Submitted, IEEE T. Control of Networked Systems (14 Nov 2013)
http://www.cds.caltech.edu/~murray/papers/xtm13-tcns.html

Specification and Synthesis of Reactive Protocols for Aircraft Electric Power Distribution

Huan Xu[1], Ufuk Topcu[2], and Richard M. Murray[3]

***Abstract*—The increasing complexity of electric power systems leads to challenges in integration and verification. We consider the problem of designing a control protocol for an aircraft electric power system that meets a set of requirements describing the correct behaviors of the system and reacts dynamically to changes in internal system states. We formalize the requirements by translating them into a temporal logic specification language and apply game-based, temporal logic formal methods to automatically synthesize a controller protocol that satisfies these overall properties and requirements. Through a case study, we perform a design exploration to show the benefits and tradeoffs between centralized and distributed control architectures.**

I. Introduction

THE move from conventional to more-electric aircraft architectures within the aerospace industry has been motivated by advancements in electronics technology that can result in aircraft with improved reliability, optimized system-level performance, and decreased life-cycle costs [1]. As more subsystems become reliant on electric power, the ability of the electric power system to function properly is critical in flight operations. However, this increased reliance also increases the complexity of the electric power system, requiring more components, actuators, and more complex architecture. At present, standard methodologies for designing such systems are costly both in terms of time and money. Past work has focused on the analysis of aircraft performance and power optimization by using modeling libraries and simulations on proposed designs [2], [3]. Analysis of all faults or errant behaviors in models, however, is difficult due to the high complexity of systems and subsystem interactions.

Because the process of verifying the correctness of a system with respect to specifications is expensive, this difficulty has led to a greater emphasis on the use of formal methods within a model-based systems engineering framework to aid in safety and performance certification. The work described in [4] provides a methodology for the overall design of an electric power system, beginning from requirements to real-time simulation. In this paper, we examine the control logic portion of the methodology and "specify and synthesize" a solution to the design problem instead of "design then verify." In other words, instead of piecing together legacy designs that are then simulated and verified, we can synthesize control logic that is provably correct with respect to system requirements.

Building on previous work [5], we apply formal synthesis of control protocols that enable dynamic reconfiguration of power in more-electric aircraft. We automatically synthesize a controller based on temporal logic specifications that satisfy system requirements while reacting to uncontrolled moves from an environment (or adversary) [6]. We begin by writing English-based specifications in linear temporal logic (LTL), and then use a combination of tools from computer science formal method domains for the automatic synthesis of control protocols. The use of synthesis methods follows from their successful integration in verification of hardware and software systems in computer science, engineering, and robotics domains [7]–[11]. Applications of synthesis tools are limited to small problems due to the state space explosion issue. To address this challenge, we utilize previous work on the compositional design of correct-by-construction, distributed protocols for an electric power system [12], [13]. Distribution of the design and implementation of the electric power system will reduce the computational complexity, as well as allow for the design of flexible control architectures in terms of modularity, fault-tolerance, and integrability [14].

The main contributions of this paper are twofold: (i) We investigate how specifications typically imposed in the design of controllers for electric power distribution can be translated into temporal logic. This formalization allows automation of synthesis of reactive control protocols that ensure the correctness of the underlying specifications. We provide an end-to-end demonstration of such a "specify and synthesize" design flow on a small example. (ii) As a step toward scaling automated controller synthesis for practical size problems, we examine the application of compositional synthesis approaches, which we developed in our prior work, for the design of distributed control protocols. We show the utility of this compositional method on master/slave and decentralized control architectures. The long- term potential for incorporating the "specify and synthesize" will be (i) faster design times, as problems can be found earlier in the design cycle, and (ii) less costly systems, due to the ability to save time and perform design space exploration in software rather than in hardware.

The remainder of the paper is structured as follows: We describe a standard electric power system, including components, connectivity, and typical design considerations in Section II. Sections III and IV presents a technical description of the linear temporal logic and the specifications for an electric power system. Section V details the synthesis procedure. A case study is presented in Section VI with results for a

This work was supported in part by the Multiscale Systems Center (MuSyC), the Boeing Corporation, and AFOSR Award FA9550-12-1-0302.
The authors are with (1) Institute for Systems Research and Aerospace Engineering, University of Maryland, College Park, MD 20740, USA, (2) Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA, and (3) Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA *mumu@umd.edu, utopcu@seas.upenn.edu, murray@cds.caltech.edu*

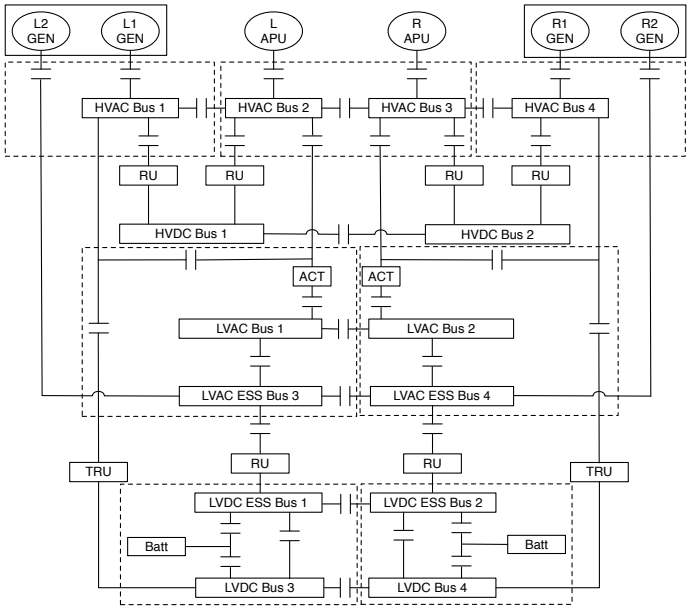


Fig. 1: Single-line diagram of an electric power system adapted from a Honeywell, Inc. patent [18]. Two high-voltage generators, two APU-mounted generators, and two low-voltage generators serve as power sources for the aircraft. Power can be routed from sources to buses through contactors, rectifier units, and transformers. Buses are connected to subsystem loads. Batteries can be used to provide emergency power.

centralized and distributed control architecture.

II. Electric Power Distribution System

A. System Components

A standard electric power system comprises generators, batteries, rectifier units, and batteries that supply power to a set of loads through buses. Figure 1 is a single-line diagram of an electric power system. The following is a brief description of components referenced throughout the paper [17].

AC and DC *buses* for high and low voltage deliver power to a sub-buses, loads, and power conversion equipment. These buses can be classified as essential and non-essential. Essential buses supply loads that should always remain powered, such as the flight actuation subsystem, while non-essential buses have loads that may be shed, such as cabin lighting, in the case of a failure. *AC Generators* can operate at either high voltages, which can connect to the high-voltage AC buses, or low voltages, which feed directly to the low-voltage buses. *Contactors* are high-power electronic switches that connect the flow of power from sources to buses and loads. Contactors provide the actuation for reconfiguration of the topology of the electric power system, hence, changing the paths through which power is delivered. *Rectifier Units* convert three-phase AC power to DC power. *Transformer Rectifier Units* combine a rectifier unit with a transformer to lower voltages.

B. Current Design Practice

Figure 2 depicts the design flow for a typical aerospace system. In current practice, there are two concurrent work streams. In the requirements work stream (left column in

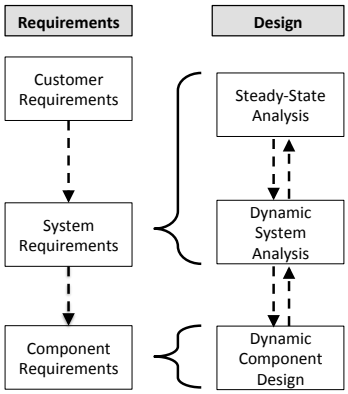


Fig. 2: A typical aerospace system design flow depicting requirements and design streams.

Figure 2), a set of English-based customer requirements are manually decomposed into system-level and component-level specifications. In the second work stream, steady-state analysis is first performed on the system. Modeling tools are used to select relevant properties or set points. The methods typically rely on reuse where possible and domain expertise to ensure the entire envelope is covered. Once this is done, a dynamic system analysis is performed on low-fidelity models to assess operational scenarios and validate control performance. Finally, detailed models are used to analyze at the component level. System and component requirements are mapped or linked to the models (from the left column to the right column). All three levels of abstraction are constructed by different groups using design reviews to ensure consistency. If an inconsistency is found, then the design gets iterated until the problem is corrected. Dashed arrows thus represent manual checks between groups to guarantee consistency.

The disadvantage to current practice is that exchanges between these design layers are manual and text-based, thus requiring design reviews, multiple iterations, and verification of results. Problems found late in the design process are difficult to diagnose, and costly to change. An alternative to designing a system and then verifying its correctness is to "specify and synthesize." Requirements are specified using a mathematical language, and then a control law is automatically synthesized to be correct with respect to those requirements. The potential benefits of such an approach are twofold. First, synthesis is efficient in terms of time by automatically constructing control logic that is guaranteed to satisfy requirements. Second, the use of formal specification languages to synthesize controllers allows for easier and systematic ways to identify improper designs, and allows designers to capture system requirements (including temporal requirements) in an unambiguous manner.

One of the major barriers to implementing synthesis and formal methods techniques has been the aversion to techniques and tools that are new or have a high learning curve. Past methods for design of systems have worked. Thus, the need for a more model-based approach was seen as unnecessary or risky. This mentality has been slowly changing as newer systems have proven to be more much complex to design and difficult to verify [15], [16], where delays in production have lasted years and cost millions of dollars. The second barrier

to synthesis is the issue of scalability. As systems increase in size and complexity, synthesis tools become time and memory intensive for industrial scale problems. This problem can be alleviated by distributed synthesis methods, as well as restricting the types of specifications with which the system must be correct. The transition from the current practice to a model-based systems engineering framework with suitable tool support for large-scale problems, and formal methods techniques have been shown to decrease design time and used to aid the finding of errant behavior early in the process.

C. Design Considerations

The control protocol design problem considers how the system shall reconfigure as a function of the changes in flight conditions and faults in the components. Typically such reconfiguration takes place in multiple layers. Generation and primary distribution involves the start-up or shut-down of high-voltage generators or APUs in addition to the reconfiguration of contactors in order to route power to high and low voltage buses and their respective loads.

The networked structure of the electric power system requires a protocol to account for all possible interleavings of faults or events throughout a flight. These combinations and sequences of failures may be rare but can be catastrophic. This structure likewise introduces and allows for redundancy in the electric power system in both hardware and software. In the remainder of this paper we focus on the dynamic reconfiguration of generation and primary distribution systems by designing a control protocol for contactors. Based on the status of generators and buses, the protocol ensures the proper switching of contactors to ensure buses will remain powered.

III. Formal Specification Using Linear Temporal Logic

Given a topology of an electric power system like that of Figure 1, the main design problem becomes determining all correct configurations of contactors for all flight conditions and faults that can occur in the system. For a configuration to be "correct" means that it satisfies system requirements, also referred to as specifications. We now discuss a formal specification language that will be utilized for the synthesis of control protocols later in this section.

In reactive systems (i.e., systems that react to a dynamic, *a priori* unknown environment), correctness will depend not only on inputs and outputs of a computation, but also on execution of the system. Temporal logic is a branch of logic that incorporates notions of temporal ordering to reason about correctness of propositions over a sequence of states, and is well-suited for problems in which the system must react to an adversary or environment. First used as a specification language by Pnueli [19] in the 1970s, it has been utilized to specify properties in a number of applications, including embedded systems, robotics, and controls [7]. In this paper we consider a version called linear temporal logic (LTL).

LTL's main building block is the atomic proposition, which is a statement on a valuation of variables that has a unique truth value (*True* or *False*). Consider, for example, the health status of generators g_1 and g_2, where $\{g_1 = healthy\}$, and $\{g_2 = unhealthy\}$ are atomic propositions. The truth values of each proposition can be determined for a given configuration of the electric power system. LTL combines logical connectives such as negation ($\neg$), disjunction ($\vee$), conjunction ($\wedge$), and material implication ($\rightarrow$) alongside temporal operators such as *always* ($\Box$), *eventually* ($\Diamond$), *next* ($\bigcirc$) and *until* ($\mathcal{U}$) to create complex specifications for a system. These logical connectives and temporal operators can be combined to specify a number of complex requirements.

Safety formulas assert that a state or sequence of states will not be reached. In particular, we use a subclass of safety formula referred to as invariants throughout this paper. Invariant formula assert that a property will remain true throughout the entire execution, and ensure that nothing bad will happen. A safety specification for the electric power system could take the form $\Box(\neg bus\_i\_unpowered)$ where i is the bus index.

Progress formulas guarantee that a property holds infinitely often in an execution This property ensures that the system will make progress. For example, always eventually ensure that Bus 1 is powered can be written as: $\Box \Diamond gen\_i\_powered.$

Response formulas state that at some point in the execution following a state where a property is true, there exists a point where a second property is true. Response properties can be used to describe how systems need to react to changes in environment or operating conditions. A response property can be used to describe how the system should react to a generator failure. If a generator fails, then at some point a corresponding contactor should open: $\Box((gen\_j\_not\_healthy) \rightarrow \Diamond(contactor\_k\_open))$ where j, k represent indices for generators and contactors, respectively.

LTL is naturally used for specifying properties with temporal ordering. For specifications involving hard-timing constraints, LTL can be used for synchronous systems in which all processes (i.e., components) proceed in a lock-step manner. The next operator has a "time" measure so that, for a given property φ, $\bigcirc\varphi$ signifies at the next time instant φ is true. To specify a property occurring at some point in the future, multiple next operators can be used, such that $\bigcirc^k\varphi \triangleq \bigcirc\bigcirc\ldots\bigcirc\varphi$ asserts that property φ holds k time instants in the future. Alternatively, the "timed" specifications in the electric power system uses a clock variable to define an equivalent property. A more detailed discussion of LTL and other temporal logics can be found in [20], [21].

IV. Formal Specifications for Aircraft Electric Power Systems

The following list details the temporal logic specifications for synthesizing control protocols for electric power systems.

Environment Assumption: Let $\mathcal{G}$ and $\mathcal{R}$ represent the set of all generators and rectifier units in the electric power system topology. Let the Boolean variables g and r denote the health status of generator $G \in \mathcal{G}$ and rectifier $R \in \mathcal{R}$, respectively. That is, the lowercase symbol represents the health status of the component denoted by the corresponding uppercase symbol. We use a similar convention between upper and lowercase symbols in the remainder of the paper. In

order for the system to be non-trivial, we assume that at least one generator and one rectifier unit must always be healthy, i.e., have a status of 1, at any given time. The environment assumption is written as

$$\square \left\{ \bigvee_{G \in \mathcal{G}} (g = 1) \ \wedge \ \bigvee_{R \in \mathcal{R}} (r = 1) \right\}. \tag{1}$$

Contactor Delays: To capture the physical delay in contactor actuation times, we introduce the Boolean variable $\tilde{c}$, denoting controller intent (command) for contactor status c. The delay between intent and contactor status is handled by an additional clock variables x_C^{close} (closing time) and x_C^{open} (opening time) for each contactor C, where each "tick" of the clock represents δ time. If the intent is to open and a contactor status is closed, then the contactor opens within time bound $[T_{o_{min}}, T_{o_{max}}]$. Similarly, if the intent is to close and a contactor status is open, the contactor closes within time bound $[T_{c_{min}}, T_{c_{max}}]$. If the contactor status does not match the intent, at the next step clock x_C increments by δ. The controlled variable is the contactor intent, while the status is modeled as an environment variable. The following specifications are thus included in the environment assumption.

If the status and intent match, then in the next step the clock resets to zero, which is written as

$$\square\{(\bigcirc c = \tilde{c}) \rightarrow (\bigcirc x_C = 0)\}. \tag{2}$$

If the intent is the same as the contactor status, then contactor status remains the same in the next step, i.e.,

$$\square\{(\tilde{c} = c) \rightarrow (\bigcirc c = c)\}. \tag{3}$$

The assumption to capture the delay in closing time between a contactor intent and status is given by

$$\begin{aligned} &\square \left\{ (\tilde{c} = 1 \wedge c = 0 \wedge (x_C^{close} < T_{c_{min}})) \rightarrow \right. \\ &\qquad \left. (\bigcirc c = 0 \wedge \bigcirc x_C^{close} = x_C^{close} + \delta) \right\}, \end{aligned} \tag{4}$$

$$\begin{aligned} &\square \left\{ (\tilde{c} = 1 \wedge c = 0 \wedge (x_C^{close} \geq T_{c_{min}})) \rightarrow \right. \\ &\qquad \left. (\bigcirc c = 1 \vee \bigcirc x_C^{close} = x_C^{close} + \delta) \right\}. \end{aligned} \tag{5}$$

To capture the delay in opening between a contactor intent and status is given by

$$\begin{aligned} &\square \{ (\tilde{c} = 0 \wedge c = 1 \wedge (x_C^{open} < T_{o_{min}})) \rightarrow \\ &\qquad (\bigcirc c = 1 \wedge \bigcirc x_C^{open} = x_C^{open} + \delta) \}, \end{aligned} \tag{6}$$

$$\begin{aligned} &\square \{ (\tilde{c} = 0 \wedge c = 1 \wedge (x_C^{open} \geq T_{o_{min}})) \rightarrow \\ &\qquad (\bigcirc c = 0 \vee \bigcirc x_C^{open} = x_C^{open} + \delta) \}. \end{aligned} \tag{7}$$

Finally, we guarantee that clocks x_C^{close} and x_C^{open} never exceed the opening and closing time bounds, written as

$$\square(x_C^{close} \leq T_{c_{max}}), \tag{8}$$

$$\square(x_C^{open} \leq T_{o_{max}}). \tag{9}$$

Unhealthy Sources: We require any contactor adjoining a generator or rectifier unit to open when that component becomes unhealthy. Let $\mathcal{C}$ represent the set of all contactors in the electric power system. Sets $\mathcal{N}(G_i) \subseteq \mathcal{C}$ and $\mathcal{N}(R_i) \subseteq \mathcal{C}$ represent the contractors directly connected to, or neighboring, generator G_i and rectifier R_i, respectively. In Figure 3, for example, sets $\mathcal{N}(G_1)$ and $\mathcal{N}(G_2)$ consist of contactors C_1 and C_2, respectively. For a contactor C, let c be its status (for example, 0 represents an open contactor, 1 a closed contactor). If a generator's status is unhealthy, then contactors connecting to it should be commanded open, i.e., the contactor intent variable should take the value of 0. This can be written as

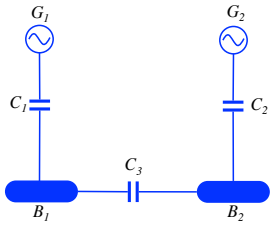


Fig. 3: A single-line diagram with two generators, two buses, and three contactors. Paralleling of AC sources can occur if the status of all three contactors C_1, C_2, and C_3 are all closed.

$$\bigwedge_{G \in \mathcal{G}} \square \left\{ (g = 0) \rightarrow \bigwedge_{C \in \mathcal{N}(G)} (\tilde{c} = 0) \right\}. \tag{10}$$

Similar specifications hold for disconnecting rectifier units.

No Paralleling of AC Sources: A mismatch in AC generator frequencies and voltages can lead to loss of availability and even damage of the generator or distribution system. To avoid difficulties of generator synchronization, we disallow any paralleling of AC sources, i.e., no bus should be powered by multiple AC generators at the same time. One way to avoid paralleling AC sources is to explicitly enumerate and eliminate all configurations in which buses can be powered from multiple sources. In the example shown in Figure 3, paralleling could occur if the status of contactors C_1, C_2, and C_3 were all closed at the same time. A specification would then be to never allow all contactors along a path to close at the same time if that path could connect two AC sources.

Let X_{G_i,G_j} represent the set of components along a path between generators $G_i, G_j \in \mathcal{G}$ and $i \neq j$. We disallow configurations in which all contactors $C \in X_{G_i,G_j}$ create a *live path* (i.e., all contractors closed along a path). These specifications are written as

$$\square \bigwedge_{G_i,G_j \in \mathcal{G}} \left\{ \neg \bigwedge_{C \in X_{G_i,G_j}} (c = 1) \right\}. \tag{11}$$

Power Status of Buses: An AC bus is powered if there exists a live path that connects the bus to a healthy generator. A DC bus is powered if there exists a live path between the DC bus and a healthy rectifier unit, which itself is connected to a powered AC bus. Let $X_{G,B}$ denote the set of all components (contactors, buses, and rectifier units) along a path between bus B and generator G, excluding B and G.

Bus B is powered if there exists a live path between B and G, written as

$$\square \left\{ \bigvee_{G \in \mathcal{G}} \left((g = 1) \wedge \bigwedge_{X \in X_{G,B}} (x = 1) \right) \rightarrow (b = 1) \right\}. \tag{12}$$

If no live path exists, then B will be unpowered, written as

$$\square \left\{ \neg \bigvee_{G \in \mathcal{G}} \left((g = 1) \wedge \bigwedge_{X \in X_{G,B}} (x = 1) \right) \rightarrow (b = 0) \right\}. \tag{13}$$

Essential Buses: Buses connected to safety-critical loads need to remain powered. Let $\mathcal{B}_s$ be the set of all safety-critical buses. Denote the allowable length of time a bus can remain unpowered as T. For example, typical values for T fall in the 50 msec range [17]. Time in this formulation is implemented through an additional clock θ_B associated with bus B, and where, again, each increment represents δ time. For each safety-critical bus in $B \in \mathcal{B}_s$, these specifications can be written as the following:

- If the status of bus B is unpowered, in the next step clock variable θ_B will increment by one unit, which is written as

$$\square \{(b = 0) \rightarrow (\bigcirc \theta_B = \theta_B + \delta)\}. \tag{14}$$

- If the status of bus B is powered, in the next step clock variable x_B is reset to 0. This is written as

$$\square \{(b = 1) \rightarrow (\bigcirc \theta_B = 0)\}. \tag{15}$$

- Clock variable x_B will never be greater than the maximum allowable unpowered time T. This is implemented by

$$\square \{\theta_B \leq T\}. \tag{16}$$

Remark 1: Specifications for DC components are the same as AC specifications except for two simplifications: (1) The non-paralleling of AC sources specification may be ignored and (2) no DC bus may ever be unpowered (i.e., unpowered time $T = 0$). In addition, rectifier units are uni-directional and thus power cannot flow from DC buses to AC buses.

V. Synthesis of Reactive Control Protocols

The overall goal is to synthesize a control protocol that ensures the controlled system satisfies the specifications discussed previously. The correctness of the system is not merely a function of the states of the controlled variables. It needs to be interpreted in conjunction with the statuses of the externalities that interact with the system that cannot be controlled. Furthermore, it is necessary to incorporate information on potential environment conditions under which the system is expected to operate. If the environment variables are not properly constrained, then the resulting control protocol may be overly conservative, and it may not be possible to construct a protocol that ensures the satisfaction of the system requirements. For example, if all the generators simultaneously stay unhealthy for a long enough time, then it is not possible to satisfy the condition that the essential buses shall not be unpowered longer than some prespecified period. Consequently, the overall goal is to design a protocol that determines how controlled variables shall move at each point of the execution as a function of the behaviors of the controlled and environment variables so far in the execution as long as the environment assumptions are satisfied.

A. Reactive Synthesis

Equipped with LTL as a specification language, we now formally state the reactive synthesis problem. Let E and P be sets of environment and controlled variables, respectively. Let $s = (e, p) \in dom(E) \times dom(P)$ be a state of the system. Consider a LTL specification φ of assume-guarantee form

$$\varphi = \varphi_e \rightarrow \varphi_s, \tag{17}$$

where φ_e is the conjunction of LTL specifications that characterizes the assumptions on the environment and φ_s is the conjunction of LTL specifications that characterizes the system requirements. The synthesis problem is then concerned with constructing a strategy, i.e., a partial function $f : (s_0 s_1 \ldots s_{t-1}, e_t) \mapsto p_t$, that chooses the move of the controlled variables based on the state sequence so far and the behavior of the environment so that the system satisfies φ_s as long as the environment satisfies φ_e. The synthesis problem can be viewed as a two-player game between the environment and controlled plant: the environment attempts to falsify the specification in (17) and the controlled plant tries to satisfy it.

For general LTL, the synthesis problem has a doubly exponential complexity in the size of the specification [7]. For a subset of LTL called Generalized Reactivity (1) (GR(1)), Piterman et al., have shown that synthesis can be solved in polynomial time (polynomial in the number of valuations of the variables in E and P) [6]. GR(1) specifications restrict φ_e and φ_s to take the following form, for $\alpha \in \{e, s\}$,

$$\varphi_\alpha := \varphi^\alpha_{\text{init}} \wedge \bigwedge_{i \in I^\alpha_1} \square \varphi^\alpha_{1,i} \wedge \bigwedge_{i \in I^\alpha_2} \square \diamond \varphi^\alpha_{2,i},$$

where $\varphi^\alpha_{\text{init}}$ is a propositional formula characterizing the initial conditions; $\varphi^\alpha_{1,i}$ are transition relations characterizing safe, allowable moves and propositional formulas characterizing invariants; and $\varphi^\alpha_{2,i}$ are propositional formulas characterizing states that should be attained infinitely often. Many interesting temporal logic specifications can be expressed or easily transformed into GR(1) specifications. See [6], [22] for a more precise treatment of GR(1) synthesis and [5], [6], [22], [23] for case studies in hardware synthesis, motion planning for autonomous vehicles, and vehicle management systems.

Given a GR(1) specification, the digital design synthesis tool implemented in JTLV (a framework for developing temporal verification algorithm) [24] generates a finite automaton that represents a switching strategy for the system. The temporal logic planning (TuLiP) toolbox, a collection of python-based code for automatic synthesis of correct-by-construction embedded control software provides an interface to JTLV [25]. For examples discussed in this paper, we use TuLiP.

B. Distributed Synthesis

The control protocols discussed in Section V-A are centralized in that the controller has access to measurements of all controlled and environment variables, and is able to determine the evolution of all controlled variables in order to satisfy a set of specifications. We now detail a few reasons for migrating to distributed control architectures.

Hardware challenges: A centralized controller onboard an aircraft requires wiring from a central processing unit to all components. Local controllers allows for shorter wires and increased efficiency due to this reduction in weight. *Increased resilience to failure:* By distributing the implementation of the controller, the electric power system can be more robust to failures, i.e., if one portion of the electric power system malfunctions, the other sections are unaffected and can still be fully operational. *Reduction of computational complexity:* With an increased number of electric components, the combination of configurations the controller must account for quickly becomes intractable for current verification and synthesis tools as well as testing. A distributed controller design correctly decomposes the design task into smaller subproblems each of which may be easier to cope with.

Advantages from the distribution of the control design come with increased importance of reasoning about the interfaces between the controlled subsystems. There is relatively extensive literature on compositional reasoning [26]–[28]. Here, we follow the exposition in recent work from [12]. Figure 4 illustrates the decomposition of global specifications into local specifications. For ease of presentation, consider the case where the system *SYS* is decomposed into two subsystems SYS_1 and SYS_2. For $i = 1, 2$, let E_i and P_i be the environment variables and controlled variables for SYS_i such that $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$. Let φ_{e_1} and φ_{e_2} be LTL formulas containing variables in E_1 and E_2, respectively. Similarly, let φ_{s_1} and φ_{s_2} be LTL formulas in terms of $E_1 \cup P_1$ and $E_2 \cup P_2$, respectively. If the following conditions hold

1) any execution of the environment that satisfies φ_e also satisfies $(\varphi_{e1} \wedge \varphi_{e2})$,
2) any execution of the system that satisfies $(\varphi_{s1} \wedge \varphi_{s2})$ also satisfies φ_s, and
3) there exist two control protocols that realize the local specifications $(\varphi_{e1} \rightarrow \varphi_{s1})$ and $(\varphi_{e2} \rightarrow \varphi_{s2})$,

then, by a result in [12], implementing these control protocols leads to a system where the global specification is met.

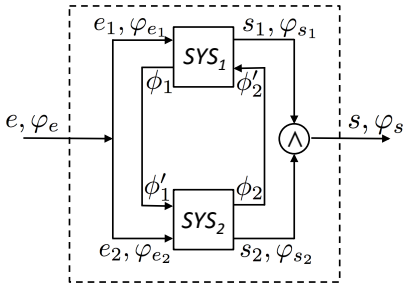


Fig. 4: A schematic for the decomposition of global specifications into distributed controllers for two subsystems. The overall environment assumptions φ_e and system guarantees φ_s are distributed into the two subsystems SYS_1 and SYS_2. Each subsystem has its own local environment assumptions and system guarantees. In addition, SYS_1 has an extra set of local guarantees ϕ_1 that interact with SYS_2 as environment assumptions ϕ'_1, while SYS_2 guarantees contained in ϕ_2 act as environment assumptions ϕ'_2 for SYS_1.

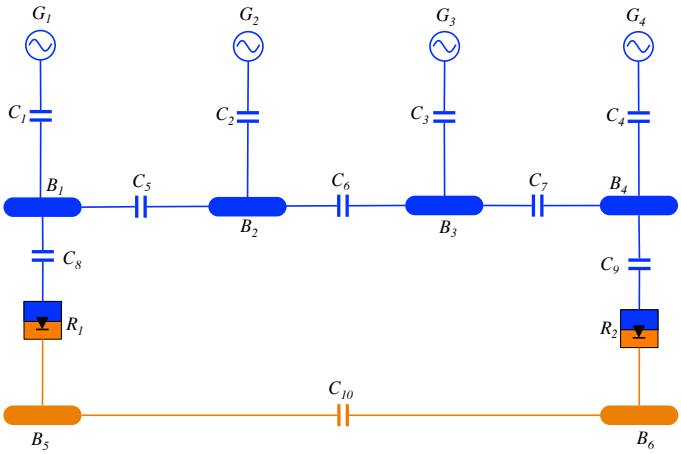


Fig. 5: A simplified single-line diagram. Four generators connect to four AC buses. Two rectifier units convert power from AC to DC, and connect to two DC buses.

It is possible that even if the centralized problem is realizable, the local distributed synthesis may be unrealizable. Subsystems may need to interact with each other through shared variables (either information or physical values) in order to become realizable. As seen in Figure 4, subsystem SYS_1 provides additional guarantees ϕ_1 to subsystem SYS_2, evaluated as an environment assumption and denoted as ϕ'_1. The same interaction applies to the interface between SYS_2, which sends its own local guarantees ϕ_2 to SYS_1. If the following local specifications (and interface refinements) hold:

$$\phi'_2 \wedge \varphi_{e_1} \rightarrow \varphi_{s_1} \wedge \phi_1, \tag{18}$$

$$\phi'_1 \wedge \varphi_{e_2} \rightarrow \varphi_{s_2} \wedge \phi_2. \tag{19}$$

Then the global specification $\varphi_e \rightarrow \varphi_s$ is realizable.

VI. Case Study

We examine a simplified single-line diagram as shown in Figure 5. For all allowable failures, the synthesis problem is to design a control logic that will reconfigure contactors so that power will be delivered to buses.

A. Variables

Environment Variables: The health statuses for generators G_1, G_2, G_3, G_4, and rectifier units R_1 and R_2 can each take values of healthy (1) or unhealthy (0). Again, we distinguish component variables and status variables by upper and lower cases, e.g., the first generator is represented by G_1, while its health status is denoted by g_1. To implement delays between a contactor intent and status, we define contactor statuses c_1, c_2, c_3, and c_4 with values of open (0) and closed (1). For simplicity, all other contactors are considered to have no delays (i.e., at each time step, the physical state of the contactor will always match the intent of the contactor).

Controlled Variables: Contactor intent $\tilde{c}_1, \tilde{c}_2, \tilde{c}_3$, and $\tilde{c}_4$ for contractors connecting generators to buses can each take values of intend to open (0) or intend to closed (1). All other contactors are directly controlled and can either be 0, denoting an open contactor, or 1, signifying a closed contactor.

Dependent Variables: The statuses of buses B_1, B_2, B_3, B_4, B_5, and B_6 can be either powered (1) or unpowered (0) depending on the status of neighboring contactors, rectifier units, and generators.

B. Specifications

Given the topology in Figure 5, the specifications described in Section IV reduce to the following specifications used in the synthesis problem for the simplified single-line diagram.

Environment Assumption: From equation (1), the assumption that at least one generator and rectifier unit is always healthy becomes

$$\square\{(g_1 = 1) \ \vee \ (g_2 = 1) \ \vee \ (g_3 = 1) \ \vee \ (g_4 = 1)\}, \quad (20)$$

$$\square\{(r_1 = 1) \ \vee \ (r_2 = 1)\}. \quad (21)$$

Because of delays between contactor intents and statuses, we prevent generators and rectifier statuses from rapidly switching between healthy and unhealthy. Without such a restriction, the intent of the contactor could constantly change before the status would have time react, thus rendering the synthesis problem unrealizable. A simple assumption is that once a generator becomes unhealthy, it will remain unhealthy. This is equivalent to assuming that even if the generator were to become healthy again, the contactor would not allow the generator to provide power to the rest of the system, which is true in practice as well. Because in this case study the contactors connected to rectifier units have no delay, there is no need to similarly restrict their environment behavior. Therefore, the additional specification is

$$\square \bigwedge_{i=1}^{4} \{(g_i = 0) \rightarrow (\bigcirc g_i = 0)\}. \quad (22)$$

Contactor Delays: We set each clock tick such that $\delta = 1$, and consider the delay times to open and close all contactors to be the same, with minimum and maximum delay times to be 0 and 2. Thus, $T_{c_{min}} = T_{o_{min}} = 0$ and $T_{c_{max}} = T_{o_{max}} = 2$, and equations (2)–(9) are used to specify contactor C_i delays, with $i \in [1, 4]$.

Unhealthy Sources: In Figure 5, the set of neighboring contactors to generators are $\mathcal{N}(G_1) = C_1$, $\mathcal{N}(G_2) = C_2$, $\mathcal{N}(G_3) = C_3$, and $\mathcal{N}(G_4) = C_3$. Neighobring contractors to rectifier units are $\mathcal{N}(R_1) = C_8$ and $\mathcal{N}(R_2) = C_9$. If, for example, the status of rectifier R_1 becomes unhealthy, its neighboring contactor status intent should be set to open (0). This is written as

$$\square\{(r_1 = 0) \rightarrow (\tilde{c}_8 = 0)\}. \quad (23)$$

Similar specifications hold for all neighboring contractors to generators and rectifier units.

No Paralleling of AC Sources: For the single-line diagram in Figure 5, there are six AC generator pairs: $\{G_1, G_2\}$, $\{G_1, G_3\}$, $\{G_1, G_4\}$, $\{G_2, G_3\}$, $\{G_2, G_4\}$, and $\{G_3, G_4\}$. A live path for the pair $\{G_1, G_3\}$ exists if contactors C_1, C_5, C_6, and C_3 are all closed. To avoid instances of paralleling AC sources, the specification to disallow the live path would be

$$\square\neg\{c_1 = 1 \ \wedge \ c_5 = 1 \ \wedge \ c_6 = 1 \ \wedge \ c_3 = 1\}. \quad (24)$$

Similar specifications to avoid non-paralleling are implemented for all AC generator pairs.

Power Status of Buses: Consider AC bus B_1 in Figure 5. Bus B_1 will be powered if a live path exists between itself and a generator. This requirement is captured by the following

$$\square\{(c_1 = 1 \wedge g_1 = 1) \rightarrow b_1 = 1\}, \quad (25)$$

$$\square\{(c_5 = 1 \wedge b_2 = 1 \wedge c_2 = 1 \wedge g_2 = 1) \rightarrow (b_1 = 1)\}, \quad (26)$$

$$\square\{(c_5 = 1 \wedge b_2 = 1 \wedge c_6 = 1 \wedge b_3 = 1 \wedge c_3 = 1 \wedge g_3 = 1) \rightarrow (b_1 = 1)\}, \quad (27)$$

$$\square\{(c_5 = 1 \wedge b_2 = 1 \wedge c_6 = 1 \wedge b_3 = 1 \wedge c_7 = 1 \wedge b_4 = 1 \wedge c_4 = 1 \wedge g_4 = 1) \rightarrow (b_1 = 1)\}. \quad (28)$$

If none of the four conditions to the left side of the implication are true, then the status of bus B_1 is unpowered. Similar specifications hold for AC buses B_2, B_3, and B_4.

DC bus B_5 will be powered if it is connected to a healthy rectifier unit and a live path exists between itself and a powered AC bus. This requirement is captured by the following

$$\square\{(r_1 = 1 \wedge c_8 = 1 \wedge b_1 = 1) \rightarrow (b_5 = 1)\}, \quad (29)$$

$$\square\{(c_{10} = 1 \wedge b_6 = 1 \wedge r_2 = 1 \wedge c_9 = 1 \wedge b_4 = 1) \rightarrow (b_5 = 1)\}. \quad (30)$$

If neither of the above two conditions are true, then the status of B_5 is unpowered. Similar specifications hold for B_6.

Essential Buses: In this problem, we consider buses B_1 and B_4 to be connected to safety-critical loads, and can be unpowered for no longer than three time steps. Each increment of the clock variable θ_{B_1} and θ_{B_4} represents one time step $\delta = 1$. A safety specification for B_1 is of the following form:

- If bus status b_1 is unpowered, then at the next time step clock θ_{B_1} increments by one, such that

$$\square\{(b_1 = 0) \rightarrow (\bigcirc\theta_{B_1} = \theta_{B_1} + 1)\}. \quad (31)$$

- If bus status b_1 is powered, then at the next time step reset clock θ_{B_1} to zero. This is written as

$$\square\{(b_1 = 1) \rightarrow (\bigcirc\theta_{B_1} = 0)\}. \quad (32)$$

- To ensure that the status of B_1 is never unpowered for more than 5 steps, (16) becomes

$$\square\{\theta_{B_1} \leq 3\}. \quad (33)$$

The requirement that all DC buses must always remain powered can be implemented by setting $T = 0$, or, more compactly, written as

$$\square\{b_5 = 1 \wedge b_6 = 1\}. \quad (34)$$

C. Results

Consider a system model with environment variables E that includes generators $G_1 - G_4$ and rectifier units R_1, R_2, system variables S consisting of contactors $C_1 - C_{10}$ and buses $B_1 - B_6$. Given environment assumption φ_e from (2)-(9), (20)-(22), and φ_s as the conjunction of all specifications from (23)-(33), the synthesis problem is finding a control protocol such that (17) holds. The output of the synthesis procedure

is a discrete planner represented as a finite-state automaton. States are pairs of system and environment states. If the system follows the transitions in the automaton, the system will satisfy its requirements under all allowable environment actions.

1) Centralized Controller Design: We now present the results for the centralized case of the electric power system design problem with variables and specifications discussed in the previous section. Figure 6 shows the simplified single-line diagram used in the problem formulation over a sequence of time. Each subfigure represents the step of the simulation, starting at step 0 and ending with step 5. For clarity of exposition, the results presented only implement a contactor delay for C_1, and all other contactors have instant actuation.

At each step, depicted in Figures 6a-6f, the statuses of environment variables can switch between healthy and unhealthy, subject to the assumption that at least one generator and one rectifier always remains healthy. Furthermore, because of the contactor delay implemented for C_1, once the status for generator G_1 becomes unhealthy, that status will remain unhealthy. A healthy generator or rectifier unit is denoted by a shaded component. Additionally, a shaded bus is powered, and a shaded contactor is closed.

Figure 6a shows the initial configuration at time $t = 0$. All four generator statuses are healthy, as are both rectifier units. Contactor statuses c_1, c_4, c_8, and c_9 are closed, and bus statuses b_1, b_4, b_5 and b_6 are powered. At time $t = 1$ (Figure 6b), the status for generator G_2 and rectifier R_1 become unhealthy. As a result, contactor status c_8 switches to open, c_{10} closes, and bus B_5 is powered through B_6. At time $t = 2$, generator status g_1 becomes unhealthy. Because of the contactor delay time for C_1, status c_1 does not open until $t = 4$. For safety-critical buses B_1 and B_4, their statuses are never unpowered for more than two time steps throughout the entire simulation sequence. This specification is not imposed on the middle two buses, however, and and thus b_3 can remain unpowered for five steps without violating any system requirements. Bus status b_4 remains unpowered for three time steps, and then becomes powered again at time $t = 5$.

The synthesis process produces a control protocol in the form of a finite state automaton. The resulting automaton for the electric power system centralized controller takes 258 seconds to solve on a 2.9 GHz Intel Core Duo processor with 8 GB memory, and has 2049 states.

2) Distributed Control Architecture: The physical decomposition of the electric power system for a distributed control architecture is shown in Figure 7. Let SYS_1 represent subsystem on the left, and SYS_2 the subsystem on the right. We now present results for two types of distributed control architectures: master/slave and bi-directional. The environment and system variables for the two subsystems are denoted by e_1, s_1, e_2 and s_2, respectively.

Master/Slave Control Architecture: For a master/slave architecture, power flow between the decomposed systems is controlled by one side. For the decomposition in Figure 7, SYS_2 is the "master" and can control the supply of power that can flow via contactor C_6 and C_{10}. SYS_1 is the "slave" and can only receive power when SYS_2 provides it. We decompose the global environment assumption, in which at least one power source must remain healthy at each step, such that

$$\varphi_{e_2} = \Box((g_3 = 1 \vee g_4 = 1) \wedge r_2 = 1); \;\; \varphi_{e_1} = \Box(true).$$

The specification for φ_{e_1} states that there are no restrictions on the behavior of φ_{e_1}. The assumption placed on φ_{e_2} ensures that for any execution $\sigma \in \Sigma$, the controller for SYS_2 is able to supply power to SYS_1 at any step. Health status information for G_1 and G_2 is sent to SYS_2 via a health status variable h_1. The variable is set to 0 if neither source is healthy, and 1 if either g_1 or g_2 is healthy. In addition, the status of rectifier R_1 from SYS_1 is also sent to SYS_2 as an information variable.

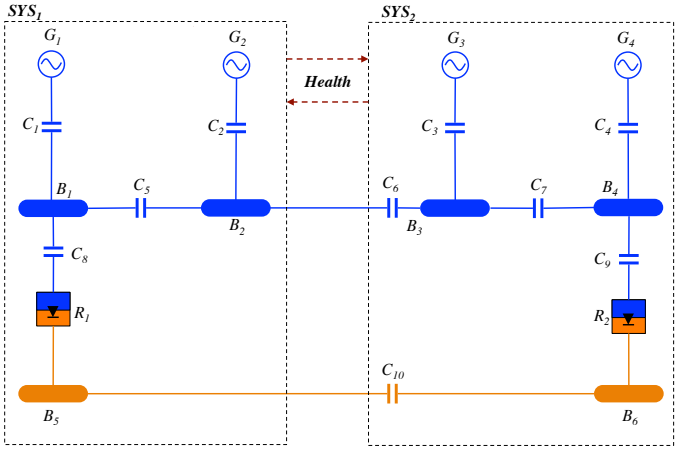
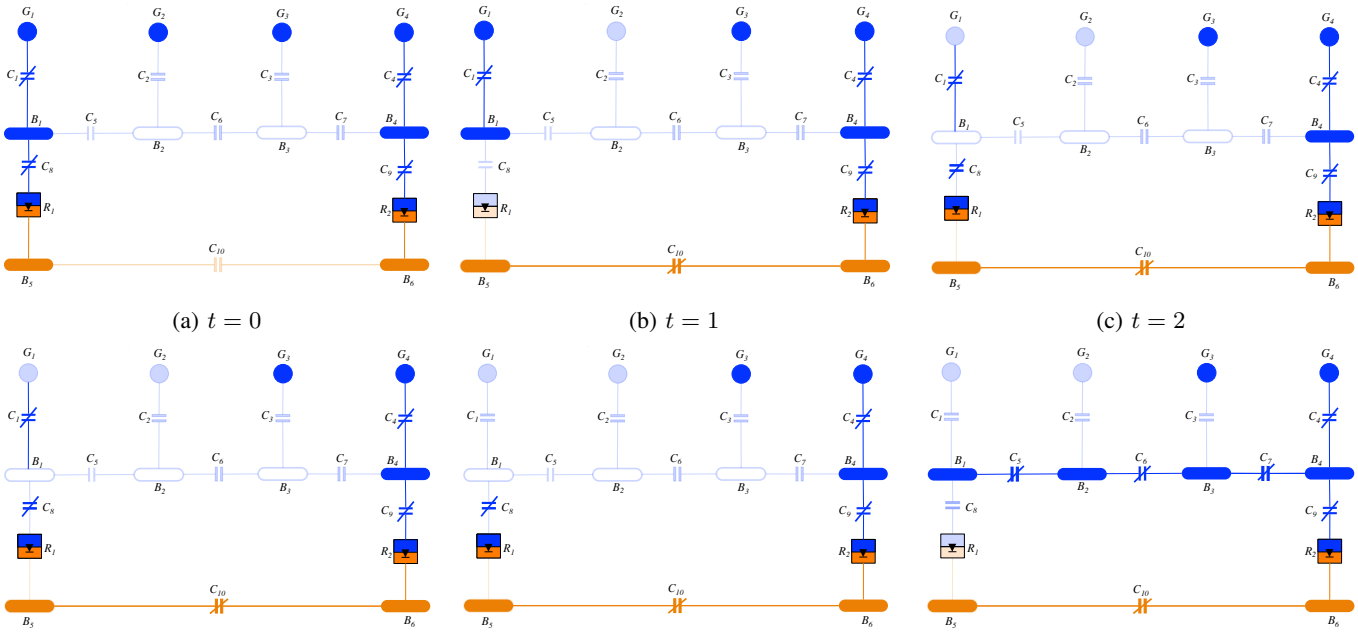


Fig. 7: A distributed controller decomposition. Components enclosed within the dashed rectangles are controlled by their own respective controllers. The dashed arrow represents information flow, in the form of a health status variable, directed from SYS_1 to SYS_2. The solid arrow represents the physical transfer of power from SYS_2 to SYS_1.

In order for the master/slave distributed synthesis problem to become realizable, additional interface refinements are needed. It is not enough for generators G_3 and G_4 to be able to generate power at all steps. The controller for SYS_2 must also be able to guarantee that power can be delivered to SYS_1. Thus, we introduce ϕ_2 as a guarantee for controller SYS_2, and denote ϕ_2' as an assumption for controller SYS_1. Because the master subsystem controls the flow of power, a single-sided refinement is sufficient for the design problem to be realizable, and we can set $\phi_1 = true$. The additional specification ϕ_2 imposes conditions on contactors C_6 and C_{10}, and on buses B_3 and B_6 (the components nearest to the interface of SYS_2 and SYS_1. These specifications are of the following form: bus status b_3 is never unpowered for a pre-specified period of time T. In other words, B_3 becomes an essential bus, and we introduce a variable θ_{B_3} that is used as a counter such that

$$\Box\{(b_3 = 0) \rightarrow (\bigcirc\theta_{B_3} = \theta_{B_3} + 1)\} \;\wedge$$
$$\Box\{(b_3 = 1) \rightarrow (\bigcirc\theta_{B_3} = 0)\} \;\wedge\; \Box\{\theta_{B_3} \leq T\}.$$

Moreover, because of the requirement that DC buses must always be powered, then maximum unpowered time $T = 0$ in order to guarantee that power can always be provided to SYS_1.



(a) $t = 0$ (b) $t = 1$ (c) $t = 2$

(d) $t = 3$ (e) $t = 4$ (f) $t = 5$

Fig. 6: A single simulation trace for a centralized controller for the electric power system single-line diagram case study. Each subfigure represents a single time step. Shaded generators and rectifier units are healthy, shaded contactors are closed, and shaded buses are powered.

If health status $h_1 = 0$, i.e., both g_1 and g_2 are unhealthy, then, whenever b_3 is powered, $\tilde{c}_6$ will be set to close

$$\square\{((h_1 = 0) \wedge (b_3 = 1)) \rightarrow (\tilde{c}_6 = 1)\}.$$

Similarly, if health status $r_1 = 0$, then $\tilde{c}_{10}$ will be set to close.

Decentralized Control Architecture: Consider again the physical decomposition in Figure 7, where power can flow from either subsystem. The physical actuation of contactor C_6 is still controlled by the right side. The environment variables for SYS_1 include statuses g_3, g_4, r_2, c_6, and c_{10}, while environment variables for SYS_2 contain statuses g_1, g_2, b_2, and r_1. Note that this differs from the master/slave control architecture with the necessary addition of b_2 as an environment variable to allow for power to flow in two directions, as well as additional environment variables. Statuses g_1 and g_2 are combined into health variable h_1, while g_3 and g_4 are combined to form h_2.

The case where there is power flow between SYS_1 and SYS_2 corresponds to an interconnection where part of the output of each system acts as an environment variable for the other, i.e., both ϕ_1 and ϕ_2 are non-trivial. In order to ensure that the interconnection is well-posed, i.e., the interconnected system avoids deadlock, environment variables should be partitioned into external and feedback parts. For SYS_1, external environment variables are g_1, g_2 and r_1, while the feedback environment is contactor status c_6 and c_{10}. In order for the system to be well-posed, decisions made by the controller for SYS_1 at step t must use the status of c_6 at the previous step $t - 1$. A deadlock situation can occur between subsystems if this time shift is not accounted for, where each subsystem waits on an action from the other subsystem before it can make a move. Due to the issue of well-posedness in the decentralized architecture, additional specifications are introduced in order to make the problem realizable. In order to successfully synthesize controllers for each subsystem, the following specifications are imposed.

- For SYS_2, if neither g_3 nor g_4 is healthy, then require bus status b_2 to be powered. This is written as

$$\square\{g_3 = 1 \vee g_4 = 1 \vee b_2 = 1\}.$$

- If neither g_1 nor g_2 is healthy, then intent status $\tilde{c}_6$ should be set to deliver power from SYS_2. This is written as

$$\square\{g_1 = 0 \wedge g_2 = 0 \rightarrow (\tilde{c}_6 = 1)\}.$$

- If either r_1 or r_2 is unhealthy, then intent status $\tilde{c}_{10}$ should be set to close.

$$\square\{r_1 = 0 \rightarrow \tilde{c}_{10} = 1\}.$$

Because power must be able to be delivered to both subsystems, safety-critical buses are moved to those buses nearest the interface, i.e., to B_2 and B_3. In order to enforce well-posedness, specifications for the controller for SYS_1 involving C_6 are defined with additional next operators to implement a shift in time step. For the decentralized synthesis problem to be realizable, contactor delays are thus omitted in this problem formulation in order avoid conflicting specifications. The resulting controllers for each subsystem take 5 seconds to synthesize and result in an automaton with 128 states.

VII. Conclusion and Future Work

The use of a specification language to capture requirements and a synthesis procedure to construct control protocols automates the design process, allowing for faster design times and more efficient ways to identify errors. This paper demonstrates how text-based specifications can be captured using a temporal

logic specification language with timing constraints in a representative case study. Given a topology for an electric power system and a set of system requirements, we automatically synthesize a control protocol for an electric power system on an aircraft. The controller reacts to changes in the environment and is guaranteed, by construction, to satisfy the desired properties even in the presence generator failures.

The computational complexity of synthesis makes solving industrial scale problems difficult for current tools. We thus examined the use of distributed control protocols, which take less computational time to synthesize due to fewer components within each subsystems, and thus smaller state spaces. They are, however, more conservative than a centralized controller in terms of length of time non-essential buses are powered. Buses closer to the interfaces between subsystems are now powered for longer lengths of time in order to anticipate power requests from the other subsystem.

From the basis of the work in this paper there are a number of directions for both practical and theoretical future work that apply to electric power systems as well as other application areas that span other networked control systems. The decomposition of overall system specifications into subsystem specifications, including interface assumptions and guarantees, is currently generated in an ad hoc manner. Future work will focus on automating this process. Timing specifications, (e.g., safety and contactor open/closing times) in the electric power system problem are addressed with the use of clocks by way of an additional counter variable. This discretization of time further adds to the difficulties arising from state space explosion. We are currently examining the use of timed verification and synthesis tools, such as UPPAAL-TIGA [29]. One open issue not addressed is the level of abstraction needed for modeling, design, and specifications of an electric power system. Control of the power quality from generators is considered at a continuous level of abstraction. Load management and load shedding are considered at a discrete low-level of abstraction. Both of these problems, although at different levels of abstraction, should be interfaced with the primary distribution problem discussed in this paper.

ACKNOWLEDGMENTS

The authors wish to thank Necmiye Ozay from the University of Michigan, and Rich Poisson and Eelco Scholte from United Technologies Aerospace Systems for their insight and helpful discussions.

REFERENCES

[1] J. Rosero, J. Ortega, E. Aldabas, and L. Romeral, "Moving towards a more electric aircraft," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 22, no. 3, pp. 3–9, 2007.

[2] J. Zumberge, J. Wolff, K. McCarthy, T. O'Connell, J. Zumberge-AFRL, J. Wolff-AFRL *et al.*, "Integrated aircraft electrical power system modeling and simulation analysis," *SAE Technical Paper*, pp. 01–1804, 2010.

[3] T. Wu, S. Bozhko, G. Asher, and D. Thomas, "Fast functional modelling of the aircraft power system including line fault scenarios," in *Power Electronics, Machines and Drives (PEMD 2010), 5th IET International Conference on*. IET, 2010, pp. 1–7.

[4] P. Nuzzo, H. Xu, N. Ozay, J. B. Finn, A. L. Sangiovanni-Vincentelli, R. M. Murray, A. Donzé, and S. A. Seshia, "A contract-based methodology for aircraft electric power system design," *IEEE Access*, under review.

[5] T. Wongpiromsarn, U. Topcu, and R. Murray, "Formal synthesis of embedded control software: Application to vehicle management systems," in *Proceedings of the AIAA Infotech Aerospace Conference*, 2011.

[6] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," in *Verification, Model Checking, and Abstract Interpretation*. Springer, 2006, pp. 364–380.

[7] A. Pnueli, "Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends," *Current trends in Concurrency*, pp. 510–584, 1986.

[8] A. Galton, *Temporal logics and their applications*. Academic Press London, 1987.

[9] G. J. Holzmann, "The model checker spin," *Software Engineering, IEEE Transactions on*, vol. 23, no. 5, pp. 279–295, 1997.

[10] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *Automatic Control, IEEE Transactions on*, vol. 53, no. 1, pp. 287–297, 2008.

[11] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Robotics, IEEE Transactions on*, vol. 25, no. 6, pp. 1370–1381, 2009.

[12] N. Ozay, U. Topcu, and R. M. Murray, "Distributed power allocation for vehicle management systems," in *Proc. IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 4841–4848.

[13] N. Ozay, U. Topcu, R. M. Murray, and T. Wongpiromsarn, "Distributed synthesis of control protocols for smart camera networks," in *Cyber-Physical Systems (ICCPS), 2011 IEEE/ACM International Conference on*. IEEE, 2011, pp. 45–54.

[14] T. Laengle, T. C. Lueth, and U. Rembold, "A distributed control architecture for autonomous robot systems," *Series in Machine Perception and Artificial Intelligence*, vol. 21, pp. 384–402, 1995.

[15] A. Shalal-Esa. (2012, March) Pentagon says F-35 fighter delayed, costs rise 4.3 percent. http://articles.chicagotribune.com/2012-03-29/news/sns-rt-us-lockheed-fighterbre82t03r-20120329_1_f-35-costs-rise-pentagon-report. Chicago Tribune.

[16] S. Creedy. (2013, October) Dreamliner's slow flight. http://www.theaustralian.com.au/news/features/dreamliners-slow-flight/story-e6frg6z6-1226733805717. The Australian.

[17] I. Moir and A. Seabridge, *Aircraft Systems: Mechanical, Electrical, and Avionics Subsystems Integration*. AIAA Education Series, 2001.

[18] R. G. Michalko, "Electrical starting, generation, conversion and distribution system architecture for a more electric vehicle," Patent US 7 439 634, 10, 2008.

[19] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*. IEEE, 1977, pp. 46–57.

[20] Z. M. A. Pnueli, "The temporal logic of reactive and concurrent systems."

[21] C. Baier, J.-P. Katoen *et al.*, *Principles of model checking*. MIT press, 2008, vol. 26202649.

[22] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive (1) designs," *Journal of Computer and System Sciences*, vol. 78, no. 3, pp. 911–938, 2012.

[23] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," *Automatic Control, IEEE Transactions on*, 2012.

[24] A. Pnueli, Y. Saar, and L. Zuck, "Jtlv: A framework for developing verification algorithms," in *Computer Aided Verification*. Springer, 2010, pp. 171–174.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*. HSCC, 2011, pp. 313–314.

[26] M. Mukund, "From global specifications to distributed implementations," *Synthesis and control of discrete event systems*, pp. 19–34, 2002.

[27] E. Filiot, N. Jin, and J.-F. Raskin, "Antichains and compositional algorithms for ltl synthesis," *Formal Methods in System Design*, vol. 39, no. 3, pp. 261–296, 2011.

[28] P. Madhusudan and P. Thiagarajan, "Distributed controller synthesis for local specifications," *Automata, languages and programming*, pp. 396–407, 2001.

[29] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. Larsen, and D. Lime, "Uppaal-tiga: Time for playing games!" in *Computer Aided Verification*. Springer, 2007, pp. 121–125.