A Case Study on Reactive Protocols for Aircraft Electric Power Distribution

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

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

I. INTRODUCTION

Advances in electronics technology have made the transition from conventional to more-electric aircraft architectures possible. Conventional architectures utilize a combination of mechanical, hydraulic, electric, and pneumatic subsystems. The move towards more-electric aircraft increases efficiency by reducing power take-offs from the engines that would otherwise be needed to run hydraulic and pneumatic components. Moreover, use of electric systems provides opportunities for system-level performance optimization and decreases life-cycle costs.

Efforts have been made to re-use previously developed systems from conventional aircraft [3], but additional high-voltage networks and electrically-powered components increase the system's complexity, and thus new design approaches need to be considered. These electric power system designs must behave according to certain properties or requirements determined by physical constraints and performance criteria. Because safety of the aircraft is solely or mostly dependent on electric power, the electric power system needs to be highly reliable, fault tolerant, and autonomously controlled. Previous work in this area has focused on the analysis of aircraft performance and power optimization by using modeling libraries and simulations [2], [4], [5]. Analysis of all faults or errant behaviors in these models is difficult due to the high complexity of these systems. This has led to a greater emphasis on the use of formal methods to aid in safety and performance certification.

System requirements are typically text-based lists, oftentimes ambiguous in intent or inconsistent with each other. The process of verifying the correctness of a system with respect to these specifications is expensive, both in terms of cost and time. Building on recent work on formal synthesis of vehicle management systems [15] and distributed synthesis of control protocols [8], [9], we take the initial step toward automatically converting text-based system requirements into a specification language, synthesizing centralized and distributed control protocols for an aircraft electric power system, and examine design tradeoffs between these different control architectures. The remainder of the paper is structured as follows: We describe the standard components in an electric power system and the general problem description in Section II. Section III gives a technical description of the specification language and synthesis procedure. Sections IV presents a case study of an electric power system, which is followed by results, and concluding remarks.

This work was supported in part by the FCRP consortium through the Multiscale Systems Center (MuSyC), the Boeing Corporation, and AFOSR Award FA9550-12-1-0302.

[1]Mechanical Engineering, California Institute of Technology
[2]Electrical and Systems Engineering, University of Pennsylvania
[3]Controls and Dynamical Systems, California Institute of Technology

II. ELECTRIC POWER DISTRIBUTION SYSTEM

Next-generation aircraft will have increased safety-criticality reliant on the electric power system and increased number of overall components in the electric power system, raising the complexity of design. In this paper, we investigate an alternative way for the design of control protocols for electric power systems.

A. System Components

The single-line diagram in Fig. 1 includes a combination of generators, contactors, buses, and loads. The following is a brief description of the components referenced in the primary power distribution single-line diagram [6].

AC and DC buses deliver power to both high-voltage and low-voltage loads. (Note: Individual loads are not depicted in the single-line diagram.) Buses can be essential or non-essential. Essential buses supply loads that should always remain powered, while non-essential buses supply loads that may be shed in the case of a fault. *Generators* supply power to buses, and can operate at either high or low-voltages. *Contactors* are electronic switches that connect the flow of power from sources to buses and loads (represented by $\dashv\vdash$). They can reconfigure (i.e., switch between open and closed) by commands from one or multiple controllers. *Rectifier units* convert AC power to DC power, *transformers* step down a high-voltage to a lower one, and a *transformer rectifier unit* both converts AC to DC and lowers the voltage. *Batteries* provide short-term power in case of an emergency.

B. Problem Description

Given the topology of an electric power system, the main design problem is determining all correct configurations of contactors for all flight conditions and faults that can occur. A

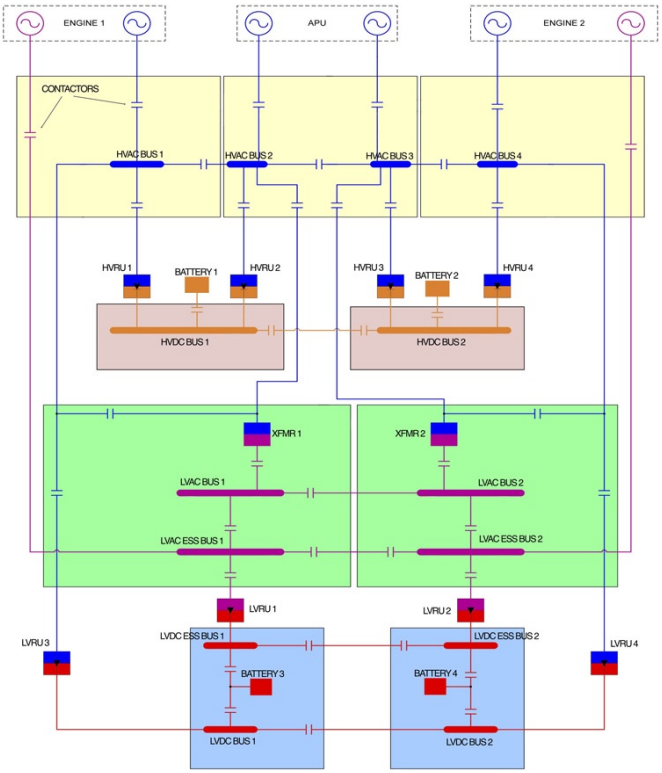

Fig. 1. Single line diagram of an electric power system adapted from a Honeywell, Inc. patent [7]. Two high-voltage generators, APU generators, and low-voltage generators serve as power sources for the aircraft. Depending on the configuration of contactors, power can be routed from sources to buses through the contactors, rectifier units, and transformers. Buses are connected to subsystem loads. Batteries can be used to provide emergency backup power to DC buses.

"correct" configuration satisfies all system requirements, also referred to as specifications. We now discuss a few sample specifications relevant to the problems found in Fig. 1.

Safety specifications characterize the way each bus can be powered and the length of time it can tolerate power losses. First, we disallow any paralleling of AC sources (i.e., no bus should be powered by multiple AC generators at the same time.) Second, essential loads, such as flight-critical actuators, should never be unpowered for more than 50 msec. Lastly, typical contactor opening and closing times can range between 10-25 msec [6]. *Performance* specifications rank desired system configurations. A generator priority list is assigned to each bus specifying the order of sources each bus should be powered. If the first priority generator is unavailable, then it will be powered from the second priority generator, and so forth.

III. Formal Specification and Synthesis

We now discuss a formal specification language utilized for the synthesis of control protocols later in this section.

A. Formal Specification Using Linear Temporal Logic

In reactive systems, correctness will depend not only inputs and outputs of a computation, but on executions of the system. Temporal logic is a branch of logic that incorporates temporal aspects to reason about propositions in time, and was first used as a specification language by Pneuli [11]. In this paper, we consider a version of temporal logic called linear temporal logic (LTL) [1]. We first define an atomic proposition, LTL's main building block.

A system consists of a set V of variables. The domain of V, denoted by $dom(V)$, is the set of valuations of V. An *atomic proposition* is a statement on a valuation of system variables that has a unique truth value (*True* or *False*) for a given value v. Let $v \in dom(V)$ be a state of the system and p be an atomic proposition. Then v satisfies p, $v \models p$, if p is *True* at the state v. Otherwise, $v \not\models p$.

For a set π of atomic propositions, any atomic proposition $p \in \pi$ is an LTL formula. Given LTL formulas φ and ψ over π, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi\,\mathcal{U}\,\psi$ are also LTL formulas. LTL formulas over π are interpreted over infinite sequences of states. Formulas involving other operators can be derived from these, including *eventually* ($\Diamond$) and *always* ($\Box$). We refer the reader to [1] and references therein for a more detailed discussion of LTL.

B. Reactive Synthesis

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 an LTL specification φ of assume-guarantee form $\varphi = (\varphi_e \rightarrow \varphi_s)$ where φ_e characterizes the assumptions on the environment and φ_s characterizes the system requirements. The synthesis problem is then concerned with constructing a strategy which 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 and environment that attempts to falsify the specification and a controlled plant that tries to satisfy it.

For general LTL, the synthesis problem has a doubly exponential complexity [12]. A subset of LTL, namely generalized reactivity (1) (GR(1)), can be solved in polynomial time [10]. GR(1) specifications restrict φ_e and φ_s to take the following form, for $\alpha \in \{e, s\}$,

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

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.

Given a GR(1) specification, the digital design synthesis tool implemented in JTLV (a framework for developing temporal verification algorithm) [13] generates a finite-state 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 as discussed in provides an interface to JTLV [16]. For examples discussed in this paper, we use TuLiP.

C. Distributed Synthesis

We follow the exposition in [9] for the distributed synthesis problem. For ease of representation, consider the case of two subsystems. The set of variables and global specification $\varphi_e \rightarrow \varphi_s$ decomposed as follows:

Let $\varphi_e, \varphi_{e_1}, \varphi_{e_2}, \varphi_s, \varphi_{s_1}$, and φ_{s_2} be LTL formulas containing variables only from their respective sets of environment variables E, E_1, E_2 and system variables S, S_1, S_2. If the following conditions hold: (1) any execution of the environment that satisfies φ_e also satisfies $(\varphi_{e_1} \wedge \varphi_{e_2})$, (2) any execution of the system that satisfies $(\varphi_{s_1} \wedge \varphi_{s_2})$ also satisfies φ_s, and (3) there exist two control protocols that make the local specifications $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ true. Then, by a result in [9], implementing these two control protocols together leads to a system where the global specification $\varphi_e \rightarrow \varphi_s$ is met. See Section V-B.2 for an example of such a refinement and [9] for restrictions on distributing local environment and system variables.

IV. Synthesis of Reactive Protocols for Aircraft Electric Power Distribution

We address the problem of primary distribution in an electric power system by examining a simplified version of the single-line diagram. Fig. 2 shows a topology that consists of the basic high-voltage AC components: two generators powered from engines and two generators powered by the APU, all of which are connected to four buses via seven contactors. Note: With abuse of notation, we refer to APU powered generators as APUs in the rest of the paper.

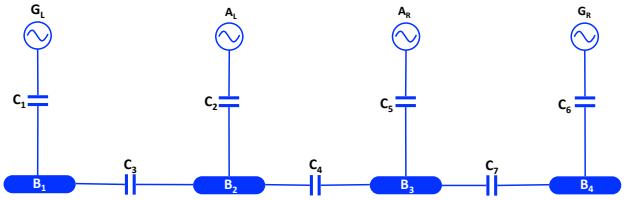

Fig. 2. Simplified single-line diagram used in the centralized problem. Four power sources connect to four buses through a series of seven contactors

A. Variables

Environment variables include the health statuses of the left and right generators (G_L, G_R) and APUs (A_L, A_R) can each take values of healthy (1) and unhealthy (0). These statuses are uncontrollable and may change at any point in time. *Controlled* variables are contactors connecting generators and APUs to buses, (C_1, C_2, C_5, C_6). They can each take values of open (0) or closed (1). A closed contactor will allow power to pass through, while an open one does not. Contactors between buses (C_3, C_4, C_7) can take three values. A value of 0 again denotes an open contactor. A value of -1 or 1 signifies a contactor is closed and that power is flowing from right to left, or left to right, respectively. *Dependent* variables are buses B_1, B_2, B_3, and B_4 that can be either powered (1) or unpowered (0). Bus values will depend on the status of their neighboring contactors, buses, as well as the health status of connecting generators or APUs.

Timing considerations play a key part in the specifications for an electric power system. LTL, however, only addresses the notion of temporal ordering of events. In order to reconcile this discrepancy, the variable $\tilde{C}_i$ for $i \in [1, 7]$ is introduced to represent the controller intent for contactor C_i. If a fault occurs, the controller sets the intent for a contactor based on the status of its neighboring generator or bus. An action on contactor status occurs immediately or one time step later. Note that timing is addressed in an ad-hoc manner. An alternative approach, currently under study, is using timed specification languages, e.g., computation tree logic (CTL), and appropriate synthesis tools, e.g., UPPAAL-Tiga [14].

B. Formal Specifications

Given the topology in Fig. 2, the following lists the temporal logic specifications used.

Environment Assumption: At least one power source is healthy at any given time, written as $\square\{(G_L = 1) \vee (A_L = 1) \vee (A_R = 1) \vee (G_R = 1)\}$.

Power Status of Buses: A bus can only be powered if a contactor is closed and its connecting generator, APU, or neighboring bus is powered. If B_1 is powered if one of two properties holds: G_L is healthy and C_1 is closed, or B_2 is powered and C_3 is closed. If neither of these two are true, then bus B_1 will be unpowered. Specifically:

- $\square\{((C_1 = 1) \wedge (G_L = 1)) \rightarrow (B_1 = 1)\}$
- $\square\{((B_2 = 1) \wedge (C_3 = -1)) \rightarrow (B_1 = 1)\}$
- $\square\{\neg((C_1 = 1) \wedge (G_L = 1)) \vee ((B_2 = 1) \wedge (C_3 = -1)) \rightarrow (B_1 = 0)\}$

A similar set of specifications is applied for B_2, B_3, and B_4.

No Paralleling of AC Sources: One way to avoid paralleling is to explicitly enumerate and eliminate all bad configurations. In Fig. 2, paralleling can occur if G_L and A_L are both healthy, and contactors C_1, C_2, and C_3 are all closed. A specification could be to disallow closing C_1, C_2, and C_3 at the same time. This "global" approach becomes difficult to scale when the number of paths and components grows large. We take a "localized" view; instead of examining entire paths, we focus on the source of power coming into each bus. To this end, we first introduce "power flow direction" to contactors C_3, C_4, and C_7. The contactors connecting generators and APUs are strictly unidirectional. We restrict the value of contactors based on the direction in which power may flow depending on the health and status of surrounding buses/sources. For these bidirectional contactors, if the neighboring two nodes (either a generator, APU or bus) is unpowered, then the contactor cannot direct power in the opposite direction those nodes. Note that directionality within the contactor is not present in the physical implementation of hardware. A contactor is either open or closed.

If the left generator is unhealthy, then contactor C_3 cannot direct power from left to right, and the intent variable $\tilde{C}_3$ should be assigned accordingly. If the following properties are not true: (1) the left generator is healthy an bus B_2 is powered, or (2) Buses B_2 and B_3 are powered, then contactor C_3 cannot direct power from right to left. This can be written:

- $\square\{\neg(G_L = 1) \rightarrow \neg(\tilde{C}_3 = 1)\}$

- $\square\{\neg(((G_L = 1) \ \wedge \ (B_2 = 1)) \vee((B_3 = 1) \ \wedge \ (B_2 = 1))) \rightarrow \neg(\tilde{C}_3 = -1)\}$

Given direction of flow in contactors, we can examine each bus and eliminate any configuration of contactors which may allow for paralleling of sources. For example, the following configurations are not allowed for bus B_2.

- $\square\{\neg((C_2 = 1) \ \wedge \ (C_3 = 1))\}$
- $\square\{\neg((C_2 = 1) \ \wedge \ (C_4 = -1))\}$
- $\square\{\neg((C_3 = 1) \ \wedge \ (C_4 = -1))\}$

Safety-Criticality of Buses: In this problem we consider buses B_1 and B_4 to be safety-critical buses, and can be unpowered for no longer than five time steps. This is implemented through an additional clock variable t for each bus, where each "tick" of the clock represents 10 msec. A safety specification for B_1 would be: If B_1 is unpowered, then at the next time step clock t_1 increases $\square\{(B_1 = 0) \rightarrow (\bigcirc t_1 = t_1 + 1)\}$. If B_1 is powered, then at the next time step reset clock t_1 $\square\{(B_1 = 1) \rightarrow (\bigcirc t_1 = 0)\}$. Then, ensure that bus B_1 is never unpowered for more than 5 steps $\square\{t_1 \leq 5\}$.

Unhealthy Buses: A bus connected to an unhealthy source will create a short-circuit failure, leading to excessive electrical currents, overheating, and possible fires. We require that a contactor open when a generator or APU becomes unhealthy to avoid such failures. An example specification for the intent of contactor C_1 would be: $\square\{(G_L = 0) \rightarrow (\tilde{C}_1 = 0)\}$.

Prioritization: We thus introduce the notion of prioritization on power sources. Generators G_L and G_R, if healthy, will always be connected and used to power left and right side buses, respectively. APUs A_L and A_R are only connected if their respective left and right generator is unhealthy. This corresponds to a notion of nearest generator (in distance). In the below example, contactor C_2 is only closed if the left generator goes unhealthy. This can be written as: $\square\{((G_L = 0) \ \wedge \ (A_L = 1)) \rightarrow (\tilde{C}_2 = 1)\}$.

V. Results

We apply the approach presented in Section III-B for a specification of the form $\varphi_e \rightarrow \varphi_s$. The output of TuLiP includes a high-level discrete planner represented as a finite-state automaton whose states are pairs of system states and environment states. This section presents some preliminary results for the formal reactive synthesis of control protocols in an electric power system for centralized and distributed controllers.

A. Centralized Controller Design

Fig. 3 shows the simplified single-line diagram overlaid with a sample simulation run. The horizontal axis of each graph in the figure represents the step of the simulation, starting at step 0 and ending with step 5. The resulting automaton for the centralized controller takes roughly one minute to solve on a Mac Powerbook with a 2 GHz Intel Core Duo processor, and has approximately 200 states.

The four graphs in row 1 correspond to the statuses of the environment variables. These values are arbitrarily input, subject to constraints imposed by the environment assumptions. At each step, generators and APUs can switch between healthy and unhealthy as long as at least one source remains healthy. Graphs in rows 2 and 3 correspond to the contactor statuses generated from the synthesized control protocol. Because power can only flow from a generator or APU, the graphs for the contactors shown in row 2 can only take values of open or closed. Row 3 graphs, however, can take three values corresponding to open, closed with power directed to the right, or closed with power flowing to the left. Graphs in row 4 correspond to the four buses, and the vertical axis represents the power status of the bus. Because buses are dependent variables, these values are determined by the environment variables as well as the contactor configurations.

To better understand the results shown in Fig. 3 let us examine the simulation graphs for a single step, namely step 2. The left generator G_L is unhealthy and contactor C_1 is open. The left APU A_L is healthy, and contactor C_2 is closed. Bus B_2 is powered because it is connected to A_L. Bus B_1 is unpowered because both neighboring contactors C_1 and C_3 are open. Meanwhile, the right generator G_R is healthy and contactor C_6 is closed. Therefore, according to the second set of specifications from Section IV-B, bus B_4 is powered. Note, however, that C_5 remains closed even though the right APU is unhealthy. In the previous step, A_R was healthy, and its intent to open $\tilde{C}_5$ in step 2 does not get implemented until step 4. In order to ensure non-paralleling of sources, contactor C_7 must remain open at step 2 because C_5 is closed, even though no power is flowing from the APU. As a result, bus B_3 is unpowered.

Safety-critical buses B_1 and B_4 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. In addition, at no time in the simulation run are AC sources paralleled. Consider, for example, power flowing to bus B_1. When contactor C_1 is closed (steps 0,1, and 4), C_3 is always open.

B. Distributed Control Architecture

In the following section, we decompose the centralized electric power system topology into two smaller subsystems and synthesize local controllers. When implemented together these controllers are guaranteed to be correct with respect to the global specification. The physical decomposition of the electric power system is shown in Fig. 4. Let S_r represent the right subsystem (enclosed in the dotted lines) and S_l represent the left subsystem. The environment and system variables for S_l and S_r are denoted by e_r, s_r, e_l and s_l, respectively. Based on the refinement technique mentioned in Section III-C, the global specification discussed in Section IV-B is satisfied if the following are true

$$\phi_r \wedge \varphi_{e_l} \rightarrow \varphi_{s_l} \wedge \phi_l, \tag{1}$$

$$\phi_l \wedge \varphi_{e_r} \rightarrow \varphi_{s_r} \wedge \phi_r, \tag{2}$$

where formulas ϕ_r and ϕ_l represent additional assumptions and guarantees made at the interface between the left and the right subsystems in order to ensure that the global system is

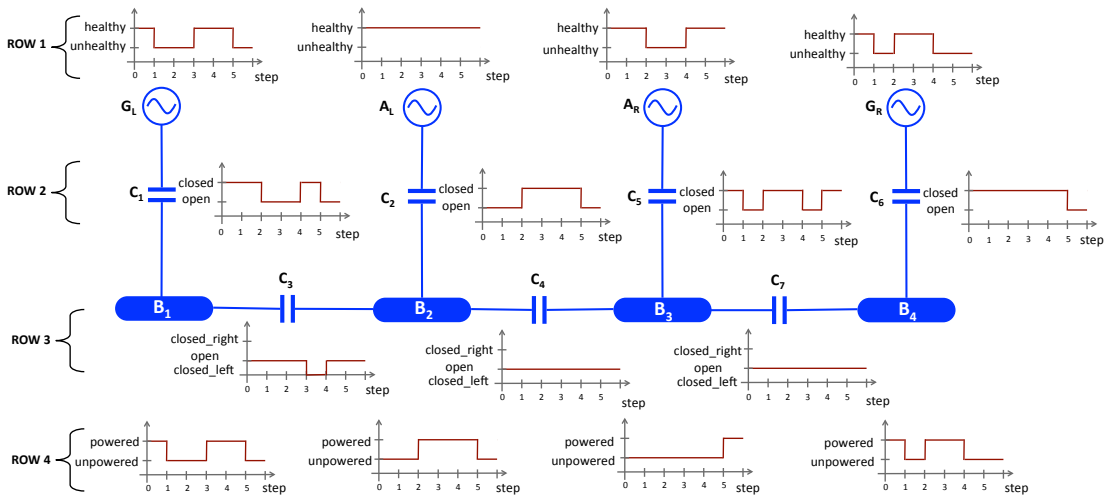

Fig. 3. A simulation result for a centralized controller for the electric power system. The horizontal axis represents the simulation step. Row 1 shows the environment inputs for generator and APU health. Based on these values, the controller values for contactors are set to either open or closed, as seen in Row 2. Additionally, Row 3 shows the direction of power flow through contactors C_3, C_4, and C_7. Row 4 shows the power status for all buses.

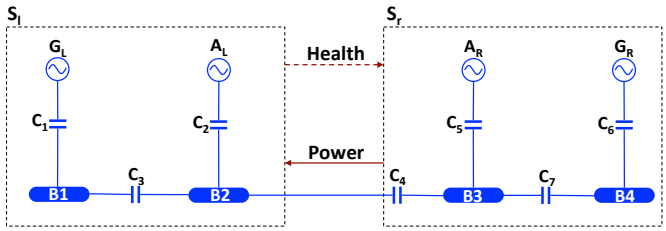

Fig. 4. A distributed controller decomposition for the electric power system. S_l sees contactor C_4 as an environment variable, provides the health status of its generator and APU as information to the right side. S_r has control of C_4, which enables the flow of power between the two subsystems.

realizable. Because local distributed controllers are limited in the information they know about other subsystems, these additional interface guarantees are necessary to avoid deadlock cases and ensure realizability. ϕ_l is a guarantee from subsystem S_l and seen as an environment assumption by the controller for subsystem S_r. Similarly, ϕ_r is a guarantee from S_r and an environment assumption in S_l. Specifications for these interface refinements will be stated in the following. We now present results for two different types of distributed control architectures: master/slave and bidirectional.

1) Master/Slave Control Architecture: For a master/slave architecture, power flow between the decomposed systems is controlled by one side, and unidirectional only. For the decomposition shown in Fig. 4, the subsystem S_r is the "master" and can control the supply of power that can flow from right to left via contactor C_4. Subsystem S_l is the "slave" and can only receive power when S_r provides it.

We decompose the global environment assumption such that $\varphi_{e_r} = \square(A_R = 1 \vee G_R = 1)$ and $\varphi_{e_l} = true$. This ensures that for any execution $\sigma \in \Sigma$, the controller for S_r is able to supply power to S_l at any step. The left generator and APU health statuses are sent to the right side via a health variable H_1. The variable is set to 0 if neither source is healthy, and is set of 1 if either G_L or A_L is healthy such that φ_{e_r} can assume knowledge about the health status of the left side.

In order for the master/slave distributed synthesis problem to become realizable, additional assumptions and guarantees (i.e., interface refinements) need to be implemented. It is not enough for power sources G_R and A_R to be able to generate power at all steps. The controller for S_r must also be able to guarantee that power can be delivered to the left. Thus, we introduce ϕ_r as a guarantee for the S_r controller, and as an assumption for S_l controller. Because the master controls the flow of power, a single-sided refinement is sufficient for the design problem to be realizable, and we can set $\phi_l = true$. The additional specification ϕ_r imposes conditions on contactor C_4 and bus B_3 (the components nearest to the interface of S_r and S_l. These specifications are

- Bus B_3 is never unpowered for a set number of time steps n. Essentially, B_3 becomes a safety-critical bus, and we introduce a clock variable t_3 to monitor the power status. $\square\{(B_3 = 0) \rightarrow (\bigcirc t_3 = t_3 + 1)\} \wedge \square\{(B_3 = 1) \rightarrow (\bigcirc t_3 = 0)\} \wedge \square\{t_3 \leq n\}$
- If health status $H_1 = 0$ (i.e., both G_L and A_L are unhealthy), then whenever B_3 is powered, C_4 will close. $\square\{((H_1 = 0) \wedge (B_3 = 1)) \rightarrow (\tilde{C}_4 = -1)\}$

A similar modification must be made for the case where unidirectional power flows from S_l to S_r. In both of the cases discussed in the master/slave architecture, all other specifications remain the same as those discussed from Section IV-B and decomposed with their respective components. Simulation results are comparable to those for the centralized controller, shown in Fig. 3, and thus omitted.

2) Bidirectional Power Flow Control Architecture: Consider again the physical decomposition shown in Fig. 4, where power is allowed to flow from either left to right, or right to left. The physical actuation of middle contactor C_4 is still controlled by the right side. The environment variables for S_l include G_L, A_L, and C_4, while environment variables for S_r contain G_R, A_R, B_2, and H_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.

The case where there is power flow between S_l and S_r corresponds to a feedback interconnection where part of the output of each system acts as an environment variable for the other (i.e., both ϕ_l and ϕ_r are non-trivial). In order to ensure that the interconnection is well-posed (i.e., the interconnected system avoids deadlocks), the environment variables should be partitioned into external and feedback parts. For subsystem S_l, external environment variables are G_L and A_L, while the feedback environment e_f is the status of contactor C_4. In order for the system to be well-posed, decisions made by the controller for S_l at step t must use the value of C_4 at the previous step $t-1$.

Realizability is more difficult to achieve for the bidirectional case due to the issue of well-posedness. In order to successfully synthesize controllers for each subsystem, the following guarantees/assumptions are imposed: For S_r, if neither G_R nor A_R is healthy, then bus B_2 is powered $\phi_r = \square\{G_R = 1 \vee A_R = 1 \vee B_2 = 1\}$. For S_l, if neither G_L nor A_L is healthy, then power will be delivered through C_4 $\phi_l = \square\{G_L = 0 \wedge A_L = 0 \rightarrow (C_4 = -1)\}$.

Because power must be able to be delivered to the other subsystem when needed, safety-critical buses are moved to those buses nearest the interface (e.g., B_2 and B_3.) In order to enforce well-posedness (i.e., to avoid deadlock), specifications for the controller for S_l involving C_4 are defined with additional next operators to implement a shift in time step. For the bidirectional synthesis problem to be realizable, contactor delays are thus omitted in this problem formulation in order avoid conflicting specifications. Each automaton in the distributed case is approximately 90 states and takes less than one minute to solve.

A centralized controller has complete knowledge of all component statuses. It can anticipate the behavior of the entire environment, and thus control protocols can be less conservative (e.g., longer delays in contactor switching times). For large-scale systems, though, distributed synthesis can be solved faster (due to the smaller number of components) and are thus more scalable. Yet additional interface refinements are required. These refinements include more conservative contactor and bus configurations. For the bidirectional distributed case in which refinements ϕ_l and ϕ_r are needed, well-posedness conditions further restrict the system. Contactor delays are no longer possible, and additional specifications are imposed on all components along the interfaces.

VI. Conclusion and Future Work

This paper demonstrates how text-based specifications can be translated into a temporal logic specification language and used to automatically synthesize a control protocol for an electric power system on a more-electric aircraft. The resulting controller is guaranteed, by construction, to satisfy the desired properties even in the presence of an adversary (i.e., changes in the environment.) We synthesized a centralized controller, and then refined the interface specifications for distributed control architectures. Distributed controllers are easier to synthesize due to fewer components, but are more conservative with respect to power usage due to lack of information of the entire system.

From the basis of the preliminary work in this paper, there are a number of potential directions. The first is addressing the full scale problem from Fig. 1. Current LTL specifications involve safety, but not liveness. Therefore, we may not need the full expressivity of GR(1), and other safety/reachability solvers may be used. Second, we explore utilizing timed synthesis tools such as UPPAAL-Tiga that may allow for larger problems to be synthesized using CTL as a specification language.

Acknowledgments

The authors wish to acknowledge Rich Poisson from Hamilton-Sundstrand and Necmiye Ozay for their helpful discussions.

References

[1] C. Baier, and J.P. Katoen, *Principles of Model Checking* MIT press, 1999.

[2] J. Bals, G. Hofer, A. Pfeiffer, and C. Schallert, "Virtual Iron Bird - a multidisciplinary modeling and simulation platform for new aircraft system architectures," in *German Aerospace Conference*, 2005.

[3] L. Faleiro, "Initial research towards a more electrical aircraft," in *More Electrical Aircraft Conference*, Royal Aeronautics Society, 2004.

[4] P. Krus, B. Johansson, and L. Austin, "Concept optimization of aircraft systems using scaling models," in *Recent Advances in Aerospace Actuation Systems and Components*, France, 2004.

[5] P. Krus and J. Nyman, "Complete aircraft system simulation for aircraft design - paradigms for modeling of complex systems," in *Proceedings of the 22nd International Congress of Aeronautical Sciences*, UK, 2000.

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

[7] R.G. Michalko, "Electrical starting, generation, conversion and distribution system architecture for a more electric vehicle," US Patent 7,439,634 B2, Oct. 21, 2008.

[8] N. Ozay, U. Topcu, T. Wongpiromsarn, and R.M. Murray, "Distributed synthesis of control protocols for smart camera networks," in *Intl. Conf. on Cyber-Physical Syst.*, 2011.

[9] N. Ozay, U. Topcu, and R.M. Murray, "Distributed power allocation for vehicle management systems," in *Proceedings of the 50th International Conference on Decision and Control*, 2011.

[10] N. Piterman, A. Pneuli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking and Abstract Interpretation*, vol. 3855, pp. 364-380. Springer-Verlag, 2006.

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

[12] A. Pnueli and R. Rosner, "Distributed reactive systems are hard to synthesize," in *Proceedings of the 31st IEEE Symposium Foundations of Comp. Sci.*, pp. 746-757, 1990.

[13] A. Pnueli, Y. Sa'ar, and L.D. Zuck, "JTLV: a framework for developing verification algorithms," in *Proceedings of the 22nd International Conference on Computer Aided Verification*, pp. 171-174, 2010.

[14] UPPAAL-Tiga, a synthesis tool for timed games. `http://people.cs.aau.dk/~adavid/tiga/`

[15] T. Wongpiromsarn, U. Topcu, and R.M. Murray,"Formal synthesis of embedded control software for vehicle management systems," in *AIAA Infotech@Aerospace*, 2011.

[16] 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*, 2011.