Towards decentralized and parameterized supervisor synthesis

Lin, Liyong

2015

Lin, L. (2015). Towards decentralized and parameterized supervisor synthesis. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/65641>

<https://doi.org/10.32657/10356/65641>

TOWARDS DECENTRALIZED AND PARAMETERIZED SUPERVISOR SYNTHESIS

NANYANG
TECHNOLOGICAL
UNIVERSITY

LIN LIYONG

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2015

This thesis is dedicated to my parents

Acknowledgements

I would like to thank my research supervisor, Prof. Rong Su, for the patient guidance, warm encouragement and thoughtful advice he has provided throughout my PhD study. A sizable part of this thesis benefits from the discussions I had with him. This work has also benefited from the discussions with Dr. Simon Ware, Prof. Weilin Wang, Prof. Alin Stefanescu and Prof. Walter Murray Wonham. The comments and suggestions of the thesis examiners have helped me greatly improve the quality of the thesis. Financial support from NTU Research Student Scholarship is also gratefully acknowledged. Finally, I would like to thank my family for their love and support. Special thanks go to my wife, Ling Lin, for her accompany in times of hardship.

Abstract

This thesis studies the decentralized and parameterized supervisor synthesis problems in the Ramadge-Wonham paradigm.

We obtain a characterization of the solvability of the decentralized realization problem, by reducing it to the problem of solving a set of language equations. The close relationship among the problems of language decomposition, supervisor decomposition and decentralized realization is then studied. Complexity-theoretic lower bound proofs are provided by reductions from the language decomposition problem. We then develop complexity reduction techniques for language decomposability verification.

A sufficient condition for the undecidability of the distributed supervisor synthesis problem is derived based on the structure of the distributed control architecture. We also present a sufficient condition for the undecidability of a language based parameterized supervisor synthesis problem.

The expressive power of the symbolic reachability relations of a class of parameterized systems is investigated and a sound state based parameterized supervisor synthesis procedure is developed in the framework of regular model checking.

Contents

Acknowledgements	ii
Abstract	iii
Contents	iv
List of Figures	vii
List of Algorithms	viii
List of Abbreviations	ix
1 Introduction	1
1.1 Control of Discrete-Event Systems	2
1.2 Contributions and Outlines of the Thesis	8
2 Preliminaries	11
2.1 Basic Notions and Notations	11
2.1.1 Languages and Automata	12
2.1.2 Trace Theory	15
2.1.3 Supervisory Control	17
2.2 Basic Results	17

3	Decentralized Realization	19
3.1	System Setup	21
3.2	Characterization with Language Equations	24
3.2.1	Characterization Result	24
3.2.2	Supervisor Decomposition as Decentralized Realization	29
3.2.3	Complexity Results	30
3.3	Verification of Language Decomposability	32
3.3.1	Reducible Distributions	33
3.3.2	Prefix-closed Specifications	40
3.4	Discussions	41
4	Decentralized Supervisor Synthesis	44
4.1	Decomposable Sublanguage Problem	46
4.1.1	Characterization of the Decidability	47
4.1.2	Sound Heuristics	48
4.2	Joint Observability Problem	53
4.2.1	Characterization of the Decidability	53
4.2.2	Prefix-closed Joint Observability Problem	54
4.3	Towards Decentralized Supervisor Synthesis	55
4.4	Parameterized Supervisor Synthesis Problem	60
4.5	Discussions	69
5	Symbolic Analysis of Parameterized Systems	71
5.1	Miscellaneous Preliminaries	75
5.2	Globally Synchronized Templates	77
5.2.1	System Setup	78
5.2.2	Expressiveness Characterization	80
5.2.3	Dynamic Systems with Entrance and Exit	87

5.2.4 Applications to Model Checking	88
5.3 Regular Supervisor Synthesis	93
5.4 Discussions	98
6 Conclusions	100
Author's Publications	106
Bibliography	106

List of Figures

2.1	Graph P_4	16
2.2	Graph C_4	16
3.1	Distribution $\Delta_c = (\{a, b\}, \{b, c\}, \{c, d\}, \{e, f\}, \{f, g\}, \{g, h\})$	38
4.1	Automaton G that recognizes $L = (ab + c)^*(a + (b + ac + aa)(a + b + c)^*$	49
4.2	Automaton G' that recognizes $L^c = (ab + c)^*$	49
5.1	Control logic for a team of UAVs	73
5.2	A globally synchronized template G and the corresponding G^∞	79
5.3	A globally synchronized template G_c	82
5.4	Construction of G'_c using Proposition 5.2	85
5.5	Construction of G''_c	86
5.6	Illustration for the length-preserving cross product operation	95

List of Algorithms

5.1	Regular Supervisor Synthesis	96
-----	--	----

List of Abbreviations

DRP	Decentralized Realization Problem
SDP	Supervisor Decomposition Problem
LDP	Language Decomposition Problem
DeSSP	Decentralized Supervisor Synthesis Problem
DiSSP	Distributed Supervisor Synthesis Problem
DSP	Decomposable Sublanguage Problem
JOP	Joint Observability Problem
PCJOP	Prefix-closed Joint Observability Problem
STSP	Specification Template Synthesis Problem
PSSP	Parameterized Supervisor Synthesis Problem
PCP	Post Correspondence Problem

Chapter 1

Introduction

The design of correct distributed (finite state) systems remains a challenging computational problem. The difficulty of this problem seems to be inherent in the very nature of these systems, as evidenced by the vast amount of proven *intractability* results in the *verification* and *synthesis* of distributed systems (see for example [1], [2], [3], [4], [5], [6] and the references therein). Intuitively, the cause of this fact is the *combinatorial explosion of states*, which makes verifying the correctness of distributed systems already difficult, let alone the synthesis task. To make it worse, there are many different frameworks within which the synthesis of distributed systems could even become *undecidable* due to the lack of complete information (see for example [6], [7], [8], [9], [10], [11], [12]), even when all the involved languages are regular. A tentative explanation of this fact is that in general computing non-regular under-approximations or an exhaustive search through an infinite set of regular sublanguages of given regular languages may become necessary when partial observation is involved¹. In the next section, we introduce the framework that we shall study throughout the thesis, i.e., *supervisory control of discrete-event systems*.

¹It will become clear in Chapter 4 that in deciding the existence of non-empty decomposable sublanguages for any given regular language, which is an important problem in the synthesis of distributed supervisors, in general we may need to compute the supremal trace closed sublanguage of the given regular language, which is not necessarily regular.

1.1 Control of Discrete-Event Systems

A discrete-event system is a dynamic system equipped with a discrete state space and a state-transition structure [13]. Each state transition is labeled by an event from a fixed, but possibly infinite, alphabet. Thus, a discrete-event system is often represented by an (incomplete) *automaton*. Unless explicitly stated otherwise, we shall only consider discrete-event systems that are represented by *finite state automata* before Chapter 5.

The way a discrete-event system interacts with other discrete-event systems is through *lockstep synchronization*, i.e., the interaction of different discrete-event systems is modeled by the *synchronous product* of automata. From an *observer's* point of view, the alphabet of a discrete-event system is partitioned into two disjoint subsets, i.e., the subset of *observable* events and the subset of *unobservable* events. From a controller's point of view, the alphabet is partitioned into the subset of *controllable* events and the subset of *uncontrollable* events. For a discrete-event system, any other discrete-event system that interacts with it could be treated as both its observer and controller, i.e., its supervisor. Synchronization implements observation and control mechanism, i.e., control and observation information is implicitly communicated between interacting discrete event systems through synchronization. A supervisor² is considered to *enable* an event σ at a particular state if at that state there is an outgoing transition labeled by σ . It is considered not to *observe* an event σ at a particular state if the existence of an outgoing transition labeled by σ at that state implies that outgoing transition is a self-loop. Given a finite state automaton representing the *plant* to be controlled, the synthesis problem requires computing a tuple of finite state automata representing a tuple of *supervisors* so that the *closed-loop system*, i.e., the synchronous product of the plant and the supervisors, *satisfies*

²see page 21 for the definition of a supervisor and more details.

a given *specification*. There are two interpretations of the statement that a finite state automaton satisfies a specification that will interest us in this thesis: 1) the finite state automaton is *language equivalent* to the given specification; 2) the finite state automaton is language equivalent to a non-empty subset of the given specification. We classify the synthesis problem into the *decentralized realization problem* and the *decentralized supervisor synthesis problem* accordingly. If no constraint is imposed on the supervisors, then the synthesis problem becomes rather trivial. A much more interesting case is when the supervisors can neither observe nor control all of the plant's alphabet. A supervisor "effectively" cannot observe σ if and only if it does not observe σ at each state. A supervisor "effectively" cannot control σ if and only if at each state of the supervisor σ is enabled.

The theory of supervisory control of discrete-event systems was originally initiated in the 1980s by Ramadge and Wonham [14], [15], [16], [17] and has been under active development for more than 30 years (see the textbooks and monographs [13], [18], [19]). The main domains of applications for supervisory control theory are the coordination of complex man-made systems such as manufacturing systems [13], [20], [21], telecommunication and network protocols [22], [23], database management systems [24] and transportation [13], [25], e.t.c., where coordination and conflict resolution [26], [27], [28] are usually needed to ensure certain safety and liveness properties. A brief list of some of the more important developments and those developments that are more closely related to ours is provided below.

- The important concepts of *controllability* and *observability*, which along with the notion of $L_m(G)$ -*closedness*³ together characterize the solvability of the *centralized realization problem* [15], [23], [29], are first formulated in [15] and [29]. In characterizing the solvability of the decentralized realization problem [30], the more general notion of *co-observability* [31] has to be used in place of

³If the supervisor is assumed to be marking, then we only need controllability and observability.

observability. When the given specification does not satisfy the above properties, (optimal) under-approximations that satisfy these properties need to be computed, giving rise to the problem of decentralized supervisor synthesis. The decentralized supervisor synthesis problem is much harder than the decentralized supervisor realization problem. In fact, there is provably no algorithm, according to the Church-Turing thesis [32], that correctly determines in finite time the existence of such an under-approximation for all instances of the decentralized supervisor synthesis problem even when all the involved automata are of finite states, i.e., the decentralized supervisor synthesis problem is undecidable [7], [8]. Because of this major hurdle, a large number of works have been devoted to coping with the undecidability and intractability in the decentralized supervisor synthesis problem (see [20], [31], [33], [34], [35], [36], [37], [38], [39], [40] for example). There are some special decidability results [9], [31], [41] and many heuristics have been developed (see [20], [33], [34], [36], [37], [38], [42] for example). There are two major classes of heuristics that have been proposed, i.e., *approximation algorithms* and *semi-algorithms*. For example, the approximation algorithm of [38] uses the notion of *relative co-observability* to approximate the notion of co-observability and modular over-approximation technique has been used in the approximation algorithm of [37]; the semi-algorithm of [36], if ever terminates, is guaranteed to compute a maximal solution, using the notion of strong Nash equilibrium. The design of heuristics for decentralized supervisor synthesis problem is an active research area of great importance. There are also many other interesting extensions of the classical framework. For example, there are works on supervisor synthesis for non-deterministic automata [43], supervisor synthesis for petri nets [44], [45], supervision on infinite behavior [46], [47] and supervisor synthesis for mu-calculus [9], [48]. For a good overview of

some recent progresses made on the decentralized supervisor synthesis problem, the readers may refer to [49]. For a general introduction and survey of existing decentralized control methods in classical control theory, e.g., decentralized control of linear, nonlinear and stochastic systems, the readers may refer to [50], [51]. For a game theoretic perspective on multi-agent decision making and the application to decentralized supervisor synthesis, the readers may refer to [36], [52]. Those developments that are more closely related to ours are discussed in the following.

- A general technique for obtaining automata-theoretic characterization results for a large class of decentralized realization problems is presented in [53]. However, it is not straightforward to translate these characterization results to their language-theoretic counterparts, since it is not clear what language operations correspond to the automata operations⁴ involved in [53]. Later, a collection of language-theoretic characterization results are obtained in [54], in which the notions of controllability and co-observability are nicely combined. However, [54] only considers the case of synthesis for the *closed behavior* of the plant and it does not mention how to generalize these results to other synthesis problems, e.g., synthesis for the marked behavior. The classical synchronization model treats supervisors and plants equally, except for the possible constraints imposed on the supervisors. This is later referred to as the *conjunctive fusion rule* [55], where the supervisors together disable an event if and only if there is at least one supervisor that does so. For a general setup where both the conjunctive fusion rule and the *disjunctive fusion rule* exist, the characterization results are obtained in [55]. The problem of decentralized realization is known to be *PSPACE-complete* even in the classical setup [2]. On

⁴It is not even straightforward to model the following simple automaton operation: for each state s of a given automaton and each uncontrollable event σ , add a self-loop labeled by σ at state s if there is no outgoing transition labeled by σ at state s .

the other hand, the decentralized supervisor synthesis problem is undecidable even for non-blocking property [7]. And the undecidability result also holds when infinite state supervisors are considered [8], modal (loop) formulas are considered [9] or priority order is to be enforced [56]. Unfortunately, the boundary between the decidability and undecidability of the decentralized supervisor synthesis problem still remains unclear.

- The notion of *language decomposability* and its importance to the decentralized supervisor synthesis problem has been studied in [30], [57]. In [57], it is used to characterize the condition under which distributed supervisor synthesis, where the global plant is given by the synchronous product of a collection of local plants, can achieve the optimal behavior achievable by a global supervisor. Recently, a top-down *supervisor localization* technique is developed for solving the distributed supervisor synthesis problem [20], which also requires the specification to be a priori language decomposable. Also, a special case of the language decomposability, called *conditional decomposability*, plays an important role in the *coordination control* of discrete event systems [58]. A complexity-theoretic study of the language decomposability has been conducted in [6] and the verification problem is proved to be PSPACE-complete even when the specification is required to be prefix-closed [6]. It is shown in [58] that there exists a polynomial time algorithm for verifying conditional decomposability. In view of the importance of the notion of language decomposability to the supervisor synthesis problems, complexity reduction techniques for its verification seem to deserve more careful study. It is also of great interest to synthesize non-empty decomposable sublanguages, if possible, from any given specification which is a priori not language decomposable, to bridge the gap of the approach used in [20]. We also hope the heuristics developed for the decomposable sublanguage problem could eventually be used for solving the

decentralized supervisor synthesis problem.

- The problems of *model checking* for *parameterized systems* are known to be undecidable (see for example [59], [60], [61], [62], [63] and the references therein). It is not difficult to see that the supervisor synthesis problems for these systems are also undecidable. The synthesis problems for some classes of parameterized systems have been previously studied in [64], [65]. There are also several works that directly address the problem of supervisor synthesis for a general class of infinite state systems, that are based on *predicates* and *predicate transformers* [66], [67], [68], [69], [70]. Although *deadlock avoidance* techniques have been well studied, the problem of enforcing *livelock freedom* for the closed-loop system has not been addressed in these works. The *binary decision diagram* based symbolic implementation [21], [71] has been quite successful in dealing with *non-blocking* control of large scale discrete event systems. However, it is not ideally suitable for *parameterized supervisor synthesis* purpose, since the parameterized supervisor in general has to be inferred from the solved instance and the inferred supervisor has to be verified/proved to be correct for an arbitrary number of modules in the system.

The main motivation of this work is to contribute to a better theoretical understanding of the boundaries between the decidability and undecidability, the tractability and intractability in the verification and synthesis of decentralized discrete-event systems (including distributed discrete-event systems) and parameterized discrete-event systems, which are still far from being understood. If a problem is proved to be intractable (respectively, undecidable), then either efficient heuristics need to be developed or tractable (respectively, decidable) fragments need to be identified. The intractability (respectively, the undecidability) proof often provides hints on how to systematically proceed in this direction of research. Since the major hurdle for practical industrial applications of automatic supervisor synthesis is the high

computational complexity and the undecidability, we believe that the theoretical investigation carried out in this work may also have practical relevance. The complexity reduction techniques for the verification of language decomposability and the heuristics for the synthesis of non-empty decomposable sublanguages developed in this work could be used to verify the assumption and bridge the gap of [20], where applications to the supervision of manufacturing systems have been nicely demonstrated. The procedure of supervisor decomposition developed in this work can also be used as a heuristic for solving the decentralized supervisor synthesis problem: an (optimal) global supervisor is first synthesized; if the global supervisor cannot be decomposed into local supervisors, then the decentralized control architecture can be suitably modified to make the global supervisor decomposable, e.g., by increasing the subsets of events each local supervisor can observe and control.

1.2 Contributions and Outlines of the Thesis

This thesis will address some of the issues raised above. The structure and contributions of the thesis are as follows.

Chapter 2 We begin with the preliminaries.

Chapter 3 Then we present a language-theoretic characterization of the solvability of the decentralized realization problem, by reducing it to the problem of solving a set of language equations. The key idea involves the use of a language equational characterization of finite state supervisors. This greatly simplifies the characterization task: characterizing the solvability of the decentralized realization problem is reduced to the straightforward problem of characterizing the solvability of the set of language equations. Our approach could be seen as a simple alternative to the approach of [54], which requires computing the infimal prefix-closed, controllable, and observable superlanguage of an arbitrary language. Apart from the difference

in the points of view, we also consider the case of synthesis for the *marked behavior* of the plant, which is different from [54]. The languages of the test supervisors in our characterization result turn out to be different from the languages of the test supervisors used in [53]. We then show that the problem of language decomposition is a special case of the *supervisor decomposition problem*, which is again a special case of the decentralized realization problem. The decentralized realization problem and the supervisor decomposition problem are then shown to be PSPACE-complete. The proof of the PSPACE-completeness of the decentralized realization problem is much simpler than that of [2]. We proceed to study complexity reduction techniques for verifying language decomposability, for the cases when the distributions satisfy certain special properties or when the test languages are *prefix-closed*, in an attempt to pave the way for reducing the complexity of solving both the supervisor decomposition problem and the decentralized realization problem.

Chapter 4 We show that the *decomposable sublanguage problem*, i.e., the problem whether an arbitrary *regular* language has a non-empty decomposable sublanguage with respect to a fixed given distribution, is decidable if and only if the *independence relation* induced by the given distribution is *transitive*. And the *joint observability problem*, i.e., the problem whether two arbitrary disjoint regular languages have strings that are indistinguishable with respect to a fixed given distribution, is decidable if and only if the independence relation induced by the given distribution is a *transitive forest*. These results are obtained using available characterization results in trace theory. We also present a new proof of the undecidability of the *prefix-closed joint observability problem*, by a reduction from the joint observability problem. Sound heuristics for computing the *trace closure* of an arbitrary regular language are proposed, which can be applied to solving the above undecidable problems. A sufficient condition on the *distributed control architecture* is then derived, under which there exist some fixed local finite state plants such that the distributed

supervisor synthesis problem is undecidable. We also show that a natural formulation of language based parameterized supervisor synthesis problem is undecidable for a fixed finite state plant template, so long as the template alphabet has at least two *private* events and one *global* event that are controllable. In particular, all the undecidability results for the synthesis problems are still valid even if the global (schematic) specifications are required to be *star free*, or equivalently, *linear temporal logic definable*.

Chapter 5 We perform symbolic reachability analysis and supervisor synthesis for parameterized systems. We begin with some miscellaneous preliminaries. We then study the *symbolic reachability relations* of a class of parameterized systems in the framework of *regular model checking*. The modules of each system are instantiated from a *globally synchronized template*, which is represented by a finite state automaton whose event set consists of global and private events. We show that the symbolic (co-)reachability relations of these systems are effectively *iteration-closed* finite unions of *atomic star expressions*. We also show that for any iteration-closed finite union of atomic star expressions, there exists a template with only global events that *realizes* it. Applications of the symbolic reachability analysis to computing the sets of reachable bad states for various notions of deadlock and blocking are then presented. A semi-algorithm for computing the supremal non-blocking supervisor that enforces state avoidance property for parameterized systems is then developed in the framework of regular model checking.

Chapter 6 Conclusions and suggestions for future research are presented.

Chapter 2

Preliminaries

This chapter is devoted to mathematical preliminaries. We present in Section 2.1 the basic notions and notations (languages and automata, *trace theory*, supervisory control). Interested readers are referred to [13], [32], [72], [73], [74], [75] for more details. Then we collect some basic results in Section 2.2 to make the thesis self-contained. Miscellaneous preliminaries will be introduced in the beginning of Chapter 5, where the symbolic framework of regular model checking is adopted.

2.1 Basic Notions and Notations

Let $\mathbb{N}$ be the set of non-negative integers. We use $[1, n]$, where $n \geq 1$, to denote the set $\{1, 2, \dots, n\}$. The *set difference* of A and B is denoted by $A \setminus B := \{x \in A \mid x \notin B\}$. For a finite set A , $|A|$ is used to denote its *cardinality*. A binary relation $R \subseteq U \times U$ is said to be *reflexive* if for any $u \in U$, $(u, u) \in R$. It is said to be *symmetric* if for any $u_1, u_2 \in U$, $(u_1, u_2) \in R$ if and only if $(u_2, u_1) \in R$, and it is said to be *transitive* if for any $u_1, u_2, u_3 \in U$, $(u_1, u_2) \in R$ and $(u_2, u_3) \in R$ implies $(u_1, u_3) \in R$. R is said to be *anti-symmetric* if for any $u_1, u_2 \in U$, $(u_1, u_2) \in R$ and $(u_2, u_1) \in R$ implies $u_1 = u_2$. R is said to be an *equivalence relation* if it is reflexive, symmetric and transitive. Given an equivalence relation R on U , the equivalence

class of $u \in U$ is defined to be the set $\{u' \in U \mid (u, u') \in R\}$. The collection of equivalence classes forms a *partition* of U . R is said to be a partial order if it is reflexive, anti-symmetric and transitive. The reflexive (respectively, symmetric, transitive) closure of a relation R over U is the smallest reflexive (respectively, symmetric, transitive) relation over U that contains R .

In this work, a singleton is usually identified with the unique element it contains.

2.1.1 Languages and Automata

For Σ a finite alphabet, we denote by Σ^k the k -th *concatenation power* of Σ , for each $k \in \mathbb{N}$. Σ^k is the collection of strings of length k over Σ . The union $\bigcup_{i=0}^{\infty} \Sigma^i$, which is denoted by Σ^* , is the *Kleene closure* of Σ , i.e. Σ^* is the set of finite strings over Σ , including the empty string ϵ . Any subset L of Σ^* is called a language over Σ . The concatenation $L_1 L_2$ of $L_1, L_2 \subseteq \Sigma^*$ is defined to be $\{l_1 l_2 \in \Sigma^* \mid l_1 \in L_1, l_2 \in L_2\}$, where $l_1 l_2$ is the *string concatenation* of l_1 and l_2 . The i th concatenation power L^i of L is defined inductively by $L^0 := \{\epsilon\}$, and $L^{i+1} := L^i \cdot L$ for $i \geq 0$. $L^* := \bigcup_{i=0}^{\infty} L^i$ is the Kleene closure of L . For $u, v \in \Sigma^*$, u is said to be a *prefix* of v if there exists a string $w \in \Sigma^*$ such that $v = uw$. u is said to be a *proper prefix* of v if u is a prefix of v and $u \neq v$. $\bar{L} := \{u \in \Sigma^* \mid \exists v \in L, \exists w \in \Sigma^*, v = uw\}$ denotes the *prefix closure* of L . L is said to be *prefix-closed* if $L = \bar{L}$. We use $|w|$ to denote the length of the string w . The complement of language L is denoted by $L^c := \Sigma^* \setminus L$. A *projection* P is defined between Σ^* and Σ'^* , where Σ' is a sub-alphabet of Σ , such that $P(\epsilon) = \epsilon$ and, recursively, for $u\sigma$ in Σ^* , $P(u\sigma) = P(u)$ if $\sigma \notin \Sigma'$ and $P(u\sigma) = P(u)\sigma$ if $\sigma \in \Sigma'$. We can easily extend the projection to map from languages over Σ to languages over Σ' . The projection P erases those symbols not in Σ' from each string of L to obtain $P(L)$. For a language L' over Σ' , we define P^{-1} as the inverse of the projection P such that $P^{-1}(L')$ is the set of strings over Σ whose P projections are in L' . For any given finite alphabet Σ , a *distribution* of Σ of size n is an n -tuple $\Delta = (\Sigma_i)_{i=1}^n$

of sub-alphabets of Σ such that $\Sigma = \bigcup_{i=1}^n \Sigma_i$ and the sub-alphabets are pairwise incomparable with respect to set inclusion. For example, let $\Sigma = \{a, b, c\}$. Then $\Delta = (\{a, b\}, \{c\})$ is a distribution of Σ of size 2, where $\Sigma_1 = \{a, b\}$ and $\Sigma_2 = \{c\}$. Given a distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ , we have n projections P_i 's from Σ^* to Σ_i^* , $i \in [1, n]$, and n corresponding inverse projections P_i^{-1} 's. The synchronous product $\|_{i=1}^n L_i$ of languages L_i 's over Σ_i 's is defined to be $\bigcap_{i=1}^n P_i^{-1}(L_i)$. Given a language L over Σ , $\|_{i=1}^n P_i(L)$ is said to be the *decomposition closure* of L with respect to Δ . We will be mainly interested in two classes of formal languages: the class of regular languages and the class of star free languages.

Definition 2.1 (Regular Languages). *The class of regular languages over Σ is defined as the smallest class of languages containing $\emptyset, \{\sigma\}$ for each $\sigma \in \Sigma$, and being closed under finite union, concatenation and Kleene closure.*

Definition 2.2 (Star Free Languages). *The class of star free languages over Σ is defined as the smallest class of languages containing $\emptyset, \{\sigma\}$ for each $\sigma \in \Sigma$, and being closed under finite union, intersection, complementation and concatenation.*

Each star free language is regular, since the class of regular languages is known to be closed under intersection and complementation [32]. However, not every regular language is star free. It is known that, when interpreted over finite words, the class of star free languages corresponds exactly to the class of linear temporal logic definable languages or, equivalently, the class of first order logic definable languages [73]. Furthermore, the problem whether an arbitrary given regular language is star free is decidable [73]. An example of a star free language is $(ab)^*$ over $\{a, b\}$. The star freeness of $(ab)^*$ follows from the fact $(ab)^* = (b\emptyset^c \cup \emptyset^c a \cup \emptyset^c aa\emptyset^c \cup \emptyset^c bb\emptyset^c)^c$. Alternatively, a linear temporal logic formula or a first order logic formula can be used to describe the following properties that together define $(ab)^*$: 1) the element in the first position is not b ; 2) the element in the last position is not a ; 3) there

cannot be two successive a 's; 4) there cannot be two successive b 's. On the other hand, $\{aa\}^*$ is regular but not star free [76].

An automaton G is a 5-tuple $(S, \Sigma, \delta, S_i, S_f)$, where S is the set of states, Σ is the alphabet, $\delta \subseteq S \times \Sigma \times S$ is the transition relation, $S_i \subseteq S$ is the set of initial states and $S_f \subseteq S$ is the set of final (marked) states. In the graphical representation of an automaton, each initial state is indicated by an inward pointing arrow without a source and each final state is indicated by an outward pointing arrow without a sink. If both S and Σ are finite, then we say G is a finite state automaton. If, in addition, $|S_i| = 1$ and $\forall s, s', s'' \in S, \sigma \in \Sigma, (s, \sigma, s') \in \delta \wedge (s, \sigma, s'') \in \delta \implies s' = s''$, then G is said to be a *deterministic finite automaton*. A finite state automaton is said to be *non-marking* if $S_f = S$. A *run* of G over $w = \sigma_1\sigma_2 \dots \sigma_n \in \Sigma^*$ is a sequence $s_1s_2 \dots s_{n+1}$ of states of G such that $s_1 \in S_i$ and for each $k \in [1, n]$, $(s_k, \sigma_k, s_{k+1}) \in \delta$. The run is said to be *accepting* if in addition $s_{n+1} \in S_f$. The set of strings over Σ , for which G has a run, is denoted by $L(G)$ and is said to be the closed behavior of G . Alternatively, we say G *generates* $L(G)$. The set of strings over Σ , for which G has an accepting run, is denoted by $L_m(G)$ and is said to be the marked behavior of G . Alternatively, we say G *recognizes* $L_m(G)$. Each regular language is recognized by a finite state automaton and each finite state automaton recognizes a regular language [32]. If G is a deterministic finite automaton, then it is said to be non-blocking if $\overline{L_m(G)} = L(G)$.

The synchronous product of two finite state automata $G_1 = (S_1, \Sigma_1, \delta_1, S_{1,i}, S_{1,f})$ and $G_2 = (S_2, \Sigma_2, \delta_2, S_{2,i}, S_{2,f})$ is the finite state automaton $G_1 \parallel G_2 = (S_1 \times S_2, \Sigma_1 \cup \Sigma_2, \delta_1 \parallel \delta_2, S_{1,i} \times S_{2,i}, S_{1,f} \times S_{2,f})$, where $\delta_1 \parallel \delta_2$ is defined as follows:

1. If $\sigma \in \Sigma_1 \cap \Sigma_2$, then $((s_1, s_2), \sigma, (s'_1, s'_2)) \in \delta_1 \parallel \delta_2$ if and only if $(s_1, \sigma, s'_1) \in \delta_1$ and $(s_2, \sigma, s'_2) \in \delta_2$.
2. If $\sigma \in \Sigma_2 \setminus \Sigma_1$, then $((s_1, s_2), \sigma, (s'_1, s'_2)) \in \delta_1 \parallel \delta_2$ if and only if $s_1 = s'_1$ and $(s_2, \sigma, s'_2) \in \delta_2$.

3. If $\sigma \in \Sigma_1 \setminus \Sigma_2$, then $((s_1, s_2), \sigma, (s'_1, s'_2)) \in \delta_1 \parallel \delta_2$ if and only if $(s_1, \sigma, s'_1) \in \delta_1$ and $s_2 = s'_2$.

We have $L_m(G_1 \parallel G_2) = L_m(G_1) \parallel L_m(G_2)$ and $L(G_1 \parallel G_2) = L(G_1) \parallel L(G_2)$.

2.1.2 Trace Theory

An independence relation $I \subseteq \Sigma \times \Sigma$ is an *irreflexive* and *symmetric* relation. It induces a binary relation $\sim_I$ on Σ^* in the following manner: $u \sim_I v$ if and only if there exist $x, y \in \Sigma^*, (a, b) \in I$ such that $u = xaby$ and $v = xbay$. Two strings w, w' over Σ are said to be *trace equivalent*¹ with respect to I , denoted by $w \equiv_I w'$, if (w, w') belongs to the symmetric, reflexive and transitive closure of $\sim_I$, i.e., the smallest equivalence relation that contains $\sim_I$. For example, let $I = \{(a, b), (b, a)\}$. The strings $caabb$ and $cbaba$ are trace equivalent, since $caabb \sim_I cabab \sim_I cbaab \sim_I cbaba$. The set of trace equivalent strings of s for an independence relation I is called the trace closure of s , denoted by $[s]_I$ or $[s]$ if I is clear from the context. The trace closure $[L]$ of a language L is defined to be the set $\bigcup_{s \in L} [s]$. A language L is said to be *trace closed* if $L = [L]$.

A distribution $\Delta = (\Sigma_i)_{i=1}^n$ naturally induces an independence relation $I(\Delta)$ in the following way. The reflexive, symmetric relation $D(\Delta) = \{(a, b) \in \Sigma \times \Sigma \mid \exists i \in [1, n], a, b \in \Sigma_i\}$ is called the *dependence relation* induced by Δ . Then $I(\Delta) = (\Sigma \times \Sigma) \setminus D(\Delta)$ is the independence relation induced by Δ . In the rest, whenever we are given a distribution Δ , we also construct its independence relation $I(\Delta)$. An independence relation $I \subseteq \Sigma \times \Sigma$ is said to be “transitive” if $\forall a, b, c \in \Sigma, a \neq c, (a, b) \in I \wedge (b, c) \in I \implies (a, c) \in I$. For example, the independence relation induced by $\Delta = (\{a, c\}, \{b, c\})$ is transitive, since $I(\Delta) = \{\{a, b\}\}$ ². I is said to be a

¹Note that in discrete event systems community a string is often called a trace, which is different from the notion of a trace in trace theory.

²That is, $I(\Delta) = \{(a, b), (b, a)\}$. We often use the set $\{a, b\}$ to represent the pairs (a, b) , (b, a) together.

transitive forest if the undirected graph (Σ, I) , where $I \subseteq \Sigma \times \Sigma$ is viewed as the set of undirected edges, contains neither the graph³ $P_4 = \{\{a, b\}, \{b, c\}, \{c, d\}\}$ nor the graph $C_4 = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, a\}\}$ as induced subgraphs [77]. Here a, b, c, d are all different vertices and the set $\{a, b\}$ represents the edge between vertices a and b . The graphs P_4 and C_4 are shown in Fig. 2.1 and Fig. 2.2 respectively. It is clear that each transitive independence relation is a transitive forest. The independence relation induced by $\Delta = (\{a\}, \{b, c, d\})$ is not transitive. But it is a transitive forest, since $I(\Delta) = \{\{a, b\}, \{a, c\}, \{a, d\}\}$ contains neither the graph P_4 nor the graph C_4 as induced subgraphs.

Figure 2.1: Graph P_4 Figure 2.2: Graph C_4

³When the vertex set is obvious, we will often use I to represent the underlying graph (Σ, I) .

2.1.3 Supervisory Control

This work will only deal with the case where each supervisor can observe each event it can disable. We only need to occasionally use the definitions of controllability and $L_m(G)$ -closedness.

Definition 2.3 (Controllability). *A language $K \subseteq \Sigma^*$ is said to be controllable with respect to $(L(G), \Sigma_u)$ if $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$.*

Definition 2.4 ($L_m(G)$ -closedness). *A language $K \subseteq \Sigma^*$ is said to be $L_m(G)$ -closed if $K = \overline{K} \cap L_m(G)$.*

2.2 Basic Results

We need the following folklore lemmas [13], [40], [78], [79], [80], [81].

Lemma 2.1 states that for any language L , its decomposition closure is an over-approximation.

Lemma 2.1. *For any language L over Σ and any distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ , we have $L \subseteq \|\|_{i=1}^n P_i(L)$.*

Proof: $L \subseteq P_i^{-1}(P_i(L))$ for each $i \in [1, n]$. Thus, $L \subseteq \bigcap_{i=1}^n P_i^{-1}(P_i(L)) = \|\|_{i=1}^n P_i(L)$. ■

The following lemma is a special case of the result of [78], [79], [80]. The proof appears in Proposition 1.1 of [80].

Lemma 2.2. *Let $\Delta = (\Sigma_i)_{i=1}^n$ be a distribution of Σ . $s \equiv_{I(\Delta)} s'$ if and only if $\forall i \in [1, n], P_i(s) = P_i(s')$.*

The following lemma states that the trace closure and the decomposition closure of a string are equal.

Lemma 2.3. $[s]_{I(\Delta)} = \parallel_{i=1}^n P_i(s)$ for any string s over Σ and any distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ .

Proof: Let $s' \in [s]_{I(\Delta)}$. We have $P_i(s') = P_i(s)$ for each $i \in [1, n]$, by Lemma 2.2, and $s' \in \parallel_{i=1}^n P_i(s') = \parallel_{i=1}^n P_i(s)$ by Lemma 2.1. Thus, $[s]_{I(\Delta)} \subseteq \parallel_{i=1}^n P_i(s)$. On the other hand, let $s' \in \parallel_{i=1}^n P_i(s)$. We have $P_i(s') \in P_i(\parallel_{i=1}^n P_i(s)) \subseteq \{P_i(s)\}$ for each $i \in [1, n]$. Thus, $P_i(s') = P_i(s)$ for each $i \in [1, n]$ and then $s' \equiv_{I(\Delta)} s$, i.e., $s' \in [s]_{I(\Delta)}$, by Lemma 2.2. That is, $\parallel_{i=1}^n P_i(s) \subseteq [s]_{I(\Delta)}$. ■

Lemma 2.4 states that the trace closure of L is a subset of the decomposition closure of L , i.e., the decomposition closure of L is an over-approximation of the trace closure of L .

Lemma 2.4. $[L]_{I(\Delta)} \subseteq \parallel_{i=1}^n P_i(L)$ for any language L over Σ and any distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ .

Proof: $[L]_{I(\Delta)} = \bigcup_{s \in L} [s]_{I(\Delta)} = \bigcup_{s \in L} \parallel_{i=1}^n P_i(s) \subseteq \parallel_{i=1}^n P_i(L)$. ■

Lemma 2.5. Let $\Delta = (\Sigma_i)_{i=1}^n$ be any distribution of Σ . Let L_i be any language over Σ_i , for each $i \in [1, n]$. Then $\parallel_{i=1}^n L_i$ is trace closed with respect to $I(\Delta)$.

Proof: Let $s \in \parallel_{i=1}^n L_i$. Clearly, $\parallel_{i=1}^n P_i(s) \subseteq \parallel_{i=1}^n L_i$ and thus $[s] \subseteq \parallel_{i=1}^n L_i$. That is, $\parallel_{i=1}^n L_i$ is trace closed with respect to $I(\Delta)$. ■

The proof of the following lemma appears in [40].

Lemma 2.6. Let $\Delta = (\Sigma_i)_{i=1}^n$ be any distribution of Σ . Let L_i be any language over Σ_i , for each $i \in [1, n]$. If $\bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) \subseteq \Sigma_0 \subseteq \Sigma$, then $P_0(\parallel_{i=1}^n L_i) = \parallel_{i=1}^n P_0(L_i)$. Here $P_0 : \Sigma^* \mapsto \Sigma_0^*$ is the natural projection.

Chapter 3

Decentralized Realization

The central theme of this chapter is the well-known decentralized realization problem¹ (**DRP**) [30], [31], which is informally stated as follows: “Given an arbitrary finite state automaton G representing the global plant, an arbitrary regular language K representing the global specification and an arbitrary decentralized control architecture $(\mathcal{A})_{i=1}^k$, synthesize a tuple of local finite state supervisors $(S_i)_{i=1}^k$ over the given architecture, if there exists one, such that the closed-loop system, i.e., the synchronous product of the global plant and the local supervisors, is non-blocking and language equivalent to the global specification, i.e., $L_m(\|_{i=1}^k S_i \| G) = K$ and $L(\|_{i=1}^k S_i \| G) = \overline{K}$ ”. We revisit this synthesis problem² from the point of view of language equations—the synthesis problem is reduced to the problem of solving a set of language equations. Then a language-theoretic characterization of the solvability of **DRP** is immediately obtained by characterizing the solvability of the set of language equations. The key idea of our approach involves the use of a language equational characterization of finite state supervisors.

We then proceed to study the supervisor decomposition problem (**SDP**): “Given

¹The term “realization” used in this thesis has to be distinguished from the same term that has been used in system theory [82].

²This synthesis problem is essentially reduced to a verification problem, as will become clear later.

an arbitrary global finite state supervisor S and an arbitrary decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$, synthesize a tuple of local finite state supervisors $(S_i)_{i=1}^k$ over the given architecture, if there exists one, such that the synchronous product of the local supervisors is language equivalent to the global supervisor, i.e., $L(\|_{i=1}^k S_i) = L(S)$. It turns out that **SDP** is just a special case of **DRP**. Thus, a language-theoretic characterization of supervisor decomposability is immediately obtained.

We then continue to investigate a special case of **SDP** (and thus also a special case of **DRP**): “Given an arbitrary distribution Δ and an arbitrary regular language K representing the global specification, synthesize a tuple of regular languages $(K_i)_{i=1}^k$ representing a tuple of local specifications over the given distribution, if there exists one, such that the synchronous product of the local specifications is language equivalent to the global specification, i.e., $K = \|\|_{i=1}^k K_i$ ”. A global specification that could be decomposed in such a manner is said to be *language decomposable* with respect to the given distribution. This problem is henceforth called the language decomposition problem (**LDP**). The notion of language decomposability is weaker than the notion of supervisor decomposability: none of the local specifications is required to be a supervisor. The PSPACE-hardness of **SDP** is established by a simple reduction from **LDP**, which is known to be PSPACE-complete [6]. Then an alternative proof of the PSPACE-hardness of **DRP** is provided via a simple reduction from **SDP**. We then propose in this work several techniques for verifying language decomposability that exploit special properties of the given distributions or the given global specifications.

The chapter is organized as follows. Section 3.1 is devoted to system setup. The characterization of solvability is obtained in Section 3.2. In the same section, the relationship among **DRP**, **SDP** and **LDP** is revealed and then complexity-theoretic lower bound results are proved. Techniques for reducing the complexity of verifying language decomposability are shown in Section 3.3. Section 3.4 is devoted to the

discussions.

3.1 System Setup

Let Σ be a finite set of events. A *control constraint* $\mathcal{A}$ over Σ is a tuple $(\Sigma_{S,o}, \Sigma_{S,c})$ of subsets of Σ . A finite state supervisor³ S over $\mathcal{A}$ is a finite state non-marking automaton over Σ with the constraint that at each state of S : 1) there is an outgoing transition labeled by σ , for each $\sigma \in \Sigma \setminus \Sigma_{S,c}$; 2) if there is an outgoing transition labeled by $\sigma \in \Sigma \setminus \Sigma_{S,o}$, then this transition is a self-loop. The observation and control mechanism of a supervisor on a plant is implemented by the synchronization of shared events. The closed-loop system is the synchronous product of the supervisor and the plant. A supervisor S is considered to enable σ at state s (of S) if there is an outgoing transition labeled by σ at that state. S “effectively” cannot control σ if and only if σ is enabled at each state of S . A supervisor S is considered not to observe σ at state s if the existence of an outgoing transition labeled by σ at that state implies that outgoing transition is a self-loop. S “effectively” cannot observe σ if and only if it does not observe σ at each state. Thus, $\Sigma_{S,uc} := \Sigma \setminus \Sigma_{S,c}$ is the subset of events supervisor S cannot control (or, disable) and $\Sigma_{S,uo} := \Sigma \setminus \Sigma_{S,o}$ is the subset of events supervisor S cannot observe. Dually, $\Sigma_{S,c}$ is the subset of events supervisor S can control (or, disable) and $\Sigma_{S,o}$ is the subset of events supervisor S can observe. We remark that constraint 1) imposed here is not an essential restriction. Any candidate supervisor defined in [13], for example, can be modified to satisfy constraint 1) by adding, for each state s and each $\sigma \in \Sigma_{uc}$, a self-loop labeled by σ at state s if there is no outgoing transition labeled by σ at state s , without changing the solvability of the synthesis instance. Intuitively, it is impossible that at the same time, the modified supervisor reaches such a (supervisor) state s and the plant reaches a (plant) state where there is an outgoing transition labeled by σ

³We have followed the definition used in [9].

(so adding these self-loops does not matter), since otherwise the original supervisor has reached a (supervisor) state where it disables an uncontrollable event, which contradicts the definition of a supervisor in [13]. As for constraint 2), if there is no outgoing transition labeled by σ at state s , then the supervisor is considered not to observe σ at state s , since σ will not be fired whenever the supervisor is at state s (so it does not observe it at state s); if at state s there is a self-loop labeled by σ , then the supervisor ‘effectively’ does not observe σ at state s , since the state and thus all future control decisions, which depend on the states, of the supervisor are unchanged after firing σ . The advantage of this definition of supervisors is that the set of states that have outgoing transitions labeled by $\sigma \in \Sigma_{uc}$ does not depend on the plant and thus a language theoretic characterization of supervisors becomes rather simple. In a centralized synthesis setting, the control constraint $\mathcal{A}$ is all the information we need to constrain the supervisor. The notion of a decentralized control architecture becomes necessary when decentralized synthesis is considered. A decentralized control architecture⁴ over Σ is a tuple $(\mathcal{A}_i)_{i=1}^k$, where each component $\mathcal{A}_i = (\Sigma_{S_i,o}, \Sigma_{S_i,c})$ is a control constraint over Σ . A tuple $(S_i)_{i=1}^k$ of supervisors is said to be over $(\mathcal{A}_i)_{i=1}^k$ if each supervisor S_i is over $\mathcal{A}_i$. We shall impose an assumption on the decentralized control architectures, the relaxation of which is left as future work (see the remark after Problem 1). Assumption 1 has been used in [30] and [36]. It can be easily realized by letting each supervisor not disable any event that it can control but cannot observe. This necessarily reduces the chance of solving the synthesis task. But as we shall show in Section 3.2.3, **DRP** is already intractable even with this assumption.

Assumption 1. $\mathcal{A}_i$ satisfies the restriction $\Sigma_{S_i,c} \subseteq \Sigma_{S_i,o}$, for each $i \in [1, k]$.

⁴The decentralized control architecture corresponds to the conjunctive architecture of [55]. In [55], the notion of an architecture refers to the way the control information from local supervisors is fused, e.g., conjunctively or disjunctively. In this work, the notion of an architecture refers to the observation and control capability of each local supervisor.

Assumption 1 states that each supervisor S_i is able to observe each event that it can disable. Clearly, by Assumption 1, if an event cannot be observed by S_i , then it cannot be disabled by S_i either. That is, each supervisor S_i satisfies the restriction that $\Sigma_{S_i,uo} \subseteq \Sigma_{S_i,uc}$. An implication is that there is a self-loop labeled by σ at each state of S_i , for each event $\sigma \in \Sigma_{S_i,uo}$. Under Assumption 1, a detailed formulation of **DRP** is now provided below.

Problem 1 (Decentralized Realization Problem). *For an arbitrary finite state automaton G over Σ , an arbitrary regular language K over Σ and an arbitrary decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$ over Σ that satisfies Assumption 1, synthesize a tuple $(S_i)_{i=1}^k$ of finite state supervisors over $(\mathcal{A}_i)_{i=1}^k$, if there exists one, such that $G\|S$ is non-blocking and $L_m(G\|S) = K$, where $S = \parallel_{i=1}^k S_k$.*

For convenience, $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$ is used to denote the instance of the synthesis problem with global plant G , global specification K and decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$.

Remark: Instead of a global specification over Σ , usually several local specifications respectively over several sub-alphabets of Σ are given as inputs of the problem. The global specification in this case is simply the synchronous product of the local specifications. The same remark is also applicable to the global plant, which could be the synchronous product of several local plants. Although both our problem formulation and the characterization results rely on Assumption 1, a general language-theoretic approach could also be developed without any assumption imposed on the decentralized control architecture, the details of which is left to the future work. $\square$

We impose the following assumption on the plant G and supervisors S_i 's.

Assumption 2. *G and S_i 's are deterministic.*

The assumption that the supervisors are deterministic is not an essential restriction, due to the following simple result⁵.

Lemma 3.1. *There exists a tuple of finite state supervisors that solves the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$ if and only if there exists a tuple of deterministic finite state supervisors that solves the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$.*

Proof: This follows from the fact that if $(S_i)_{i=1}^k$ solves the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$, then $(\mathbf{det}(S_i))_{i=1}^k$ also solves the same instance. Here $\mathbf{det}$ is the usual determinization operator [32]. ■

By Assumption 1 and 2, each supervisor S_i could also be treated as a deterministic finite non-marking automaton over $\Sigma_{S_i,o}$, by eliminating the self-loops labeled by events in $\Sigma_{S_i,uo}$.

3.2 Characterization with Language Equations

After introducing the system setup in previous section, we now establish some results in order to obtain a characterization of the solvability of **DRP** in Theorem 3.1.

3.2.1 Characterization Result

In order to establish Theorem 3.1, we need a language equational characterization of any finite state supervisor over $\mathcal{A}_i$, which is shown in Lemma 3.2. Recall that each supervisor over $(\Sigma_{S,o}, \Sigma_{S,c})$ could be treated as a deterministic finite non-marking automaton over $\Sigma_{S,o}$.

Lemma 3.2 (Characterization of Supervisors). *For each finite state supervisor S_i over $(\Sigma_{S_i,o}, \Sigma_{S_i,c})$, $L(S_i)$ is a prefix-closed regular language over $\Sigma_{S_i,o}$ that satisfies*

⁵In this work, we do not consider probabilistic or stochastic automata [83], [84], [85].

$L(S_i) = L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. Conversely, for each prefix-closed regular language L_i over $\Sigma_{S_i,o}$ that satisfies $L_i = L_i(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, there exists a finite state supervisor S_i over $(\Sigma_{S_i,o}, \Sigma_{S_i,c})$ such that $L(S_i) = L_i$.

Proof: Let S_i be a finite state supervisor over $(\Sigma_{S_i,o}, \Sigma_{S_i,c})$. Clearly, $L(S_i)$ is a prefix-closed regular language over $\Sigma_{S_i,o}$. By definition of a supervisor, there is a transition labeled by σ at each state of S_i , for each event $\sigma \in \Sigma_{S_i,o} \setminus \Sigma_{S_i,c}$. Thus, clearly $L(S_i) = L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$.

Conversely, let L_i be a prefix-closed regular language over $\Sigma_{S_i,o}$ that satisfies $L_i = L_i(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. There is a finite state non-marking automaton S_i over $\Sigma_{S_i,o}$ that generates L_i , i.e., $L(S_i) = L_i$. We only need to show that S_i is indeed a supervisor over $(\Sigma_{S_i,o}, \Sigma_{S_i,c})$, i.e., to show that there is an outgoing transition labeled by σ at each state of S_i , for each event $\sigma \in \Sigma_{S_i,o} \setminus \Sigma_{S_i,c}$. Suppose to the contrary, there is a state q in S_i and an event $\sigma \in \Sigma_{S_i,o} \setminus \Sigma_{S_i,c}$ such that $\delta_{S_i}(q, \sigma)$ is undefined, where δ_{S_i} is the extended (partial) transition function of S_i . Let q_0 be the initial state of S_i and let $s \in L(S_i)$ be any string such that $\delta_{S_i}(q_0, s) = q$. Clearly $s\sigma \notin L(S_i)$ but $s\sigma \in L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, which violates $L(S_i) = L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. ■

Let $P_{S_i,o} : (\bigcup_{i=1}^k \Sigma_{S_i,o})^* \mapsto \Sigma_{S_i,o}^*$ denote the natural projection. The following lemma shows that the synchronous product of supervisors S_i 's over $\mathcal{A}_i$'s is a supervisor over $\bigcup (A_i)_{i=1}^k := (\bigcup_{i=1}^k \Sigma_{S_i,o}, \bigcup_{i=1}^k \Sigma_{S_i,c})$.

Lemma 3.3. *Let $(S_i)_{i=1}^k$ be a tuple of finite state supervisors over $(\mathcal{A}_i)_{i=1}^k$. The automaton $\|_{i=1}^k S_i$ is a finite state supervisor over $(\bigcup_{i=1}^k \Sigma_{S_i,o}, \bigcup_{i=1}^k \Sigma_{S_i,c})$. Let $H = L(\|_{i=1}^k S_i)$. We have $H = H((\bigcup_{i=1}^k \Sigma_{S_i,o}) \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^* = \|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$.*

Proof: Clearly $\|_{i=1}^k S_i$ is a finite state non-marking automaton over $\bigcup_{i=1}^k \Sigma_{S_i,o}$. It is not difficult to see that there is an outgoing transition labeled by σ at each state of $\|_{i=1}^k S_i$, for each event $\sigma \in (\bigcup_{i=1}^k \Sigma_{S_i,o}) \cap \bigcap_{i=1}^k \Sigma_{S_i,uc} = (\bigcup_{i=1}^k \Sigma_{S_i,o}) \setminus \bigcup_{i=1}^k \Sigma_{S_i,c}$.

Thus, $\|_{i=1}^k S_i$ is a finite state supervisor over $(\bigcup_{i=1}^k \Sigma_{S_i,o}, \bigcup_{i=1}^k \Sigma_{S_i,c})$ and $L(\|_{i=1}^k S_i) = L(\|_{i=1}^k S_i)((\bigcup_{i=1}^k \Sigma_{S_i,o} \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^*)$. Let $H = L(\|_{i=1}^k S_i)$. Immediately we have $H = H((\bigcup_{i=1}^k \Sigma_{S_i,o} \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^*)$. We now show that $H = \|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. Indeed, since $H = L(\|_{i=1}^k S_i) = \|_{i=1}^k L(S_i)$ and $\|_{i=1}^k L(S_i) \subseteq P_{S_i,o}^{-1}(L(S_i))$ for each $i \in [1, k]$, we have $P_{S_i,o}(H) = P_{S_i,o}(\|_{i=1}^k L(S_i)) \subseteq P_{S_i,o}(P_{S_i,o}^{-1}(L(S_i))) = L(S_i)$. Thus, $P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \subseteq L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$ and also $\|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \subseteq \|_{i=1}^k L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* = H$. It holds that $H \subseteq \|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. Thus, $H = \|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. $\blacksquare$

Now we are finally ready to state the main characterization theorem, Theorem 3.1, by reducing **DRP** to the problem of solving a set of language equations.

Theorem 3.1. *The following four statements are equivalent.*

1. *The instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$ of **DRP** has a solution.*
2. *There exists a tuple $(S_i)_{i=1}^k$ of finite state non-marking automata such that*

$$(a) \quad \|_{i=1}^k L(S_i) \| L_m(G) = K$$

$$(b) \quad \|_{i=1}^k L(S_i) \| L(G) = \overline{K}$$

$$(c) \quad \forall i \in [1, k], L(S_i) = L(S_i)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \subseteq \Sigma_{S_i,o}^*$$

3. *There exists a prefix-closed regular language H over $\bigcup_{i=1}^k \Sigma_{S_i,o}$ such that*

$$(a) \quad H \| L_m(G) = K$$

$$(b) \quad H \| L(G) = \overline{K}$$

$$(c) \quad H = H((\bigcup_{i=1}^k \Sigma_{S_i,o} \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^*) = \|_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$$

4. *K satisfies the following two conditions:*

$$(a) \quad K = \|_{i=1}^k P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \| L_m(G)$$

$$(b) \ \overline{K} = \parallel_{i=1}^k P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \parallel L(G)$$

Proof: 1) and 2) are obviously equivalent. We first show that 2) and 3) are equivalent. Suppose 2) holds. Let $H := \parallel_{i=1}^k L(S_i)$. Clearly H is a prefix-closed regular language over $\bigcup_{i=1}^k \Sigma_{S_i,o}$ that satisfies $H \parallel L(G) = \overline{K}$ and $H \parallel L_m(G) = K$. From Lemma 3.3, we have $H = H((\bigcup_{i=1}^k \Sigma_{S_i,o}) \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^* = \parallel_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. Thus, 3) holds. On the other hand, suppose 3) holds, it is then apparent that the tuple $(S_i)_{i=1}^k$ of finite state non-marking automata, with $L(S_i) := P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, satisfies 2).

It is straightforward to see that 4) implies 2), since the tuple $(S_i)_{i=1}^k$, with $L(S_i) := P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, satisfies 2). Thus, 4) implies 3). We now show that 3) also implies 4). Suppose 3) holds. From 3. a) and 3. b), it is clear that $K \subseteq \parallel_{i=1}^k P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \parallel L_m(G)$ and $\overline{K} \subseteq \parallel_{i=1}^k P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \parallel L(G)$. Also we have $P_{S_i,o}(\overline{K}) \subseteq P_{S_i,o}(H)$. Thus, $P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \subseteq P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$ and $\parallel_{i=1}^k P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* \subseteq \parallel_{i=1}^k P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^* = H$. It is then straightforward to see that 4) holds. ■

Statement 2) of Theorem 3.1 provides the promised reduction from **DRP** to the problem of solving a set of language equations. The key idea of the reduction is a language-equational characterization of finite state supervisors shown in Lemma 3.2. Statement 4) is the desired language-theoretic characterization result. It is straightforward to prove Statement 4) from Statement 2), even without going through Statement 3). Nevertheless, Statement 3) provides a rather intuitive interpretation of the solvability of **DRP**, which deserves some discussions. It essentially states that the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$ is solvable if and only if there is a global finite state supervisor S , with $L(S) = H((\bigcup_{i=1}^k \Sigma_{S_i,o}) \setminus \bigcup_{i=1}^k \Sigma_{S_i,c})^*$, over $(\bigcup_{i=1}^k \Sigma_{S_i,o}, \bigcup_{i=1}^k \Sigma_{S_i,c})$ that solves the centralized synthesis instance $\langle G, K, \bigcup (A_i)_{i=1}^k \rangle$ and this supervisor S could be decomposed into the tuple $(S_i)_{i=1}^k$ of local finite state supervi-

sors, with $L(S_i) = P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, over the decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$. The condition that $L(S_i) = P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$ is not essential, as it could be easily shown that S could be decomposed into a tuple of local finite state supervisors if and only if it could be decomposed into the tuple $(S_i)_{i=1}^k$, with $L(S_i) = P_{S_i,o}(H)(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, over $(\mathcal{A}_i)_{i=1}^k$. Thus, the notion of supervisor decomposition, which we shall further study in Section 3.2.2, plays a useful role in understanding the solvability of **DRP**. However, Statement 3) is difficult to use in practice since it is an *existential* statement: we need not only to find a global finite state supervisor that solves the centralized synthesis instance, but also to ensure that it could be decomposed into a tuple of local finite state supervisors. This difficulty is completely resolved by Statement 4): if there exists a tuple of local finite state supervisors that solves the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$, then the tuple $(S_i)_{i=1}^k$, where $L(S_i) = P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$, solves the same instance.

We now look at Statement 4) from an automata perspective. The only possible difficulty that may arise is a proper interpretation of the finite state supervisor that generates $P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$. Given a finite state automaton $S = (Q, \Sigma, \delta, q_0, Q_f)$ that recognizes K , it is easy to see that $P_{S_i,o}(\overline{K})(\Sigma_{S_i,o} \setminus \Sigma_{S_i,c})^*$ is the closed behavior of the non-deterministic finite non-marking automaton with ϵ -moves [32] $S_i = (Q \cup \{q_\#\}, \Sigma_{S_i,o}, \delta_i, q_0, Q \cup \{q_\#\})$, where $q_\# \notin Q$ is a new state and $\delta_i : (Q \cup \{q_\#\}) \times (\Sigma_{S_i,o} \cup \{\epsilon\}) \mapsto \mathcal{P}(Q \cup \{q_\#\})$ is the transition function such that

1. $\delta_i(q, \sigma) = \{\delta(q, \sigma)\}$, if $q \in Q$ and $\sigma \in \Sigma_{S_i,o}$
2. $\delta_i(q, \epsilon) = \{\delta(q, \sigma) \mid \sigma \notin \Sigma_{S_i,o}\} \cup \{q_\#\}$, if $q \in Q$
3. $\delta_i(q_\#, \sigma) = \{q_\#\}$, if $\sigma \in \Sigma_{S_i,o} \setminus \Sigma_{S_i,c}$

By using synchronous product of non-deterministic finite automata with ϵ -moves, the tuple $(S_i)_{i=1}^k$ constructed above could directly be used to verify Statement 4), i.e. determine the solvability of the instance $\langle G, K, (\mathcal{A}_i)_{i=1}^k \rangle$, without the need of

determinization. In order to implement the control logic of the tuple $(S_i)_{i=1}^k$ and at the same time avoid explicit determinization of S_i 's, we use the tuple $(S_i)_{i=1}^k$ to on-line simulate the tuple $(\mathbf{det}(S_i))_{i=1}^k$, by keeping track of the set $\mathcal{R}(q_0, P_{S_i,o}(s))$ of states S_i may reach (with ϵ -moves) upon reading $P_{S_i,o}(s)$, for each string s generated by the closed-loop system. An event $\sigma \in \Sigma_{S_i,c}$ is disabled by S_i , after reading $P_{S_i,o}(s)$, if and only if there is no outgoing transition labeled by σ among the set of states in $\mathcal{R}(q_0, P_{S_i,o}(s))$. An event $\sigma \in \bigcup_{i=1}^k \Sigma_{S_i,c}$ is disabled by the tuple $(S_i)_{i=1}^k$, after the generation of each string s by the closed-loop system, if and only if at least one supervisor S_i disables σ after reading $P_{S_i,o}(s)$.

3.2.2 Supervisor Decomposition as Decentralized Realization

In this section, we will reveal the relation between **DRP** and **SDP**. To that end, we shall formally define **SDP** below.

Problem 2 (Supervisor Decomposition Problem). *For an arbitrary control constraint $\mathcal{A}$ over Σ , an arbitrary global finite state supervisor S over $\mathcal{A}$ and an arbitrary decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$ over Σ , synthesize a tuple $(S_i)_{i=1}^k$ of local finite state supervisors over $(\mathcal{A}_i)_{i=1}^k$, if there exists one, such that $L(\parallel_{i=1}^k S_i) = L(S)$.*

For convenience, $\langle S, \mathcal{A}, (\mathcal{A}_i)_{i=1}^k \rangle$ is used to denote the instance of **SDP** with global supervisor S , control constraint $\mathcal{A}$ and decentralized control architecture $(\mathcal{A}_i)_{i=1}^k$. We immediately have the following simple reduction result.

Proposition 3.1. ***SDP** is polynomial time reducible to **DRP**.*

Proof. It is easy to see that the instance $\langle S, \mathcal{A}, (\mathcal{A}_i)_{i=1}^k \rangle$ of **SDP** has a solution if and only if $\mathcal{A} = \bigcup (\mathcal{A}_i)_{i=1}^k$ and the instance $\langle G_0, L(S), (\mathcal{A}_i)_{i=1}^k \rangle$ of **DRP** has a solution, where $L_m(G_0) = L(G_0) = \Sigma^*$. ■

We have the following language-theoretic characterization of supervisor decomposability, which essentially corresponds to Statement 3. c) of Theorem 3.1.

Corollary 3.1. *The instance $\langle S, \mathcal{A}, (\mathcal{A}_i)_{i=1}^k \rangle$ has a solution if and only if $\mathcal{A} = \bigcup (\mathcal{A}_i)_{i=1}^k$ and S satisfies the condition $L(S) = \parallel_{i=1}^k P_{S_i, o}(L(S))(\Sigma_{S_i, o} \setminus \Sigma_{S_i, c})^*$.*

The theoretical advantage of considering **SDP** will become obvious when we deal with the decentralized supervisor synthesis problem, where we have to first compute a global supervisor in the worst case if any attempt of decentralized synthesis fails due to the undecidability reason. The procedure of supervisor decomposition studied in this work is quite different from and much simpler than the supervisor localization and decomposition techniques developed in [20], [42], where we do not need to take the plant's behavior into consideration and the *conflictingness* of the local supervisors need not be considered when carrying out supervisor decomposition. A practical reason for decomposing a global supervisor, however, may come from the necessity of implementing a global supervisor in a distributed manner due to the physical constraints.

3.2.3 Complexity Results

In this subsection, we will establish complexity-theoretic lower bounds for **SDP** and **DRP**, based upon the available lower bound result on **LDP**. We first introduce the notion of language decomposability.

Formally, a language L over Σ is said to be language decomposable with respect to a distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ if there exists a tuple $(L_i)_{i=1}^n$ of languages over Δ such that $L = \parallel_{i=1}^n L_i$. The problem of language decomposition is formulated below.

Problem 3 (Language Decomposition Problem). *For an arbitrary regular language L over Σ and an arbitrary distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ , synthesize a tuple $(L_i)_{i=1}^n$ of regular languages over Δ , if there exists one, such that $L = \parallel_{i=1}^n L_i$.*

The following proposition is straightforward and well-known [57].

Proposition 3.2. *A language L over Σ is language decomposable with respect to distribution $\Delta = (\Sigma_i)_{i=1}^n$ if and only if $L = \bigcup_{i=1}^n P_i(L)$.*

We remark that if L is decomposable with respect to $\Delta = (\Sigma_i)_{i=1}^n$, then in general there may exist an infinite number of decompositions $(L_i)_{i=1}^n$ and the tuple $(P_i(L))_{i=1}^n$ is the smallest decomposition with respect to $\Delta = (\Sigma_i)_{i=1}^n$ [57], [58]. The notion of supervisor decomposability could be seen as a generalization of the notion of language decomposability. In fact, by setting $\Sigma_{S_i,o} = \Sigma_{S_i,c} = \Sigma_i$ for each control constraint $\mathcal{A}_i$, supervisor decomposability reduces to language decomposability. We denote by $\langle L(M), (\Sigma_i)_{i=1}^n \rangle$ the instance of **LDP** with specification $L(M)$, where M is a deterministic finite automaton, and distribution $\Delta = (\Sigma_i)_{i=1}^n$.

Proposition 3.3. *Any instance $\langle L(M), (\Sigma_i)_{i=1}^n \rangle$ of **LDP** can be reduced to an instance $\langle M, \mathcal{A} = (\Sigma, \Sigma), (\mathcal{A}_i = (\Sigma_i, \Sigma_i))_{i=1}^n \rangle$ of **SDP**.*

Since **LDP** is PSPACE-complete even when L is required to be prefix-closed [6], we immediately obtain the result that **SDP** is PSPACE-hard. From Corollary 3.1, we could easily see that **SDP** is in PSPACE (In fact, **DRP** is already in PSPACE due to Theorem 3.1). Thus, we have the following.

Corollary 3.2. ***SDP** is PSPACE-complete.*

By Proposition 3.1 and Theorem 3.1, it is not difficult to see the following.

Corollary 3.3. ***DRP** is PSPACE-complete. The problem is PSPACE-hard even when the specification K is prefix-closed and the plant G satisfies $L_m(G) = L(G) = \Sigma^*$, and $\Sigma_{S_i,o} = \Sigma_{S_i,c} = \Sigma_i$ for each control constraint $\mathcal{A}_i$.*

Unless $P=PSPACE$, which is highly unlikely and still remains an intriguing open problem, there is no algorithm that can solve every instance of the problem with

running time above bounded by a polynomial of the input sizes. It is thus of interest to study tractable fragments of the problem and develop heuristics that work efficiently on most input instances. In view of the fundamental importance of **LDP**, the next section is devoted to a detailed study of language decomposability and the complexity reduction techniques for its verification.

3.3 Verification of Language Decomposability

Suppose L is regular and the given deterministic finite automaton M that recognizes it has m states. An algorithm of time complexity⁶ $\mathcal{O}((n+1)|\Sigma| - \sum_{i=1}^n |\Sigma_i|)m^{n+1}$ for testing the language decomposability of L was shown in [57]. When the distribution $\Delta = (\Sigma_i)_{i=1}^n$ possesses the special property $\exists \Sigma_0 \subseteq \Sigma, \forall i, j \in [1, n], (i \neq j \implies \Sigma_i \cap \Sigma_j = \Sigma_0)$, the complexity for language decomposability verification could be reduced to $\mathcal{O}(n|\Sigma|m^3)$ [58]. Indeed, for distributions that satisfy the above property, L is language decomposable with respect to Δ if and only if L is language decomposable with respect to $\Delta_i = (\Sigma_i, \bigcup_{j \neq i} \Sigma_j)$ for each $i \in [1, n]$ [58]. By setting $\Sigma_0 = \emptyset$, this immediately implies that if $\Delta = (\Sigma_i)_{i=1}^n$ forms a *partition* of Σ , then the time complexity of checking language decomposability is $\mathcal{O}(n|\Sigma|m^3)$, which generalizes Algorithm 4.2 of [57].

In the following subsections, further generalizations and complexity improvements will be provided for language decomposability verification based on special properties of the distributions. An alternative technique is also provided to verify the language decomposability of L when L is prefix-closed.

⁶Step 2) of Algorithm 4.1 in [57] has to be eliminated. Otherwise, the complexity of the given algorithm is $\mathcal{O}((\sum_{\sigma \in \Sigma} m^{|I_\sigma|})m^{n+1})$, where $I_\sigma = \{i \in [1, n] \mid \sigma \in \Sigma_i\}$ for each $\sigma \in \Sigma$.

3.3.1 Reducible Distributions

Our goal in this subsection is to show a technique that could be used to identify a large class of distributions for which better complexity than $\mathcal{O}(((n+1)|\Sigma| - \sum_{i=1}^n |\Sigma_i|)m^{n+1})$ could be achieved. To that end, we study an order-theoretic structure of the set of distributions with respect to which a given language is language decomposable. We first define a relation $\leq_\Sigma$ on the set of distributions of Σ as follows.

Definition 3.1. *Given two distributions $\Delta = (\Sigma_i)_{i=1}^n$ and $\Delta' = (\Sigma'_i)_{i=1}^k$ of Σ , Δ is said to refine Δ' , denoted by $\Delta \leq_\Sigma \Delta'$, if and only if there exists a function $f : [1, n] \mapsto [1, k]$ such that $\forall i \in [1, n], \Sigma_i \subseteq \Sigma'_{f(i)}$.*

Let $\Delta(\Sigma)$ denote the collection of all the distributions of Σ . It is not difficult to verify that the relation of refinement $\leq_\Sigma$ is a partial order. Indeed, $(\Delta(\Sigma), \leq_\Sigma)$ is a complete upper semi-lattice [13] under the joint operation **join**, defined such that **join** (Δ_1, Δ_2) is the distribution whose components are exactly the maximal elements (with respect to set inclusion) of $\Delta_1 \cup \Delta_2$, viewing both Δ_1 and Δ_2 as sets of sub-alphabets of Σ . In particular, the unique supremal element of $\Delta(\Sigma)$ is $\Delta^\uparrow = (\Sigma_i)_{i=1}^{|\Sigma|}$ with $\Sigma_i := \Sigma \setminus \{\sigma_i\}$, where Σ is enumerated as $\{\sigma_i \mid i \in [1, |\Sigma|]\}$. It is not difficult to see that the following holds.

Observation 1. *Given any two distributions Δ, Δ' of Σ and any language L over Σ , if L is language decomposable with respect to Δ and $\Delta \leq_\Sigma \Delta'$, then L is also language decomposable with respect to Δ' .*

Proof. Let $\Delta = (\Sigma_i)_{i=1}^n, \Delta' = (\Sigma'_i)_{i=1}^k$ be two distributions of Σ such that $\Delta \leq_\Sigma \Delta'$. Suppose L is language decomposable with respect to Δ . For each $i \in [1, k]$, let $E(i) = \Sigma'_i \setminus \bigcup_{j \in f^{-1}(i)} \Sigma_j$. Clearly $\Sigma'_i = \bigcup_{j \in f^{-1}(i)} \Sigma_j \cup E(i)$. It is not difficult to see that $P'_i(L) \subseteq \bigcup_{j \in f^{-1}(i)} P_j(L) \parallel P_{E(i)}(L)$ if $E(i) \neq \emptyset$ and $P'_i(L) \subseteq \bigcup_{j \in f^{-1}(i)} P_j(L)$ if

$E(i) = \emptyset$. It follows that $\|_{i=1}^k P'_i(L) \subseteq \|_{i=1}^k (\|_{j \in f^{-1}(i)} P_j(L)) = \|_{j=1}^n P_j(L) = L$, since $\{f^{-1}(i) \mid i \in [1, k]\}$ is a partition of $[1, n]$. ■

In particular, Observation 1 implies that there is a distribution Δ of Σ such that L is language decomposable with respect to Δ if and only if L is language decomposable with respect to $\Delta^\uparrow$. The following corollary is then immediate.

Corollary 3.4. *Given an arbitrary regular language $L \subseteq \Sigma^*$, the problem of determining whether there is a distribution Δ of Σ with respect to which L is language decomposable is of complexity $\mathcal{O}(|\Sigma|m^{|\Sigma|+1})$.*

Observation 1 will be frequently used in the following manner: we look for a set of distributions Δ_i 's, where $\Delta \leq_\Sigma \Delta_i$, such that the language decomposability of L with respect to these Δ_i 's implies the language decomposability of L with respect to Δ . Then, by Observation 1, L is language decomposable with respect to Δ if and only if L is language decomposable with respect to these Δ_i 's. If the sizes of these distributions Δ_i 's are smaller than the size of Δ , then the worst case verification complexity could be improved. We now show that in some sense the converse of Observation 1 also holds, which justifies the consideration of distributions Δ_i 's satisfying $\Delta \leq_\Sigma \Delta_i$.

Proposition 3.4. *Let $\Delta = (\Sigma_i)_{i=1}^n$, $\Delta' = (\Sigma'_i)_{i=1}^k$ be any two distributions of Σ . If the language decomposability of L with respect to Δ implies the language decomposability of L with respect to Δ' , for any language $L \subseteq \Sigma^*$, then $\Delta \leq_\Sigma \Delta'$.*

Proof: Suppose to the contrary it is not the case that $\Delta \leq_\Sigma \Delta'$. Then there exists $i \in [1, n]$ such that for any $j \in [1, k]$, $\Sigma_i \not\subseteq \Sigma'_j$. Thus, $\Sigma_i \cap \Sigma'_j \subsetneq \Sigma_i$ for each $j \in [1, k]$. Clearly, any language over Σ_i is language decomposable with respect to $\Delta = (\Sigma_i)_{i=1}^n$. Let Σ_i be enumerated as $\{\sigma_1, \sigma_2, \dots, \sigma_m\}$. It is not that difficult to see that $L := \{\sigma_1^{\delta_{1,j}} \sigma_2^{\delta_{2,j}} \dots \sigma_m^{\delta_{m,j}} \mid j \in [1, k]\}$ over Σ_i is not language decomposable

with respect to $\Delta' = (\Sigma'_i)_{i=1}^k$, where $\delta_{h,j} := 1$ if $\sigma_h \in \Sigma_i \cap \Sigma'_j$ and $\delta_{h,j} := 2$ otherwise. In fact, $s = \sigma_1 \sigma_2 \dots \sigma_m \in \prod_{j=1}^k P'_j(L)$ but $s \notin L$. The reasoning is as follows. Indeed, since $\Sigma_i \cap \Sigma'_j \subsetneq \Sigma_i$, for each $j \in [1, k]$ there exists $h \in [1, m]$ such that $\sigma_{h,j} \notin \Sigma_i \cap \Sigma'_j$, i.e. $\delta_{h,j} = 2$. This implies that $s \notin L$. Since $L \subseteq \Sigma_i^*$, we have $P'_j(L) \subseteq (\Sigma_i \cap \Sigma'_j)^*$. Furthermore $P'_j(\sigma_1 \sigma_2 \dots \sigma_m) = P'_j(\sigma_1^{\delta_{1,j}} \sigma_2^{\delta_{2,j}} \dots \sigma_m^{\delta_{m,j}}) \in P'_j(L)$ for each $j \in [1, k]$, with the definition of $\delta_{h,j}$. It follows that $s \in \prod_{j=1}^k P'_j(s) \subseteq \prod_{j=1}^k P'_j(L)$. ■

In this thesis, we only consider those distributions Δ_i 's that are obtained from Δ by merging some sub-alphabets Σ_j 's together. Apparently, there are distributions where this technique cannot be applied, e.g. $\Delta^\uparrow$. We say Δ is *reducible* if there exists such a set of distributions Δ_i 's and Δ is *k-reducible* if there exists such a set of distributions Δ_i 's that the maximum of the sizes of these Δ_i 's is k . This particular set of Δ_i 's is said to be a *k-reduction* for Δ . Rephrased in our terminology, it was shown in [58] that $\Delta = (\Sigma_i)_{i=1}^n$ is 2-reducible if $\exists \Sigma_0 \subseteq \Sigma, \forall i, j \in [1, n], (i \neq j \implies \Sigma_i \cap \Sigma_j = \Sigma_0)$; and $\{(\Sigma_i, \bigcup_{j \neq i} \Sigma_j) \mid i \in [1, n]\}$ is a 2-reduction for Δ .

There are many properties that could be used to identify those distributions that are reducible. As illustrations of the above idea, we will only exploit two special properties of the distributions: the existence of *domination*, multiple *connected components*. The reductions that we provide are by no means the best reductions that could be achieved using our technique, since other special properties could be explored to possibly render a more efficient reduction.

Domination

Any subset $\Sigma_0 \subseteq \Sigma$ is said to *dominate* the distribution $\Delta = (\Sigma_i)_{i=1}^n$ if $\bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) \subseteq \Sigma_0$. That is, Σ_0 dominates Δ if Σ_0 contains all the shared events in Δ . We observe the fact that if there exists a dominant sub-alphabet Σ_k in Δ , i.e., $\bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) \subseteq \Sigma_k$ for some $k \in [1, n]$, then to check the language decomposability

of L with respect to Δ we only need to perform $n - 1$ language decomposability verifications each of which costs $\mathcal{O}(|\Sigma|m^3)$. This is stated in Proposition 3.5 below, which considerably generalizes Theorem 8 of [58]. We note that the fact that Σ_k dominates the distribution Δ implies that $(\Sigma_i, \bigcup_{j \neq i} \Sigma_j)$ is also a distribution⁷ of Σ for each $i \neq k$. For any set $I \subseteq [1, n]$, $P_I : \Sigma^* \mapsto (\bigcup_{i \in I} \Sigma_i)^*$ is used to denote the natural projection.

Proposition 3.5. *If there exists some $k \in [1, n]$ such that Σ_k dominates $\Delta = (\Sigma_i)_{i=1}^n$, then L is language decomposable with respect to Δ if and only if for each $i \in [1, n] \setminus \{k\}$, L is language decomposable with respect to $\Delta_i = (\Sigma_i, \bigcup_{j \neq i} \Sigma_j)$. The complexity⁸ of language decomposability verification with respect to Δ is $\mathcal{O}(n|\Sigma|m^3)$.*

Proof. Suppose L is language decomposable with respect to Δ . By Lemma 2.6, for any $i \in [1, n]$, $P_{[1, n] \setminus \{i\}}(L) = P_{[1, n] \setminus \{i\}}(\|_{j=1}^n P_j(L)) = \|_{j=1}^n P_{[1, n] \setminus \{i\}}(P_j(L))$ since $\bigcup_{j \neq i} \Sigma_j$ dominates Δ . We observe that, for each $j \neq i$, $P_{[1, n] \setminus \{i\}}(P_j(L)) = P_j(L)$. It follows that $P_i(L) \| P_{[1, n] \setminus \{i\}}(L) = P_1(L) \| \dots \| P_k(L) \| \dots \| P_n(L) \| P_{[1, n] \setminus \{i\}}(P_i(L)) = L$, since $(\bigcup_{j \neq i} \Sigma_j) \cap \Sigma_i \subseteq \Sigma_k$.

Suppose for each $i \in [1, n] \setminus \{k\}$, $L = P_i(L) \| P_{[1, n] \setminus \{i\}}(L)$ and assume, without loss of generality, that $k = n$. From $L = P_1(L) \| P_{[1, n] \setminus \{1\}}(L)$ and $L = P_2(L) \| P_{[1, n] \setminus \{2\}}(L)$, we have $L = P_1(L) \| P_{[1, n] \setminus \{1\}}(P_2(L) \| P_{[1, n] \setminus \{2\}}(L)) = P_1(L) \| P_2(L) \| P_{j \in [1, n] \setminus \{1, 2\}}(L)$ by Lemma 2.6, since $\Sigma_2 \subseteq \bigcup_{j \neq 1} \Sigma_j$ and $(\bigcup_{j \neq 1} \Sigma_j) \cap (\bigcup_{j \neq 2} \Sigma_j) = \bigcup_{j \in [1, n] \setminus \{1, 2\}} \Sigma_j \cup (\Sigma_1 \cap \Sigma_2) = \bigcup_{j \in [1, n] \setminus \{1, 2\}} \Sigma_j$. Together with $L = P_3(L) \| P_{[1, n] \setminus \{3\}}(L)$, similar analysis as above shows that $L = P_1(L) \| P_2(L) \| P_3(L) \| P_{j \in [1, n] \setminus \{1, 2, 3\}}(L)$. By successive substitutions of $L = P_i(L) \| P_{[1, n] \setminus \{i\}}(L)$ for $i = 3, 4, \dots, n - 1$, it is not difficult to obtain that $L = \|_{i=1}^{n-1} P_i(L) \| P_{[1, n] \setminus \{1, 2, \dots, n-1\}}(L) = \|_{i=1}^n P_i(L)$. ■

⁷Otherwise, it could be shown that $\Sigma_i \subseteq (\bigcup_{j \neq i} \Sigma_j) \cap \Sigma_i \subseteq \Sigma_k$, violating the definition of a distribution.

⁸Since the complexity of verifying language decomposability is $\mathcal{O}(((n+1)|\Sigma| - \sum_{i=1}^n |\Sigma_i|)m^{n+1})$ (see page 32) for distributions of size n , the complexity becomes $\mathcal{O}(|\Sigma|m^3)$ when the distribution is of size 2. Now since $n - 1$ decomposability verifications, where the size of each distribution is 2, have to be carried out, the complexity of language decomposability verification in this case is $\mathcal{O}(n|\Sigma|m^3)$.

Example: We now present an example to illustrate the notion of domination. Let $\Sigma = \{a, b, c, d, e, f\}$. Consider the distribution $\Delta = (\{b, c\}, \{b, d\}, \{a, b\}, \{c, e\}, \{c, f\})$. It is clear that the sub-alphabet $\{b, c\}$ dominates Δ . Thus, for any language L over Σ , L is decomposable with respect to Δ if and only if L is decomposable with respect to $\Delta_1 = (\{b, d\}, \{a, b, c, e, f\})$, $\Delta_2 = (\{a, b\}, \{b, c, d, e, f\})$, $\Delta_3 = (\{c, e\}, \{a, b, c, d, f\})$ and $\Delta_4 = (\{c, f\}, \{a, b, c, d, e\})$, according to Proposition 3.5. $\square$

Multiple Connected Components

We now present another special property of distributions that could be exploited to obtain a reduced verification complexity, which is based on the existence of multiple connected components.

Let $G(\Delta) = (V, E)$ be an undirected graph associated with the distribution Δ , with the vertex set $V = \{v_1, v_2, \dots, v_n\}$. We define a bijective map $f : V \mapsto \{\Sigma_1, \Sigma_2, \dots, \Sigma_n\}$ such that $f(v_i) = \Sigma_i$. Let $E = \{(v, v') \in V \times V \mid v \neq v' \wedge f(v) \cap f(v') \neq \emptyset\}$. Let $\{C_1, C_2, \dots, C_k\}$ be the set of connected components of $G(\Delta)$, where C_i is also abused to denote the vertices of the i th connected component. Clearly $C_i \cap C_j = \emptyset$, for $i \neq j$, and $\bigcup_{i=1}^k C_i = V$, i.e., the collection $\{C_1, C_2, \dots, C_k\}$ is a partition of V . For any $V' \subseteq V$, let $I(V') = \{j \in [1, n] \mid v_j \in V'\}$ be the index set for the vertices of V' . For example, let $\Sigma = \{a, b, c, d, e, f, g, h\}$. Consider the distribution $\Delta_c = (\{a, b\}, \{b, c\}, \{c, d\}, \{e, f\}, \{f, g\}, \{g, h\})$ of Σ . The undirected graph $G(\Delta_c)$ associated with Δ_c is shown in Fig. 3.1. Here there are two connected components, $C_1 = \{v_1, v_2, v_3\}$ and $C_2 = \{v_4, v_5, v_6\}$. We have $I(C_1) = \{1, 2, 3\}$ and $I(C_2) = \{4, 5, 6\}$.

It is straightforward to extract a verification algorithm based on the following result. We note that all the involved tuples of sub-alphabets of Σ in Proposition 3.6

Figure 3.1: Distribution $\Delta_c = (\{a, b\}, \{b, c\}, \{c, d\}, \{e, f\}, \{f, g\}, \{g, h\})$

are distributions of Σ .

Proposition 3.6. *Suppose there are at least two connected components in the graph $G(\Delta)$ of the distribution $\Delta = (\Sigma_i)_{i=1}^n$. Then L is language decomposable with respect to Δ if and only if $L = \parallel_{j \in I(C_i)} P_j(L) \parallel P_{I(V \setminus C_i)}(L)$ for each $i \in [1, k]$, where k is the number of connected components of $G(\Delta)$. If $|C_h| = 1$ for some $h \in [1, k]$, then L is language decomposable with respect to Δ if and only if $L = \parallel_{j \in I(C_i)} P_j(L) \parallel P_{I(V \setminus C_i)}(L)$ for each $i \in [1, k] \setminus \{h\}$.*

Proof: Suppose there are at least two connected components in the graph $G(\Delta)$. Then $V \setminus C_i \neq \emptyset$ for each $i \in [1, k]$. Suppose L is language decomposable with respect to Δ , then $P_{I(V \setminus C_i)}(L) = P_{I(V \setminus C_i)}(\parallel_{j=1}^n P_j(L)) = \parallel_{j \in I(V \setminus C_i)} P_j(L)$, since $\forall j \in I(C_i), \forall l \in I(V \setminus C_i), \Sigma_j \cap \Sigma_l = \emptyset$. Thus, we have for each $i \in [1, k]$,

$$\parallel_{j \in I(C_i)} P_j(L) \parallel P_{I(V \setminus C_i)}(L) = \parallel_{j=1}^n P_j(L) = L$$

In fact, the above equality also immediately follows from the decomposability of L with respect to Δ and Observation 1.

Suppose $L = \parallel_{j \in I(C_i)} P_j(L) \parallel P_{I(V \setminus C_i)}(L)$ holds for each $i \in [1, k]$. In particular, $L = \parallel_{j \in I(C_1)} P_j(L) \parallel P_{I(V \setminus C_1)}(L)$ and $L = \parallel_{j \in I(C_2)} P_j(L) \parallel P_{I(V \setminus C_2)}(L)$. Then we have $L = \parallel_{j \in I(C_1)} P_j(L) \parallel P_{I(V \setminus C_1)}(\parallel_{j \in I(C_2)} P_j(L) \parallel P_{I(V \setminus C_2)}(L))$. By Lemma 2.6, it is not difficult to see that

$$L = (\|_{j \in I(C_1)} P_j(L)) \| (\|_{j \in I(C_2)} P_j(L)) \| P_{I(V \setminus (C_1 \cup C_2))}(L)$$

since $I(C_2) \subseteq I(V \setminus C_1)$ and $\forall j \in I(C_1), l \in I(C_2), \Sigma_j \cap \Sigma_l = \emptyset$. We continue the above analysis by substitutions of $L = \|_{j \in I(C_i)} P_j(L) \| P_{I(V \setminus C_i)}(L)$, for $i = 3, 4, \dots, k-1$. At the $(k-1)$ th iteration, we have the equality that

$$L = (\|_{j \in I(C_1)} P_j(L)) \| \dots \| (\|_{j \in I(C_{k-1})} P_j(L)) \| P_{I(C_k)}(L)$$

Upon the final substitution $L = \|_{j \in I(C_k)} P_j(L) \| P_{I(V \setminus C_k)}(L)$, we have that $L = \|_{i=1}^k (\|_{j \in I(C_i)} P_j(L)) = \|_{j=1}^n P_j(L)$. Note that if $|C_h| = 1$ for some $h \in [1, k]$ (we may assume, without loss of generality, that $|C_k| = 1$), then at the $(k-1)$ th iteration the equality $L = \|_{j=1}^n P_j(L)$ has been obtained. $\blacksquare$

Example: As an illustration, Proposition 3.6 implies that for any language L over $\Sigma = \{a, b, c, d, e, f, g, h\}$, L is decomposable with respect to Δ_c if and only if L is decomposable with respect to the distributions $\Delta_1 = (\{a, b\}, \{b, c\}, \{c, d\}, \{e, f, g, h\})$ and $\Delta_2 = (\{a, b, c, d\}, \{e, f\}, \{f, g\}, \{g, h\})$. The complexity of language decomposability verification with respect to Δ_c reduces to $\mathcal{O}(|\Sigma|m^5)$ (instead of $\mathcal{O}(|\Sigma|m^7)$). $\square$

The complexity analysis of the verification algorithm based on Proposition 3.6 is shown below. For each connected component C_i of $G(\Delta)$, let $c(C_i) := -|\bigcup_{j \in I(C_i)} \Sigma_j| + \sum_{j \in I(C_i)} |\Sigma_j| \geq 0$. If there is only one connected component, or if there are two connected components in which one component is of cardinality one, then the complexity of testing language decomposability is $\mathcal{O}((n+1)|\Sigma| - \sum_{i=1}^n |\Sigma_i|)m^{n+1}$. Otherwise, using Proposition 3.6, the complexity is $\mathcal{O}(\sum_{i=1}^k ((|C_i| + 1)|\Sigma| - c(C_i))m^{|C_i|+2})$, where k is the number of connected components and $|C_i|$ is the size of the i th connected component. Let $d = \max_{i=1}^k |C_i|$ be the size of the largest connected components. We could see that, in the latter case, a crude upper bound for the complexity of checking language decomposability is $\mathcal{O}(n|\Sigma|m^{d+2})$. And we note that $d \leq n-2$.

3.3.2 Prefix-closed Specifications

It turns out that if L is prefix-closed, then there is an alternative technique for language decomposability verification. The key is the following result, which is essentially a generalization of Lemma 4.1 of [57].

Proposition 3.7. *Let L be any prefix-closed language over Σ and L_i any prefix-closed language over Σ_i for $i \in [1, n]$. If it holds that $L \subseteq \parallel_{i=1}^n L_i$, then the following three statements are equivalent:*

1. $L \neq \parallel_{i=1}^n L_i$.
2. $\exists \sigma \in \Sigma, \exists s \in L, s\sigma \notin L \wedge (\forall i \in I_\sigma, P_i(s\sigma) \in L_i)$
3. $\exists \sigma \in \Sigma, (L\sigma \cap L^c) \cap (\bigcap_{i \in I_\sigma} P_i^{-1}(L_i)) \neq \emptyset$

Proof: The equivalence of 2) and 3) is straightforward. We first show that 1) implies 2): suppose $L \neq \parallel_{i=1}^n L_i$, then $W := \{\hat{s} \in \Sigma^* \mid \hat{s} \in \parallel_{i=1}^n L_i, \hat{s} \notin L\} \neq \emptyset$, since it holds that $L \subseteq \parallel_{i=1}^n L_i$. We choose a minimal string (in terms of the prefix order) $s' \in W$. Clearly $\epsilon \notin W$ and thus $s' \neq \epsilon$. Thus, $\exists s \in \Sigma^*, \exists \sigma \in \Sigma, s' = s\sigma$. It follows from the fact that $s' \in \parallel_{i=1}^n L_i$ and each L_i is prefix-closed that $s \in \parallel_{i=1}^n L_i$. It follows from the definition of s' that $s \notin W$ and thus $s \in L$. Also $s\sigma = s' \notin L$. Finally $s\sigma = s' \in \parallel_{i=1}^n L_i$ implies that $P_i(s\sigma) \in L_i$. We then show that 2) implies 1): Let $s \in L, \sigma \in \Sigma$ be such that $s\sigma \notin L \wedge (\forall i \in I_\sigma, P_i(s\sigma) \in L_i)$. From $s \in L \subseteq \parallel_{i=1}^n L_i$, it follows that $P_i(s) \in L_i$ for each $i \in [1, n]$. Clearly $\forall i \notin I_\sigma, P_i(s\sigma) = P_i(s) \in L_i$. Thus, we have $s\sigma \in \parallel_{i=1}^n P_i(s\sigma) = (\parallel_{i \in I_\sigma} P_i(s\sigma)) \parallel (\parallel_{i \notin I_\sigma} P_i(s)) \subseteq \parallel_{i=1}^n L_i$. From the fact that $s\sigma \in \parallel_{i=1}^n L_i$ and $s\sigma \notin L$ we have $L \neq \parallel_{i=1}^n L_i$. ■

We still need the following result.

Lemma 3.4. *Let L be a prefix-closed regular language over Σ and generated by a deterministic finite automaton M . The deterministic finite automaton that recognizes $L\sigma \cap L^c$ is linear time computable from M , for each $\sigma \in \Sigma$.*

Proof. Let $M = (Q, \Sigma, \delta, q_0, Q)$ be a deterministic finite automaton that generates L . For each state $q \in Q$, if $\delta(q, \sigma)$ is not defined, then add an extra state $\bar{q} \notin Q$ to $\bar{Q}$ and the corresponding transition $(q, \sigma, \bar{q})$ to $\bar{\delta}$. It is clear that the deterministic finite automaton $\bar{M} = (Q \cup \bar{Q}, \Sigma, \delta \cup \bar{\delta}, q_0, \bar{Q})$ recognizes $L\sigma \cap L^c$, i.e., $L_m(\bar{M}) = L\sigma \cap L^c$. ■

From Proposition 3.7 and Lemma 3.4, we have the following.

Corollary 3.5. *Let L be a prefix-closed regular language over Σ and generated by a deterministic finite automaton M . The complexity of verifying the language decomposability of L with respect to Δ is $\mathcal{O}(\sum_{\sigma \in \Sigma} ((|I_\sigma| + 1)|\Sigma| - \sum_{i \in I_\sigma} |\Sigma_i|)m^{|I_\sigma|+1})$.*

Corollary 3.6. *Let L be a prefix-closed regular language over Σ and generated by a deterministic finite automaton M . The complexity of deciding the existence of a distribution Δ , with respect to which L is language decomposable, is $\mathcal{O}(|\Sigma|^2 m^{|\Sigma|})$.*

It turns out that verifying $L = \bigvee_{i=1}^n P_i(L)$ costs $\mathcal{O}(|\Sigma|^2 m^2)$ if the sub-alphabets of Δ are pairwise disjoint and L is a prefix-closed regular language. Since usually $|\Sigma| \ll m$, this is an improvement of the complexity $\mathcal{O}(n|\Sigma|m^3)$ (see Proposition 3.5) when L is only assumed to be a regular language.

Example: Consider the distribution $\Delta = (\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\})$ of $\Sigma = \{a, b, c, d, e\}$. The complexity of decomposability verification for prefix-closed specifications is $\mathcal{O}(|\Sigma|^2 m^3)$ (instead of $\mathcal{O}(|\Sigma|m^6)$) with respect to Δ . □

3.4 Discussions

In this chapter, we have presented a method to obtain the sufficient and necessary condition for the solvability of the decentralized realization problem, i.e., by reduc-

ing it to the problem of solving a set of language equations. The set of language equations is obtained by a direct translation of the problem and the key idea relies on a language equational characterization of finite state supervisors. This characterization of supervisors is possible because the definition of a supervisor does not take any plant into account, and is only based on a given control constraint. Apart from its simplicity, the advantage of this approach is that appropriate combinations of most of the important notions, e.g., controllability, normality, observability, $L_m(G)$ -closeness, can be encoded in language equations. And this shall facilitate the computation of supremal sublanguages for solving the synthesis problem, e.g. supremal controllable and normal sublanguage [94]. Also, the problem of supervisor decomposition can be easily defined and solved using our approach. The connection between the language decomposition, supervisor decomposition and decentralized realization problems also becomes more transparent. This facilitates the proof of lower bound results and reveals the importance of the language decomposability problem. In Section 3.3, complexity reduction techniques for the verification of language decomposability have been studied. We have paid special attention to the structure of the distributions and provided some structures for which reductions of the distributions can be carried out. We believe the notion of reducible distributions is of interest at least for the following reasons: 1) worst case complexity of language decomposability verification is reduced for a large class of distributions; 2) parallel verification is supported; 3) other efficient techniques for language decomposability verification or even language inclusion checking can be readily integrated, as verifications of language decomposability of L with respect to Δ_i 's still need to be performed and could be optimized accordingly; 4) it paves a way towards a better understanding of the boundary between the intractability and tractability of **LDP**, **SRP** and **DRP**. Note that since the definition of a supervisor is non-standard, the supervisor decomposition problem defined in this work is new and different from the

supervisor localization problem of [20], where the behavior of the plant needs to be explicitly considered.

The problems considered in this chapter are essentially reduced to finite state verification problems. If the given specification is verified to satisfy the corresponding properties, i.e., the set of language equations, then the local specifications or supervisors are often obtained as a byproduct of the verification process. A fundamental problem arises when specifications do not satisfy the desired properties. Some related problems will be addressed in the next chapter.

Chapter 4

Decentralized Supervisor

Synthesis

The decentralized realization problem (**DRP**), which was presented in Chapter 3, is too restrictive. In most cases, the answers to instances of the realization problem would be negative, that is, no tuple of supervisors would exist so that the closed-loop system is exactly language equivalent to the specification. This is because the specification has to satisfy the strong properties listed in Theorem 3.1 in order for the answer to be positive. In this chapter, we study the *decentralized supervisor synthesis problem* (**DeSSP**): “Given a global plant G , a global specification K and a decentralized control architecture $(\mathcal{A}_i)_{i=1}^n$, synthesize a tuple of local supervisors $(S_i)_{i=1}^n$ over the given architecture, if there exists one, such that the closed-loop system is non-blocking and language equivalent to a non-empty subset of the global specification K , i.e., $\emptyset \neq L_m(\|_{i=1}^n S_i \| G) \subseteq K$ and $\|_{i=1}^n S_i \| G$ is non-blocking.” Clearly, for the same problem inputs, **DeSSP** allows more solutions than **DRP**.

In view of the fundamental importance of **LDP** to **DRP**, instead of directly addressing **DeSSP**, we first study a relaxation of **LDP**, i.e., the *decomposable sub-language problem* (**DSP**), which is informally stated as follows: “Given a regular

language K representing the global specification and a distribution Δ , synthesize a tuple of regular languages $(K_i)_{i=1}^n$ representing a tuple of local specifications over the given distribution, if there exists one, such that the synchronous product of the local specifications is language equivalent to a non-empty subset of the specification K , i.e., $\emptyset \neq \prod_{i=1}^n K_i \subseteq K$.”

Another problem that is related to **DSP** is the joint observability problem (**JOP**), which is also an important problem in the area of decentralized supervisor synthesis. Two disjoint languages are said to be jointly observable with respect to a distribution if any string in one language could be distinguished from any string in the other language, by at least one local observation point. Put in other words, the property of joint observability requires any “good” string to be distinguishable from any “bad” string by at least one local observation point. Since control decisions made by the supervisors crucially rely on the information they obtain through observation, it is not difficult to see the importance of **JOP** to the decentralized synthesis problem. In fact, **JOP** has been used to establish the undecidability of **DeSSP** [8].

We provide a characterization of the decidability of **DSP** and **JOP**, based on results that available in the theory of traces. Then, an alternative proof of the undecidability of the prefix-closed joint observability problem (**PCJOP**) is provided by a reduction from **JOP**. Heuristics to solving these undecidable problems are also proposed.

Then, we obtain a sufficient condition for the undecidability of the distributed supervisor synthesis problem (**DiSSP**), which is a special case of **DeSSP**, using the characterization result for **DSP**. A decidable fragment of **DiSSP** is also presented. Then we establish the undecidability of a parameterized version of the decomposable sublanguage problem, referred to as the specification template synthesis problem (**STSP**). This is a key step towards showing the undecidability of a (language based) parameterized supervisor synthesis problem (**PSSP**), where both the plant template

and the supervisor template have only global and private events. Indeed, we show that **STSP** is undecidable if the template alphabet has at least two private events and **PSSP** is undecidable for a fixed finite state plant template if the plant template alphabet has at least two private events and one global event that are all controllable, even when the (schematic) specification is required to be symmetric. In particular, it will be shown that all the undecidability results for the synthesis problems are still valid even if (schematic) star free specifications or, equivalently, (schematic) linear temporal logic formulas are considered.

This chapter is organized as follows. In Section 4.1, we present the characterization result for **DSP** and propose some sound, but incomplete, heuristics for solving it. **JOP** and **PCJOP** are treated in Section 4.2. The application of the characterization result of **DSP** to **DiSSP** is shown in Section 4.3. Section 4.4 is devoted to showing the undecidability of **STSP** and **PSSP**. Discussions are provided in Section 4.5.

4.1 Decomposable Sublanguage Problem

In this section, we investigate **DSP**, which is formally stated below.

Problem 4 (Decomposable Sublanguage Problem). *Let Δ be a fixed distribution of Σ . For an arbitrary regular language L over Σ , determine whether L has a non-empty (regular) sublanguage that is decomposable with respect to Δ .*

It is not difficult to see that L has a non-empty decomposable sublanguage with respect to Δ if and only if there is a word s in L such that the decomposition closure $\|_{i=1}^n P_i(s)$ of s is a subset of L . Since the decomposition closure and the trace closure of a word coincide by Lemma 2.3, L has a non-empty decomposable sublanguage with respect to Δ if and only if there is a word s in L such that the trace closure $[s]_{I(\Delta)}$ of s is a subset of L if and only if L has a non-empty trace closed sublanguage with respect to $I(\Delta)$, where $I(\Delta)$ is the independence relation

induced by distribution Δ (see page 15). It follows from the undecidability of the existence of a non-empty trace closed sublanguage that **DSP** is undecidable for the distribution $\Delta = (\{a, b\}, \{c\})$ of $\{a, b, c\}$ [81], [86].

4.1.1 Characterization of the Decidability

In this subsection, we present a characterization of the decidability of **DSP**. The key observation is that there exists a closed form expression representing the unique supremal trace closed sublanguage of any language [87].

Lemma 4.1. *For any independence relation $I \subseteq \Sigma \times \Sigma$ and any language L over Σ , the supremal trace closed sublanguage of L with respect to I is $[L^c]^c$.*

It is straightforward to see that $\exists s \in L, [s] \subseteq L$ if and only if $[L^c]^c \neq \emptyset$ if and only if $[L^c] \neq \Sigma^*$. Thus, **DSP** is equivalent to the non-universality problem (**NUP**) in trace theory [88]. Given a fixed independence relation $I \subseteq \Sigma \times \Sigma$, the universality problem (**UP**) asks, for an arbitrary regular language L over Σ , whether the trace closure $[L]$ of L is equal to Σ^* . It is known that **UP** is decidable if and only if I is transitive [88]. Furthermore, in view of the undecidability of **UP** for star free languages [89], we have the following result.

Theorem 4.1. ***DSP** is decidable if and only if $I(\Delta)$ is transitive. If $I(\Delta)$ is non-transitive, the undecidability result is still valid even if L is required to be star free.*

Remark for Theorem 4.1: A view of this result, directly from the perspective of **DSP**, is as follows. On the one hand, if $I(\Delta)$ is not transitive, then there exist three letters $a, b, c \in \Sigma$ such that $(a, c), (c, b) \in I(\Delta)$ but $(a, b) \notin I(\Delta)$. It is known that each instance of the post correspondence problem (**PCP**) can be encoded as an instance of **DSP** with $L \subseteq (a + b + c)^*$ and $\Delta = (\{a, b\}, \{c\})$ [81], [86], [90]. By restricting $L \subseteq (a + b + c)^*$, the problem of existence of a non-empty trace closed sublanguage, when $I(\Delta)$ is not transitive, reduces to the case when $\Delta = (\{a, b\}, \{c\})$.

Thus, if $I(\Delta)$ is not transitive, **DSP** is undecidable. On the other hand, if $I(\Delta)$ is transitive, then for every regular language L there exists a regular language K such that $[L]^c = [K]$ and the finite state automaton that recognizes K is effectively computable [87]. In this case, the supremal trace closed sublanguage $[L^c]^c$ of L is equal to the trace closure of some regular language K , i.e., $[L^c]^c = [K]$. It is then obvious that the answer for the instance of **DSP** is no if and only if K is empty. Thus, if $I(\Delta)$ is transitive, then **DSP** is decidable. $\square$

Example: Consider the distribution $\Delta = (\{a, c\}, \{b, c\})$ whose induced independence relation is transitive (see page 15). It follows from Theorem 4.1 that it is decidable whether an arbitrary given regular language has a non-empty decomposable sublanguage with respect to Δ . For example, let $L = (ab + c)^*(a + (b + ac + aa)(a + b + c)^*)$ be a regular language over $\Sigma = \{a, b, c\}$, which is recognized by the automaton G shown in Fig. 4.1. L^c is recognized by the automaton G' shown in Fig. 4.2. Thus, $L^c = (ab + c)^* = ((ab)^* + c)^*$. It then follows that $[L^c] = (C_0 + c)^*$, where $C_0 = \{w \in (a + b)^* \mid |w|_a = |w|_b\}$, since $[(ab)^*] = C_0$. It can be shown that the supremal trace closed sublanguage $[L^c]^c$ of L is $(\epsilon + (a + b + c)^*c)C_1(c(a + b + c)^* + \epsilon)$, where $C_1 = \{w \in (a + b)^* \mid |w|_a \neq |w|_b\}$. In fact, any string in the complement of $(C_0 + c)^*$ must contain a substring from C_1 that is not immediately surrounded by a or b . Let $K = (\epsilon + (a + b + c)^*c)((ab)^*a^+ + (ab)^*b^+)(c(a + b + c)^* + \epsilon)$. It is clear that $[K] = [L^c]^c$. Since K is non-empty, we conclude that L has a non-empty decomposable sublanguage. In fact, for any string $s \in K$, $[s]$ is a non-empty decomposable sublanguage of L . $\square$

4.1.2 Sound Heuristics

The decidability results of decision problems about the trace closures of regular languages easily translate to the results for the corresponding decision problems

Figure 4.1: Automaton G that recognizes $L = (ab + c)^*(a + (b + ac + aa)(a + b + c))^*$

Figure 4.2: Automaton G' that recognizes $L^c = (ab + c)^*$

about the supremal trace closed sublanguages of regular languages. In particular, it is undecidable whether the supremal trace closed sublanguage of an arbitrary regular language is regular, since it is undecidable whether the trace closure of an arbitrary regular language is regular [88]. This implies operations that preserve regularity are in general not sufficient for computing the supremal trace closed sublanguage of an arbitrary regular language.

To circumvent this difficulty, it is worth studying heuristics for computing regular approximations of it and, for the purpose of further synthesis and verification, it is preferable for the regular approximations to be trace closed as well. Firstly, such a heuristics is useful for the synthesis of control and communication scheme for robot motion planning, where the specification is required to be trace closed [81], by synthesizing a trace closed subset from an arbitrary given specification. Secondly, it could be directly translated to a heuristics for the computation of the trace closure of an arbitrary regular language, which finds applications in many decision problems

in trace theory, in the partial order reduction based model checking and symbolic verification of some classes of mutual exclusion protocols [91]. We investigate two such heuristics.

1) Based on Lemma 2.4, it is straightforward to see that $(\|_{i=1}^n P_i(L^c))^c$, denoted by L_{amt} , is a regular under-approximation¹ of the supremal trace closed sublanguage of L . Such an under-approximation indeed provides a heuristics for **DSP**: If $(\|_{i=1}^n P_i(L^c))^c$, which is effectively regular, is non-empty, then L has a non-empty decomposable sublanguage with respect to Δ . Note that L_{amt} itself may not be decomposable with respect to Δ , since the class of decomposable languages is not closed under complementation.

2) Let $L_k = P_k(L) \setminus P_k((\|_{i=1}^n P_i(L)) \setminus L)$ for some $k \in [1, n]$ and $L_i = P_i(L)$ for $i \neq k$. Denote $L_k \| (\|_{i \neq k} L_i)$ by L_{amd}^k . It is easy to see that the following holds.

Proposition 4.1. $L_{amd}^k \subseteq L$.

Proof: $L_k \| (\|_{i \neq k} L_i) = P_k^{-1}(P_k(L) \cap P_k((\|_{i=1}^n P_i(L)) \cap L^c)^c) \cap \bigcap_{i \neq k} P_i^{-1}(P_i(L))$ by definition. Now $P_k^{-1}(P_k(L) \cap P_k((\|_{i=1}^n P_i(L)) \cap L^c)^c) = P_k^{-1}(P_k(L)) \cap P_k^{-1}(P_k((\|_{i=1}^n P_i(L)) \cap L^c)^c) = P_k^{-1}(P_k(L)) \cap P_k^{-1}((P_k(\|_{i=1}^n P_i(L)) \cap L^c)^c)$, since inverse projection distributes over intersection operator and commutes with complementation operator. Thus, $L_{amd}^k = P_k^{-1}(P_k((\|_{i=1}^n P_i(L)) \cap L^c)^c) \cap \bigcap_{i=1}^n P_i(L) \subseteq ((\|_{i=1}^n P_i(L)) \cap L^c)^c \cap \bigcap_{i=1}^n P_i(L) = L$. ■

Thus, L_{amd}^k is a (possibly empty) decomposable sublanguage of L , which can be used in [20] to synthesize a sub-specification that is language decomposable. There exists one degree of freedom in this scheme, i.e., the particular k could be arbitrarily chosen among $[1, n]$. Note that L_{amd}^k is a trace closed regular language by Lemma 2.5.

As approximations for the supremal trace closed sublanguage, $L_{amd}^k = L_k \| (\|_{i \neq k} L_i)$ and $L_{amt} = (\|_{i=1}^n P_i(L^c))^c$ are incomparable.

¹A similar idea has also appeared in [81], but in a slightly different setting.

Example: Let $L = \{ab, ba\}$ and $\Delta = (\{a\}, \{b\})$. We have $(\|_{i=1}^n P_i(L^c))^c = \emptyset$ but $L_k \| (\|_{i \neq k} L_i) = L$ for either $k = 1, 2$. Let $L = \Sigma^* \setminus \{ab\}$. We have $(\|_{i=1}^n P_i(L^c))^c = \Sigma^* \setminus \{ab, ba\}$ but $L_k \| (\|_{i \neq k} L_i)$ is equal to $\Sigma^* \setminus \Sigma^* a \Sigma^*$ and $\Sigma^* \setminus \Sigma^* b \Sigma^*$ for $k = 1$ and 2 , respectively. $\square$

In practice, an approximation for the supremal trace closed sublanguage of a regular language could be chosen from the $n+1$ candidates $L_{amt}, L_{amd}^k, k \in [1, n]$. Dually, a regular over-approximation of the trace closure of a regular language L could be chosen from the $n+1$ candidates $\|_{i=1}^n P_i(L), [(P_k(L^c) \setminus P_k(\|_{i=1}^n P_i(L^c)) \setminus L^c) \| (\|_{i \neq k} P_i(L^c))]^c, k \in [1, n]$. If L is prefix-closed, then it is desirable for each $L_i, i \in [1, n]$, to be also prefix-closed. For that purpose, we could modify L_k to be $P_k(L) \setminus P_k(\|_{i=1}^n P_i(L)) \setminus L \Sigma_k^*$, which is the supremal prefix-closed sublanguage of $P_k(L) \setminus P_k(\|_{i=1}^n P_i(L) \setminus L)$, with $L_i = P_i(L)$ kept unchanged for $i \neq k$.

Remark: There exists another technique for computing an upper-approximation of the trace closure of an arbitrary regular language, namely, the regular widening technique [92]. Compared with our method, the advantage of regular widening technique is obviously its wider applicability. Our heuristics is more geared towards the computation of the trace closures of regular languages and is simpler than regular widening technique. It attacks the approximation problem from a rather interesting point of view, i.e., the duality between the trace closure and the supremal trace closed sublanguage of a regular language. $\square$

We say an approximation of the supremal trace closed sublanguage (the trace closure) of a regular language is *exact* if it is equal to the supremal trace closed sublanguage (the trace closure) of the given language. We show that the exactness of the approximations L_{amt} or L_{amd}^k is undecidable. This is implied by the following stronger result. Indeed, the exactness of any regular under-approximation of the supremal trace closed sublanguage (any regular over-approximation of the trace closure) of an arbitrary regular language is decidable if and only if I is transitive.

We remark that the undecidability result is not implied by the undecidability of the regularity of $[L]$. For example, the problem whether an arbitrary regular language is trace closed, i.e., $[L] = L$, is decidable [93].

Proposition 4.2. *Let R be an operator on languages such that R preserves regularity and a finite automaton that recognizes $R(L)$ is effectively computable whenever L is regular, and $[L] \subseteq R(L)$, then for an arbitrary regular language L it is decidable whether $[L] = R(L)$ if and only if I is transitive.*

Proof: The proof idea is inspired by [89]. On one hand, if I is transitive, then it is decidable whether the trace closure of an arbitrary regular language is regular [88]. Moreover, a finite automaton that recognizes $[L]$ is effectively computable whenever $[L]$ is regular [89]. Thus, if I is transitive, it is decidable whether $[L] = R(L)$.

On the other hand, let A, B be two disjoint alphabets and f, g be two homomorphisms from A^* to B^* that encode a given instance of **PCP**, such that $\exists w \in A^+, f(w) = g(w)$ if and only if the answer to that instance is yes. Let $W_f = \{wf(w)c^{|f(w)|} \mid w \in A^+\}$, $W_g = \{wg(w)c^{|g(w)|} \mid w \in A^+\}$ and $\Delta = (A \cup B, \{c\})$, where c is a new symbol. Then $\exists w \in A^+, f(w) = g(w)$ if and only if $W_f \cap W_g \neq \emptyset$. It is possible to show that there exist two regular languages L_f, L_g such that $[L_f] = [W_f]^c$ and $[L_g] = [W_g]^c$ [89]. Let $L = L_f \cup L_g$. If $W_f \cap W_g = \emptyset$, then $[L] = [L_1 \cup L_2] = ([W_f] \cap [W_g])^c = [W_f \cap W_g]^c = \Sigma^*$. Since $\Sigma^* = [L] \subseteq R(L)$, $[L] = R(L)$. If $W_f \cap W_g \neq \emptyset$, then $[L]$ is not regular [89]. Thus, $[L] \neq R(L)$ since $R(L)$ is regular. Any decision procedure for verifying $[L] = R(L)$ then provides a decision procedure for **PCP**, thus the verification problem $[L] = R(L)$ is undecidable. Note that it is possible to encode the set $A \cup B$ using $\{a, b\}$. Thus, the undecidability result indeed holds when I is not transitive. ■

It turns out that **DSP** is related to the problem of joint observability [65], another important decision problem that will be discussed in the next section.

4.2 Joint Observability Problem

Formally, two disjoint languages G, B over Σ are said to be *jointly observable* with respect to a distribution $\Delta = (\Sigma_i)_{i=1}^n$ of Σ if $\forall s \in G, \forall s' \in B, \exists i \in [1, n], P_i(s) \neq P_i(s')$. The problem whether two arbitrary disjoint regular languages are jointly observable with respect to a given distribution is henceforth called the *joint observability problem* (**JOP**). **JOP** is known to be an undecidable problem [8]. We present a characterization of the decidability of **JOP** in the next subsection.

4.2.1 Characterization of the Decidability

As observed in [8], **JOP** is equivalent to the disjointness problem (**DP**) in trace theory². **DP** asks whether the trace closures of two arbitrary disjoint regular languages are disjoint for a given independence relation I , i.e., whether $[G] \cap [B] = \emptyset$. It is known that **DP** is decidable if and only if I is a transitive forest [77], [88]. Thus, the characterization result for **JOP** follows.

Theorem 4.2. ***JOP** is decidable if and only if $I(\Delta)$ is a transitive forest.*

Remark: This result subsumes the undecidability result established in [8]. If we use $\Sigma_1 = \{a, c\}$ to encode the **PCP** alphabet and $\Sigma_2 = \{b, d\}$ to encode the set of new symbols $\{a_1, a_2, \dots, a_n\}$ used in [8], then the independence relation induced by the distribution $\Delta = (\{a, c\}, \{b, d\})$ is indeed not a transitive forest [77]. $\square$

Example: Consider the distribution $\Delta = (\{a, b\}, \{a, c\}, \{a, d\})$ whose induced independence relation is a transitive forest (see page 16). It follows from Theorem 4.2 that **JOP** is decidable with respect to Δ . For example, let $G = dc^*ab^*$ and $B = d^*ac^*b$ be two regular languages over $\Sigma = \{a, b, c, d\}$. Clearly, $[G] = c^*dc^*ab^*$ and $[B] = d^*ac^*bc^*$. Thus, $[G] \cap [B] = \{dab\} \neq \emptyset$. G and B are not jointly observable

²However, [8] only shows the undecidability.

with respect to Δ . □

The heuristics developed for computing the trace closures of regular languages could also be used for solving **JOP** as well as the prefix closed joint observability problem introduced in the next subsection.

4.2.2 Prefix-closed Joint Observability Problem

It is possible to show that the prefix-closed joint observability problem (**PCJOP**) as formulated in [8] is equivalent to the following form: “Given two arbitrary disjoint (with the exception of the empty string ϵ) prefix-closed regular languages G', B' , i.e. $G' \cap B' = \{\epsilon\}$, determine whether $[G'] \cap [B'] = \{\epsilon\}$ ”. Clearly **PCJOP** is decidable if I is a transitive forest, since $[G'] \cap [B'] = \{\epsilon\}$ if and only if $[G' \setminus \{\epsilon\}] \cap [B' \setminus \{\epsilon\}] = \emptyset$. With $\Sigma_1 = \{a, c\}$ that encodes the **PCP** alphabet, $\Sigma_2 = \{b, d\}$ that encodes $\{a_1, a_2, \dots, a_n\}$ and $\Sigma_3 = \{e, m\}$ that encodes $\{b_1, b_2, \dots, b_n\}$, the result of [8] shows that each instance of **PCP** can be encoded as an instance of **PCJOP** if the graph of the independence relation contains an induced subgraph that corresponds to the distribution $\Delta = (\{a, c\}, \{b, d\}, \{e, m\})$. Let $G, B \subseteq \Sigma_1^*$ be any two disjoint languages. Let Δ_1 be any distribution of Σ_1 and let $\Delta_2 = (\Delta_1, \{e\}, \{m\})$ be a distribution of $\Sigma_2 = \Sigma_1 \cup \{e\} \cup \{m\}$ ³, where e, m are two new symbols not in Σ_1 . Let I_1 and I_2 be the induced independence relations of Δ_1 and Δ_2 , respectively. Then, the following reduction holds, which permits a straightforward extension of the undecidability result of the joint observability problem to the prefix-closed case.

Proposition 4.3. $[G]_{I_1} \cap [B]_{I_1} \neq \emptyset$ if and only if $[\overline{eGm}]_{I_2} \cap [\overline{mBe}]_{I_2} \neq \{\epsilon\}$.

Proof: If $[G] \cap [B] \neq \emptyset$, say $s \in G, s' \in B$ and $[s] = [s']$, then $esm \in \overline{eGm}, ms'e \in \overline{mBe}$ and $[esm] = [ms'e]$. Thus, $[\overline{eGm}] \cap [\overline{mBe}] \neq \{\epsilon\}$. If $[\overline{eGm}] \cap [\overline{mBe}] \neq \{\epsilon\}$,

³ Δ_2 augments the tuples of Δ_1 with $\{e\}, \{m\}$.

then $\exists s \in G, s' \in B$ such that $[\overline{esm}] \cap [\overline{ms'e}] \neq \{\epsilon\}$. This is only possible if and only if $[esm] = [ms'e]$ if and only if $[s] = [s']$. Then $[G] \cap [B] \neq \emptyset$. ■

This shows that **PCJOP** is undecidable if the graph of the independence relation contains an induced subgraph that corresponds to the distribution $\Delta = (\{a, c\}, \{b, d\}, \{e\}, \{m\})$. We remark that with above reduction step, it is also possible to establish the undecidability result even when one of the prefix-closed regular languages is fixed. In fact, with $\Delta = (\{a, c\}, \{b, d\}, \{e\}, \{m\})$, $G' = \overline{e(ab + cd)^+m}$ and $B' = \overline{m(u_1w_1 + \dots + u_nw_n)^+e}$, where u_i is over $\{a, c\}$ and w_i is its isomorphic copy over $\{b, d\}$, together encodes the **PCP** instance, i.e. $[G'] \cap [B'] \neq \{\epsilon\}$ if and only if $\exists i_1, i_2, \dots, i_k$ in $[1, n]$, $u_{i_1}u_{i_2} \dots u_{i_k}$ is isomorphic to $w_{i_1}w_{i_2} \dots w_{i_k}$. Note that G' is a fixed prefix-closed regular language that is independent of the **PCP** instance. This is essentially a slight modification of the construction suggested in [77].

4.3 Towards Decentralized Supervisor Synthesis

In this section, we study an undecidable fragment of **DeSSP**, in which the global finite state plant $G = \parallel_{k=1}^n G_k$ is the synchronous product of local finite state plants G_k 's over Σ_k 's and $\Sigma_{S_k, o} = \Sigma_k$ for each control constraint $\mathcal{A}_k$. That is, each local supervisor S_k observes all of the behavior of G_k . The corresponding supervisor synthesis problem is henceforth called the distributed supervisor synthesis problem (**DiSSP**)⁴. We now define the notion of a distributed control architecture, which is simply a specialized notion of the decentralized control architecture for **DiSSP**, that will be used in this section.

⁴In decentralized supervisor synthesis, each local supervisor observes and controls part of the behavior of the centralized plant. However, even within the discrete-event systems community there are different groups with different opinions on the notion of distributed supervisor synthesis. In distributed supervisor synthesis formulated in this work, each local supervisor observes all and controls part of the behavior of each local plant. The definition of “distributed supervisor synthesis” in this thesis follows the work of [20] and [39]. Note that distributed supervisor synthesis formulated in this work is often called “modular supervisor synthesis” in the literature [33], [35].

Definition 4.1. A distributed control architecture over Σ is a tuple $\mathcal{A} = ((\Sigma_k, \Sigma_{k,c}))_{k=1}^n$ of sub-alphabets of Σ , such that $\Delta(\mathcal{A}) := (\Sigma_k)_{k=1}^n$ is a distribution of Σ and $\Sigma_{k,c} \subseteq \Sigma_k$ for each $k \in [1, n]$.

DiSSP is now formulated below.

Problem 5 (Distributed Supervisor Synthesis Problem). Let $((\Sigma_k, \Sigma_{k,c}))_{k=1}^n$ be a fixed distributed control architecture over Σ . Let G_k be a fixed finite state plant over Σ_k , for each $k \in [1, n]$. For an arbitrary regular language L over Σ , determine whether there exists a tuple $(S_k)_{k=1}^n$ of finite state supervisors over $((\Sigma_k, \Sigma_{k,c}))_{k=1}^n$ such that: 1) $\|_{k=1}^n (S_k \| G_k)$ is non-blocking, and 2) $\emptyset \neq \|_{k=1}^n L_m(S_k \| G_k) \subseteq L$.

In above formulation of **DiSSP**, the distributed control architecture $\mathcal{A}$ and the local plants G_k , $k \in [1, n]$, are fixed a priori, while L is the only input of the decision problem. In this section, we reduce **DSP** to **DiSSP**. A sufficient condition for the undecidability of **DiSSP** is then obtained. Note that each local supervisor observes all the events of each local plant. Thus, each event is at least observed by one local supervisor. The undecidability of this restrictive formulation implies the undecidability of formulations of the problem that are more general. The idea of the reduction is explained below: If non-blockingness is not of concern, e.g., when the local sub-alphabets Σ_k 's are mutually disjoint, then **DiSSP** asks for a tuple $(S_k)_{i=1}^n$ of local finite state supervisors such that $\emptyset \neq \|_{k=1}^n L_m(S_k \| G_k) \subseteq L$. Note that each local closed-loop marked behavior $L_m(S_k \| G_k)$ has to be $L_m(G_k)$ -closed and controllable with respect to $(L(G_k), \Sigma_k \setminus \Sigma_{k,c})$. Then the instance of the synthesis problem has a solution if and only if for each $k \in [1, n]$, there exists an $L_m(G_k)$ -closed and controllable sublanguage L_k of $L_m(G_k)$, such that $\emptyset \neq \|_{k=1}^n L_k \subseteq L$. If $\Sigma_{k,c} = \Sigma_k$ for each $k \in [1, n]$, then the synthesis problem has a solution if and only if for each $k \in [1, n]$, there exists a string s_k in $L_m(G_k)$ such that $\emptyset \neq \|_{k=1}^n (\overline{s_k} \cap L_m(G_k)) \subseteq L$. Although the supervisor cannot eliminate the proper prefixes in $\overline{s_k} \cap L_m(G_k)$,

the plant G_k can eliminate them when $L_m(G_k)$ is constructed to be prefix free (A language is prefix free if each string in it is not a proper prefix of any other string in the same language). With this trick, $\overline{s_k} \cap L_m(G_k)$ will be equal to the singleton $\{s_k\}$. In this way, we have the required connection between **DSP** and **DiSSP**.

We say $\mathcal{A}$ satisfies condition (C) if there are $i, j \in [1, n]$ and four letters $a, b, c, d \in \Sigma$ such that $\{a, b, d\} \subseteq \Sigma_{i,c}$, $\{c, d\} \subseteq \Sigma_{j,c}$, $(a, c), (b, c) \in I(\Delta(\mathcal{A}))$ and for any $\sigma \in \Sigma \setminus \{a, b, c, d\}$, there exists $k \in [1, n]$ such that $\sigma \in \Sigma_{k,c}$. We have the following result.

Theorem 4.3. *Let $\mathcal{A} = ((\Sigma_k, \Sigma_{k,c}))_{k=1}^n$ be a fixed distributed control architecture over Σ that satisfies condition (C) with the four letters a, b, c, d as above. For each $k \in [1, n]$, if $d \in \Sigma_k$, then let $L_m(G_k) = (\Sigma_k \setminus \{d\})^* d$ and $L(G_k) = \overline{L_m(G_k)}$; if $d \notin \Sigma_k$, then let $L_m(G_k) = L(G_k) = \Sigma_k^*$. **DiSSP** is undecidable for the given distributed control architecture $\mathcal{A}$ and local plants G_k , $k \in [1, n]$. The undecidability result is still valid even if L is required to be star free.*

Proof: $\Delta(\mathcal{A})$ is not transitive. It will be shown that for any language $L' \subseteq \{a, b, c\}^*$, L' has a non-empty decomposable sublanguage with respect to $\Delta(\mathcal{A})$ if and only if there exists a solution for the instance of **DiSSP** with $L = L'd$, for the given $\mathcal{A}$ and local plants G_k , $k \in [1, n]$. It then follows that for any distributed control architecture $\mathcal{A}$ that satisfies condition (C), there exist n fixed finite state plants G_k , $k \in [1, n]$, such that **DiSSP** is undecidable. From Theorem 4.1 and the above reduction, the undecidability result is still valid even if L is required to be star free.

Indeed, if for the given $\mathcal{A}$ and G_k , $k \in [1, n]$, the instance of **DiSSP** with $L = L'd$ has a solution, then there exists a tuple $(S_k)_{k=1}^n$ of finite state supervisors over $\mathcal{A}$ such that $\emptyset \neq \bigcap_{k=1}^n L_m(S_k \| G_k) \subseteq L$. And then $\bigcap_{k=1}^n L_m(S_k \| G_k)$ is a non-empty decomposable sublanguage of $L = L'd$ with respect to $\Delta(\mathcal{A})$. It follows that there exists a word $s \in \{a, b, c, d\}^*$ such that $[s] \subseteq L'd$. Since $(a, d), (b, d), (c, d) \in D(\Delta(\mathcal{A}))$, it follows that $s = s'd$ for some word $s' \in \{a, b, c\}^*$ and $[s] = [s'd] = [s']d \subseteq L'd$.

It follows that $[s'] \subseteq L'$, i.e., L' has a non-empty decomposable sublanguage with respect to $\Delta(\mathcal{A})$. On the other hand, if L' has a non-empty decomposable sublanguage with respect to $\Delta(\mathcal{A})$, then there exists a word $s \in \{a, b, c\}^*$ such that $\|_{k=1}^n P_k(s) \subseteq L'$. Let $s_k = P_k(s)d$ if $d \in \Sigma_k$ and $s_k = P_k(s)$ if $d \notin \Sigma_k$. It is easy to see that $\emptyset \neq \|_{k=1}^n s_k \subseteq L'd = L$. We construct the supervisors S_k , $k \in [1, n]$, as follows: for each event $\sigma \in \Sigma \setminus \{a, b, c, d\}$, let one of the supervisors S_k with $\sigma \in \Sigma_{k,c}$ always disable it, which is possible due to condition (C). It follows that the closed-loop system can only execute events in $\{a, b, c, d\}$. For the particular $i, j \in [1, n]$ such that $\{a, b, d\} \subseteq \Sigma_{i,c}$, $\{c, d\} \subseteq \Sigma_{j,c}$, let supervisor S_i enforce the execution of closed behavior $\overline{P_i(s)d}$ by G_i and let supervisor S_j enforce the execution of closed behavior $\overline{P_j(s)d}$ by G_j . All the other supervisors S_k are not allowed to disable events in $\{a, b, c, d\} \cap \Sigma_{k,c}$. By construction, $\emptyset \neq \|_{k=1}^n L_m(S_k \| G_k) = \|_{k=1}^n s_k \subseteq L$. Also $\|_{k=1}^n (S_k \| G_k)$ is non-blocking. ■

The negative result established above is also applicable to the more general problem formulation, where both the local plants and the specification are the inputs of the decision problem. **DiSSP**, where the global plant G is given locally as interacting plants over local sub-alphabets, may be considered as a special case of **DeSSP** [57]. It follows that the above undecidability result immediately implies the undecidability of **DeSSP**⁵. In fact, condition (C) easily translates to a sufficient condition for the undecidability of **DeSSP**, for the fixed centralized plant $\|_{i=1}^n G_i$ given in Theorem 4.3. It turns out that only four letters a, b, c, d are needed in order to establish the undecidability of **DiSSP**. It is still open, when Σ has three letters, whether **DiSSP** is decidable for any given $\mathcal{A}$ over Σ and any local plants. However, it is possible to show that with $\Sigma = \{a, b\}$, **DiSSP** is always decidable. In fact, it will be shown that for any fixed $\mathcal{A}$ that satisfies the condition that $\exists i, (\forall j \neq i, |\Sigma_j| = 1)$,

⁵However, this does not imply that our result is stronger than the results of [7], [8], [9]. In [8], the supervisor is not required to be finite state. In [7], uncontrollable events are used. In [9], both the supervisors and the closed loop system are required to satisfy modal (loop) formulas.

DiSSP is decidable for any fixed local plants. The following lemma is needed.

Lemma 4.2. *Given any regular language L_3 over Σ_3 and any regular language L_2 over $\Sigma_2 \subsetneq \Sigma_3$, the supremal language $L_1^\uparrow$ over $\Sigma_1 := \Sigma_3 \setminus \Sigma_2$ such that $L_1^\uparrow \| L_2 \subseteq L_3$ is effectively regular. Indeed $L_1^\uparrow = P_1(L_3 \cap P_2^{-1}(L_2)) \setminus P_1((P_1(L_3 \cap P_2^{-1}(L_2)) \| L_2) \setminus L_3)$, where $P_1 : \Sigma_3^* \mapsto \Sigma_1^*$ and $P_2 : \Sigma_3^* \mapsto \Sigma_2^*$ are the natural projections.*

Proof: It is clear that the supremal element $L_1^\uparrow$ of $\{L_1 \subseteq \Sigma_1^* \mid L_1 \| L_2 \subseteq L_3\}$ exists and is unique. For notational convenience, we denote $K = P_1(L_3 \cap P_2^{-1}(L_2)) \subseteq \Sigma_1^*$. On one hand, $(K \setminus P_1((K \| L_2) \setminus L_3)) \| L_2 = P_1^{-1}(K \setminus P_1((K \| L_2) \setminus L_3)) \cap P_2^{-1}(L_2) = (P_1^{-1}(K) \setminus P_1^{-1}(P_1((K \| L_2) \setminus L_3))) \cap P_2^{-1}(L_2) \subseteq K \| L_2 \cap ((K \| L_2) \setminus L_3)^c \subseteq L_3$. That is, $K \setminus P_1((K \| L_2) \setminus L_3) \subseteq L_1^\uparrow$.

On the other hand, let s be any word over Σ_1 such that $\{s\} \| L_2 \subseteq L_3$. Clearly, $\{s\} \| L_2 \subseteq L_3 \cap P_2^{-1}(L_2)$. Thus, $s \in K$. It is not difficult to see that $s \notin P_1((K \| L_2) \setminus L_3)$. In fact, $(K \| L_2) \setminus L_3 = ((\{s\} \cup (K \setminus \{s\})) \| L_2) \setminus L_3 = ((K \setminus \{s\}) \| L_2) \setminus L_3$. Thus, we have $P_1((K \| L_2) \setminus L_3) \subseteq P_1((K \setminus \{s\}) \| L_2) \subseteq K \setminus \{s\}$. It follows that $L_1^\uparrow = \{s \in (\Sigma_3 \setminus \Sigma_2)^* \mid \{s\} \| L_2 \subseteq L_3\} \subseteq K \setminus P_1((K \| L_2) \setminus L_3)$. ■

Essentially, Lemma 4.2 computes the unique supremal decomposable sublanguage of L_3 , when component L_2 is fixed. Different techniques for computing different supremal sublanguages that arise in supervisory control theory are also available in the literature [94], [95], [96].

Proposition 4.4. *DiSSP is decidable, for any fixed distributed control architecture $\mathcal{A}$ that satisfies the condition $\exists i, (\forall j \neq i, |\Sigma_j| = 1)$ and for any fixed local plants G_k , $k \in [1, n]$.*

Proof: Let $\Sigma = \{a_k \mid k \in [1, n + m]\}$, where $m \geq 0$, and we assume, without loss of generality, $\Sigma_k = \{a_k\}$ for $k \in [1, n - 1]$ and $\Sigma_n = \{a_n, \dots, a_{n+m}\}$. Let G_k be any fixed local plant over Σ_k , $k \in [1, n]$. Let L be an arbitrary regular language over Σ . If there is any $k \in [1, n - 1]$ such that a_k is uncontrollable and G_k is blocked,

then there is no solution to the instance of **DiSSP**. Otherwise the test for the existence of a solution could be carried out as follows: For each $k \in [1, n-1]$, if a_k is uncontrollable, let S_k be a finite state supervisor such that $L_m(S_k \| G_k) = L_m(G_k)$; if a_k is controllable, let S_k be a finite state supervisor such that $L_m(S_k \| G_k) = \{s_k\}$, where s_k is the shortest word of $L_m(G_k)$. Let $L_k := L_m(S_k \| G_k)$ for $k \in [1, n-1]$. We compute the supremal language $L_n^\uparrow$ such that $L_1 \| L_2 \| \dots \| L_{n-1} \| L_n^\uparrow \subseteq L$, with Lemma 4.2. It is not difficult to see that the instance of **DiSSP** has a solution if and only if the supremal controllable and $L_m(G_n)$ -closed sublanguage of $L_n^\uparrow \cap L_m(G_n)$ is non-empty, the latter of which is decidable [97]. ■

Remark: An implication of Proposition 4.4 is that **DiSSP** is decidable for any $\mathcal{A}$, with $\Delta(\mathcal{A}) = (\{a, b\}, \{c\})$, and any local plants G_1, G_2 . **DSP** is undecidable for this distribution. □

4.4 Parameterized Supervisor Synthesis Problem

The standard setup for **DiSSP** assumes no structural relations between the local plants and supervisors. In this section, an important subclass of **DiSSP**, where the given local plants are assumed to be isomorphic and the supervisors are required to be isomorphic, is considered.

The idea of parameterized supervisor synthesis is to use instantiated supervisors from a designed supervisor template to control a parameterized family of finite state plants in a uniform manner. In this section, we explore the possibility of parameterized supervisor synthesis in a language based framework. A natural top-down approach for (language based) parameterized supervisor synthesis starts with a schematic regular language (user-given specification) $L(n)$ (see [59]), parameterized by the number of local plants n , and a plant template G (represented by a deterministic finite automaton) over the template alphabet) to be controlled; then,

a non-empty specification template T is synthesized such that the composition of n isomorphic copies of T satisfies the specification $L(n)$, for any $n \geq 2$; finally, a supervisor template S is synthesized according to the specification template T and plant template G .

It will be shown that the problem of determining whether there exists a non-empty specification template T as above, i.e., the specification template synthesis problem (**STSP**), is undecidable. Indeed, the result holds even when $L(n)$ is required to be symmetric and star free for each n , as long as the template alphabet has at least two private events. Note that any reasonable modeling formalism is capable of expressing two private events. So this undecidability result is rather strong. Moreover, **STSP** is a parameterized version of **DSP**. The above result is then used to establish the undecidability of a natural formulation of parameterized supervisor synthesis problem (without being restricted to the top-down approach). Some further terminology and notations are introduced below to facilitate the proof.

Let $\Sigma_T = \Sigma_g \cup \Sigma_p$ be the *template alphabet* [98], where Σ_g is the *global* event set and Σ_p is the *private* event set. A distribution $\Delta_T(n) = (\Sigma_i)_{i=1}^n$ of size n based on template alphabet Σ_T is a distribution of $\Sigma(n) := \bigcup_{i=1}^n \Sigma_i$, where $\Sigma_i = \Sigma_g \cup (\Sigma_p \times \{i\})$ for every $i \in [1, n]$. Let $h_i : \Sigma_T \mapsto \Sigma_i$ be a bijective map that maps events in Σ_T to events in Σ_i for each $i \in [1, n]$, such that $h_i(\sigma) = \sigma$, if $\sigma \in \Sigma_g$ and $h_i(\sigma) = (\sigma, i)$, if $\sigma \in \Sigma_p$. To simplify the notation, (σ, i) is denoted by σ_i . Depending on the context, h_i may also be regarded as a bijective map from Σ_T^* to Σ_i^* or a bijective map from $2^{\Sigma_T^*}$ to $2^{\Sigma_i^*}$. A language L over $\Sigma(n)$ is said to be *symmetric* with respect to $\Delta_T(n)$ if $\forall i, j \in [1, n], h_i^{-1}(P_i(L)) = h_j^{-1}(P_j(L))$.

Problem 6 (Specification Template Synthesis Problem). *Let $\Sigma_T = \Sigma_g \cup \Sigma_p$ be a fixed template alphabet. For an arbitrary schematic regular language $L(n)$ over $\bigcup_{i=1}^n \Sigma_i$ parameterized by n , where Σ_i 's are isomorphic copies of the template alphabet Σ_T , i.e., $\Sigma_i = h_i(\Sigma_T)$, determine whether there exists a non-empty specification template*

$T \subseteq \Sigma_T^*$ such that the composition $\|_{i=1}^n h_i(T)$ of n isomorphic copies of T satisfies the specification $L(n)$, i.e., $\|_{i=1}^n h_i(T) \subseteq L(n)$, for all $n \geq 2$.

STSP is essentially a parameterized version of **DSP**. By modifying the reduction used in [89] to work in the parameterized setting, it will be shown that **STSP** is undecidable. The construction is provided below.

Let A, B be two disjoint alphabets. Let $\Sigma_T = A \cup B$ and $h_i : \Sigma_T \mapsto \Sigma_i$ be a bijective map that maps events in Σ_T to events in Σ_i such that $h_i(\sigma) = \sigma_i$ for any $\sigma \in \Sigma_T$, for any $i \in [1, n]$. That is, $\Sigma_T = \Sigma_p$ and $\forall i \in [1, n], \Sigma_i = A_i \cup B_i$. Clearly, the independence relation $I(n)$ induced by $\Delta_T(n)$ is $\{(x, y) \in \Sigma(n) \times \Sigma(n) \mid x \in \Sigma_i \text{ and } y \in \Sigma_j \text{ for different } i \text{ and } j\}$. Let $f, g : A^* \mapsto B^*$ be two given homomorphisms such that $|f(\sigma)| \geq 1$ and $|g(\sigma)| \geq 1$ for any $\sigma \in A$, and f_i, g_i be two homomorphisms from A_i^* to B_i^* such that $f_i(\sigma_i) = h_i(f(\sigma))$ and $g_i(\sigma_i) = h_i(g(\sigma))$ for each $\sigma \in A$ and $i \in [1, n]$. For any $w \in A^+$, let $s_i^f(w) = h_i(wf(w)) \in (A_i \cup B_i)^*$ and $s_i^g(w) = h_i(wg(w)) \in (A_i \cup B_i)^*$, for each $i \in [1, n]$. $W_f(n) = \{s_1^f(w)s_2^f(w) \dots s_n^f(w) \mid w \in A^+\}$ and $W_g(n) = \{s_1^g(w)s_2^g(w) \dots s_n^g(w) \mid w \in A^+\}$ are constructed for each $n \geq 2$. Then the following key lemma holds.

Lemma 4.3. *There exist two star free languages $L_f(n)$ and $L_g(n)$ over $\Sigma(n)$ such that $[L_f(n)]^c = [W_f(n)]$ and $[L_g(n)]^c = [W_g(n)]$ with respect to $I(n)$; and moreover, $L(n) = \Sigma(n)^* \setminus (L_f(n) \cup L_g(n))$ is symmetric with respect to $\Delta_T(n)$ and star free, for each $n \geq 2$.*

Proof: The construction spirit described in [89] is adopted here and the following sets are constructed, whose union defines $L_f(n)$. The basic idea is that there are a finite number of rules which determine whether a word $s \in \Sigma(n)^*$ is in $[W_f(n)]$. They are listed below. A word $s \in [W_f(n)]$ if and only if the following conditions hold,

1. for each i , letters from each A_i must appear in s ;

2. for each i , no letter of B_i comes before letters of A_i ;
3. for each i, j , $h_i^{-1}(P_{A_i}(s)) = h_j^{-1}(P_{A_j}(s))$, where $P_{A_i} : \Sigma(n)^* \rightarrow A_i^*$ is the natural projection;
4. for each i, j , $h_i^{-1}(P_{B_i}(s)) = h_j^{-1}(P_{B_j}(s))$, where $P_{B_i} : \Sigma(n)^* \rightarrow B_i^*$ is the natural projection;
5. for each i , $P_{A_i}(s)$ matches $P_{B_i}(s)$ in the form of $h_i(wf(w))$ with $h_i(w) \in A_i^*$ and $h_i(f(w)) \in B_i^*$ for some $w \in A^+$.

A regular language $L_i(n)$ is then constructed such that each word in its trace closure $[L_i(n)]$ violates condition (i), for each $i \in [1, 5]$. It follows that the complement of the union of $[L_i(n)]$'s ($i \in [1, n]$) will be the set of words satisfying all conditions, i.e., the set $[W_f(n)]$. Then by setting $L_f(n) = \bigcup_{i=1}^5 L_i(n)$, in view of the property that trace closure distributes over union, it follows that $[L_f(n)]^c = [\bigcup_{i=1}^5 L_i(n)]^c = \bigcap_{i=1}^5 [L_i(n)]^c = [W_f(n)]$. The constructions of $L_i(n)$'s are given below.

1. Let $L_1(n) := \bigcup_{i=1}^n (\bigcup_{j \in [1, n], j \neq i} A_j \cup (\bigcup_{k=1}^n B_k))^*$, the complement of whose trace closure ensures that each word contains letters from each A_i .
2. Let $L_2(n) := \bigcup_{i=1}^n \Sigma(n)^* B_i \Sigma(n)^* A_i \Sigma(n)^*$, the complement of whose trace closure ensures that, for each $i \in [1, n]$, no letter of B_i comes before letters of A_i .
3. Regarding the third condition, let

$$L_3(n) := \bigcup_{i, j \in [1, n], i \neq j} (\bigcup_{k \in [1, n], k \neq i \wedge k \neq j} A_k)^* [(A_i A_j)^* (A_i^+ \cup A_j^+) \cup (A_i A_j)^* \{yy' \mid y \in A_i \wedge y' \in A_j \wedge y' \neq h_j(h_i^{-1}(y))\} (A_i A_j)^*] (\bigcup_{k=1}^n B_k)^*.$$

The meaning of $L_3(n)$ can be explained as follows. Two integers i and j in $[1, n]$ with $i \neq j$ are arbitrarily picked, and due to the independence relation $I(n)$,

all other A_k 's can be moved to the leftmost position, and all B_k 's (including B_i 's and B_j 's) to the rightmost position. The expression in the square bracket is to enumerate two cases which lead to $h_i^{-1}(P_{A_i}(s)) \neq h_j^{-1}(P_{A_j}(s))$. These are: (1) more letters from A_i or more letters from A_j appear in s , which is captured by the expression $(A_i A_j)^*(A_i^+ \cup A_j^+)$; (2) an equal number of letters from both A_i and A_j but some do not match, i.e., there exist $y \in A_i$ and $y' \in A_j$ such that y is paired with y' but $y' \neq h_j(h_i^{-1}(y))$.

4. Similarly a regular language $L_4(n)$ could be constructed, the complement of whose trace closure ensures the fourth condition:

$$L_4(n) := \bigcup_{i,j \in [1,n]: i \neq j} (\bigcup_{k=1}^n A_k)^* [(B_i B_j)^* (B_i^+ \cup B_j^+) \cup (B_i B_j)^* \{yy' \mid y \in B_i \wedge y' \in B_j \wedge y' \neq h_j(h_i^{-1}(y))\} (B_i B_j)^*] (\bigcup_{k \in [1,n]: k \neq i \wedge k \neq j} B_k)^*.$$

5. To come up with $L_5(n)$ for the last condition the following construction is performed:

$$\begin{aligned} L_5(n) = & \bigcup_{i,j \in [1,n]: i \neq j} (\bigcup_{k \in [1,n]: k \neq i} A_k)^* [\{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| = \\ & |f_i(x)|\}^* (\{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| < |f_i(x)|\} A_i^* \cup B_j^+) \cup \{xs \mid x \in A_i \wedge s \in \\ & B_j^* \wedge |s| = |f_i(x)|\}^* \{ys' \mid y \in A_i \wedge s' \in B_j^* \wedge |s'| = |f_i(y)| \wedge s' \neq \\ & h_j(h_i^{-1}(f_i(y)))\} \{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| = |f_i(x)|\}^*] (\bigcup_{k \in [1,n]: k \neq j} B_k)^*. \end{aligned}$$

By condition (4) it is known that the word in B_i^* and the word in B_j^* are isomorphic. Thus, the word in B_j^* is used to compare with the word in A_i^* . The main construction is similar to the constructions for $L_3(n)$ and $L_4(n)$, i.e., all cases that violate condition (5) are enumerated, which are described in the square bracket. That is: either more letters from A_i (but note that the matching is from one letter in A_i to a word in B_i^* instead of one letter in A_i to one letter in B_i due to the usage of $f_i : A_i^* \rightarrow B_i^*$) or more letters from B_j (which corresponds to B_i due to condition (4)) will violate the matching

condition, or the number of letters from A_i and the number of letters from B_j match with each other, but some $y \in A_i$ does not match $s' \in B_j^*$ in the sense that $s' \neq h_j(h_i^{-1}(f_i(y)))$.

It can be shown that all above five languages are indeed star free. Our first observation is that for any alphabet $\Sigma' \subseteq \Sigma(n)$, Σ'^* is star free. The star freeness of Σ'^* follows from the fact that $\Sigma'^* = (\emptyset^c(\Sigma(n) \cap \Sigma'^c)\emptyset^c)^c$. This immediately implies that $L_1(n)$ and $L_2(n)$ are star free. And $(\bigcup_{k \in [1, n]: k \neq i \wedge k \neq j} A_k)^*$, $(\bigcup_{k=1}^n B_k)^*$ are also star free. A_i^+ is star free since $A_i^+ = A_i A_i^*$. $\{yy' \mid y \in A_i \wedge y' \in A_j \wedge y' \neq h_j(h_i^{-1}(y))\}$ is by construction a finite language, thus also star free. The star freeness of $(A_i A_j)^*$ follows from the fact that $(A_i A_j)^* = ((\Sigma(n) \setminus A_i)\emptyset^c \cup \emptyset^c(\Sigma(n)^2 \setminus (A_i A_j \cup A_j A_i))\emptyset^c \cup \emptyset^c(\Sigma(n) \setminus A_j))^c$, which is similar to the construction of the star free expression for $(ab)^*$. It follows that $L_3(n)$ is star free. Similarly, $L_4(n)$ is also star free. To show that $L_5(n)$ is star free, we only need to show $\{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| = |f_i(x)|\}^*$ is star free. A star free expression for $\{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| = |f_i(x)|\}^*$ is

$$\begin{aligned} & ((\Sigma(n) \setminus \bigcup_{i=1}^n A_i)\emptyset^c \cup \bigcup_{x \in A_i} \emptyset^c(x \Sigma(n)^{\leq |f_i(x)|-1}) \cup \bigcup_{x \in A_i} \emptyset^c(x(\Sigma(n)^{|f_i(x)|} \setminus B_j^{|f_i(x)|}))\emptyset^c \cup \\ & \quad \bigcup_{x \in A_i} \emptyset^c(x B_j^{|f_i(x)|})(\Sigma(n) \setminus \bigcup_{i=1}^n A_i)\emptyset^c)^c. \end{aligned}$$

A more straightforward approach for showing the above languages star free is to provide linear temporal logic formulas or, equivalently, first order logic formulas, that define them [73]. For $\{xs \mid x \in A_i \wedge s \in B_j^* \wedge |s| = |f_i(x)|\}^*$, we remark that a linear temporal logic formula or first order logic formula can be used to express the following properties: 1) the element of the first position shall not be an element of $\Sigma(n) \setminus \bigcup_{i=1}^n A_i$; 2) for each position whose corresponding element σ belongs to A_i , the next $|f_i(\sigma)|$ positions correspond to elements from B_j and the element of the $(|f_i(\sigma)| + 1)$ -th position shall not be an element of $\Sigma(n) \setminus \bigcup_{i=1}^n A_i$. It follows that $L_f(n)$ is star free language. Now by suitably enlarging $L_f(n)$ it is possible to make it become symmetric with respect to $\Delta_T(n)$ while preserving $[L_f(n)]^c = [W_f(n)]$ and the star freeness of $L_f(n)$. Indeed, the relative positions of the subwords

$s_i^f(w)$ in $s(n) := s_1^f(w)s_2^f(w)\dots s_n^f(w)$ do not make a difference in $[W_f(n)]$ due to the independence relation $I(n)$. Thus, any procedure to construct $L_f(n)$ could be extended to a procedure that constructs a symmetric $L_f(n)$, by taking the union of $n!$ copies of $L_f(n)$ with indexes permuted. The meaning of index permutation is formalized by a permutation $\pi : [1, n] \mapsto [1, n]$. Abusing the notation, we write $\pi(\sigma_i) = \sigma_{\pi(i)}$ for any $\sigma \in \Sigma_p$, for any $i \in [1, n]$ and extend π to a homomorphism between words and languages over $\bigcup_{i=1}^n (\Sigma_p \times \{i\})$. Similarly, a symmetric and star free $L_g(n)$ can be constructed in this manner, by replacing every occurrence of f with g throughout the expression for $L_f(n)$. By construction, both of the new $L_f(n)$ and $L_g(n)$ are invariant under index permutation, i.e., for any permutation π , $\pi(L_f(n)) = L_f(n)$ and $\pi(L_g(n)) = L_g(n)$. Thus, $L(n) = \Sigma(n)^* \setminus (L_f(n) \cup L_g(n))$ is also invariant under index permutation. Invariance under index permutation is stronger than symmetry property. It follows that $L(n)$ is also symmetric with respect to $\Delta_T(n)$. Clearly, $L(n)$ is also star free. ■

Now it is possible to establish the undecidability of **STSP**.

Theorem 4.4. ***STSP** is undecidable if Σ_T has at least two private events. The undecidability result holds even when $L(n)$ is required to be symmetric with respect to $\Delta_T(n)$ and star free, for each $n \geq 2$.*

Proof: Let A, B be two disjoint alphabets. Any instance of Post's Correspondence Problem (**PCP**) [32], [89] is encoded by two homomorphisms $f, g : A^* \mapsto B^*$. The given **PCP** instance has a solution if and only if $\exists w \in A^+, f(w) = g(w)$. Let $\Sigma_T = A \cup B$ be a template alphabet with $\Sigma_T = \Sigma_p$ and $h_i : \Sigma_T \mapsto \Sigma_i$ be the bijective map. For each $n \geq 2$, the star free language $L(n)$ over $\Sigma(n)$ is constructed, which is symmetric with respect to $\Delta_T(n)$, from $L_f(n)$ and $L_g(n)$ that are constructed, respectively, from $W_f(n) = \{s_1^f(w)s_2^f(w)\dots s_n^f(w) \mid w \in A^+\}$ and $W_g(n) = \{s_1^g(w)s_2^g(w)\dots s_n^g(w) \mid w \in A^+\}$ as in Lemma 4.3. It will be shown that $\exists w \in A^+, f(w) = g(w)$ if and only if there exists a non-empty specification template

$T \subseteq \Sigma_T^*$ such that $\|_{i=1}^n h_i(T) \subseteq L(n)$, for any $n \geq 2$.

Suppose $w_0 \in A^+$ such that $f(w_0) = g(w_0)$. Then clearly, $s_i^f(w_0) = h_i(w_0 f(w_0)) = h_i(w_0 g(w_0)) = s_i^g(w_0)$ for each $i \in [1, n]$ and it follows that $s_1^f(w_0) s_2^f(w_0) \dots s_n^f(w_0) = s_1^g(w_0) s_2^g(w_0) \dots s_n^g(w_0)$ for any $n \geq 2$. It follows that $s(n) = s_1^f(w_0) s_2^f(w_0) \dots s_n^f(w_0) \in W_f(n) \cap W_g(n) \subseteq [W_f(n)] \cap [W_g(n)] = \Sigma(n)^* \setminus [L_f(n) \cup L_g(n)] \subseteq L(n)$ and at the same time $[s(n)] \subseteq [W_f(n) \cap W_g(n)] \subseteq [W_f(n)] \cap [W_g(n)] \subseteq L(n)$, for any $n \geq 2$. In view of Lemma 2.3, this implies $\|_{i=1}^n P_i(s(n)) \subseteq L(n)$, where $P_i : \Sigma(n)^* \rightarrow \Sigma_i^*$ is the natural projection. The special form $s(n) = s_1^f(w_0) s_2^f(w_0) \dots s_n^f(w_0)$ ensures that $\forall i \in [1, n], P_i(s(n)) = h_i(w_0 f(w_0))$. By setting $T = \{w_0 f(w_0)\} \subseteq \Sigma_T^*$, it follows that $\|_{i=1}^n h_i(T) \subseteq L(n)$, for any $n \geq 2$.

On the other hand, suppose T_0 is a non-empty specification template over Σ_T such that $\|_{i=1}^n h_i(T_0) \subseteq L(n)$, for any $n \geq 2$. Since $\|_{i=1}^n h_i(T_0)$ is trace closed with respect to $I(n)$ by Lemma 2.5, it follows that $\|_{i=1}^n h_i(T_0)$ is a non-empty subset of the supremal trace closed sublanguage of $L(n)$, i.e., $\|_{i=1}^n h_i(T_0) \subseteq [L(n)^c]^c = \Sigma(n)^* \setminus [L_f(n) \cup L_g(n)] = [W_f(n)] \cap [W_g(n)]$. Let s be any word of $[W_f(n)] \cap [W_g(n)]$. There exist $w_0, w_1 \in A^+$ such that $P_i(s) = s_i^f(w_0) = s_i^g(w_1)$ for $i \in [1, n]$. This implies that $s_1^f(w_0) s_2^f(w_0) \dots s_n^f(w_0) = s_1^g(w_1) s_2^g(w_1) \dots s_n^g(w_1) \in W_f(n) \cap W_g(n)$ and then $s \in \|_{i=1}^n P_i(s) \subseteq [W_f(n) \cap W_g(n)]$. It follows that $W_f(n) \cap W_g(n) \neq \emptyset$ and therefore $\exists w \in A^+, f(w) = g(w)$.

We finally note that the alphabets A and B can be encoded with the set $\{a, b\}$ [86], [89]. Thus, the above reduction works if Σ_T has at least two private events. ■

Any reasonable modeling formalism is capable of expressing two private events. So the above undecidability result is rather strong. By reduction from the undecidability of **STSP**, we are now able to establish the undecidability of a natural formulation of **PSSP** (without being restricted to the top-down approach), to be formally defined below.

Problem 7 (Parameterized Supervisor Synthesis Problem). *Let Σ_T be a fixed tem-*

plate alphabet and let $\Sigma_{T,c} \subseteq \Sigma_T$ be a fixed sub-alphabet. Let G be a fixed finite state plant template over Σ_T . For an arbitrary schematic regular language $L(n) \subseteq \Sigma(n)^*$, determine whether there exists a supervisor template S over Σ_T , where S observes all of Σ_T and controls all of $\Sigma_{T,c}$, such that for any $n \geq 2$: 1) $\|_{i=1}^n (S_i \| G_i)$ is non-blocking, and 2) $\emptyset \neq \|_{i=1}^n L_m(S_i \| G_i) \subseteq L(n)$. Here each S_i (respectively, G_i) over Σ_i is an isomorphic copy of S (respectively, G) and Σ_i is an isomorphic copy of Σ_T .

Theorem 4.5. *Let Σ_T and $\Sigma_{T,c}$ be given such that there are two private events a, b and one global event g with $\{a, b, g\} \subseteq \Sigma_{T,c}$. Let G be a fixed finite state plant template such that $L_m(G) = (a + b)^* g$ and $L(G) = \overline{L_m(G)}$. **PSSP** is undecidable in this setting, even if $L(n)$ is required to be symmetric with respect to $\Delta_T(n)$ and star free, for each $n \geq 2$.*

Proof: Let $\Sigma'_T = \{a, b\}$ and $h'_i : \{a, b\} \mapsto \{a_i, b_i\}$ be the corresponding bijective map. Let $L'(n)$ be an arbitrary schematic regular language over $\bigcup_{i=1}^n \{a_i, b_i\}$. Similar to the proof of Theorem 4.3, there exists a non-empty specification template $T' \subseteq \{a, b\}^*$ such that $\|_{i=1}^n h'_i(T') \subseteq L'(n)$ for each $n \geq 2$ if and only if there exists a solution for the instance of **PSSP** with $L(n) = L'(n)g$, for the given alphabets Σ_T , $\Sigma_{T,c}$ and plant template G . From Theorem 4.4 and the above reduction, the undecidability result is still valid even if $L(n)$ is required to be symmetric with respect to $\Delta_T(n)$ and star free, for each $n \geq 2$. ■

The above proof also works for the setting when n is a fixed integer, for any $n \geq 2$, or there is an upper bound on the number of plants [99]. It is noteworthy that only global and private events are needed in our problem formulation. Templates with only global and private events are very limited in expressiveness, since the problems of verifying state reachability and non-blocking properties are decidable for these templates [98], [100]. We will turn to a state based parameterized supervisor synthesis approach in the next chapter. Note that the undecidability of the language based approach cannot be translated to an undecidability result for the state based

approach, for templates with global and private events, since the global schematic specification used in our language based approach in general cannot be decomposed into a template.

4.5 Discussions

As a step towards a better understanding of the boundary between the decidability and undecidability of the decentralized supervisor synthesis problem (including the distributed supervisor synthesis problem), we have provided a sufficient condition for the undecidability based on the decentralized control architecture (distributed control architecture) in Section 4.3. It turns out that only four symbols are needed to establish the undecidability. In our current formulation of the distributed supervisor synthesis problem, each local supervisor observes the alphabet of exactly one local plant. It is of interest to study the decidability in a more general setup, where each local supervisor can possibly observe and control more than one local plant and each local plant can possibly be observed and controlled by more than one local supervisor. A sufficient condition for the undecidability in this general formulation can also be readily obtained. However, there is still a large gap between what is known to be decidable and what is known to be undecidable. Up to now, the “seed” undecidable problem that we have used for reduction is the decomposable sublanguage problem. It may be profitable to investigate a weak sufficient condition for the undecidability by a reduction from the joint observability problem, since a characterization of its decidability is available (Section 4.2).

The condition for the undecidability of the distributed supervisor synthesis problem is relatively mild. Thus, instead of identifying a decidable fragment of the synthesis problem, a practically more promising approach is to develop efficient approximation algorithms or semi-algorithms for synthesizing distributed supervisors.

The heuristics developed in Section 4.1.2 for computing decomposable sublanguages and trace closures of regular languages are potentially beneficial for that purpose.

The sufficient condition obtained for the undecidability of the language based parameterized supervisor synthesis problem seems to be quite weak. The synchronization mechanism of global events is restrictive. In fact, this class of parameterized systems is even unrealistic for modeling most systems involving low level behaviors and more discussions about the expressive power of these systems shall be carried out in the next chapter. However, the undecidability result for such a weak class of parameterized systems may indicate that any more expressive models would also have an undecidable synthesis problem. It is thus of particular interest to develop approximation algorithms and semi-algorithms for solving parameterized supervisor synthesis problem. In the next chapter, we shall turn to a state based parameterized supervisor synthesis framework, where the property to be enforced is that of state avoidance.

Chapter 5

Symbolic Analysis of Parameterized Systems

Parameterized systems such as telecommunication protocols and sensor networks, where an indefinite number of instantiations of the same finite state template interact in order to fulfill some goals, are becoming more and more pervasive and critical in modern society. However, the correctness of parameterized systems is in general rather difficult to analyze, due to the algorithmic undecidability and high computational complexity [59], [63], [101]. The fundamental problem of *reachability* is known to be undecidable for many classes of parameterized systems [63]. Both the model checking and parameterized supervisor synthesis problems are also undecidable for these classes of parameterized systems, via a reduction from the reachability problem [59], [61], [62]. In order to identify decidable fragments of the parameterized supervisor synthesis problem, it is natural to first identify subclasses of parameterized systems where the reachability problem is decidable. This chapter is mainly devoted to symbolic reachability analysis in the framework of regular model checking, where sets of states are represented by regular languages and system successor relations by regular relations [92], [102], [103], [104], [105], [106], [107]. Regular

model checking has been proposed as a generic symbolic framework for algorithmic verification of infinite state systems based on automata theory. The most fundamental problem considered in this framework is (symbolic) state space exploration, which relies on computing the reflexive and transitive closure of the (system) successor relations, i.e., computing the (system) reachability relations. However, symbolic reachability relations of most parameterized systems are either not effectively computable or non-regular. Partial algorithms for computing (over-approximations of) system reachability relations have been well investigated, using the techniques of widening [92], [106], inference of regular languages [108], abstraction-refinement of automata [109] and etc. [102], [110], [111].

In this work, we identify a class of parameterized systems whose symbolic reachability relations are effectively regular. The modules of each system are instantiated from a globally synchronized template, which is represented by a finite state automaton whose event set consists of global and private events, by a relabeling of the private events. All *isomorphic* modules in the system must synchronize on common global events while each private event is allowed to be executed independently. The class of globally synchronized templates is potentially useful for high level logic modeling of formation switching protocols in multi-agent systems [112], where the tick of clock and formation switching events are essentially global. The tick event “ t ” can be treated as global since usually a global clock is available or the local clocks have already been synchronized in multi-agent systems. Formation switching events usually need to be executed simultaneously to ensure safety during the formation transition phases and thus can be considered as global. To implement simultaneous formation switching, low level communication protocols are usually needed for negotiation and ensuring fault tolerance under agent failures, e.t.c., which shall not be the concern of and cannot be modeled by the high level formation switching protocols. A toy example of a globally synchronized template modeling the control

Figure 5.1: Control logic for a team of UAVs

logic for a team of UAVs (Unmanned Aerial Vehicles) is shown in Fig. 5.1, where $\Sigma_g = \{t, g\}$ and $\Sigma_p = \{r, b, s\}$. Here event b denotes the breakdown of an UAV, event r denotes the repair of an UAV, event s denotes the start of surveillance mode and event g denotes the global event of returning to the base station in formation.

[98] studies several verification and control problems for these systems. It is shown there that with a *quotient* construction, state reachability test, *system-blocking* and *system-deadlock* freeness verification problems are all decidable. Later in [100], algorithms that avoid explicit synchronization of modules are developed to verify component-level as well as system-level deadlock freeness and blocking freeness properties, in an attempt to reduce the verification complexity. Given n isomorphic modules, an algorithm is also provided in [98] that determines, for an arbitrary regular language specifying the legal behavior of n modules, the existence of n isomorphic local supervisors that ensures the language equivalence of the closed-loop system and the specification. If the marked behavior of the closed-loop system is allowed to be a non-empty subset of the global specification, then the decision problem becomes undecidable [65], as shown in Section 4.4. A more natural and efficient alternative is to synthesize a state-based supervisor that solves the parameterized control problem in a uniform manner, regardless of the number of isomorphic modules in the system [64].

The results and contributions of this chapter are as follows: we obtain a characterization of the expressive power of these systems in terms of their symbolic reachability relations and show that they correspond exactly to the set of iteration-closed finite unions of atomic star expressions over $S \times S$, where S is the finite state set of the template. More specifically, we show that the reachability relations of these systems are effectively iteration-closed finite unions of atomic star expressions, and for any iteration-closed finite union of atomic star expressions, there exists a template with only global events that realizes it, i.e., the reachability relation of this template is equal to the given iteration-closed finite union of atomic star expressions. In this work, we propose to use symbolic reachability relations for characterizing the expressive power of parameterized systems. This is motivated by the following considerations: 1) parameterized systems can be viewed as transducers that compute symbolic reachability relations, i.e., devices that provide output states when given input states. Relational characterization of computation has also been explored in [113], [114]; 2) symbolic reachability relations can potentially be used for comparing different classes of parameterized systems. It may be used to provide an explanation why some protocols cannot be modeled using certain classes of parameterized systems, by computing the set of reachable states. For example, globally synchronized templates cannot be used for modeling mutual exclusion protocol, since the set of reachable states is a finite union of atomic star expressions over S (a corollary of the characterization result). If an isomorphic module is in the critical section, then any other module can also be in the critical section at the same time; 3) symbolic reachability relations are fundamental to model checking. To the best of our knowledge, this is the first available result in characterizing the expressive power of classes of parameterized systems using symbolic reachability relations. We also perform symbolic reachability analysis for systems allowing entrance and exit of modules. For each template, regular expressions representing the sets of bad

states for various notions of deadlock and blocking properties are shown. It is then possible to compute these sets of reachable bad states and identify the minimum numbers of isomorphic modules that may cause the violations of these properties. A symbolic technique for computing the supremal non-blocking supervisor that enforces state avoidance property is then developed in the framework of regular model checking.

This chapter is organized as follows. Section 5.1 is devoted to miscellaneous preliminaries that are necessary for the understanding of this chapter. In Section 5.2.1, we formally define globally synchronized templates and set up the symbolic framework. The main expressiveness characterization result is obtained in Section 5.2.2. We then consider dynamic systems allowing entrance and exit of modules in Section 5.2.3. The applications of symbolic reachability analysis to the model checking problems, including the verification of various deadlock and blocking freeness properties, are shown in Section 5.2.4. The technique of regular supervisor synthesis is then developed in Section 5.3. Section 5.4 is devoted to the discussions.

5.1 Miscellaneous Preliminaries

We use $w_1 \sqcup w_2 := \{u_1v_1u_2v_2 \dots u_kv_k \mid w_1 = u_1u_2 \dots u_k, w_2 = v_1v_2 \dots v_k, u_i, v_i \in \Sigma^*\}$ to denote the shuffle of two strings w_1 and w_2 , and similarly $L_1 \sqcup L_2 := \bigcup_{w_1 \in L_1, w_2 \in L_2} w_1 \sqcup w_2$ for the shuffle of two languages L_1, L_2 . The *commutative closure* $[w]_{\sqcup}$ of a string $w = \sigma_1\sigma_2 \dots \sigma_n$, where $\sigma_i \in \Sigma$, is the set $\sigma_1 \sqcup \sigma_2 \dots \sqcup \sigma_n$. The commutative closure $[L]_{\sqcup}$ of L is defined to be $\bigcup_{w \in L} [w]_{\sqcup}$. L is said to be closed under commutation if $[L]_{\sqcup} = L$. The Cartesian product of A and B is denoted by $A \times B$. Given a set U , for a relation $R \subseteq U \times U$, its inverse R^{-1} is $\{(u, u') \in U \times U \mid (u', u) \in R\}$. For two relations $A, B \subseteq U \times U$, their relational composition (in that order) is $A \circ B = \{(u, u'') \in U \times U \mid \exists u' \in U, (u, u') \in A, (u', u'') \in B\}$.

$A \wedge (u', u'') \in B\}$. The i th iteration power $R^{i(\circ)}$ of R is defined inductively by $R^{0(\circ)} := \{(u, u) \in U \times U \mid u \in U\}$, and $R^{(i+1)(\circ)} := R^{i(\circ)} \circ R$ for $i \geq 0$. $R^{0(\circ)}$ is the diagonal relation over U , denoted by $\text{copy}(U)$, and is the identity for relational composition, i.e., $R \circ \text{copy}(U) = \text{copy}(U) \circ R = R$ for any relation $R \subseteq U \times U$. $R^{*(\circ)}$ stands for the reflexive and transitive closure, or the iteration closure, of R , i.e., $R^{*(\circ)} := \bigcup_{i=0}^{\infty} R^{i(\circ)}$. R is said to be iteration-closed if $R^{*(\circ)} = R$. Recall that an automaton G over Σ is a 5-tuple $(S, \Sigma, \delta, S_i, S_f)$, or a 3-tuple (S, Σ, δ) when S_i, S_f are not important. For each $a \in \Sigma$, we use $E_a := \{(s, s') \in S \times S \mid (s, a, s') \in \delta\}$ to denote the successor relation (of G) associated with a , for each event $a \in \Sigma$. Thus, G is also given as $(S, \Sigma, \{E_a \mid a \in \Sigma\}, S_i, S_f)$. We call $E_G := \bigcup_{a \in \Sigma} E_a$ the (system) successor relation and $E_G^{\circ(*)}$ the (system) reachability relation of G . Let $G^T := (S, \Sigma, \{E_a^{-1} \mid a \in \Sigma\}, S_f, S_i)$ denote the transposed automaton of G . The successor relation of G^T is $E_{G^T} = E_G^{-1} = \bigcup_{a \in \Sigma} E_a^{-1}$. The reachability relation $(E_G^{-1})^{\circ(*)}$ of G^T is the co-reachability relation of G . G is said to be non-blocking if $E_G^{\circ(*)}[S_i] \subseteq (E_G^{-1})^{\circ(*)}[S_f]$. Let $S(a) := \{s \in S \mid \exists s' \in S, (s, s') \in E_a\}$ denote the subset of states (of G) out of which there exists a transition labeled by a . For each state $s \in S$, let $\Sigma(s)$ be the set of events labeling the transitions out of state s , i.e. $\{a \in \Sigma \mid \exists s', (s, s') \in E_a\}$. For $\Sigma' \subseteq \Sigma$, let $S_{\Sigma'}$ be the set of states out of which there are only transitions labeled by events of Σ' , i.e., $S_{\Sigma'} := \{s \in S \mid \emptyset \neq \Sigma(s) \subseteq \Sigma'\}$. Let $S^{\perp}$ denote the set of deadlocked states, i.e., $S^{\perp} := \{s \in S \mid \Sigma(s) = \emptyset\}$.

We will mainly work with the free monoids S^* and $(S \times S)^*$ generated by S and $S \times S$ respectively, the latter monoid being embedded¹ in $S^* \times S^*$. A relation R over S^* is said to be length-preserving if whenever $(w_1, w_2) \in R \subseteq S^* \times S^*$, we have $|w_1| = |w_2|$, i.e., $R \subseteq (S \times S)^*$. We shall only focus on length-preserving relations in this work. A (length-preserving) relation $R \subseteq (S \times S)^*$ is said to be regular iff R is a regular language over the alphabet $S \times S$. The class of regular

¹We shall identify $(S \times S)^*$ with $\{(w_1, w_2) \in S^* \times S^* \mid |w_1| = |w_2|\} \subseteq S^* \times S^*$ and not distinguish monoids that are isomorphic in this work.

(length-preserving) relations is closed under intersection and relational composition. Every regular relation $R \subseteq (S \times S)^*$ is recognized by a finite state automaton over $S \times S$, often called a finite state transducer. For $L \subseteq S^*$, the image of L under the transduction of R is $R[L] = \{w \in S^* \mid \exists w' \in L, (w', w) \in R\}$. Given a regular relation $R \subseteq (S \times S)^*$ and a regular language L over S , $R[L]$ is effectively regular [72]. We call the relations in $Star(S \times S) := \{D^* \mid \emptyset \neq D \subseteq S \times S\}$ the atomic star expressions over $S \times S$. $Star(S \times S)$ is a finite set since $S \times S$ is a finite set. $\bigcup Star(S, S)$ is used to denote the collection of finite unions of atomic star expressions over $S \times S$. Since there are only a finite number of different unions of atomic star expressions over $S \times S$, $\bigcup Star(S, S)$ is also a finite set. Clearly $copy(S^*) = copy(S)^* \subseteq (S \times S)^*$ is an atomic star expression over $S \times S$. Every regular relation $R \subseteq (S \times S)^*$ is recognized by a finite state automaton over $S \times S$, often called a *finite state transducer*.

The algebraic structure $(2^{U \times U}, \cup, \circ, {}^{*(\circ)}, \emptyset, copy(U))$ is a Kleene algebra [115]. Thus, we have the following result.

Lemma 5.1. *Let each R_i be a relation over U . It holds that $(\bigcup_{i \in [1, k]} R_i)^{*(\circ)} = (R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)})^{*(\circ)}$.*

Proof. Clearly, $R_i \subseteq R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)}$ for each $i \in [1, k]$ and then $\bigcup_{i \in [1, k]} R_i \subseteq R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)}$. We have $(\bigcup_{i \in [1, k]} R_i)^{*(\circ)} \subseteq (R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)})^{*(\circ)}$. On the other hand, $R_i^{*(\circ)} \subseteq (\bigcup_{i \in [1, k]} R_i)^{*(\circ)}$ for each $i \in [1, k]$. Thus, $R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)} \subseteq (\bigcup_{i \in [1, k]} R_i)^{*(\circ)}$ and then $(R_k^{*(\circ)} \circ R_{k-1}^{*(\circ)} \circ \dots \circ R_1^{*(\circ)})^{*(\circ)} \subseteq (\bigcup_{i \in [1, k]} R_i)^{*(\circ)}$. ■

5.2 Globally Synchronized Templates

In the next subsection, we shall formally define globally synchronized templates, the class of parameterized systems studied in this work, and set up the symbolic framework for reasoning about them.

5.2.1 System Setup

Let $G = (S, \Sigma, \{E_a \mid a \in \Sigma\}, S_i, S_f)$ be a finite state automaton, where the finite event set Σ is partitioned into the global event set Σ_g and the private event set Σ_p , i.e., $\Sigma = \Sigma_g \dot{\cup} \Sigma_p$. Let G_i be an isomorphic module of G , obtained through relabeling the private event set Σ_p to $\Sigma_{p,i} := \Sigma_p \times \{i\}$ [98], [100]. For each number $n \geq 1$ of isomorphic modules, G generates a finite state automaton $\parallel_{i=1}^n G_i$. G also generates an infinite state automaton $G^\infty = \bigsqcup_{n \geq 1} h(\parallel_{i=1}^n G_i) = h(G_1) \sqcup h(\parallel_{i=1}^2 G_i) \sqcup h(\parallel_{i=1}^3 G_i) \sqcup \dots$, where h denotes the map that eliminates the indexes of private events from $\parallel_{i=1}^n G_i$. Here $\sqcup$ is the disjoint union operation defined on automata such that $G \sqcup G' := (S \cup S', \Sigma \cup \Sigma', \delta \cup \delta', S_i \cup S'_i, S_f \cup S'_f)$ of G and G' . Essentially, G^∞ is an infinite state automaton that simultaneously contains the transition structures of $\parallel_{i=1}^n G_i$, for all $n \geq 1$. The set of states of G^∞ is the union of the set of states of $\parallel_{i=1}^n G_i$, $n \geq 1$. For illustration, a globally synchronized template G , where $\Sigma_g = \{r, g\}$ and $\Sigma_p = \{a, b\}$, and the corresponding G^∞ are shown in Fig. 5.2. Note that G^∞ contains $G = h(G_1)$, thus only G^∞ is shown in the figure. The set of initial and the set of final states are not important here, thus they are not drawn in the figure.

We call G a *globally synchronized template*, as the only synchronization between different modules is through the global events. The states of G^∞ are encoded using strings over S , with the state $(s_1, s_2, \dots, s_n)$ of $h(\parallel_{i=1}^n G_i)$ encoded symbolically by the string $w = s_1 s_2 \dots s_n \in S^n$. For technical convenience, we use S^* instead of S^+ to encode the state space of G^∞ . Thus, G^∞ is represented by the tuple $(S^*, \Sigma, \{R_a \mid a \in \Sigma\}, S_i^*, S_f^*)$, where R_a encodes the successor relation (over S^*) of G^∞ associated with a . Our next result establishes the relationship between R_a and E_a (for an illustration of the lemma, see the example on page 82).

Lemma 5.2. *For $a \in \Sigma_p$, $R_a = \text{copy}(S^*)E_a\text{copy}(S^*)$. For $a \in \Sigma_g$, $R_a = E_a^*$*

Proof: For any private event a , $(s_1 s_2 \dots s_n, s'_1 s'_2 \dots s'_m) \in R_a$ if and only if $m = n$

Figure 5.2: A globally synchronized template G and the corresponding G^∞

and there exists an index i such that $(s_i, s'_i) \in E_a$ and $s_j = s'_j$ for $j \neq i$ if and only if $(s_1 s_2 \dots s_n, s'_1 s'_2 \dots s'_m) \in \text{copy}(S^*) E_a \text{copy}(S^*)$.

For any global event a , $(s_1 s_2 \dots s_n, s'_1 s'_2 \dots s'_m) \in R_a$ if and only if $m = n$ and $(s_i, s'_i) \in E_a$ for each i if and only if $(s_1 s_2 \dots s_n, s'_1 s'_2 \dots s'_m) \in E_a^*$. ■

Essentially, Lemma 5.2 is a formalization of the synchronization rules for the private and the global events in the framework of regular model checking, using the successor relations R_a 's of G^∞ . Each R_a , where $a \in \Sigma$, and thus the system successor relation $R_{G^\infty} = \bigcup_{a \in \Sigma} R_a$, is a regular relation and represented by a finite state transducer. One important question is whether $R_{G^\infty}^{*(\circ)}$ is regular for any globally synchronized template G . We provide an affirmative answer² and study the expressive power of the symbolic reachability relations of these templates in the next subsection.

²Note that we do not require G to be deterministic. Systems with multiple templates could also be dealt with in the symbolic framework in a similar way.

5.2.2 Expressiveness Characterization

To obtain an expressiveness characterization, we first show that for any globally synchronized template G , the reachability relation $R_{G^\infty}^{*(\circ)}$ is a finite union of atomic star expressions over $S \times S$, i.e., $R_{G^\infty}^{*(\circ)} \in \bigcup \text{Star}(S \times S)$ (Proposition 5.1). And then we show that for each iteration closed relation T in $\bigcup \text{Star}(S \times S)$, there exists a globally synchronized template G that realizes it, i.e., $R_{G^\infty}^{*(\circ)} = T$. (Proposition 5.2).

The following technical lemmas are needed in order to establish the first proposition. The first lemma essentially states that for subsets of $S \times S$, Kleene closure distributes over relational composition. The second lemma states that $R_a^{*(\circ)} \in \bigcup \text{Star}(S \times S)$ for each $a \in \Sigma$.

Lemma 5.3. *Let $D_1, D_2, \dots, D_n \subseteq S \times S$, then $D_n^* \circ \dots \circ D_2^* \circ D_1^* = (D_n \circ \dots \circ D_2 \circ D_1)^*$.*

Proof: We prove the statement by induction on n . The statement holds trivially when $n = 1$. Suppose it holds for some $k \geq 1$ that $D_k^* \circ \dots \circ D_2^* \circ D_1^* = (D_k \circ \dots \circ D_2 \circ D_1)^*$. Consider the case $n = k+1$. $D_{k+1}^* \circ D_k^* \circ \dots \circ D_2^* \circ D_1^* = D_{k+1}^* \circ (D_k \circ \dots \circ D_2 \circ D_1)^*$ by the induction hypothesis. We only need to show that for any $D, D' \subseteq S \times S$, $D^* \circ D'^* = (D \circ D')^*$. It then follows that $D_{k+1}^* \circ (D_k \circ \dots \circ D_2 \circ D_1)^* = (D_{k+1} \circ \dots \circ D_2 \circ D_1)^*$ and the proof is done.

Let $(\alpha, \beta) \in (D \circ D')^*$. By definition, there exists some $k \in \mathbb{N}$ such that $(\alpha, \beta) = (s_1, s'_1)(s_2, s'_2) \dots (s_k, s'_k) \in (D \circ D')^k$. For each $(s_i, s'_i) \in D \circ D'$, there exists $s''_i \in S$ such that $(s_i, s''_i) \in D$ and $(s''_i, s'_i) \in D'$. Thus, $(s_1 \dots s_{k-1} s_k, s''_1 \dots s''_{k-1} s''_k) \in D^*$ and $(s''_1 \dots s''_{k-1} s''_k, s'_1 \dots s'_{k-1} s'_k) \in D'^*$. This implies that $(s_1 \dots s_{k-1} s_k, s'_1 \dots s'_{k-1} s'_k) \in D^* \circ D'^*$. It follows that $(\alpha, \beta) \in D^* \circ D'^*$.

Let $(\alpha, \beta) \in D^* \circ D'^*$. Then $|\alpha| = |\beta|$, since both D^* and D'^* are length-preserving. There exists some $k \in \mathbb{N}$ such that $\alpha = s_1 s_2 \dots s_k$ and $\beta = s'_1 s'_2 \dots s'_k$. Also, there exist $s''_1, s''_2, \dots, s''_k \in S$ such that $(s_1 s_2 \dots s_k, s''_1 s''_2 \dots s''_k) \in D^*$ and $(s''_1 s''_2 \dots s''_k, s'_1 s'_2 \dots s'_k) \in D'^*$. Thus, $(s_i, s''_i) \in D$, $(s''_i, s'_i) \in D'$ and then $(s_i, s'_i) \in D \circ D'$.

$D \circ D'$. We then have $(\alpha, \beta) = (s_1 s_2 \dots s_k, s'_1 s'_2 \dots s'_k) \in (D \circ D')^*$ $\blacksquare$

Lemma 5.4. *For $a \in \Sigma$, $R_a^{*(\circ)} \in \bigcup \text{Star}(S \times S)$.*

Proof: For $a \in \Sigma_p$, $R_a^{*(\circ)} = (E_a^{*(\circ)})^* \in \bigcup \text{Star}(S \times S)$, where $\emptyset \neq \text{copy}(S) \subseteq E_a^{*(\circ)} \subseteq S \times S$. In fact, $E_a^{*(\circ)}$ is the reachability relation of G along a -paths, i.e., the set of state tuples (s, s') of G such that s' can be reached from s by a finite sequence (including zero sequence) of a 's. Then $(E_a^{*(\circ)})^*$ is the reachability relation of G^∞ along a -paths, i.e., $R_a^{*(\circ)}$, which captures the effect of executing sequences of a 's to all the modules. For $a \in \Sigma_g$, $R_a^{*(\circ)} = \bigcup_{i=0}^\infty R_a^{i(\circ)}$ corresponds to the supremum of the ascending chain $C_0 \subseteq C_1 \subseteq C_2 \subseteq \dots$, where $C_n = \bigcup_{i=0}^n R_a^{i(\circ)}$. It is not difficult to see that $R_a^{i(\circ)} = (E_a^{i(\circ)})^*$ and $E_a^{i(\circ)} \subseteq S \times S$. In fact, $E_a^{i(\circ)}$ is the reachability relation of G along a -paths of length i , i.e., the set of state tuples (s, s') of G such that s' can be reached from s by a finite sequence of a 's of length i . $R_a^{i(\circ)} = (E_a^{i(\circ)})^*$ then captures the effect that all the modules synchronously execute a finite sequence of a 's of length i . Note that $R_a^{0(\circ)} = \text{copy}(S)^*$ and $\emptyset \neq \text{copy}(S) \subseteq S \times S$. Thus, each $C_n \in \bigcup \text{Star}(S \times S)$. It follows that the chain $C_0 \subseteq C_1 \subseteq C_2 \subseteq \dots$ stabilizes, i.e., $\exists k \in \mathbb{N}, (\forall m \geq k, C_m = C_k)$, since $\bigcup \text{Star}(S \times S)$ is a finite set. Thus, $R_a^{*(\circ)} = C_k \in \bigcup \text{Star}(S \times S)$. $\blacksquare$

Lemma 5.4 essentially shows that the reachability relation of G^∞ along a -paths belongs to $\bigcup \text{Star}(S \times S)$, for any $a \in \Sigma$. Now we are ready to show Proposition 5.1, which says that the reachability relation of G^∞ also belongs to $\bigcup \text{Star}(S \times S)$.

Proposition 5.1. *For any globally synchronized template G , $R_{G^\infty}^{*(\circ)} \in \bigcup \text{Star}(S \times S)$.*

Proof: Let $\Sigma = \{a_1, a_2, \dots, a_{|\Sigma|}\}$. For each i , there exists some finite $k(i)$ such that $R_{a_i}^{*(\circ)} = \bigcup_{j=1}^{k(i)} D_{i,j}^*$ for some non-empty $D_{i,j} \subseteq S \times S$, by Lemma 5.4. Using Lemma 5.1, $R_{G^\infty}^{*(\circ)} = (\bigcup_{a \in \Sigma} R_a)^{*(\circ)} = (R_{a_{|\Sigma|}}^{*(\circ)} \circ R_{a_{|\Sigma|-1}}^{*(\circ)} \circ \dots \circ R_{a_1}^{*(\circ)})^{*(\circ)}$ corresponds to the supremum of the ascending chain $c_0 \subseteq c_1 \subseteq c_2 \subseteq \dots$, where $c_n = \bigcup_{i=0}^n (R_{a_{|\Sigma|}}^{*(\circ)} \circ R_{a_{|\Sigma|-1}}^{*(\circ)} \circ \dots \circ R_{a_1}^{*(\circ)})^{i(\circ)}$. We observe that $\text{copy}(S)^* \subseteq (R_{a_{|\Sigma|}}^{*(\circ)} \circ R_{a_{|\Sigma|-1}}^{*(\circ)} \circ \dots \circ R_{a_1}^{*(\circ)})^{i(\circ)} \in \bigcup \text{Star}(S \times S)$.

S) for each i , using the fact that $R_{a_i}^{*(o)} = \bigcup_{j=1}^{k(i)} D_{i,j}^*$ and union operation distributes over relational composition and using Lemma 5.3. Thus, $c_n \in \bigcup \text{Star}(S \times S)$. It follows from the fact that $\bigcup \text{Star}(S \times S)$ is finite that the chain $c_0 \subseteq c_1 \subseteq c_2 \subseteq \dots$ stabilizes, i.e., $\exists k \in N, (\forall m \geq k, c_k = c_m)$. Thus, $R_{G^\infty}^{*(o)} = c_k \in \bigcup \text{Star}(S \times S)$. ■

The same conclusion also holds for the co-reachability relation $(R_{G^\infty}^{-1})^{*(o)}$ of G^∞ , which is the reachability relation of $(G^\infty)^T$ (see page 76). In fact, $(R_{G^\infty}^{-1})^{*(o)} = (R_{G^\infty}^{*(o)})^{-1}$. We notice that the above proof is constructive. In general, the length k of the chain depends on the order upon which the reachability relations of G_c^∞ along σ -paths are composed. Since the complexity of computing the reachability relation relies on the length of the chain, it is of interest to develop techniques that could optimize the ordering of composition³.

Example: We shall illustrate the use of Proposition 5.1 by computing the (co-)reachability relation of a globally synchronized template G_c shown in Fig. 5.3.

Figure 5.3: A globally synchronized template G_c

In this example we have the finite state set $\{s_1, s_2, s_3, s_4\}$, the global event set $\{g_1, g_2, g_3\}$, and the private event set $\{a, b, c\}$. By Lemma 5.2, the successor relations of G_c^∞ associated with events a, b, c, g_1, g_2, g_3 are shown as follows:

³However, this problem will not be addressed in this thesis.

$$\begin{aligned}
R_a &= \text{copy}(S^*)(s_2, s_3)\text{copy}(S^*) \\
R_b &= \text{copy}(S^*)(s_2, s_4)\text{copy}(S^*) \\
R_c &= \text{copy}(S^*)(s_3, s_3)\text{copy}(S^*) \\
R_{g_1} &= (s_1, s_2)^* \\
R_{g_2} &= (s_1, s_3)^* \\
R_{g_3} &= ((s_2, s_1) \cup (s_4, s_1))^*
\end{aligned}$$

From the proof of Lemma 5.4 we then have the following:

$$\begin{aligned}
R_a^{*(\circ)} &= (\text{copy}(S) \cup (s_2, s_3))^* \\
R_b^{*(\circ)} &= (\text{copy}(S) \cup (s_2, s_4))^* \\
R_c^{*(\circ)} &= \text{copy}(S^*); \\
R_{g_1}^{*(\circ)} &= (s_1, s_2)^* \cup \text{copy}(S^*) \\
R_{g_2}^{*(\circ)} &= (s_1, s_3)^* \cup \text{copy}(S^*) \\
R_{g_3}^{*(\circ)} &= ((s_2, s_1) \cup (s_4, s_1))^* \cup \\
&\quad \text{copy}(S^*)
\end{aligned}$$

For G_c , if we compose the reachability relations of G_c^∞ along σ -paths in the order of a, b, c, g_1, g_2, g_3 , then the ascending chain in Proposition 5.1 turns out to stabilize at c_3 . Thus, the reachability relation $R_{G_c^\infty}^{*(\circ)}$ is equal to $\bigcup_{i=0}^3 (R_a^{*(\circ)} \circ R_b^{*(\circ)} \circ R_c^{*(\circ)} \circ R_{g_1}^{*(\circ)} \circ R_{g_2}^{*(\circ)} \circ R_{g_3}^{*(\circ)})^{i(\circ)}$, which is equal to

$$\begin{aligned}
&(\text{copy}(S) \cup (s_2, s_3) \cup (s_2, s_4))^* \cup ((s_2, s_1) \cup (s_4, s_1))^* \cup ((s_1, s_2) \cup (s_1, s_3) \cup (s_1, s_4))^* \cup \\
&\quad ((s_2, s_2) \cup (s_2, s_3) \cup (s_2, s_4) \cup (s_4, s_2) \cup (s_4, s_3) \cup (s_4, s_4))^*
\end{aligned}$$

Thus, the co-reachability relation $(R_{G_c^\infty}^{-1})^{*(\circ)}$ is equal to

$$\begin{aligned}
&(\text{copy}(S) \cup (s_3, s_2) \cup (s_4, s_2))^* \cup ((s_1, s_2) \cup (s_1, s_4))^* \cup ((s_2, s_1) \cup (s_3, s_1) \cup (s_4, s_1))^* \cup \\
&\quad ((s_2, s_2) \cup (s_3, s_2) \cup (s_4, s_2) \cup (s_2, s_4) \cup (s_3, s_4) \cup (s_4, s_4))^*
\end{aligned}$$

□

The reachability relation $R_{G^\infty}^{*(\circ)}$ of any globally synchronized template G is iteration closed by definition and belongs to $\bigcup \text{Star}(S \times S)$ by Proposition 5.1. In the following, we show that the reverse also holds.

Proposition 5.2. *For each iteration closed $T \in \bigcup \text{Star}(S \times S)$, there exists a globally synchronized template G with only global events such that $R_{G^\infty}^{*(\circ)} = T$.*

Proof: Suppose $T \in \bigcup \text{Star}(S \times S)$ is iteration closed. Then $T = \bigcup_{i \in I} K_i^*$ for some finite index set I , where $\emptyset \neq K_i \subseteq S \times S$ for each i . For each K_i^* , we create a global event g_i and its corresponding successor relation $E_{g_i} := K_i$. Let $G = (S, \{g_i \mid i \in I\}, \{E_{g_i} \mid i \in I\})$ be the constructed globally synchronized template. It is not difficult to see that T is realized by G , i.e., $R_{G^\infty}^{*(\circ)} = T$. Indeed $R_{g_i} = K_i^*$ and thus $R_{G^\infty}^{*(\circ)} = (\bigcup_{i \in I} R_{g_i})^{*(\circ)} = (\bigcup_{i \in I} K_i^*)^{*(\circ)} = T^{*(\circ)} = T$, since T is iteration closed. ■

Remark: It is not difficult to see that the number of global events used in the previous constructive proof may be reduced by deleting the redundant global events, where a global event g is redundant if there exists another global event g' such that $R_g^{*(\circ)} \subseteq R_{g'}^{*(\circ)}$. □

Since for any template G , $R_{G^\infty}^{*(\circ)}$ is an iteration-closed finite union of atomic star expressions over $S \times S$, there exists a template G' with only global events such that $R_{G'^\infty}^{*(\circ)} = R_{G^\infty}^{*(\circ)}$. This is stated in Corollary 5.1 below.

Corollary 5.1. *For each globally synchronized template G , there exists another globally synchronized template G' such that G' has only global events and $R_{G'^\infty}^{*(\circ)} = R_{G^\infty}^{*(\circ)}$.*

Example: Fig. 5.4 shows the corresponding G'_c that realizes $R_{G_c}^{*(\circ)}$ based on Proposition 5.2 (see G_c in Fig. 5.3). However, note that the four global events g_1, g_2, g_3, g_4 are obtained according to Proposition 5.2 and events g_1, g_2, g_3 in Fig. 5.4 do not correspond to events g_1, g_2, g_3 in Fig. 5.3. The four events g_1, g_2, g_3, g_4 and the corresponding successor relations $E_{g_1}, E_{g_2}, E_{g_3}, E_{g_4}$ are created to match the four atomic star expressions in $R_{G_c}^{*(\circ)}$, i.e., $E_{g_1} := \text{copy}(S) \cup (s_2, s_3) \cup (s_2, s_4)$, $E_{g_2} := (s_2, s_1) \cup (s_4, s_1)$, $E_{g_3} := (s_1, s_2) \cup (s_1, s_3) \cup (s_1, s_4)$ and $E_{g_4} := (s_2, s_2) \cup (s_2, s_3) \cup (s_2, s_4) \cup (s_4, s_2) \cup (s_4, s_3) \cup (s_4, s_4)$ (see the proof of Proposition 5.2 for details).

Figure 5.4: Construction of G'_c using Proposition 5.2

□

Essentially, the corollary says that the class of globally synchronized templates with only global events is as expressive as the class of globally synchronized templates (with both global and private events). The procedure in Proposition 5.2 to construct an equivalent globally synchronized template G' with only global events relies on computing the reachability relation of G^∞ . A more straightforward construction of an equivalent globally synchronized template G'' with only global events is to replace each private event transition $(s, a, s') \in \delta$, where $a \in \Sigma_p$, with a new global event $g_{(s,a,s')} \notin \Sigma_g$, by adding a loop labeled by $g_{(s,a,s')}$ for each state and replacing the transition (s, a, s') with $(s, g_{(s,a,s')}, s')$. It is easy to see that after replacement of each private event transition (s, a, s') with the global event $g_{(s,a,s')}$ in above manner, the new globally synchronized template G'' has only global events

Figure 5.5: Construction of G''_c

and $R_{G_\infty}^{*(\circ)} = R_{G''_\infty}^{*(\circ)}$. Essentially, we use a new global event to simulate each private event transition.

Example: Fig. 5.5 shows the corresponding G''_c that is equivalent to G_c using the above simulation approach. Note that we have ignored the effect of private event c in G_c because self-loop of private event transition does not affect the (co-)reachability relation. We use the global events g_a, g_b directly, instead of $g_{(s_2, a, s_3)}, g_{(s_2, b, s_4)}$, since there is only one transition for each private event a and b . $\square$

To complete the expressiveness characterization theorem, we need the following lemma, whose proof is straightforward.

Lemma 5.5. $\text{copy}(S^*) \subseteq T$ and $T \circ T = T$ iff $T^{*(\circ)} = T$.

Combining the above results, we have our main characterization theorem below.

Theorem 5.1. For $T \subseteq (S \times S)^*$, the following four statements are equivalent:

- 1) T is realizable by a globally synchronized template G with only global events.
- 2) T is realizable by a globally synchronized template G .
- 3) $T \in \bigcup \text{Star}(S \times S)$ and $T^{*(\circ)} = T$.
- 4) $T \in \bigcup \text{Star}(S \times S)$, $\text{copy}(S^*) \subseteq T$ and $T \circ T = T$.

Given an arbitrary regular relation $T \subseteq (S \times S)^*$, it is decidable whether $T \in \bigcup \text{Star}(S \times S)$. By Theorem 5.1, we have the following two immediate decidability results as corollaries.

Corollary 5.2. *It is decidable whether an arbitrary regular relation T is realizable by a globally synchronized template G .*

Corollary 5.3. *It is decidable whether two arbitrary globally synchronized templates are equivalent, i.e., whether they have the same reachability relation.*

Further applications of Theorem 5.1 will be discussed in Section 5.2.4. In the next subsection, we shall deal with dynamic systems where new modules may enter and existing modules may exit.

5.2.3 Dynamic Systems with Entrance and Exit

During the evolution of the system, some modules may enter or exit and all these modules currently in the system begin to evolve together. After the execution of some atomic transitions, it is possible that other modules would also enter or exit the system, and so on and so forth.

To stay within the framework of length-preserving relations, we introduce a new symbol $\# \notin \Sigma \cup S$ that acts as a placeholder. Then the length-preserving successor relation corresponding to the entrance of a new module in a state of S_i is simply $R_{in} = \{(x\#y, xsy) \mid x, y \in S^*, s \in S_i\}$. Clearly $R_{in}^{*(\circ)} = (\text{copy}(S) \cup \bigcup_{s \in S_i} (\#, s))^*$. Similarly, the length-preserving successor relation corresponding to the exit of a module is $R_{out} = \{(xsy, x\#y) \mid x, y \in S^*, s \in S\}$ and it is easy to see that $R_{out}^{*(\circ)} = (\text{copy}(S) \cup \bigcup_{s \in S} (s, \#))^*$. We observe that both $R_{in}^{*(\circ)}$ and $R_{out}^{*(\circ)}$ are in $\text{Star}((S \cup \{\#\}) \times (S \cup \{\#\}))$. Now each atomic star expression D^* in $R_a^{*(\circ)}$, where $a \in \Sigma$ and $\emptyset \neq D \subseteq S \times S$, becomes the atomic star expression $(D \cup (\#, \#))^*$ over $(S \cup \{\#\}) \times (S \cup \{\#\})$. Clearly, following Proposition 5.1, the reachability relation

$T_e = (\bigcup_{a \in \Sigma} R_a \cup R_{in} \cup R_{out})^{*(\circ)}$ is effectively a finite union of atomic star expressions over $(S \cup \{\#\}) \times (S \cup \{\#\})$. Essentially, the reachability relation T_e is realized by $G_e = (S \cup \{\#\}, \Sigma \cup \{\text{enter}, \text{exit}\}, \delta \cup \{(\#, \text{enter}, s) \mid s \in S_i\} \cup \{(s, \text{exit}, \#) \mid s \in S\} \cup \{(\#, a, \#) \mid a \in \Sigma_g\}$, treating the new events “enter” and “exit” as private. That is, the dynamic system generated by G is equal to G_e^∞ . The sets of initial states and final states of G_e are then $S_i \cup \{\#\}$ and $S_f \cup \{\#\}$ respectively.

5.2.4 Applications to Model Checking

It is straightforward to see that the set of reachable states $R_{G^\infty}^{*(\circ)}[S_i^*]$ of G^∞ is effectively regular. In fact, $R_{G^\infty}^{*(\circ)}[S_i^*]$ is effectively a finite union of atomic star expressions over S by Proposition 5.1, i.e., $R_{G^\infty}^{*(\circ)}[S_i^*] = \bigcup_{i \in I} K_i^*$, where $\emptyset \neq K_i \subseteq S$.

Corollary 5.4. *Given a globally synchronized template G , the set of reachable states of G^∞ is effectively a finite union of atomic star expressions over S .*

The same conclusion also holds for the set of co-reachable states $(R_{G^\infty}^{*(\circ)})^{-1}[S_f^*]$ of G^∞ .

Example: Coming back again to the template G_c , it is straightforward to see that the set of reachable states of G_c^∞ is $s_1^* \cup (s_2 \cup s_3 \cup s_4)^*$ and the set of co-reachable states of G_c^∞ is $s_1^* \cup (s_2 \cup s_4)^*$. $\square$

We have the following result concerning the model checking of safety properties⁴.

Corollary 5.5. *Given any globally synchronized template G and any set of bad states of G^∞ represented by a regular language L over S , it is decidable whether G is safe with respect to L (whether no bad state is reachable), i.e., whether the set $R_{G^\infty}^{*(\circ)}[S_i^*] \cap L$ of reachable bad states is empty.*

Many important safety properties are captured by regular languages over S , representing the sets of bad states to be avoided. Thus, the above corollary says that

⁴We mainly consider the state avoidance properties.

the problems of verifying these properties are all decidable and, furthermore, the sets of reachable bad states are effectively regular. For each safety property if the set $R_{G^\infty}^{*(\circ)}[S_i^*] \cap L$ of reachable bad states is non-empty, then one could easily calculate the minimum number of isomorphic modules that may cause the violation of the safety property, which is equal to the length of the shortest strings in $R_{G^\infty}^{*(\circ)}[S_i^*] \cap L$. Alternative ways to obtain these numbers have been shown in [100] for various deadlock and blocking freeness properties. A technique based on *weak invariant simulation* has recently been proposed for computing the set of reachable deadlocked states [116]. However, the main purpose of that technique is to identify a subclass of parameterized ring networks where deadlock-freeness is decidable. Neither the problem of computing the (co)-reachability relations nor the problem of computing the set of reachable blocked states is addressed in [116].

To illustrate the application of our result, we will show in a uniform manner in the remaining of this subsection that for various notions of deadlock and blocking, the sets of reachable bad states are effectively regular. In view of the effective regularity of the set of reachable states, we only need to show that the sets of bad states are effectively regular, summarized by the following proposition.

Proposition 5.3. *Given any globally synchronized template G , the sets of system-deadlock, system-blocking, component-deadlock, component-blocking, universal-blocking states of G^∞ are effectively regular.*

We give the definitions and provide the constructions that prove the above proposition.

System Deadlock

A state $w \in S^*$ of G^∞ is said to be system-deadlocked if $R_a[w] = \emptyset$ for any $a \in \Sigma$. The set of system-deadlocked states is denoted by L_{SD} . G is said to be system-deadlocked if there exists a reachable state of G^∞ that is system-deadlocked. G is

said to be system-deadlock free if it is not system-deadlocked. It is not difficult to see that the set $L_{NSD} = L_{SD}^c$ of non-system-deadlocked states is a regular language over S , which is shown below.

$$L_{NSD} = S^*(\bigcup_{a \in \Sigma_p} S(a))S^* \cup (\bigcup_{a \in \Sigma_g} S(a)^*)$$

Intuitively, each state in $S^*(\bigcup_{a \in \Sigma_p} S(a))S^*$ is able to execute at least an private event and each state in $\bigcup_{a \in \Sigma_g} S(a)^*$ is able to execute at least a global event.

Example: For G_c , the set of system-deadlocked states of G_c^∞ is $L_{SD} = L_{NSD}^c = (S^*(s_2 \cup s_3)S^* \cup s_1^* \cup (s_2 \cup s_4)^*)^c$, which is equal to $(s_1 \cup s_4)^*s_1(s_1 \cup s_4)^*s_4(s_1 \cup s_4)^* \cup (s_1 \cup s_4)^*s_4(s_1 \cup s_4)^*s_1(s_1 \cup s_4)^*$. Then the set of reachable system-deadlocked states is $(s_1^* \cup (s_2 \cup s_3 \cup s_4)^*) \cap L_{SD} = \emptyset$. In other words, G_c is system-deadlock free. $\square$

System Blocking

A state w of G^∞ is said to be system-blocked if $R_{G^\infty}^{*(\circ)}[w] \cap S_f^* = \emptyset$. The set of system-blocked states is denoted by L_{SB} . G is said to be system-blocked if there exists a reachable state of G^∞ that is system-blocked. G is said to be system-blocking free if G is not system-blocked. Since the set of co-reachable, or equivalently, the set of non-system-blocked states of G^∞ is $L_{SB}^c = L_{NSB} = (R_{G^\infty}^{-1})^{*(\circ)}[S_f^*]$, the problem of verifying system-blocking freeness reduces to the problem of checking whether $R_{G^\infty}^{*(\circ)}[S_i^*] \subseteq (R_{G^\infty}^{-1})^{*(\circ)}[S_f^*]$.

Example: For the template G_c , the set of system-blocked states of G_c^∞ is simply $L_{SB} = L_{NSB}^c = (s_1^* \cup (s_2 \cup s_4)^*)^c$. Thus, $(s_1^* \cup (s_2 \cup s_3 \cup s_4)^*) \cap (s_1^* \cup (s_2 \cup s_4)^*)^c = (s_2 \cup s_3 \cup s_4)^*s_3(s_2 \cup s_3 \cup s_4)^*$ is the set of reachable system-blocked states. The minimum number of isomorphic modules that may cause system-blocking is 1, when the only module executes g_2 and goes from s_1 into the blocking state s_3 . $\square$

Component Deadlock

A state $w = s_1 s_2 \dots s_n$ of G^∞ is said to be *component-deadlocked* if there exists $i \in [1, n]$ such that $R_a[s_i] = \emptyset$ for any $a \in \Sigma_p$ and $R_a[R_{G^\infty}^{*(\circ)}[w]] = \emptyset$ for any $a \in \Sigma_g$. Intuitively, a state is component-deadlocked if there exists a module that is not able to execute any event in any future evolution of the system. The set of component-deadlocked states of G^∞ is denoted by L_{CD} . G is said to be component-deadlocked if there exists a reachable state of G^∞ that is component-deadlocked. G is said to be component-deadlock free if G is not component-deadlocked. Due to symmetry, it suffices to consider only the set of component-deadlocked states with the first module being deadlocked, denoted by L_{FCD} , and then $L_{CD} = [L_{FCD}]_\sqcup$. It is not difficult to see that $L_{FCD} = (\bigcup_{s \in S_{\Sigma_G}} s(\bigcup_{g \in \Sigma(s)} (\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[S(g)^*])^c) \cup S^\perp S^*$. In fact, there are two cases where the first module is deadlocked. The first case is when the first module could only execute global events, and the rest of the modules could not execute any of these global events, even after they execute a finite sequence, including zero sequence, of private events. The second case is when the first module is in a deadlock state of G . Now it is easy to see that $L_{CD} =$

$$(\bigcup_{s \in S_{\Sigma_G}} (\bigcup_{g \in \Sigma(s)} (\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[S(g)^*])^c \sqcup s) \cup S^* S_d S^*$$

is effectively regular since the shuffle of two regular languages is regular. Note that $(\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}$ is also a finite union of atomic star expressions over $S \times S$.

Example: For G_c , the set of first-component-deadlocked states of G_c^∞ is $L_{FCD} = s_1((\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[S(g_1)^*] \cup (\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[S(g_2)^*])^c \cup s_4((\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[S(g_3)^*])^c$. It could be easily shown that for G_c , $(\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)} = (\text{copy}(S) \cup (s_3, s_2) \cup (s_4, s_2))^*$. Since $S(g_1) = S(g_2) = \{s_1\}$ and $S(g_3) = \{s_2, s_4\}$, L_{FCD} is

$$s_1((\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[s_1^*] \cup (\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[s_1^*])^c \cup s_4((\bigcup_{a \in \Sigma_p} R_a^{-1})^{*(\circ)}[(s_2 \cup s_4)^*])^c = s_1(s_1^*)^c \cup s_4((s_2 \cup s_4)^*)^c.$$

Thus, the set of reachable first-component-deadlocked states is $(s_1^* \cup (s_2 \cup s_3 \cup s_4)^*) \cap (s_1(s_1^*)^c \cup s_4((s_2 \cup s_4)^*)^c) = s_4(s_2 \cup s_3 \cup s_4)^* s_3(s_2 \cup s_3 \cup s_4)^*$. Then $(s_2 \cup s_3 \cup s_4)^* s_3(s_2 \cup s_3 \cup s_4)^* \sqcup s_4 = (s_2 \cup s_3 \cup s_4)^* s_3(s_2 \cup s_3 \cup s_4)^* s_4(s_2 \cup s_3 \cup s_4)^* \cup (s_2 \cup s_3 \cup s_4)^* s_4(s_2 \cup s_3 \cup s_4)^* s_3(s_2 \cup s_3 \cup s_4)^*$ is the set of reachable component-deadlocked states, which is effectively regular, and the minimum number of isomorphic modules that may cause component-deadlock is 2. $\square$

Component Blocking

A state w of G^∞ is said to be *component-blocked* if there exists a module that is not able to enter any one of its final states S_f in any future evolution of the system. The set of component-blocked states of G^∞ is denoted by L_{CB} . G is said to be component-blocked if there exists a reachable state of G^∞ that is component-blocked. G is said to be component-blocking free if G is not component-blocked. Without loss of generality, we focus on the set of states where the first module is component-blocked. A state w is first-component-blocked if $R_{G^\infty}^{*(o)}[w] \cap (S_f S^* \cup \epsilon) = \emptyset$. To verify first-component-blocking freeness, it suffices to check whether every reachable state could reach a state whose first module is in its final state. That is, we only need to check whether $R_{G^\infty}^{*(o)}[S_i^*] \subseteq (R_{G^\infty}^{-1})^{*(o)}[S_f S^* \cup \epsilon]$. The set of first-component-blocked states is $((R_{G^\infty}^{-1})^{*(o)}[S_f S^* \cup \epsilon])^c$. Since $R_{G^\infty}^{*(o)}$ is a finite union of atomic star expressions over $S \times S$, it is not difficult to see that $(R_{G^\infty}^{-1})^{*(o)}[S_f S^*]$ is a finite union of expressions $S_1 S_2^*$'s, where $\emptyset \neq S_1, S_2 \subseteq S$. We note that $(R_{G^\infty}^{-1})^{*(o)}[S_f S^* \cup \epsilon] = (R_{G^\infty}^{-1})^{*(o)}[S_f S^*] \cup \epsilon$. The set of component-blocked states is $[((R_{G^\infty}^{-1})^{*(o)}[S_f S^* \cup \epsilon])^c]_\sqcup$.

Example: For the template G_c , the set of first-component-blocked states of G_c^∞ is $((R_{G_c^\infty}^{-1})^{*(o)}[s_1 S^* \cup \epsilon])^c = (s_1 S^* \cup (s_2 \cup s_4)(s_2 \cup s_4)^* \cup \epsilon)^c$. Thus, the set of reachable first-component-blocked states is $(s_1^* \cup (s_2 \cup s_3 \cup s_4)^*) \cap (\epsilon \cup s_1 S^* \cup (s_2 \cup s_4)^+)^c = (s_2 \cup s_3 \cup s_4)^+ - (s_2 \cup s_4)^+ = (s_2 \cup s_3 \cup s_4)^* s_3(s_2 \cup s_3 \cup s_4)^*$. The minimum number of isomorphic

modules that may cause component-blocking is 1. $(s_2 \cup s_3 \cup s_4)^* s_3 (s_2 \cup s_3 \cup s_4)^*$ is in fact also the set of reachable component-blocked states since it is closed under commutation. $\square$

Universal Blocking

A state w of G^∞ is said to be *universally-blocked* if no module is able to enter any one of its final states S_f in any future evolution of the system. The set of universally-blocked states of G^∞ is denoted by L_{UB} . G is said to be universally-blocked if there exists a reachable state of G^∞ that is universally-blocked. G is said to be universal-blocking free if G is not universally-blocked. The problem of verifying universal-blocking freeness reduces to the problem of checking whether $R_{G^\infty}^{*(\circ)}[S_i^*] \subseteq (R_{G^\infty}^{-1})^{*(\circ)}[S^* S_f S^* \cup \epsilon]$. The set of universally-blocked states is $L_{UB} = ((R_{G^\infty}^{-1})^{*(\circ)}[S^* S_f S^* \cup \epsilon])^c$. It is straightforward to see that $(R_{G^\infty}^{-1})^{*(\circ)}[S^* S_f S^*]$ is a finite union of expressions $S_1^* S_2 S_1^*$'s, where $\emptyset \neq S_1, S_2 \subseteq S$. We note that $(R_{G^\infty}^{-1})^{*(\circ)}[S^* S_f S^* \cup \epsilon] = (R_{G^\infty}^{-1})^{*(\circ)}[S^* S_f S^*] \cup \epsilon$.

Example: For the template G_c , the set of universally-blocked states of G_c^∞ is simply $((R_{G_c}^{-1})^{*(\circ)}[S^* s_1 S^* \cup \epsilon])^c$, which is equal to $(S^* s_1 S^* \cup (s_2 \cup s_4)^* (s_2 \cup s_4) (s_2 \cup s_4)^* \cup \epsilon)^c$. Thus, the set of reachable universally-blocked states is $(s_1^* \cup (s_2 \cup s_3 \cup s_4)^*) \cap (\epsilon \cup S^* s_1 S^* \cup (s_2 \cup s_4)^+)^c = (s_2 \cup s_3 \cup s_4)^+ - (s_2 \cup s_4)^+ = (s_2 \cup s_3 \cup s_4)^* s_3 (s_2 \cup s_3 \cup s_4)^*$. The minimum number of isomorphic modules that may cause universal-blocking is 1. $\square$

5.3 Regular Supervisor Synthesis

In this section, we shall develop a semi-algorithm in the framework of regular model checking for computing the supremal non-blocking supervisor that enforces the state avoidance property for globally synchronized templates. Let $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$

be the disjoint union of the set Σ_c of controllable events and the set Σ_{uc} of uncontrollable events. We assume that any private event is controllable if and only if all of its instantiations are controllable. The general idea is very similar to the centralized supervisor synthesis procedure for finite state systems [13], that is,

1. those states that could lead to the set of bad states (including the set of blocked states) under a sequence of uncontrollable events are pruned
2. while there exist blocked states in the resulting transition system, repeat the following: those states that could lead to the set of new blocked states under a sequence of uncontrollable events in the new transition system are pruned

We need the definition of length-preserving *cross product* of automata to iteratively update the successor relation of the controlled system.

Definition 5.1 (Length-Preserving Cross Product). *The length-preserving cross product of automata $G_1 = (S_1, \Sigma_1, \delta_1, S_{1,i}, S_{1,f})$ and $G_2 = (S_2, \Sigma_2, \delta_2, S_{2,i}, S_{2,f})$ is the transducer $G_1 \times G_2 = (S_1 \times S_2, \Sigma_1 \times \Sigma_2, \delta_1 \times \delta_2, S_{1,i} \times S_{2,i}, S_{1,f} \times S_{2,f})$, where $\delta_1 \times \delta_2$ is the transition relation such that for each $(q_1, q_2), (q'_1, q'_2) \in Q_1 \times Q_2$ and each $(\sigma_1, \sigma_2) \in \Sigma_1 \times \Sigma_2$, $((q_1, q_2), (\sigma_1, \sigma_2), (q'_1, q'_2)) \in \delta_1 \times \delta_2$ iff $(q_1, \sigma_1, q'_1) \in \delta_1$ and $(q_2, \sigma_2, q'_2) \in \delta_2$.*

The relationship between the languages recognized by G_1, G_2 and their length-preserving cross product $G = G_1 \times G_2$ is shown below.

Lemma 5.6. $L_m(G) = \{(w_1, w_2) \in L_m(G_1) \times L_m(G_2) \mid |w_1| = |w_2|\}$.

Example: As an illustration, Fig. 5.6 shows the length-preserving cross product of automata G_1 and G_2 . Here $L_m(G_1) = (ab)^*$, $L_m(G_2) = (cd)^*$ and $L_m(G_1 \times G_2) = (ab, cd)^*$. □

An implementation of the symbolic supervisor synthesis algorithm in the framework of regular model checking is presented in the next subsection.

Figure 5.6: Illustration for the length-preserving cross product operation

Implementation

Given any globally synchronized template G and any regular language $L \subseteq S^*$ representing the set of bad states of G^∞ to be avoided, we keep track of the set P_k of states that need to be pruned by the supervisor and the resultant successor relation R_k in each step. The total set of states to be pruned by the supremal non-blocking supervisor is $P := \bigcup_{k \geq 1} P_k$. Thus, the parameterized supervisor synthesis problem has a solution iff $P \cap S_i^* = \emptyset$. It follows that only those transitions in $R_{G^\infty} \cap (P^c \times P)$ need to be disabled by the supremal non-blocking supervisor, if $P \cap S_i^* = \emptyset$. Since the system successor relation R_{G^∞} of G^∞ is available (see page 79), the main task of the synthesis problem reduces to computing P . A pseudo-code implementation of regular supervisor synthesis for globally synchronized templates is shown in Algorithm 5.1.

Algorithm 5.1 Regular Supervisor Synthesis

1. **Input:** template G and a regular language $L \subseteq S^*$
 2. Let $P_1 := (R_{G^\infty}^{*(\circ)}[S_i^*])^c \cup (R_{\Sigma_{uc}}^{*(\circ)})^{-1}[L \cup L_{SB}]$;
 3. If $P_1 \cap S_i^* \neq \emptyset$, then **Output:** no solution;
 4. Let $R_1 := R_{G^\infty} \cap (P_1^c \times P_1^c)$;
 5. Set $k := 1$;
 6. **While** $((R_k^{-1})^{*(\circ)}[P_k^c \cap S_f^*])^c \cap P_k^c \neq \emptyset$
 - {
 - (a) Let $P_{k+1} := P_k \cup (R_{\Sigma_{uc}}^{*(\circ)})^{-1}[(R_k^{-1})^{*(\circ)}[P_k^c \cap S_f^*])^c \cap P_k^c \cup (R_k^{*(\circ)}[S_i^*])^c$;
 - (b) If $P_{k+1} \cap S_i^* \neq \emptyset$, then **Output:** no solution;
 - (c) Let $R_{k+1} := R_{G^\infty} \cap (P_{k+1}^c \times P_{k+1}^c)$;
 - (d) Let $k := k + 1$;
 - }
 7. **Output:** $P := \bigcup_{k \geq 1} P_k$
-

In Algorithm 5.1, $R_{\Sigma_{uc}} := \bigcup_{a \in \Sigma_{uc}} R_a$ is the system successor relation associated with uncontrollable events and $(R_{\Sigma_{uc}}^{*(o)})^{-1}$ is the co-reachability relation along paths labeled by uncontrollable events. L_{SB} is the set of system-blocked states (see Section 5.2.4). Thus, $(R_{\Sigma_{uc}}^{*(o)})^{-1}[L \cup L_{SB}]$ is the set of states that can lead to L or the set L_{SB} of system blocked states under a sequence of uncontrollable events. $(R_{G^\infty}^{*(o)}[S_i^*])^c$ is the set of unreachable states before any pruning. $R_{\Sigma_{uc}}^{*(o)}, R_{G^\infty}^{*(o)}$ are regular relations over $S \times S$ and L_{SB} is a regular language over S . Thus, P_1 is a regular language over S . The resultant successor relation after pruning P_1 is $R_1 = R_{G^\infty} \cap (P_1^c \times P_1^c)$, which is again a regular relation over $S \times S$. $((R_k^{-1})^{*(o)}[P_k^c \cap S_f^*])^c \cap P_k^c$ is the set of new blocked states in the k -th iteration after pruning P_k . The set P_{k+1} of bad states in the $(k+1)$ -th iteration is the union of the set P_k of bad states in the k -th iteration, the set $(R_{\Sigma_{uc}}^{*(o)})^{-1}[(R_k^{-1})^{*(o)}[P_k^c \cap S_f^*])^c \cap P_k^c$ of states that can lead to the new blocked states under a sequence of uncontrollable events and the set $(R_k^{*(o)}[S_i^*])^c$ of unreachable states in the k -th iteration. Some remarks have to be made regarding the computations involved in Algorithm 5.1.

Remark: The implementation of Algorithm 5.1 is not yet amenable to computation, since the updated reachability relations $R_k^{*(o)}$'s have to be computed in step 6) and it is still unknown whether $R_k^{*(o)}$'s are regular relations. A method to circumvent this difficulty is to use regular over-approximation techniques developed in the theory of regular model checking [92], [102], [108], [110] and [111] to over-approximate $R_k^{*(o)}$'s with regular relations. However, if a strict over-approximation is computed, then the resulting non-blocking supervisor may not be supremal any more. Also, there is no guarantee that Algorithm 5.1 will ever terminate. $\square$

Remark: Algorithm 5.1 is easily amenable to generalizations. By replacing L_{SB} with $L_{CB} \cup L_{SB}$ in Algorithm 5.1, the resultant closed-loop system is guaranteed to be component-blocking free as well as system-blocking free. It is also applica-

ble to dynamic systems with entrance and exit, by treating both entrance and exit events as private (see pages 87, 88). Algorithm 5.1 is also generally applicable to other classes of parameterized systems. Again, regular over-approximation techniques have to be used and termination of the algorithm is not guaranteed. $\square$

5.4 Discussions

In this chapter, we have adopted the regular model checking framework to study the symbolic reachability relations of globally synchronized templates. The main result obtained is that the symbolic reachability relations of these templates correspond exactly to the class of iteration-closed finite unions of atomic star expressions. The main novelty is that, instead of simply showing the regularity of the symbolic reachability relations, the result is sharpened considerably so that for each regular relation it becomes possible to check for its realizability with respect to globally synchronized templates. The result also has immediate applications to the model checking problems, including the verification of deadlock and blocking freeness, and is potentially useful for comparing different classes of templates. A parameterized supervisor synthesis semi-algorithm that enforces state avoidance and non-blocking properties is also developed in the framework of regular model checking.

There are several grand challenges left, however. The proposed program of characterizing the expressive power of classes of parameterized systems using symbolic reachability relations is rather difficult to extend. For most classes of parameterized systems, the symbolic reachability relations are neither computable nor regular. It seems that more fundamental study on non-regular relations and advanced techniques need to be carried out and developed in order to make serious progress in this direction. As for the parameterized supervisor synthesis semi-algorithm, it is

still unknown whether it terminates for the globally synchronized templates. Since in general the semi-algorithm will not terminate for most classes of parameterized systems, developing sound acceleration techniques to ensure finite termination are necessary.

Chapter 6

Conclusions

In this thesis, we have studied several important problems in the area of decentralized and parameterized supervisor synthesis. The main achievements of this thesis are listed as follows.

1. We have brought up a new idea for obtaining a language-theoretic characterization of the solvability of the decentralized realization problem. From the obtained characterization result, the close relationship among the decentralized realization, supervisor decomposition and language decomposition problems becomes immediately clear (see pages 26, 27, 28). It is then straightforward to obtain complexity-theoretic lower bound proofs for the supervisor decomposition problem and the decentralized realization problem, by reductions from the language decomposition problem. The main advantage of this approach is the simplicity, where the definition of a supervisor is independent from that of a plant. We also have proposed the notion of reducible distributions as a main tool towards reducing the complexity of language decomposability verification. In particular, it easily supports parallel verification and can be used to explore the structural redundancy of distributions, as far as language decomposability is concerned.

2. An undecidable fragment and a rather limited decidable fragment of the distributed supervisor synthesis problem have been identified based on the distributed control architecture. An undecidable fragment of a language based parameterized supervisor synthesis problem has also been studied for the globally synchronized templates. The approaches undertaken all involve considering the existence problem of non-empty decomposable sublanguages and the variations. Another distinctive feature of this work is that the boundary between the decidability and undecidability of the synthesis problem is based on the structure of the control architectures. This seems to be reasonable since it is the distributed control architecture that may give rise to partial observation, which is the cause of the undecidability.
3. The symbolic reachability relations of the globally synchronized templates have been studied in the framework of regular model checking. The result has immediate applications to the model checking problems, including the verification of deadlock and blocking freeness, and is also potentially useful for comparing different classes of templates. A distinctive feature of this work is that we propose to use the symbolic reachability relations to characterize the expressive power of parameterized systems. In particular, we are able to conclude that the class of globally synchronized templates is as expressive as the class of globally synchronized templates with only global events, based on their symbolic reachability relations. Finally, a parameterized supervisor synthesis technique is developed in the framework of regular model checking.

Both the decentralized and parameterized supervisor synthesis problems are still far from being fully understood. It appears that the research work could be extended in the following directions.

- One direction for future research is to fully work out the details for a charac-

terization of the solvability of the decentralized realization problem without any assumption. This requires a more general language equational characterization of finite state supervisors. Currently, we need to assume that each local supervisor can observe each event that it can disable.

- Another interesting direction is to further explore other potential reductions of distributions. In particular, it is of interest to consider the decision problem of determining the reducibility of an arbitrary distribution and characterize the reducibility of an arbitrary distribution. This would help us better understand the boundary between the tractability and intractability of the language decomposition problem, which is closely related to the supervisor decomposition and decentralized realization problems. It is also practically useful to develop algorithms that could compute (optimal) reductions for any given distribution, if they exist.
- In this work, the definition of a supervisor is independent from that of a plant. Thus, the supervisor decomposition algorithm developed in this work is quite different from the supervisor localization algorithm developed in [20] (Note that the supervisor localization algorithm developed in [20] allows the subsets of observable events to be increased and thus does not consider fixed control constraints). More fundamental study on the supervisor decomposition problem could be carried out. In particular, it is of interest to develop techniques to reduce the complexity of supervisor decomposability verification based on the structures of the distributed control architectures. If the global supervisor is not decomposable with respect to a distributed control architecture, then the subsets of controllable events and observable events need to be increased for the local supervisors to make the global supervisor decomposable. Efficient heuristics need to be developed for this purpose. In the future work, we would

like to apply the (optimized) supervisor decomposition algorithm (plus the heuristics mentioned above) to the benchmark examples of [20] to study the applicability of the algorithm and compare its performance, in terms of the computational costs, with that of the supervisor localization algorithm.

- Clearly, the most fundamental but challenging task in considering the decentralized supervisor synthesis problem is to obtain a characterization of its decidability based on the decentralized control architectures. In this work we only have a sufficient condition for the undecidability and a rather limited decidability result.
- For the regular supervisor synthesis technique developed in this thesis, there are still several tasks and challenges left. Firstly, regular over-approximation techniques need to be used to carry out the computation using finite state automata and transducers. Experimental validations of our approach are planned to be conducted in the future work and acceleration techniques such as widening [92] and inference of regular languages [108] are planned to be used. Since the parameterized supervisor synthesis procedure is generally applicable to many classes of templates, we also would like to perform experimental validations on the broadcast and the rendezvous templates, which can be used to model mutual exclusion protocols and the dining philosopher problems. Secondly, the parameterized supervisor synthesis procedure provided in this work is centralized and the global supervisor is assumed to have the full observation. It is of interest to relax this assumption and extend the synthesis procedure to template based supervisor synthesis.

AUTHOR'S PUBLICATIONS

1. L. Lin, A. Stefanescu, W. Wang, R. Su. “Symbolic reachability analysis of globally synchronized templates”, submitted to *Automatica*.
2. L. Lin, A. Stefanescu, R. Su. “On distributed and parameterized supervisor synthesis problems”, to appear in *IEEE Transactions on Automatic Control*, 2016.
3. S. Sami, L. Lin, A. R. Shehabinia, R. Su, C. S. Chong, S. Jeon. “Discrete-event based vehicle dispatching and scheduling in multicommodity transportation”, *In Proc. of 13th International Conference on Control, Automation, Robotics, and Vision*, pp. 169-174, 2014.
4. A. R. Shehabinia, L. Lin, R. Su, C. S. Chong. “Supervisory control for a class of discrete event systems for optimal throughput”, *the 12th International Workshop on Discrete Event Systems*, pp. 201-207, 2014.
5. L. Lin, A. R. Shehabinia, R. Su, B. Brandin. “Automaton-based timed supervisory control for operational planning and scheduling under multiple hard and soft job deadlines”, *IEEE 53rd Annual Conference on Decision and Control*, pp. 5943-5948, 2014.
6. L. Lin, A. Stefanescu, R. Su, W. Wang, A. R. Shehabinia. “Towards decentralized synthesis: decomposable sublanguage and joint observability problems”, *American Control Conference*, pp. 2047-2052, 2014.

7. R. Su, L. Lin. “Synthesis of control protocols for multi-agent systems with similar actions”, *In Proc. of 52th Annual Conference on Decision and Control*, pp. 6986-6991, 2013.
8. W. Wang, R. Su, L. Lin. “On analysis of deadlock and blocking freeness in isomorphic module systems”, *American Control Conference*, pp. 923-928, 2013.
9. L. Lin, R. Su, A. Stefanescu. “Remarks on the difficulty of top-down supervisor synthesis”, *In Proc. of 12th International Conference on Control, Automation, Robotics, and Vision*, pp. 270-275, 2012.

Bibliography

- [1] K. Rohloff, T. S. Yoo, S. Lafortune. “Deciding co-observability is PSPACE-complete”, *IEEE Transactions on Automatic Control*, 48(11): 1995-1999, 2003.
- [2] K. Rohloff, S. Lafortune. “PSPACE-completeness of modular supervisory control problems”, *Discrete Event Dynamic Systems*, 15(2): 145-167, 2005.
- [3] S. Kashyap, V. K. Garg. “Intractability results in predicate detection”, *Information Processing Letters*, 94(6): 277-282, 2005.
- [4] E. M. Clarke, E. A. Emerson, A. P. Sistla. “Automatic verification of finite-state concurrent systems using temporal logic specifications”, *ACM Transactions on Programming Languages and Systems*, 8(2): 244-263, 1986.
- [5] P. Gohari, W. M. Wonham. “On the complexity of supervisory control design in the RW framework”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(5): 643-652, 2000.
- [6] A. Stefanescu, K. Heljanko. “Complexity results for checking distributed implementability”, In *Proc. of the 5th International Conference on Application of Concurrency to System Design*, pp. 78-87. IEEE Computer Society, 2005.
- [7] J. G. Thistle. “Undecidability in decentralized supervision”, *Systems & Control Letters*, 54(5): 503-509, 2005.

- [8] S. Tripakis. “Undecidable problems of decentralized observation and control on regular languages”, *Information Processing Letters*, 90(1): 21-28, 2004.
- [9] A. Arnold, A. Vincent, I. Walukiewicz. “Games for synthesis of controllers with partial observation”, *Theoretical Computer Science*, 303(1): 7-34, 2003.
- [10] A. Pnueli, R. Rosner, “Distributed reactive systems are hard to synthesize”, 31st Annual Symposium on Foundations of Computer Science, Vol. 2, pp. 746-757, 1990.
- [11] P. Madhusudan, P. S. Thiagarajan. “A decidable class of asynchronous distributed controllers” In *Proc. of the 13th International Conference on Concurrency Theory*, *Lecture Notes in Computer Science*, pp. 145-160, 2002.
- [12] S. Schewe. “Distributed synthesis is simply undecidable”, *Information Processing Letters*, 114(4): 203-207, 2014.
- [13] W. M. Wonham. *Supervisory control of discrete-event systems*, Systems Control Group, Dept. of ECE, University of Toronto, Toronto, Canada, 2010.
- [14] P. J. Ramadge, W. M. Wonham. “Supervisory control of discrete event processes”, *Feedback Control of Linear and Nonlinear Systems*, *Lecture Notes in Control and Information Sciences*, Vol. 39, pp. 202-214, 1982.
- [15] P. J. Ramadge, W. M. Wonham. “Supervision of discrete event processes”, *IEEE Conference on Decision and Control*, pp. 1228-1229, 1982.
- [16] P. J. Ramadge, W. M. Wonham. “Supervisory control of a class of discrete event processes”, *SIAM Journal on Control and Optimization*, 25(1): 206-230, 1987.

- [17] W. M. Wonham, P. J. Ramadge. “On the supremal controllable sublanguage of a given language”, *SIAM Journal on Control and Optimization*, 25(3): 637-659, 1987.
- [18] C. Cassandras, S. Lafortune. *Introduction to Discrete Event Systems*, Springer, 2008.
- [19] R. Kumar, V. K. Garg. *Modeling and control of logical discrete event systems*, Springer, 1995.
- [20] K. Cai, W. M. Wonham. “Supervisor localization: a top-down approach to distributed control of discrete-event systems”, *IEEE Transactions on Automatic Control*, 55(3): 605-618, 2010.
- [21] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, G. F. Franklin. “Supervisory control of a rapid thermal multiprocessor”, *IEEE Transactions on Automatic Control*, 38(7): 1040-1059, 1993.
- [22] K. C. Wong, J. G. Thistle, R. P. Malham, H. H. Hoang. “Supervisory Control of Distributed Systems: Conflict Resolution”, *Discrete Event Dynamic Systems*, 10(1-2): 131-186, 2000.
- [23] R. Cieslak, C. Desclaux, A. S. Fawaz, P. Varaiya. “Supervisory control of discrete-event processes with partial observations”, *IEEE Transactions on Automatic Control*, 33(3): 249-260, 1988.
- [24] S. Lafortune, E. Wong. “A state model for the concurrency control problem in database management systems”, *IEEE Conference on Decision and Control*, pp. 441-442, 1985.

- [25] M. H. Zibaeenejad, J. G. Thistle. “Dependency graph: An algorithm for analysis of generalized parameterized networks”, American Control Conference, pp. 696-702, 2015.
- [26] J. Goryca, R. C. Hill. “Formal synthesis of supervisory control software for multiple robot systems”, American Control Conference, pp. 125-131, 2013.
- [27] E. Roszkowska. “Decentralized motion-coordination policy for cooperative mobile robots”, IFAC Workshop on Discrete-Event System Design, pp. 364-369, 2008.
- [28] K. T. Seow, M. T. Pham, C. Ma, M. Yokoo. “Coordination planning: Applying control synthesis methods for a class of distributed agents”, IEEE Transactions on Control Systems Technology, 17(2): 405-415, 2009.
- [29] F. Lin, W. M. Wonham. “On observability of discrete-event systems”, Information Sciences, 44(3): 173-198, 1988.
- [30] F. Lin, W. M. Wonham. “Decentralized supervisory control of discrete-event systems”, Information Sciences, 44(3): 199-224, 1988.
- [31] K. Rudie, W. M. Wonham. “Think globally, act locally: Decentralized supervisory control”, IEEE Transactions on Automatic Control, 37(11): 1692-1708, 1992.
- [32] J. E. Hopcroft, J. D. Ullman. *Introduction to automata theory, languages, and computation*, Addison-Wesley, Reading, Massachusetts, 1979.
- [33] J. Komenda, J. H. van Schuppen, B. Gaudin, H. Marchand. “Modular supervisory control with general indecomposable specification languages”, IEEE CDC-ECC, pp. 3474-3479, 2005.

- [34] S. Pinchinat, S. Riedweg. “You can always compute maximally permissive controllers under partial observation when they exist”, American Control Conference, pp. 2287-2292, 2005.
- [35] M. H. de Queiroz, J. E. R. Cury. “Modular supervisory control of large scale discrete event systems”, Discrete Event Systems, The Springer International Series in Engineering and Computer Science, Vol. 569, pp. 103-110, 2000.
- [36] A. Overkamp, J. H. van Schuppen, “Maximal solutions in decentralized supervisory control”, SIAM Journal on Control and Optimization, 39(2): 492-511, 2000.
- [37] J. Komenda, H. Marchand, S. Pinchinat. “A constructive and modular approach to decentralized supervisory control problems”, IFAC Workshop on Discrete-Event System Design, pp. 111-116, 2006.
- [38] K. Cai, R. Zhang, W. M. Wonham. “Relative observability of discrete-event systems and its supremal sublanguages”, IEEE Transactions on Automatic Control, 60(3): 659-670, 2014.
- [39] R. Su and J. G. Thistle. “A distributed supervisor synthesis approach based on weak bisimulation, IFAC Workshop on Discrete-Event System Design, pp. 64-69, 2006.
- [40] L. Feng. “Computationally efficient supervisor design in discrete-event systems”, Ph.D. dissertation, University of Toronto, 2007.
- [41] S. Pinchinat, S. Riedweg. “A decidable class of problems for control under partial observation”, Information Processing Letters, 95(4): 454-460, 2005.

- [42] P. Krishnan. “Decomposing controllers into non-conflicting distributed controllers”, *Theoretical Aspects of Computing, Lecture Notes in Computer Science*, Vol. 3407, pp. 511-526, 2004.
- [43] A. Arnold and I. Walukiewicz. “Nondeterministic controllers of nondeterministic processes”, *Logic and Automata, Texts in Logic and Games*, Amsterdam University Press, 2007.
- [44] L. E. Holloway, B. H. Krogh. “Synthesis of feedback control logic for a class of controlled Petri nets”, *IEEE Transactions on Automatic Control*, 35(5): 514-523, 1990.
- [45] B. Lacerda, P. U. Lima. “Designing petri net supervisors for multi-agent systems from LTL specifications (Extended Abstract)”, In *Proceedings of 10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.
- [46] J. G. Thistle, W. M. Wonham. “Supervision of infinite behavior of discrete-event systems”, *SIAM Journal on Control and Optimization*, 32(4): 1098-1113, 1994.
- [47] J. G. Thistle, W. M. Wonham. “Control of infinite behavior of finite automata”, *SIAM Journal on Control and Optimization*, 32(4), 1075-1097, 1994.
- [48] S. Riedweg, S. Pinchinat. “Quantified mu-calculus for control synthesis”, *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science*, Vol. 2747, pp 642-651, 2003.
- [49] S. Lafortune. “On decentralized and distributed control of partially-observed discrete event systems”, *Lecture Notes in Control and Information Sciences*, Vol. 353, pp. 171-184, 2007.

- [50] N. R. Sandell, P. Varaiya, M. Athans, M. G. Safonov. “Survey of decentralized control methods for large scale systems”. IEEE Transactions on Automatic Control, 23(2): 108-128, 1978.
- [51] D. D. Siljak. *Decentralized control of complex systems*. Dover Books on Electrical Engineering, 2011.
- [52] T. Basar, G. J. Olsder. *Dynamic noncooperative game theory*, SIAM Classics in Applied Mathematics, 1999.
- [53] A. Bergeron. “A unified approach to control problems in discrete event processes”, RAIRO-Theoretical Informatics and Applications, 27(6): 555-573, 1993.
- [54] S. Jiang, 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, 30(5): 653-660, 2000.
- [55] T. S. Yoo, S. Lafortune. “A general architecture for decentralized supervisory control of discrete-event systems”, Discrete Event Dynamic Systems, 12(3): 335-377, 2002.
- [56] S. Graf, D. Peled, S. Quinton. “Achieving distributed control through model checking”, Formal Methods in System Design, 40(2): 263-281, 2012.
- [57] Y. Willner, M. Heymann. “Supervisory control of concurrent discrete event systems”, Int. Journal of Control, 54(5): 1143-1169, 1991.
- [58] J. Komenda, T. Masopust, J. H. van Schuppen. “On conditional decomposability”, Systems & Control Letters, 61(12): 1260-1268, 2012.
- [59] K. R. Apt, D. C. Kozen. “Limits for automatic verification of finite-state concurrent systems”, Information Processing Letters, 22(6): 307-309, 1986.

- [60] I. Suzuki. “Proving properties of a ring of finite-state machines”, *Information Processing Letters*, 28(4): 213-214, 1988.
- [61] J. Esparza, A. Finkel, R. Mayr. “On the verification of broadcast protocols”, *14th Annual IEEE Symposium on Logic in Computer Science*, pp. 352-359, 1999.
- [62] K. Nazari, J. G. Thistle. “Blocking in fully connected networks of arbitrary size”, *IEEE Transactions on Automatic Control*, 57(5): 1233-1242, 2012.
- [63] J. Esparza. “Keeping a crowd safe: on the complexity of parameterized verification”, *International Symposium on Theoretical Aspects of Computer Science*, pp. 1-10, 2014.
- [64] H. Bherer, J. Desharnais, R. St-Denis. “Control of parameterized discrete event systems”, *Discrete Event Dynamic Systems*, 19(2): 213-265, 2009.
- [65] L. Lin, A. Stefanescu, R. Su, W. Wang, A. R. Shehabinia. “Towards decentralized synthesis: decomposable sublanguage and joint observability problems”, *American Control Conference*, pp. 2047-2052, 2014.
- [66] R. Kumar, V. K. Garg. “Predicates and predicate transformers for supervisory control of discrete event dynamical systems”, *IEEE Transactions on Automatic Control*, 38(2): 232-247, 1993.
- [67] T. Le Gall, B. Jeannet, H. Marchand. “Supervisory control of infinite symbolic systems using abstract interpretation”, *IEEE CDC-ECC*, pp. 30-35, 2005.
- [68] G. Kalyon, T. Le Gall, H. Marchand, T. Massart. “Decentralized control of infinite systems”, *Discrete Event Dynamic Systems*, 21(3): 359-393, 2011.

- [69] G. Kalyon, T. Le Gall, H. Marchand, T. Massart. “Symbolic supervisory control of infinite transition systems under partial observation using abstract interpretation”, *Discrete Event Dynamic Systems*, 22(2): 121-161, 2012.
- [70] R. Kumar, V. K. Garg. “On computation of state avoidance control for infinite state systems in assignment program framework”, *IEEE Trans. on Automation Science and Engineering*, 2(1): 87-91, 2005.
- [71] C. Ma, W. M. Wonham. “Nonblocking supervisory control of state tree structures”, *IEEE Transactions on Automatic Control*, 51(5): 782-793, 2006.
- [72] M. A. Harrison. *Introduction to formal language theory*, Addison-Wesley Publishing Company, 1978.
- [73] V. Diekert, P. Gastin. “First-order definable languages”, *Logic and Automata: History and Perspectives*, Texts in Logic and Games, pp. 261-306, 2008.
- [74] V. Diekert, G. Rozenberg, eds. *The book of traces*, World Scientific, 1995.
- [75] J. E. Pin. *Varieties of formal languages*, North Oxford, London and Plenum, New-York, 1986.
- [76] J. E. Pin, H. Straubing, D. Therien. “Some results on the generalized star-height problem”, *Information and Computation*, 101(2): 219-250, 1992.
- [77] I. J. Aalbersberg, H. J. Hoogeboom. “Characterizations of the decidability of some problems for regular trace languages”, *Mathematical Systems Theory*, 22(1): 1-19, 1989.
- [78] S. Tripakis. “Two-phase distributed observation problems”, In *Proc. of the 5th International Conference on Application of Concurrency to System Design*, pp. 98-105, 2005.

- [79] C. Duboc. “Mixed product and asynchronous automata”, Theoretical Computer Science, Vol. 48, pp. 183-199, 1986.
- [80] R. Cori, D. Perrin. “Automates et commutations partielles”, RAIRO-Theoretical Informatics and Applications, 19(1): 21-32, 1985.
- [81] Y. Chen, X. C. Ding, A. Stefanescu, C. Belta. “Formal approach to the deployment of distributed robotic teams”, IEEE Trans. on Robotics, 28(1): 158-171, 2012.
- [82] R. W. Brockett. *Finite Dimensional Linear Systems*, John Wiley & Sons, 1970.
- [83] A. Paz. *Introduction to probabilistic automata*, Academic Press, New York, 1971.
- [84] C. Baier, J. P. Katoen. *Principles of model checking*, MIT Press, Cambridge (MA), 2008.
- [85] P. R. D’Argenio, J. P. Katoen. “A theory of stochastic systems. Part I: Stochastic automata”, Information and Computation, 203(1): 1-38, 2005.
- [86] A. Stefanescu. “Automatic synthesis of distributed transition systems”, Ph.D. dissertation. University of Stuttgart, 2006.
- [87] I. J. Aalbersberg, E. Welzl. “Trace languages defined by regular string languages”, RAIRO-Theoretical Informatics and Applications, 20(2): 103-119, 1986.
- [88] J. Sakarovitch. “The “last” decision problem for rational trace languages”. LATIN, Lecture Notes in Computer Science, Vol. 583, pp. 460-473, 1992.
- [89] A. Muscholl, H. Petersen. “A note on the commutative closure of star-free languages”, Information Processing Letters, 57(2): 71-74, 1996.

- [90] L. Lin, R. Su, A. Stefanescu. “Remarks on the difficulty of top-down supervisor synthesis”, In Proc. of 12th International Conference on Control, Automation, Robotics, and Vision, pp. 270-275, 2012.
- [91] A. Bouajjani, A. Muscholl, T. Touili. “Permutation rewriting and algorithmic verification”, *Information and Computation*, 205(2): 199-224, 2007.
- [92] T. Touili. “Regular model checking using widening techniques”, *Electronic Notes in Theoretical Computer Science*, 50(4): 342-356, 2001.
- [93] D. Peled, T. Wilke, P. Wolper. “An algorithmic approach for checking closure properties of temporal logic specifications and omega-regular languages”, *Theoretical Computer Science*, 195(2): 183-203, 1998.
- [94] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, W. M. Wonham. “Formulas for calculating supremal controllable and normal sublanguages”, *Systems & Control Letters*, 15(2): 111-117, 1990.
- [95] R. Kumar, V. K. Garg. “Extremal solutions of inequations over lattices with applications to supervisory control”, *Theoretical Computer Science*, 148(1): 67-92, 1995.
- [96] J. Komenda. “Computation of supremal sublanguages of supervisory control using coalgebra”, *IFAC Workshop on Discrete-Event System Design*, pp. 26-33, 2002.
- [97] R. M. Ziller, J. E. R. Cury. “On the supremal L_m -closed and the supremal L_m -closed and L -controllable sublanguages of a given language”, In Proc. of the 11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems, *Lecture Notes in Control and Information Sciences*, Vol. 199, pp. 80-85, 1994.

- [98] K. Rohloff, S. Lafortune. “The verification and control of interacting similar discrete event systems”, *SIAM Journal on Control and Optimization*, 45(2): 634-667, 2006.
- [99] R. Su, L. Lin. “Synthesis of control protocols for multi-agent systems with similar actions”, *IEEE Conference on Decision and Control*, pp. 6986-6991, 2013.
- [100] W. Wang, R. Su, L. Lin, “On analysis of deadlock and blocking freeness in isomorphic module systems”, *American Control Conference*, pp. 923-928, 2013.
- [101] B. Aminof, T. Kotek, S. Rubin, F. Spegni, H. Veith. “Parameterized model checking of rendezvous systems”, *Concurrency Theory, Lecture Notes in Computer Science*, Vol. 8704, pp. 109-124, 2014.
- [102] B. Jonsson, M. Nilsson. “Transitive closures of regular relations for verifying infinite state systems”, *Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science*, Vol. 1785, pp. 220-235, 2000.
- [103] G. Delzanno, A. Rezzina. “A lightweight regular model checking approach for parameterized systems”, *International Journal on Software Tools for Technology Transfer*, 14(2): 207-222, 2012.
- [104] P. A. Abdulla, B. Jonsson, M. Nilsson, M. Saksena. “A survey of regular model checking”, *Concurrency Theory, Lecture Notes in Computer Science*, Vol. 3170, pp. 35-48, 2004.
- [105] A. Bouajjani, A. Legay, P. Wolper. “Handling liveness properties in $(\omega-)$ regular model checking”, *Electronic Notes in Theoretical Computer Science*, 138(3): 101-115, 2005.

- [106] A. Bouajjani, B. Jonsson, M. Nilsson, T. Touili. “Regular model checking”, Computer Aided Verification, Lecture Notes in Computer Science, Vol. 1855, pp. 403-418, 2000.
- [107] A. W. To, L. Libkin. “Recurrent reachability analysis in regular model checking”, International Conference on Logic for Programming Artificial Intelligence and Reasoning, Lecture Notes in Computer Science, Vol. 5330, pp. 198-213, 2008.
- [108] P. Habermehl, T. Vojnar. “Regular model checking using inference of regular languages”, Proceedings of the 6th International Workshop on Verification of Infinite-State Systems, 138(3): 21-36, 2005.
- [109] A. Bouajjani, P. Habermehl, T. Vojnar. “Abstract regular model checking”, Computer Aided Verification, Lecture Notes in Computer Science, Vol. 3114, pp. 372-386, 2004.
- [110] D. Dams, Y. Lakhnech, M. Steffen. “Iterating transducers”, Computer Aided Verification, Lecture Notes in Computer Science, Vol. 2102, pp. 286-297, 2001.
- [111] B. Boigelot, A. Legay, P. Wolper. “Iterating transducers in the large”, Computer Aided Verification, Lecture Notes in Computer Science, Vol. 2725, pp. 223-235, 2003.
- [112] W. Meng, Z. He, R. Teo, R. Su, A. R. Shehabinia, L. Lin, L. Xie. “Decentralized control of multi-UAVs for target search, tasking and tracking”, the 19th World Congress The International Federation of Automatic Control, pp. 10048-10053, 2014.
- [113] J. Engelfriet, “A Kleene characterization of computability”, Information Processing Letters, 101(4): 139-140, 2007.

- [114] A. Terlutte, D. Simplot, “Iteration of rational transductions”, *RAIRO, Theoretical Informatics and Applications*, 34(2): 99-129, 2000.
- [115] D. Kozen. “A completeness theorem for Kleene algebras and the algebra of regular events”, 6th IEEE Symposium on Logic in Computer Science, pp. 214-225, 1991.
- [116] M. H. Zibaeenejad, J. G. Thistle. “Weak invariant simulation and analysis of parameterized networks”, *IEEE Transactions on Automatic Control*, 59(8): 2024-2037, 2014.
- [117] D. Peled. “A generalized closure and complement phenomenon”, *Discrete Mathematics*, Vol. 50, pp. 285-293, 1984.
- [118] Z. Pawlak. “Rough sets”, *International Journal of Computer & Information Sciences*, 11(5): 341-356, 1982.
- [119] L. Lin, A. Stefanescu, R. Su. “On distributed and parameterized supervisor synthesis problems”, to appear in *IEEE Transactions on Automatic Control*, 2016.

Appendix: Semi-topological Structure

Let Σ be a finite alphabet. An operator $\square : 2^{\Sigma^*} \rightarrow 2^{\Sigma^*}$ is said to be a semi-topological closure operator on 2^{Σ^*} if it satisfies the axioms (S1), (S2), (S3) of [117], in addition to a new axiom (S4) : $\emptyset^\square = \emptyset$. That is, if it satisfies:

For any $A, B \in 2^{\Sigma^*}$,

$$(S1): A \subseteq A^\square$$

$$(S2): (A^\square)^\square = A^\square$$

$$(S3): A^\square \cup B^\square \subseteq (A \cup B)^\square$$

$$(S4): \emptyset^\square = \emptyset$$

The semi-topological closure operator $\square$ on 2^{Σ^*} induces a unique semi-topology $(\Sigma^*, \mathcal{T})$ with $\mathcal{T} = \{L^{\square^c} \mid L \in 2^{\Sigma^*}\}$ such that (T1), (T2), (T3) in [117] and (T4) : $\Sigma^* \in \mathcal{T}$ hold. That is,

$$(T1): \mathcal{T} \subseteq 2^{\Sigma^*}$$

$$(T2): \emptyset \in \mathcal{T}$$

$$(T3): \mathcal{T} \text{ is closed under arbitrary unions}$$

$$(T4): \Sigma^* \in \mathcal{T}$$

$L \subseteq \Sigma^*$ is said to be closed with respect to $\square$ if $L = L^\square$. $\{L^\square \mid L \in 2^{\Sigma^*}\}$ is the family of closed sets and $\mathcal{T} = \{L^{\square c} \mid L \in 2^{\Sigma^*}\}$ is the family of open sets for the semi-topologies induced by $\square$.

The following holds. A similar observation is also known in [118] in the area of artificial intelligence.

Lemma 6.1. *There exists a unique supremal $\square$ -closed subset of L for any language $L \subseteq \Sigma^*$ if the family of open sets corresponds exactly to the family of closed sets for the semi-topology induced by $\square$. Indeed, the supremal $\square$ -closed subset of L is $L^{\square c}$.*

Proof: Suppose the family of open sets corresponds exactly to the family of closed sets for the semi-topology (Σ^*, τ) , with $\tau = \{L^{\square c} \mid L \subseteq \Sigma^*\}$, induced by $\square$. It is easy to see that the family of closed sets is closed under arbitrary union in the induced semi-topology, so that there exists a unique supremal $\square$ -closed subset for any given language.

Clearly, $L^{\square c}$ is the interior of L and the supremal open subset of L . Since the family of closed sets corresponds exactly to the family of closed sets, $L^{\square c}$ is the supremal $\square$ -closed subset of L . ■