[5] J. Dai and H. Lin, "Automatic discrete-event supervisor synthesis for unknown plants," submitted to *2014 American Control Conference*.

[6] S. Jiang and R. Kumar, "Decentralized control of discrete event systems with specializations to local control and concurrent systems." *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*,vol. 30, no. 5, pp. 653-660, 2000.

[7] P. Kozak W. M. Wonham, "Fully decentralized solutions of supervisory control problems," *IEEE Trans. Autom. Control*, vol. 40, no. 12, pp. 2094-2097, 1995.

[8] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Boston: Kluwer, 1995.

[9] F. Lin, W. M. Wonham, "On observability of Discrete-event Systems," *Inform. Sci.*, vol. 44, pp. 173-198, 1988.

[10] F. Lin, "Robust and Adaptive Supervisory Control of Discrete Event Systens," *IEEE Trans. Autom. Control*, vol. 38, no. 12, pp. 1848-1852, 1993.

[11] —, "Decentralized control and coordination of discrete-event systems with partial observation," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 1330C1337, Dec. 1990.

[12] F. Liu and H. Lin, "Reliable supervisory control for general architecture of decentralized discrete event systems," *Automatica*, vol. 46, no. 9, pp. 1510-1516, 2010.

[13] A. Overkamp and J. H. van Schuppen, "Maximal solutions in decentralized supervisory control," *SIAM J. Control Optim.*, vol. 39, no. 2, pp. 492-511, 2000.

[14] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206-230, 1987.

[15] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. of the IEEE*, vol. 77, no. 1, pp. 81-98, 1989.

[16] R. L. Rivest and R. E. Schapire. "Inference of finite automata using homing sequences." *Machine Learning: From Theory to Applications*, Springer Berlin Heidelberg, pp. 51-73, 1993.

[17] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692C1708, Nov. 1992.

[18] Q. Wen, R. Kumar, J. Huang and H. Liu, "A Framework for Fault-Tolerant Control of Discrete Event Systems," *IEEE Tran. Autom. Control*, vol. 53, no. 8, pp. 1839-1849, 2008.

[19] Y. Willner and M. Heymann, "Supervisory control of concurrent discrete-event systems," *Int. J. Contr.*, vol. 54, no. 5, pp. 1143C1169, 1991.

[20] X. Yang, M. D. Lemmon, and P. Antsaklis, "Inductive inference of optimal controllers for uncertain logical discrete event systems," *Proc. American Control Conference*, Seattle, Jun. 1995,vol. 5, pp. 3163-3167.

[21] X. Yang, M. D. Lemmon, and P. Antsaklis, "Inductive inference of logical DES controllers using the L^* algorithm," *Proc. 1995 IEEE International Symposium on Intelligent Control*, pp. 585-590, 1995.