Remarks on the Difficulty of Top-Down Supervisor Synthesis

Liyong Lin and Rong Su
School of Electrical and Electronics Engineering
Nanyang Technological University
Singapore

Alin Stefanescu
Department of Computer Science
University of Pitesti
Pitesti, Romania

***Abstract*—This paper shows that language based top-down supervisor synthesis of Ramadge-Wonham supervisory control theory is in general not feasible. We show this as a direct consequence of the undecidability result of Decomposable Subset problem (and its prefix closed version), which in turn is a corollary of the undecidability result of Trace Closed Subset problem. Essentially, it is in general not possible to decide in finite amount of time whether there exists a string in a regular language such that all those strings indistinguishable from it are contained in the same language. We bring these results to the attention of control community and investigate the decidability status of some other related problems.**

Keywords*: distributed control, decomposable subset, supervisory control, top-down design*

I. Introduction

The Ramadge-Wonham paradigm for supervisory control [1],[2],[3] provides a theoretical framework for synthesizing the optimal (minimally restrictive) supervisor for controlled discrete event dynamical systems such that a given specification is satisfied.

However, supervisory control of large scale discrete event dynamical system inevitably faces the state explosion problem when several system components are composed together. Thus, the monolithic synthesis strategy becomes computationally too expensive even though synthesizing the optimal supervisor only involves a polynomial time algorithm [1],[2],[4],[5]. To circumvent this difficulty, various control architectures such as hierarchical, decentralized and distributed based approaches have been exploited [1],[6]. However, some important concepts and conditions are computationally expensive to verify. For example, to verify co-observability that is critical for decentralized control may take exponential time [7].

The main focus of this paper is to investigate the difficulty of top-down distributed supervisor synthesis and the underlying concepts in a language based framework. Top-down design in distributed control, where the global specification is decomposed into local specifications (such that we will obtain exactly the global specification after the composition of these local specifications) and each local supervisor makes decisions solely for its own local system so as to satisfy its local specification, is a divide and conquer approach. It is attractive from system development point of view since it is correct by construction [6]. The first and key step of top-down supervisor synthesis is to automatically decompose a global specification. But it is a rather strict condition for a global specification to be exactly decomposable. In most cases we are satisfied if the global specification has a non-empty decomposable subset so that after composing those local specifications the global specification is not violated, formulated in terms of language inclusion. Obviously if a global specification is decomposable then it also has a decomposable subset.

We formulate the Decomposable Subset problem as a question about whether a given global specification has a non-empty decomposable subset. It is proved in [8] that Decomposable Subset problem is undecidable when the underlying languages are required to be prefix closed. We shall henceforth call this problem prefix closed version of Decomposable Subset Problem. To avoid trivial decomposable subset $\{\epsilon\}$, in [8] all symbols that appear in the global specification are required to appear in the desired subset. That result suggests that it is not always possible to check whether a given prefix closed global specification has a prefix closed subset that can be decomposed into local specifications, where all the alphabets need appear in the desired subset. While the prefix closed version of Decomposable Subset Problem is concerned with safety specifications, Decomposable Subset problem is important for specifications related to task completion.

Another problem related to Decomposable Subset problem is the Trace Closed Subset problem, which is known to be undecidable [9]. Trace Closed Subset problem is a question about the existence of a non-empty subset where the trace closure of each string is also contained. We show in this paper that Decomposable Subset problem is in fact equivalent to the Trace Closed Subset problem. Indeed, both Traced Closed Subset and Decomposable Subset problems boil down to the same problem of whether there exists a single string whose trace closure is contained in the same language, which we shall call String Observation Closure problem in this paper. Although this is not new, see for example [8] and [10], it seems that this undecidability result is not well known by the supervisory control community. Both undecidability results for the two Decomposable Subset problem together suggest that language based top-down design approach in distributed supervisor synthesis of the Ramadge-Wonham supervisory control paradigm is in general not feasible. Indeed if the global specification is not decomposable which is usually the case, then no useful progress can be guaranteed to be obtained further.

We provide characterizations of decomposable languages under usual set theoretic operations. Some interesting problems and results related to Decomposable Subset problem are discussed in our paper as well.

The organization of this paper is as follows. Section 2 is devoted to technical preliminaries. Decomposable Subset problem is discussed in Section 3. Its related problems and their undecidability results are discussed in Section 4. Conclusions in Section 5 end the paper.

II. Preliminaries

For Σ a finite alphabet set, we denote by Σ^k the kth power of Σ, i.e. the elements of Σ^k are all the strings of length k over Σ. The union of all natural power of Σ, Σ^*, is the Kleene closure of Σ, i.e. all the finite strings over Σ including the empty string ε. Any subset L of Σ^* is called a *language*. We will use s, t, u to denote strings. For $s, t \in \Sigma^*$, s is called a *prefix* of t if there exists a string $u \in \Sigma^*$ such that $t = su$, where su is the string concatenation of s and u. Similarly, for $L_1, L_2 \subseteq \Sigma^*$, language *concatenation* L_1L_2 is defined as the set of strings formed from the concatenation of strings from L_1 and L_2. $L_1 \cup L_2$ denotes *union* of L_1 and L_2. Elements of L^k are all the strings that can be concatenated from k (possibly distinct) strings of L. Similarly we define L^* as the union of all natural powers of L. The complement of a language L is denoted L^c.

A *projection* P is defined between Σ^* and Σ_1^*, where Σ_1 is a subset of Σ, such that $P(\varepsilon) = \varepsilon$ and, recursively, for sa in Σ^*, $P(sa) = P(s)$ if $a \notin \Sigma_1$ and $P(sa) = P(s)a$ if $a \in \Sigma_1$. We can easily extend the projection to map from a language over Σ to a language over Σ_1. Thus, $P(L)$ erases those symbols not in Σ_1 from each string of L over Σ. For a language L_1 over Σ_1, we define P^{-1} as the inverse of projection P, i.e. $P^{-1}(L_1)$ is the set of strings over Σ whose P projections are in L_1.

In the rest of the paper, when we talk about two sub-alphabets Σ_1, Σ_2 of Σ, we will assume $\Sigma = \Sigma_1 \cup \Sigma_2$. Also, for two sets A and B, by $A \subsetneqq B$ we mean A is a strict subset of B, and by $A \subset B$ we mean either $A \subsetneqq B$ or $A = B$.

Given two sub-alphabets Σ_1, Σ_2 of Σ, we have two projections P_1 from Σ^* to Σ_1^* and P_2 from Σ^* to Σ_2^*. Then, we define *synchronous product* $L_1 \| L_2$ of language L_1 over Σ_1 and language L_2 over Σ_2 as $P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$. Thus, elements of $L_1 \| L_2$ are the strings over Σ whose P_i projections are in L_i for $i = 1, 2$. The *trace closure* of a single string s is $P_1(s) \| P_2(s)$ and the *decomposition closure* of L is $P_1(L) \| P_2(L)$. The *prefix closure* of a language L is denoted $\overline{L}$ consisting of strings that are prefix of some string in L. Both projection operator and inverse projection operator commute with prefix closure operator i.e. $P(\overline{L}) = \overline{P(L)}$ and $P^{-1}(\overline{L}) = \overline{P^{-1}(L)}$. L_1 and L_2 are called *non-conflicting* if $\overline{L_1} \cap \overline{L_2} = \overline{L_1 \cap L_2}$. L_1 and L_2 are called *synchronous non-conflicting* if $\overline{L_1} \| \overline{L_2} = \overline{L_1 \| L_2}$. Obviously L_1 and L_2 are synchronous non-conflicting if and only if $P_1^{-1}(L_1)$ and $P_2^{-1}(L_2)$ are non-conflicting. If L_1 and L_2 are prefix closed, then L_1 and L_2 are non-conflicting.

Following facts will be useful and most of them are standard [1].

Lemma 1: $P(L_1 \cap L_2) \subset P(L_1) \cap P(L_2)$, $P(L_1 \cup L_2) = P(L_1) \cup P(L_2)$, $P^{-1}(L_1 \cup L_2) = P^{-1}(L_1) \cup P^{-1}(L_2)$, $P^{-1}(L_1 \cap L_2) = P^{-1}(L_1) \cap P^{-1}(L_2)$.

Lemma 2: $L \subset P^{-1}(P(L))$ and $L = P(P^{-1}(L))$.

Lemma 3: Let Σ_1, Σ_2 be two sub-alphabets of Σ, and L a language over Σ, then L is a subset of $P_1(L) \| P_2(L)$. That is, every language is a subset of its decomposition closure.

Lemma 4: Let Σ_1, Σ_2 be two sub-alphabets of Σ, and L a language over Σ, then $P_i(P_1(L) \| P_2(L)) = P_i(L)$ for $i = 1, 2$. And $P_1(P_1(L) \| P_2(L)) \| P_2(P_1(L) \| P_2(L)) = P_1(L) \| P_2(L)$. Thus $P_1(L) \| P_2(L)$ is a solution of the language equation $P_1(X) \| P_2(X) = X$, where X is a language over Σ, for arbitrary language L over Σ. Interpreted differently, $P_1(L) \| P_2(L)$ is decomposable.

III. Decomposable Subset problem

A. Decomposable Language

Let language L be given as the global specification over Σ. We say that L is decomposable with respect to two sub-alphabets Σ_1, Σ_2, if it holds that $L = P_1(L) \| P_2(L)$. It is not difficult to verify whether a given regular language is decomposable with respect to two sub-alphabets since projection and synchronization operators preserve regularity of languages [11]. Note that in case L is not decomposable with respect to Σ_1, Σ_2, it is possible to redesign two sub-alphabets $\Sigma_1{}'$, $\Sigma_2{}'$ such that it holds that $L = P_1{}'(L) \| P_2{}'(L)$ [13], i.e, L is decomposable with respect to the two new sub-alphabets. This is possible because there are only finite choices for $\Sigma_1{}'$, $\Sigma_2{}'$ when Σ is a finite set.

In the rest of this subsection, properties of decomposable languages will be given in terms of usual set theoretic operations, i.e. union, intersection, and complement.

Proposition 1: If $L_a = P_1(L_a) \| P_2(L_a)$ and $L_b = P_1(L_b) \| P_2(L_b)$, where both L_a and L_b are over Σ, then $L_a \cap L_b = P_1(L_a \cap L_b) \| P_2(L_a \cap L_b)$. Thus, the class of decomposable languages is closed under intersection.

Proof: $L_a \cap L_b = (P_1(L_a) \| P_2(L_a)) \cap (P_1(L_b) \| P_2(L_b)) = P_1^{-1}(P_1(L_a)) \cap P_2^{-1}(P_2(L_a)) \cap P_1^{-1}(P_1(L_b)) \cap P_2^{-1}(P_2(L_b)) = (P_1^{-1}(P_1(L_a)) \cap P_1^{-1}(P_1(L_b))) \cap (P_2^{-1}(P_2(L_a)) \cap P_2^{-1}(P_2(L_b))) = P_1^{-1}(P_1(L_a) \cap P_1(L_b)) \cap P_2^{-1}(P_2(L_a) \cap P_2(L_b)) = (P_1(L_a) \cap P_1(L_b)) \| (P_2(L_a) \cap P_2(L_b))$ (Lemma 1).

But we have $P_1(L_a \cap L_b) \| P_2(L_a \cap L_b) \subset (P_1(L_a) \cap P_1(L_b)) \| (P_2(L_a) \cap P_2(L_b)) = L_a \cap L_b \subset P_1(L_a \cap L_b) \| P_2(L_a \cap L_b)$ (Lemma 1 and Lemma 3). So $L_a \cap L_b = P_1(L_a \cap L_b) \| P_2(L_a \cap L_b)$. □

However, the class of decomposable languages is not closed under union or complement. An easy counterexample is $L_m = \{bba, bab, abb\}$ and $L_n = \{aab, aba, baa\}$ over $\Sigma = \{a, b\}$, where $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$. We have $L_m = P_1(L_m) \| P_2(L_m)$ and $L_n = P_1(L_n) \| P_2(L_n)$ but $L_m \cup L_n = \{aab, aba, baa, bba, bab, abb\}$ is not decomposable. Since the class of decomposable language is closed under intersection and not closed under union, it is not closed under complement. We provide below a sufficient condition under which the union of two decomposable languages is still a decomposable language.

Proposition 2: Given $L_a = P_1(L_a) \| P_2(L_a)$ and $L_b = P_1(L_b) \| P_2(L_b)$, where both L_a and L_b are over Σ. Then, $L_a \cup L_b = P_1(L_a \cup L_b) \| P_2(L_a \cup L_b)$ if $P_1(L_a \cup L_b) \| P_2(L_a \cup L_b) \subset (L_a \cup L_b) \cup P_1^{-1}(P_1(L_a) \cap P_1(L_b)) \cup P_2^{-1}(P_2(L_a) \cap P_2(L_b))$.

Proof: The proof relies on Lemma 1 and is omitted for brevity. $\square$

Before we derive a sufficient condition for the complement of a decomposable language to be decomposable, we present several useful lemmas.

The projection operator P does not commute with complement operator. We have $P(L)^c \subset P(L^c)$ since P is surjective and not injective. However, the inverse projection operator P^{-1} commutes with the complement operator.

Lemma 5: $P^{-1}(L^c) = (P^{-1}(L))^c$.

Proof: $P^{-1}(L) \cap P^{-1}(L^c) = P^{-1}(L \cap L^c) = \emptyset$ and $P^{-1}(L) \cup P^{-1}(L^c) = P^{-1}(L \cup L^c) = \Sigma^*$. Thus we have $P^{-1}(L^c) = (P^{-1}(L))^c$. $\square$

Moreover, since projection operator is not injective, it is not true that $L = P^{-1}(P(L))$ for any language L. However, the following holds:

Lemma 6: $P(L)^c = P(L^c)$ if and only if $L = P^{-1}(P(L))$.

Proof: On the one hand, suppose we have $P(L)^c = P(L^c)$, then $P(L) \cap P(L^c) = \emptyset$. Then, for $\forall s$ such that $P(s) \in P(L)$, we have $\{P(s)\} \cap P(L^c) = \emptyset$, so $P(s) \notin P(L^c)$. This implies $s \notin L^c$, i.e. $s \in L$. Therefore, $\forall s \in P^{-1}(P(L)), s \in L$, i.e. $P^{-1}(P(L)) \subset L$. Since naturally $L \subset P^{-1}(P(L))$, we deduce that $L = P^{-1}(P(L))$.

Suppose on the other hand that $L = P^{-1}(P(L))$. Applying complement operator on both sides to obtain $L^c = P^{-1}(P(L)^c)$. We then apply P operator to get $P(L^c) = P(P^{-1}(P(L)^c)) = P(L)^c$ (Lemma 2). Thus, $P(L)^c = P(L^c)$. $\square$

With these lemmas, we are now able to provide a sufficient condition for the complement of a decomposable language to be decomposable.

Proposition 3: Given $L = P_1^{-1}(P_1(L))$ and $L = P_2^{-1}(P_2(L))$, then both L and L^c are decomposable.

Proof: Since $L = P_1^{-1}(P_1(L))$ and $L = P_2^{-1}(P_2(L))$, we have $L = P_1^{-1}(P_1(L)) \cap P_2^{-1}(P_2(L)) = P_1(L) \| P_2(L)$. Thus L is decomposable.

Since $L = P_1^{-1}(P_1(L)) \cup P_2^{-1}(P_2(L))$, by complementing both sides, we obtain $L^c = (P_1^{-1}(P_1(L)))^c \cap (P_2^{-1}(P_2(L)))^c = P_1^{-1}((P_1(L))^c) \cap P_2^{-1}((P_2(L))^c) = P_1(L)^c \| P_2(L)^c = P_1(L^c) \| P_2(L^c)$ since $L = P_1^{-1}(P_1(L))$ and $L = P_2^{-1}(P_2(L))$. $\square$

In the following, we provide a sufficient condition for the prefix closure of a decomposable language to be decomposable.

Proposition 4: Suppose $P_1(L)$ and $P_2(L)$ are synchronous non-conflicting. If L is decomposable with respect to Σ_1 and Σ_2, then $\overline{L}$ is decomposable with respect to Σ_1 and Σ_2.

Proof: Since $L = P_1(L) \| P_2(L)$, we have $\overline{L} = \overline{P_1(L) \| P_2(L)} = \overline{P_1(L)} \| \overline{P_2(L)} = P_1(\overline{L}) \| P_2(\overline{L})$. Thus $\overline{L}$ is decomposable with respect to Σ_1 and Σ_2. $\square$

The following proposition characterizes nicely the relationship between language L and its decomposition closure.

Proposition 5: The decomposition closure $P_1(L) \| P_2(L)$ of L is the minimal superset of L that is decomposable with respect to Σ_1 and Σ_2.

Proof: The proposition is obviously true for the case when L is decomposable with respect to Σ_1 and Σ_2. Thus we only have to consider the case when L is a strict subset of $P_1(L) \| P_2(L)$. Clearly $P_1(L) \| P_2(L)$ is decomposable with respect to Σ_1 and Σ_2 (Lemma 4). Thus we only have to show that there exists no language L', such that $L \subsetneqq L' \subsetneqq P_1(L) \| P_2(L)$ and at the same time L' is decomposable with respect to Σ_1 and Σ_2. Suppose to the contrary L' is decomposable with respect $L' = P_1(L') \| P_2(L')$, we then have $L' = P_1(L') \| P_2(L') \supset P_1(L) \| P_2(L)$ leading to a contradiction. So $P_1(L) \| P_2(L)$ is the minimal superset of L that is decomposable with respect to Σ_1 and Σ_2. $\square$

It is easy to see that any decomposable language containing L must also contain the decomposition closure of L.

B. Decomposable Subset

Naturally, if the global specification L is not decomposable with respect to Σ_1, Σ_2, we may ask is there any subset L' of L that is decomposable with respect to Σ_1, Σ_2, i.e., $L' = P_1(L') \| P_2(L')$. If this condition holds, we say that L has a *decomposable* subset with respect to Σ_1, Σ_2. Interpreting L as the set of total tasks to be completed, L' is one possible set of tasks which can be decomposed to obtain a distributed implementation. In case L is the maximally legal behavior of the system, L' is one possible legal behavior. Both L and L' have to be prefix closed in the later interpretation.

If the answer to above problem is yes, then we can continue to design local supervisors such that the supremal controllable sublanguage of $P_i(L')$ for $i = 1, 2$ is achieved under the Ramadge-Wonham paradigm [1]. The resultant composed system will surely be restricted as a subset of $L' \subset L$ and thus complete part of the tasks or behave in a legal way.

C. Decomposable Subset Problem

We provide now the formal definition of the central problem that we are interested in, i.e. the Decomposable Subset problem:

Problem 1: Given a regular language L, decide if $\exists L' \subset L$, such that $L' = P_1(L') \| P_2(L')$. Or equivalently, decide if $\exists$ $L'' \subset L$ such that $P_1(L'') \| P_2(L'') \subset L$, where L' and L'' are non-empty.

The equivalence of the two definitions in above problem is justified by the following proposition. Intuitively the proposition says that a language L has a decomposable subset if and only if there exists a subset of L whose decomposition closure is still a subset of L.

Proposition 6: There exists a subset L' of L such that $L' = P_1(L') \| P_2(L')$ if and only if there exists a subset L'' of L such that $P_1(L'')\|P_2(L'')$ is a subset of L, where L' and L'' are non-empty.

Proof: Necessity: if there exists a non-empty subset L' of L such that $L' = P_1(L')\|P_2(L')$, we can pick $L'' = L'$ so that $P_1(L'')\|P_2(L'') = P_1(L')\|P_2(L') = L' \subset L$. L'' is also non-empty.

Sufficiency: if there exists a non-empty subset L'' of L such that $P_1(L'')\|P_2(L'') \subset L$ holds, we just simply set $L' = P_1(L'')\|P_2(L'')$. Then we have $P_1(L')\|P_2(L') = P_1(P_1(L'')\|P_2(L''))\|P_2(P_1(L'')\|P_2(L''))$ $= P_1(L'')\|P_2(L'') = L' \subset L$ due to Lemma 4. It is easy to see that $L' = P_1(L'')\|P_2(L'')$ is also non-empty. □

The formal definition of prefix closed version of Decomposable Subset problem is given in [8] and reproduced below:

Problem 2: Given a prefix closed regular language L, decide if there exists a prefix closed subset L' of L , such that $L' = P_1(L')\|P_2(L')$ and all those symbols in Σ must appear in L'. Or equivalently, decide if there exists a prefix closed subset L'' of L such that $P_1(L'')\|P_2(L'') \subset L$ and all the symbols in Σ must appear in L''.

Remark: The purpose of introducing the alphabet constraint is to avoid trivial solutions since $\epsilon \in L$, for every prefix closed language L and $\{\epsilon\}$ is a decomposable language trivially.

The equivalence of the two definitions in the above problem is justified by the next proposition relying on the following lemma:

Lemma 7: If L is prefix closed, then $P_1(L)$ and $P_2(L)$ are synchronous non-conflicting and $P_1(L)\|P_2(L) = \overline{P_1(L)\|P_2(L)}$, i.e. $P_1(L)\|P_2(L)$ is prefix closed.

Proof: Straightforward since projection and inverse projection operators preserve prefix closure, and prefix closed languages are non-conflicting. □

Proposition 7: There exists a prefix closed subset L' of L such that $L' = P_1(L')\|P_2(L')$ if and only if there exists a prefix closed subset L'' of L such that $P_1(L'')\|P_2(L'') \subset L$, where all the symbols of Σ appear in both L' and L''.

Proof: Most of the proof follows directly from that of proposition 6. The only extra effort is to prove that L' is prefix closed if and only if L'' is prefix closed. This fact is obvious with the aid of Lemma 7. The proof is complete by observing that all the symbols of Σ appear in L' if and only if all of them appear in L''. □

The undecidability of prefix closed version of Decomposable Subset problem is estabilished in [8]. The undecidability of Decomposable Subset problem can be easily obtained from the undecidability of Trace Closed Subset problem [9] discussed in the next section.

IV. Problems related to the Decomposable Subset Problem

A. Trace Closed Language

A language L is called *trace closed* with respect to two sub-alphabets Σ_1 and Σ_2 of Σ if $\forall s \in L$, $P_1(s)\|P_2(s) \subset L$.

The concepts of trace closed language and decomposable language are different. Decomposable language is trace closed but trace closed language may not be decomposable [9]. However, these two concepts become the same when there is only one observationally distinguished string in L, i.e. $L = P_1(s)\|P_2(s)$ for some $s \in \Sigma^*$. This is also presented in [10]. It is interesting to note that the trace closure of a regular language can be irregular and the trace closure of an irregular language is always irregular. The decomposition closure of a regular language is always regular and the decomposition of a irregular language can be regular. Given a regular language, it is decidable whether it is trace closed [10].

B. Related Problems

The Trace Closed Subset problem is given in [9] and reproduced next:

Problem 3: Given a regular language L over Σ, decide whether $\exists L' \subset L$ such that $\forall s \in L', P_1(s)\|P_2(s) \subset L'$, where L' is non-empty.

It is pointed out in [9] that Trace Closed Subset problem is undecidable. Prefix closed version of Trace Closed Subset problem is given in [8] and also shown there that the problem is undecidable:

Problem 4: Given a prefix closed regular language L over Σ, decide if there exists a non-empty prefix closed subset L' of L such that $\forall s \in L', P_1(s)\|P_2(s) \subset L'$, where all the symbols of Σ must appear in L'.

The notion of String Observation Closure is fundamental in both Trace Closed and Decomposable Subset problems. String Observation Closure problem asks whether the given regular language L has a string whose trace closure is contained in L:

Problem 5: Given a regular language L, decide if $\exists s \in L$, such that $P_1(s)\|P_2(s) \subset L$.

String Observation Closure problem is equivalent to both Trace Closed Subset problem and Decomposable Subset problem. However, this is not the case when the languages are required to be prefix closed. In the rest of the section, we will only investigate the general case. Note that in the general case, it's not required that all the symbols in Σ appears in the decomposable subset.

Proposition 8: There exists a non-empty subset L' of L such that $L' = P_1(L')\|P_2(L')$ if and only if $\exists s \in L$, such that $P_1(s)\|P_2(s) \subset L$.Thus Decomposable Subset problem is equivalent to String Observation Closure problem.

Proof: The proof is exactly the same as the proof for proposition 6 except L'' is changed to $\{s\}$. □

Proposition 9: Given language L, $\exists L' \subset L$ such that $\forall s \in L', P_1(s)\|P_2(s) \subset L'$ if and only if $\exists s \in L$, such that $P_1(s)\|P_2(s) \subset L$, where L' is non-empty. Thus Trace Closed Subset problem is equivalent to String Observation problem.

Proof: The proof is straightforward. □

The essence of proposition 8 and proposition 9 is that to check whether a regular language L has a non-empty trace closed subset or decomposable subset, it is equivalent to checking whether there is a string whose trace closure is contained in L. Thus all these problems are undecidable [8],[9]. The undecidability result can be easily extended to the case of more than two sub-alphabets.

Remark: Informally, the difficulty of this class of problems lies in the fact that there are generally infinite number of strings s that must be checked (and there is no finite procedure that avoids the verification of them all) [8]. It is interesting to see that Trace Closed Subset Problem is undecidable but there is algorithm to decide whether a given regular language is trace closed, since both problems are dual to each other. A tentative explanation to this phenomenon is that it is possible to examine all strings up to some lengths in L and conclude that all strings have their trace closure contained in L if those short strings have. However, if all those short strings up to any bounded lengths do not have their trace closure contained in L, we cannot conclude that there is not any longer string having its trace closure in L. An example is given in the proof of [8].

Finally, we provide two problems that are related to Decomposable Subset problem. Both undecidability results are trivial consequences of undecidability of Decomposable Subset problem.

Problem 6: (*Conditionally Decomposable Subset problem*, see [14] for *Conditionally Decomposability*) Given a regular language L over Σ, decide whether $\exists L' \subset L$, such that $L' = P_{1+k}(L') \| P_{2+k}(L')$ where $\Sigma_1 \cap \Sigma_2 \subset \Sigma_k \subset \Sigma_1 \cup \Sigma_2 = \Sigma$ and L' is non-empty. Here P_{1+k} is a map from 2^{Σ^*} to $2^{(\Sigma_1 \cup \Sigma_k)^*}$ and P_{2+k} from 2^{Σ^*} to $2^{(\Sigma_2 \cup \Sigma_k)^*}$.

Proposition 10: Conditional Decomposable Subset problem is undecidable for empty Σ_k.

Proof: Transforming the Conditional Decomposable Subset problem to its corresponding String Observation Closure problem, the proof follows exactly as in [8], [9]. Indeed, $\Sigma_1 \cap \Sigma_2 = \emptyset$ in their proof. □

However, the problem is still open when we require $\Sigma_k \neq \emptyset$, which is a more useful and meaning case. See [14] for the motivation of introducing the concept of Conditional Decomposability.

Problem 7: (*Weakly Decomposable Subset problem*, see [15] for Weak Decomposability) Given a pair of regular languages L and M, decide whether $\exists L' \subset L$, such that $L' = M \cap (P_1(L') \| P_2(L'))$, where L' is non-empty.

Proposition 11: Weakly Decomposable Subset problem is undecidable.

Proof: By setting $M = \Sigma^*$, the undecidability of Weakly Decomposable Subset problem follows from the undecidability of Decomposable Subset problem. □

C. Further Observations and Two General Problems

We will discuss a connection between the concepts of Joint Observability and Trace Closure with respect to two given subalphabets.

Problem 8: (*Joint Observability* [15]) Given a pair of languages $L \subset M$, decide whether it is true that $\forall s \in L$, $(P_1(s) \| P_2(s)) \cap M \subset L$.

If the above holds, then L is said to be *jointly observable* with respect to M. Intuitively, L is jointly observable with respect to M if any string in L can be distinguished from any string in $M \cap L^c$ observed through P_1 and P_2. In other words, there can not be two indistinguishable strings, one in L and the other in $M - L$.

Proposition 12: L is jointly observable with respect to Σ^* if and only if L is trace closed.

Proof: Straightforward from definitions. □

Although in general joint observability of a pair of regular languages L with respect to M is undecidable [15], the joint observability of regular language L with respect to Σ^* is decidable.

The following two problems are general enough to summarize most of the above problems. One is joint observability problem and the other is its dual form.

Problem 9: Decide whether it is true that $\forall s \in K, (P_1(s) \| P_2(s)) \cap L \subset K$, where K is a regular language.

Case 1: When L is Σ^*, this problem becomes Trace Closed problem.

Case 2: When L is a regular language containing K, this problem becomes Joint Observability problem.

Case 3: When L is a prefix closed regular language containing K, this problem coincides with the open problem of [15].

Problem 10: Decide whether it is true that $\exists s \in K, (P_1(s) \| P_2(s)) \cap L \subset K$, where K is a regular language. This problem is undecidable for all the three cases.

V. Conclusion

In this paper we point out that language based top-down supervisor synthesis is in general not effective in Ramadge-Wonham framework. It is then of interest to investigate what kind of structural properties can we impose on the specification so that the problem becomes decidable and useful progress can be obtained. It is worth noting that most of the discussed problems are specific forms of joint observability problem and its dual form.

Acknowledgements: The third author was supported by the Romanian National Authority for Scientific Research grant no. PN-II-ID-PCE-2011-3-0688 (project MuVet).

References

[1] W. M. Wonham, *Supervisory Control of Discrete-Event Systems*, Systems Control Group, Dept. of ECE, University of Toronto, Toronto, Canada, 2010.

[2] W. M. Wonham, P. J. Ramadge, "On the supremal controllable sublanguage of a given language", SIAM Journal on Control Optimization 25(3), pp. 637–659, 1987.

[3] P. J. Ramadge, W. M. Wonham, "Supervisory control of a class of discrete event processes", SIAM Journal on Control Optimization 25(1), pp. 206–230, 1987.

[4] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages", Systems and Control Letters 15(2), pp. 111–117, 1990.

[5] F. Lin, A. F. Vaz, W. M. Wonham, "Supervisor specfication and synthesis for discrete event systems", International Journal of Control 48(1), pp. 321–332, 1988.

[6] M. Mohammad, H. Lin, "Guaranteed global performance through local coordinations", Automatica, Volume 47 Issue 5, May, 2011.

[7] K. Rohloff, T.-S. Yoo, S. Lafortune, "Deciding co-observability is PSPACE-complete", IEEE Transactions of Automatic Control 48(11), pp. 1995–1999, 2003.

[8] A. Stefanescu, "Automatic synthesis of distributed transition systems", Ph.D. dissertation. University of Stuttgart, 2006.

[9] Y. Chen, X. C. Ding, A. Stefanescu, C. Belta, “Formal approach to the deployment of distributed robotic teams”, IEEE Transactions on Robotics 28(1), pp. 158–171, 2012.

[10] S. Tripakis. “Two-phase distributed observation problems”, In Proc. of 5th Conf. on Application of Concurrency to System Design (ACSD’05), pp. 98–105, IEEE Computer Society 2005.

[11] J. E. Hopcroft, J. D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, Massachusetts, 1979.

[12] K. Heljanko, A. Stefanescu. “Complexity results for checking distributed implementability”, In Proc. of 5th Conf. on Application of Concurrency to System Design (ACSD’05), pp. 78–87, IEEE Computer Society 2005.

[13] M. Mohammad, H. Lin, “Communicate only when necessary: Cooperative tasking for multi-agent systems”, CoRR abs/1106.3134: 2011

[14] J. Komenda, T. Masopust, J. H. van Schuppen, “On conditional decomposability”, CoRR abs/1201.1733: 2012.

[15] S. Tripakis. “Undecidable problems of decentralized observation and control on regular languages”, Information Processing Letters 90(1), pp. 21–28, 2004.