2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Distributed Power Allocation for Vehicle Management Systems

Necmiye Ozay Ufuk Topcu Richard M. Murray

Abstract— **We consider the problem of designing distributed control protocols -for aircraft vehicle management systems- that cooperatively allocate electric power while meeting certain higher level goals and requirements, and dynamically reacting to the changes in the internal system state and external environment. A decentralized control problem is posed where each power distribution unit is equipped with a controller that implements a local protocol to allocate power to a certain subset of loads. We use linear temporal logic as the specification language for describing correct behaviors of the system (e.g., safe operating conditions) as well as the admissible dynamic behavior of the environment due to, for example, wind gusts and changes in system health. We start with a global specification and decompose it into local ones. These decompositions allow the protocols for each local controller to be separately synthesized and locally implemented while guaranteeing the global specifications to hold. Through a design example, we show that by refining the interface rules between power distribution units, it is possible to reduce the total power requirement.**

I. Introduction and Motivation

Vehicle management systems (VMS) control and coordinate a number of subsystems of aerial vehicles, e.g., flight controllers, electrical systems, fuel management, environmental control systems, deicing units, and landing gear, and interface with additional aircraft subsystems, e.g., sensor pointing, data acquisition, and pilot and ground interfaces [1], [2]. Traditional VMS are typically based on federated architectures in which integrated hardware and software components realize independent or loosely interconnected functions [3]. Next generation VMS are expected to incorporate distributed computation, advanced networking, and increased levels of autonomous operations to manage physical resources, e.g., requirements on electric power generation and flows. Additionally, the move to autonomous flight will require the VMS to be interactive in dynamically changing environments and reconfigurable. Integrated modular avionics (IMA) architectures driven by these trends provide an alternative to federated architectures. The IMA architectures utilize high integrity, partitioned platforms that host multiple avionics functionalities of different criticalities. The IMA architecture is based on highly-integrated resource management among the functionalities that share the existing resources [4], [5] and leads to two competing trends: possibilities for system-level optimization by dynamically allocating resources (potentially leading to reductions in weight) come at the expense of extra layers of integration complexities.

Due to the increasing complexity of VMS, certification of safety and performance properties will necessitate use of formal specifications. Furthermore, systematic methods for verifying systems against these specifications and alternatively for synthesizing correct-by-construction control protocols will likely reduce the amount of costly validation experiments and tests. In this paper, building on our recent work [6], [7], we consider distributed synthesis of distributed control protocols that enable dynamic configuration for integrated power management in VMS. In particular, motivated by the transition to "more-electric" technologies, we focus on electric power management between a subset of the subsystems, namely flight controllers, deicing units, and environmental control. The main design considerations include: (i) Real-time reconfiguration in a dynamic environment: The subsystems interact with their environment (e.g., due to outside temperature variations and changes in flight conditions); hence, they need to react to the changes in their environment in real time. (ii) Fault tolerance: The power management systems should be able to reconfigure in the presence of faults or failures to satisfy its safety and performance requirements. (iii) Resource constraints: With the increase in the electric loads and introduction of integrated architectures, the subsystems share limited electric power resources. (iv) Mixed-criticality subsystems: The subsystems have varying levels of flight-criticality, e.g., flight controllers are highly critical whereas environmental control is of lower criticality. Therefore, the control protocol for power management needs to account for the prioritization of the loads from these subsystems while maintaining non-flight-critical criteria, e.g., certain measures of passenger comfort, within acceptable bounds.

We utilize linear temporal logic (LTL) as a formal specification language and expand our previous work on the synthesis of protocols for embedded controllers [8], [6], [7] to the compositional design of correct-by-construction, distributed protocols for dynamic configuration of integrated power management. Distribution of both the design and the implementation is considered to facilitate modularity of design, e.g., in contract based design [9], to reduce onboard power and information flows, and to alleviate the computational complexity of the synthesis procedure [7]. The output of the synthesis procedure is a hierarchical control protocol with a discrete planner responsible for the satisfaction of certain high-level specifications and a continuous control that implements the discrete plan at the lower level. In this paper, we focus on the high-level design problem; and assume that

This work was supported in part by the Multiscale Systems Center and the Boeing Corporation.

Authors are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA. necmiye, utopcu, murray@cds.caltech.edu

abstract discrete models of underlying continuous variables and dynamics are available. Such discrete models can be obtained by using abstraction techniques that have been proposed for representing the behavior of a continuous system with finitely many states and transitions among them which capture the relevant dynamics (see, for instance, [10], [11], [8]).

Synthesizing distributed implementations from global specifications is generally hard [12], [13], [14]. However for certain architectures, nonconservative distributed controller synthesis results exist both for reactive [15] and cooperative systems [16]. For motion planning of robotic teams, a centralized method for synthesizing controllers for each robot is proposed in [17]. Our distributed design procedure is based on decomposing the global specification into local ones in order to enable distributed synthesis and implementation of local control protocols. The feasibility of the proposed distributed design procedure depends on the choice of the decomposition structure of the underlying system, the strength of the coupling (through the exchange of physical resources and information) between them, and the expressiveness of the interface models. In section V, we present three compositional synthesis results, which, essentially, illustrate how refining the interface models, either by tightening the constraints on the inter-system flow of physical resources or by increasing the amount of information exchange, suppress the conservatism of distributed synthesis. In section VI, we demonstrate the utility of these results on a case study of distributed power allocation for VMS.

II. Prelimineries

A. Notation and Definitions

Definition 1: A system consists of a set V of variables. The *domain* of V, denoted by $dom(V)$, is the set of valuations of V. A *state* of the system is an element $v \in dom(V)$. Sets of states are denoted by calligraphic capital letters (i.e., $v \in \mathcal{V} \subseteq dom(V)$).

Definition 2: An *atomic proposition* is a statement on system variables v that has a unique truth value ($True$ or $False$) for a given value of v. Let $v \in dom(V)$ be a state of the system and p be an atomic proposition. We write $v \Vdash p$ if p is $True$ at the state v. Otherwise, we write $v \nVdash p$.

Definition 3: A *finite transition system* is a tuple $\mathbb{T} = (\mathcal{V}, \mathcal{V}_0, \mathcal{R})$ where $\mathcal{V}$ is a finite set of states, $\mathcal{V}_0 \subseteq \mathcal{V}$ is a set of initial states, and $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ is a transition relation.

An *execution* of a finite transition system is an infinite sequence of its states $\sigma = v_0 v_1 v_2 \dots$ where $v_0 \in \mathcal{V}_0$, for each $i > 0$, $v_i \in \mathcal{V}$ and $(v_{i-1}, v_i) \in \mathcal{R}$.

Given a state $v \in dom(V)$ and a subset X of variables, the *projection* of v on X, denoted by $v|_X \in dom(X)$, is the part of the state v that contains the valuations of variables only from X. For an execution σ, $\sigma|_X$ is defined similarly as $\sigma|_X \doteq v_0|_X v_1|_X v_2|_X \dots$.

LTL has two kinds of operators: logical connectives and temporal modal operators. The logic connectives are those used in propositional logic: *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$) and *material implication* ($\rightarrow$). The temporal modal operators include *next* ($\bigcirc$), *always* ($\Box$), *eventually* ($\Diamond$) and *until* ($\mathcal{U}$). An LTL formula is defined inductively as follows:

1) any atomic proposition p is an LTL formula; and
2) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas.

Other operators can be defined as follows: (i) $\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi)$, (ii) $\varphi \rightarrow \psi \triangleq \neg\varphi \vee \psi$, (iii) $\Diamond\varphi \triangleq True\, \mathcal{U}\, \varphi$, and (iv) $\Box\varphi \triangleq \neg\Diamond\neg\varphi$.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution $\sigma = v_0 v_1 v_2 \dots$ and an LTL formula φ, we say that φ *holds at position* $i \geq 0$ of σ, written $v_i \models \varphi$, if and only if (iff) φ holds for the remainder of the execution σ starting at position i. The semantics of LTL is defined inductively as follows: (i) For an atomic proposition p, $v_i \models p$ iff $v_i \Vdash p$; (ii) $v_i \models \neg\varphi$ iff $v_i \not\models \varphi$; (iii) $v_i \models \varphi \vee \psi$ iff $v_i \models \varphi$ or $v_i \models \psi$; (iv) $v_i \models \bigcirc\varphi$ iff $v_{i+1} \models \varphi$; and (v) $v_i \models \varphi \,\mathcal{U}\, \psi$ iff there exists $j \geq i$ such that $v_j \models \psi$ and $\forall k \in [i, j), v_k \models \varphi$.

Based on this definition, $\bigcirc\varphi$ holds at position i of σ iff φ holds at the next state v_{i+1}, $\Box\varphi$ holds at position i iff φ holds at every position in σ starting at position i, and $\Diamond\varphi$ holds at position i iff φ holds at some position $j \geq i$ in σ.

Definition 4: An execution of a system $\sigma = v_0 v_1 v_2 \dots$ *satisfies* φ, denoted by $\sigma \models \varphi$, if $v_0 \models \varphi$.

Definition 5: Let Σ be the set of all executions of a system. The system is said to be *correct* with respect to its specification φ, written $\Sigma \models \varphi$, if all its executions satisfy φ.

Definition 6: Given an execution $\sigma = v_0 v_1 \dots$ of a finite transition system, its set of *prefixes* is a set of finite sequence of states, denoted by $pref(\sigma) \doteq \{v_0 v_1 \dots v_n : \text{for some finite integer } n \geq -1\}$[1]. The prefixes of an LTL formula φ are given by the set of prefixes of all executions that satisfy φ, and denoted by $pref(\varphi) = \{\hat{\sigma} \in pref(\sigma) : \sigma \models \varphi\}$.

Definition 7: A finite sequence of states $\hat{\alpha} = v_0, \dots, v_n$ is a *bad prefix* for an LTL formula φ if and only if for all infinite state sequences $y = v_{n+1} v_{n+2} \dots$, the concatenation $\hat{\alpha} \cdot y \doteq v_0, v_1, \dots$ does not satisfy φ; that is $\hat{\alpha} \cdot y \not\models \varphi$.

Definition 8: An LTL formula φ is called a *safety* formula if and only if any execution σ that does not satisfy φ has a bad prefix.

B. Synthesis of Control Protocols: a Two-Player Game Approach

In many applications, systems need to interact with their environments and whether they satisfy the desired properties depends on the behavior of the environments. For example, in an aircraft vehicle management system, it is important to maintain safe operation in a wide range of environment and system health conditions. In this section, we briefly describe the work of Piterman, et al. and Bloem et al. [18], [19] on

[1] $n = -1$ corresponds to the empty sequence ϵ.

synthesis of control protocols in the presence of adversarial environment.

We refer the controllable part of the system as *plant*. When we say *system*, we refer the combined behavior of the environment and the controlled plant. From Definition 5, for a system to be correct, its specification must be satisfied by all of its executions regardless of the behavior of the environment in which it operates. Thus, the synthesis problem can be viewed as a two-player game between the plant and the environment: the environment attempts to falsify the specification while the plant attempts to satisfy it. Let E and P be the variables of the environment and the plant respectively. A state $s = (e, p)$ of the game is in $dom(E) \times dom(P)$. A transition of the game is a move of the environment $\mathcal{R}_e$ followed by a move of the plant $\mathcal{R}_p$. A *strategy* for the plant is a partial function $f : (s_0 s_1 \dots s_{t-1}, e_t) \mapsto p_t$ which chooses a move of the plant among its allowable moves based on the state sequence so far and the behavior of the environment. In this sense a *control protocol* is a winning strategy for the plant such that for all behaviors of the environment the specification is met. We say that φ is *realizable* if such a control protocol exists, that is the system can satisfy φ no matter what the environment does. It is important to note that for the class of games we consider, there is always a finite memory strategy, i.e., $f : (m_{t-1}, s_{t-1}, e_t) \mapsto (m_t, p_t)$ where m is a memory variable, internal to the control protocol, that takes values from a finite set [19].

We consider specifications of the form

$$\varphi \doteq (\varphi_e \rightarrow \varphi_s), \tag{1}$$

where for $\alpha \in \{e, s\}$ both φ_α have the following structure :

$$\varphi_\alpha \doteq \theta^\alpha_{init} \wedge \bigwedge_{i \in I_{t_\alpha}} \Box \psi^\alpha_i \wedge \bigwedge_{i \in I_{g_\alpha}} \Box \Diamond J^\alpha_i.$$

For specifications of this form, known as *Generalized Reactivity(1)* (GR(1)) formulas, Piterman, et al. show that checking its realizability and synthesizing the corresponding control protocol can be performed in polynomial time in the number of states of the system. Roughly speaking, in Eq. (1), φ_e characterizes the assumptions on the environment and φ_s describes the correct behavior of the system. In particular, θ^α_{init} is an atomic proposition characterizing the initial conditions; ψ^α_i are transition relations (i.e., expressions over a and $\bigcirc a$ for atomic proposition a) characterizing safe, allowable moves and atomic propositions characterizing invariants; and J^α_i are atomic propositions characterizing states that should be attained infinitely often. Many interesting temporal specifications can be transformed into this form. We refer the reader to [18] for precise definitions and extensions.

If the specification is realizable, the digital design synthesis tool implemented in JTLV [18] generates a finite state automaton that represents the control protocol. Assuming that the environment satisfies φ_e, then at any instance of time, there exists a node in the automaton that represents the current state of the system and the system can follow the transition from this node to the next based on the current knowledge about the environment. However, if φ_e is violated, the automaton is no longer valid, meaning that there may not exist a node in the automaton that represents the current state of the system, or even though such a node exists and the system follows the transitions in the automaton, the correct behavior φ_s is not guaranteed. Note that, in this case, φ still holds since Eq. (1) is satisfied whenever the assumptions on the environment φ_e are violated.

III. Problem Setup

We consider a vehicle management system that dynamically allocates electric power among flight controllers, active deicing units, and cabin pressure and temperature controls. In [6], we investigated a similar setting and demonstrated that dynamically allocating power is capable of reducing the peak power generation requirements (consequently leading to potential weight reduction) compared to maintaining power sources dedicated to each functionality at all times. In this paper, we focus on the structure shown in Fig. 1 where the primary power distribution controller (in switch 1) routes the power from generator 1 and the battery to the flight controllers and active deicing units. The secondary power distribution controller (in switch 2) routes the power from generator 2 to the cabin pressure and temperature control. Power and information exchanges between primary and secondary power distribution are limited to those between the switches 1 and 2.

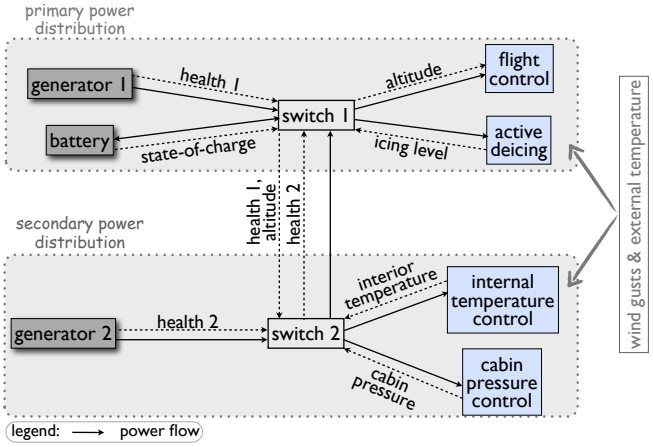

Fig. 1. Distributed power distribution model: Primary distribution routes power to the flight controllers and active deicing and the secondary distribution routes power to the controllers for internal temperature and cabin pressurization.

The control protocols proposed in [6] were centralized in the sense that the embedded controller has access to measurements of all the controlled and environment variables and determines the evolution of all the controlled variables to satisfy a (central) specification of the form (1). Here, we consider distributed synthesis of control protocols implemented in a distributed manner. For example, in the context of Fig. 1, this procedure constructs two control protocols by solving

the synthesis questions

$$(\phi_2' \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1) \quad (2)$$

$$(\phi_1' \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2) \quad (3)$$

for the primary and secondary power distribution, respectively. Here, φ_{e_1} and φ_{e_2} give a decomposition of φ_e complying with the splitting of the environment variables between the primary and secondary power distribution, and, similarly, φ_{s_1} and φ_{s_2} give a decomposition of φ_2. The formulas ϕ_i and ϕ_i', for $i = 1, 2$, model the restrictions on the power and information exchange between switches 1 and 2. The challenge in distributed synthesis is twofold: finding suitable decompositions of φ_e and φ_s and interface models that are rich enough to render (2) and (3) realizable and structured enough to avoid circular reasoning when protocols are simultaneously implemented. We discuss precise characterizations of these decompositions and conditions on the interface models in section V.

IV. Overview of modeling aspects

We now discuss the modeling aspects adopted from [6]. We use crude discretizations of the ranges in which the variables take values, relations between these variables, and finite state models that govern the time evolution of the variables hereafter.

In-flight icing affects the aircraft by changing aerodynamic properties in multiple ways including decreased lift, increased drag, decreased stall angle, and reduced controllability [20]. The amount of ice accumulation is primarily determined by the distance and time flown in icing clouds, the concentration of liquid water in the clouds, and a factor called the collection efficiency (the higher the collection efficiency the greater the rate of accumulation) [20]. The concentration of liquid water is a function of the temperature and altitude. In the range between $0°$C and $-40°$C, the concentration (i.e., the likelihood of icing) increases with decreasing temperature [21]. The accumulation of ice is faster in low-altitude cumulus-type clouds compared to higher-altitude stratiform clouds. The collection efficiency increases with increasing airspeed.

Based on the above discussion we use simple characterizations of power requests from flight controllers, deicing subsystem, and cabin pressure and temperature control as functions of the pressure altitude, level of icing, severity of wind gusts, and outside temperature:

- The power request from the flight controller increases with increasing levels of wind gusts, pressure altitude, and icing.
- The power required by the deicing subsystems increases with decreasing outside temperature and pressure altitude.
- The power required by the cabin pressure control subsystem increases with increasing pressure altitude and increasing outside temperature.
- The power required by the cabin temperature control subsystem increases with decreasing outside temperature.

We also consider an energy storage unit, whose state-of-charge is denoted by b and limited by B, in the primary power distribution (as shown in Fig. 1). Let the primary and secondary power generation are limited by $\bar{P}_1$ and $\bar{P}_2$.

At each time instant, the control protocol determines the pressure altitude h and assigns (allocates) power p_f, p_d, p_c and p_T to the four operations based on the availability of power and the prioritization determined by the flight-criticality of the operations to ensure system correctness. We assume that the flight actuators have priority over the remaining three control operations. Based on the description above, we define the independent and dependent variables needed to specify the problem.

Independent variables: Independent variables can be classified as environment or controlled variables. The environment variables are those related to factors over which the system does not have control such as the level of wind gusts and the outside temperature. At any given time, the control protocol determines the values of the controlled variables to ensure system correctness (with respect to its specification). The values of the environment variables may change arbitrarily over an execution, subject to their assumptions.

a) Environment variables: These are variables that the system does not have any control over such as the external temperature $T_{ext}(t) \in dom(T_{ext})$ and the severity of the wind gust $w(t) \in dom(w)$ and the health status $H_1(t)$ and $H_2(t)$ of generators. $H_1(t)$ and $H_2(t)$ are Boolean variables that have the value $True$ if the corresponding generator works in full capacity and $False$ otherwise.

b) Controlled variables: These are the variables that control protocol determines at each step in order to meet the specifications such as altitude h and the amount of power $p_f(t) \in dom(p_f)$, $p_d(t) \in dom(p_d)$, $p_T(t) \in dom(p_T)$ and $p_c(t) \in dom(p_c)$ supplied, respectively, to flight controllers, deicing, cabin temperature controls and cabin pressurization systems and the amount of power $p_x(t) \in dom(p_x)$ that is exchanged between power distribution systems.

Dependent variables: In addition to controlled and environment variables, there are dependent variables whose evolution is a function of environment and controlled variables. These are the amount of ice accumulation $a(t) \in dom(a)$, the internal temperature level $T_{int}(t) \in dom(T_{int})$, the cabin pressure level $c(t) \in dom(c)$, the power requirement $r_f(t) \in dom(r_f)$ of the flight control system, the battery charge level $b(t) \in dom(b)$ and the power outputs $\bar{P}_1(t) \in dom(\bar{P}_1)$ and $\bar{P}_2(t) \in dom(\bar{P}_2)$ of the generators. In particular, we have the following (at most) first order dependencies:

$$\begin{aligned} a(t+1) &= g_f(a(t), h(t), T_{ext}(t), p_d(t)) \\ T_{in}(t+1) &= g_T(T_{in}(t), T_{ext}(t), p_t(t)) \\ c(t+1) &= g_c(c(t), h(t), T_{ext}(t), p_c(t)) \\ b(t+1) &= g_b(b(t), p_f(t), p_d(t), p_x(t)) \\ r_f(t) &= g_r(a(t), w(t), h(t)) \\ \bar{P}_1(t) &= g_{\bar{P}_1}(H_1(t)) \\ \bar{P}_2(t) &= g_{\bar{P}_2}(H_2(t)). \end{aligned} \quad (4)$$

The set E of environment variables for the power distribution system is the environment variables and those dependent

variables that are functions of only the environment variables. That is, $E = \{T_{ext}, w, H_1, H_2, \bar{P}_1, \bar{P}_2\}$. The set P of plant variables are the controlled variables together with the dependent variables that are functions of controlled and environment variables. For the power distribution system, we have $P = \{h, p_f, p_d, p_T, p_c, p_x, r_f, a, T_{in}, c, b\}$.

Remark 1: Note that most of these variables correspond to continuous quantities. As a crude discretization, all continuous quantities are partitioned into bins (levels) to obtain the finite set of system variables for synthesis. One can as well use abstraction techniques for systematic discretization as mentioned in section I. For notational convenience, for each variable, we express its levels by a set of consecutive non-negative integers that start with zero and that are consistent with the ordering in the continuous domain. That is, for a variable x with N_x levels, $dom(x)$ is of the form $\{0, 1, \ldots, N_x - 1\}$. Hence, arithmetic operations on the variables as well as orders are defined in this integer domain by using the regular arithmetic operations and standard order for integers.

Specifications: System specifications include physical resource constraints and safety and performance requirements for the system as well as assumptions on the environment. The following list contains the specifications of interest in this paper:

1) Limit on total power imposes a constraint on the amount of power that can be allocated to each component (i.e., $p_d(t) + p_f(t) \leq \bar{P}_1(t) + b(t)$ and $p_c(t) + p_t(t) + p_x(t) \leq \bar{P}_2(t)$ for all t)
2) Safety requirements: (i) flight controller should always get the power it requests (i.e., $p_f(t) \geq r_f(t)$ for all t); (ii) the pressure altitude cannot change more than two levels between any consecutive time instances; (iii) ice accumulation cannot be severe (i.e., $a(t) < \mathrm{a}_u$ for all t); (iv) high amounts of ice accumulation limit the ability to change the pressure altitude (i.e., $|h(t+1) - h(t)| \leq \mathrm{a}_u - a(t)$).
3) Requirements on cabin temperature and pressure controls: (i) the cabin pressure should never exceed a certain upper limit c_u (i.e., $c(t) \leq \mathrm{c}_u$ for all t); (ii) the cabin temperature should always remain within certain comfort limits (i.e., $T_{int}(t) \in [\mathrm{T}_{int,l}, \mathrm{T}_{int,u}]$ for all t).
4) Performance requirements: (i) although variations from the desirable flight altitude range h_{nom} are allowed, this nominal value should be acquired infinitely often (i.e., $\square\diamond(h = \mathrm{h}_{nom})$); (ii) similarly, the nominal internal temperature level $\mathrm{T}_{int,nom}$ should be attained infinitely often (i.e., $\square\diamond(T_{int} = \mathrm{T}_{int,nom})$).
5) Environment assumptions: (i) wind gust w cannot be severe for more than N_w consecutive steps[2]; (ii) wind gust always eventually stops (i.e., $\square\diamond(w = 0)$); (iii) abrupt changes in external temperature are not admissible (i.e., $|T_{ext}(t-1) - T_{ext}(t)| \leq 1$ for all t); (iv) health of both generators cannot be degraded simultaneously.

The specifications listed above can be expressed in terms of LTL formulas; that is, one can construct φ_s and φ_e based on (4) and specifications 1-4 above for the system and specification 5 above for the environment, respectively. Moreover, these formulas lead to the global specification $\varphi_e \rightarrow \varphi_s$ which is a GR(1) specification.

V. Main Results

In this section, we discuss how the global specification can be decomposed into local ones so that it is possible to synthesize local control protocols for primary and secondary power distribution system separately. We present conditions under which these protocols can be implemented locally while guaranteeing that the global specification is met.

Let us denote the primary and secondary power distribution systems by G_1 and G_2, respectively, and assume that both G_1 and G_2 are equipped with local controllers on the corresponding switch in Fig. 1. We decouple these systems and define local specifications $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ for each. Initially, we assume that there is no power flow between G_1 and G_2. However, note that the internal temperature (T_{in}) and cabin pressure (c) variables that need to be regulated by the local controller in G_2 are affected by the altitude h which can be considered as an output of G_1. This leads to a serial interconnection between two systems for which we give the formal decomposition result in the following proposition.

Proposition 1: [Serial Interconnection] Given a system with the set of variables S, let φ_e, φ_{e_1}, φ_{e_2}, φ_s, φ_{s_1} and φ_{s_2} be LTL formulas that contain variables only from the respective sets of environment variables E, $E_1 \subseteq E$, $E_2 \subseteq (E \cup (S_1 \backslash E))$ and system variables S, $S_1 \subseteq S$, $S_2 \subseteq S$. Let P, P_1, P_2 be the sets of all controllable variables in S, S_1, S_2 that satisfy $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$. If

1.1 any execution σ such that $\sigma|_E \models \varphi_e$; also satisfies $\sigma|_E \models (\varphi_{e_1} \wedge \varphi_{e_2})$,
1.2 any execution σ such that $\sigma \models (\varphi_{s_1} \wedge \varphi_{s_2})$; also satisfies $\sigma \models \varphi_s$,
1.3 and, 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 implementing these two control protocols together leads to a system where the global specification $(\varphi_e \rightarrow \varphi_s)$ is met.

Proof: The conditions on P, P_1 and P_2 ensure that the synthesized local control protocols do not regulate the same controllable variables and can be implemented separately at the same time[3]. Let $\nu_1 \doteq ((\varphi_{e_1} \rightarrow \varphi_{s_1}) \wedge (\varphi_{e_2} \rightarrow \varphi_{s_2}))$ and $\nu_2 \doteq ((\varphi_{e_1} \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_1} \wedge \varphi_{s_2}))$. Note that any execution of the system that satisfies ν_1, also satisfies ν_2. That is, if there exist control protocols as in condition 1.3 of the proposition, the system meets the specification ν_2 when these control

[2]Note that by introducing a new environment variable $n_w \in \{0, 1, \ldots, N_w - 1\}$, this assumption can be expressed with first order relations between w and n_w.

[3]If the same controllable variable is included in two different local specifications, the corresponding local control protocols might assign different moves to this variable. Hence, these protocols can not be implemented simultaneously if these assignments conflict.

protocols are implemented simultaneously. Conditions 1.1 and 1.2 respectively mean that $\varphi_e \rightarrow (\varphi_{e_1} \wedge \varphi_{e_2})$ and $(\varphi_{s_1} \wedge \varphi_{s_2}) \rightarrow \varphi_s$ are tautologies (i.e., they evaluate to $True$ for any execution). Hence, it follows that for all executions that satisfy ν_2, $(\varphi_e \rightarrow \varphi_s)$ is satisfied. Therefore, for all executions that satisfy ν_1, specification φ is met. ■

In the power distribution system, $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ correspond to local specifications for G_1 and G_2, respectively. Now, assume $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ is realizable and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ is not realizable, hence condition 1.3 in Proposition 1 fails. Since G_1, by appropriately regulating h, can affect the environment behavior of G_2, we can use this interface between two systems to refine the local specifications so that the global specification would be met. Next proposition formalizes this idea.

Proposition 2: [Serial Interconnection Refinement] Let the formulas φ_e, φ_{e_1}, φ_{e_2}, φ_s, φ_{s_1} and φ_{s_2} and sets E, E_1, E_2, S, S_1, S_2, P, P_1, P_2 be defined as in Proposition 1. Also assume conditions 1.1 and 1.2 in Proposition 1 are satisfied. If

2.1 there exist two formulas ϕ_1 and ϕ_1', containing variables from $I_{1,2} \subseteq (S_1 \backslash E) \cap E_2$, such that $(\phi_1 \rightarrow \phi_1')$ is a tautology,

2.2 and, there exist two control protocols that make

$$\varphi_{e_1} \rightarrow (\varphi_{s_1} \wedge \phi_1), \tag{5}$$

$$(\phi_1' \wedge \varphi_{e_2}) \rightarrow \varphi_{s_2} \tag{6}$$

realizable. Then, implementing these two control protocols together leads to a system where the global specification $(\varphi_e \rightarrow \varphi_s)$ is met.

Proof: Assume for a given execution σ of the system, $\sigma|_E \models \varphi_e$. Then, by condition 1.1 in Proposition 1, $\sigma|_{E_1} \models \varphi_{e_1}$ and $\sigma|_{E_2} \models \varphi_{e_2}$. When a control protocol that meets the specification in (5) is implemented, $\sigma|_{S_1} \models \varphi_{s_1}$ and $\sigma|_{I_{1,2}} \models \phi_1$ whenever $\sigma|_{E_1} \models \varphi_{e_1}$. Since $(\phi_1 \rightarrow \phi_1')$ is a tautology, $\sigma|_{I_{1,2}} \models \phi_1'$. Hence, the left side of (6) is true for σ, which implies that the synthesized control protocol will guarantee $\sigma|_{S_2} \models \varphi_{s_2}$. Finally, from condition 1.2 in Proposition 1, we have $\sigma \models \varphi_s$. Consequently, the global specification is met. ■

The case where there is power flow between G_1 and G_2 as in Fig. 1 corresponds to a feedback interconnection where part of the output of each system acts as an environment variable for the other. In this case, the local variables are contained in the sets $E_1 \subseteq (E \cup (S_2 \backslash E))$, $E_2 \subseteq (E \cup (S_1 \backslash E))$ and $S_1 \subseteq S$, $S_2 \subseteq S$. Similar to Proposition 1, it is again required that P, P_1, P_2 satisfy $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$. Moreover, in order to ensure that the interconnection is well-posed and the interconnected system avoids deadlocks, additional conditions must be imposed. In particular, let for $i \in \{1,2\}$, the environment variables be partitioned into external and feedback parts; i.e., $E_i = E_{i,(l)} \cup E_{i,(f)}$ with $E_{i,(l)} \cap E_{i,(f)} = \emptyset$ and $E_{i,(f)} \subseteq (S_j \backslash E)$, $j \neq i$. Then, the formulas involving feedback variables should be such that, at least for one of the subsystems i, the value $e_t^{i,(f)} \in dom(E_{i,(f)})$ of the feedback variables at step t can be inferred from $s_{t-1}^{(i)} \in dom(S_i)$ and $e_t^{i,(l)} \in dom(E_{i,(l)})$. A refinement rule for feedback interconnections, inspired by the assume-guarantee reasoning in software verification [22], [23], [24], is given next.

Proposition 3: [Feedback Interconnection Refinement] For a feedback interconnection system, let the formulas φ_e, φ_{e_1}, φ_{e_2}, φ_s, φ_{s_1} and φ_{s_2} contain variables only from the respective sets of environment variables E, $E_1 \backslash S_2$, $E_2 \backslash S_1$ and system variables S, S_1, S_2. $P \doteq S \backslash E$, P_1 and P_2 satisfy $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$. Also assume (possibly unrealizable) local specifications $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ satisfy the conditions 1.1 and 1.2 in Proposition 1. If

3.1 there exist safety formulas ϕ_1, ϕ_1', ϕ_2 and ϕ_2', containing variables respectively from S_1, E_2, S_2 and E_1, such that for any execution σ with $\sigma|_E \models \varphi_e$, for all n and for $i \in \{1,2\}$, if $\hat{\alpha} = s_0 s_1 \ldots s_n$ is not a bad prefix for ϕ_i, $\hat{\alpha} \cdot s_{n+1}$ is not a bad prefix for ϕ_i'; and s_0 is not a bad prefix for ϕ_i',

3.2 and, there exist two control protocols that render the following local specifications true:

$$(\phi_2' \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1), \tag{7}$$

$$(\phi_1' \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2), \tag{8}$$

then implementing these two control protocols together leads to a system where the global specification $(\varphi_e \rightarrow \varphi_s)$ is met.

Proof: Assume for a given execution $\sigma = s_0 s_1 \ldots$ of the system, $\sigma|_E \models \varphi_e$. Then, by condition 1.1 in Proposition 1, $\sigma|_{E_1} \models \varphi_{e_1}$ and $\sigma|_{E_2} \models \varphi_{e_2}$. First, we show that $\sigma|_{S_j} \models \phi_i'$. Assume by contradiction that $\sigma|_{S_j} \not\models \phi_i'$. Since ϕ_i' is a safety formula, there exists a prefix of $\sigma|_{S_j}$ which is a bad prefix for ϕ_i'. Let $\hat{\sigma}|_{S_j}$, the shortest prefix of $\sigma|_{S_j}$ which is a bad prefix for ϕ_i', be of length k. According to condition 3.1, the initial state s_0 is not a bad prefix of ϕ_i' hence $k > 1$. Then, for a control protocol realizing (7)-(8), we have $f_j(\epsilon, e_0) = p_0^{(j)}$ which guarantees that $s_0 = (e_0, p_0^{(1)}, p_0^{(2)})$ is not a bad prefix of ϕ_j for $i \neq j$; $i, j \in \{1,2\}$. Note that otherwise for an execution that satisfies the assumptions in (7)-(8), the requirements would be violated for the protocol f_j. From condition 1 above, whenever s_0 is not a bad prefix for ϕ_j, $s_0 s_1$ is not a bad prefix for ϕ_j' for $j \in \{1,2\}$. By induction on time t, it can be shown that $\hat{\sigma} = s_0 \ldots s_k$ cannot be a bad prefix of ϕ_i' for any finite k which is a contradiction. Hence, when the environment satisfies its assumption, a control protocol that satisfies (7)-(8) renders the left side of the formulas $True$. Therefore, the right side has to evaluate to $True$, meaning $\sigma|_{S_1} \models \varphi_{s_1}$ and $\sigma|_{S_2} \models \varphi_{s_2}$. Then, it follows from Condition 1.2 in Proposition 1, that $\sigma \models \varphi_s$. ■

Remark 2: Using the safety closure arguments in [23], it is possible to strengthen the refinement rules given in (7)-(8) to accommodate one-sided liveness properties. That is, instead of (7)-(8), one could have $(\phi_2' \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1 \wedge \bigwedge_{i \in I_1} \square\lozenge J_i)$ and $((\bigwedge_{i \in I_2} \square\lozenge J_i) \wedge \phi_1' \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2)$ with $(\bigwedge_{i \in I_1} \square\lozenge J_i \rightarrow \bigwedge_{i \in I_2} \square\lozenge J_i)$ where J_i's are atomic propositions.

VI. EXAMPLES

Using the specifications described in section IV, and assigning initial values to the variables, a global specification of the form (1) can be derived. Our goal in this section is to synthesize control protocols that would meet this global specification. We investigate several interconnection structures, as discussed in section V, in terms of the total peak power requirement and robustness against system health degradation. We also demonstrate how reductions in peak power requirements and improvements in robustness can be achieved by refining the interface rules.

A crude discretization is employed to obtain the finite set of system variables. The variables and their discrete levels used in our examples are summarized in Table I. Table II lists the values of constants that appear in specifications.

Dedicated generation without dynamic allocation: The first thing to observe is that if the power distribution system was designed to meet the peak power requirements with dedicated generators and without taking into account the dynamics of the loads and allowable behavior of the environment, the generator capacities should be $\bar{P}_1 \geq \max(p_f + p_d) = 6$ and $\bar{P}_2 \geq \max(p_c + p_a) = 5$ units at all times. As we demonstrate next, by employing control protocols for dynamic power allocation, these number can be reduced significantly.

variable	domain	variable	domain
T_{ext}	$\{0,1,2,3\}$	w	$\{0,1,2\}$
H_1	$\{True, False\}$	H_2	$\{True, False\}$
h	$\{0,1,\ldots,4\}$	p_f	$\{0,1,2,3\}$
p_d	$\{0,1,2,3\}$	p_T	$\{0,1,2,3\}$
p_c	$\{0,1,2\}$	p_x	$\{0,1\}$
c	$\{0,1,\ldots,7\}$	a	$\{0,1,\ldots,5\}$
T_{in}	$\{0,1,\ldots,5\}$	r_f	$\{0,1,2,3\}$
b	$\{0,1,2,3\}$	$\bar{P}_1$, $\bar{P}_2$	design variables

TABLE I
VARIABLES AND CORRESPONDING DISCRETE DOMAINS FOR THE EXAMPLE.

$B = 3$	$N_w = 3$	$\mathrm{T}_{int,l} = 2$	$\mathrm{T}_{int,u} = 4$
$\mathrm{T}_{int,nom} = 4$	$\mathrm{h}_{nom} = 3$	$\mathrm{a}_u = 4$	$\mathrm{c}_u = 6$

TABLE II
CONSTANTS USED FOR SPECIFYING THE ASSUMPTIONS AND REQUIREMENTS.

Serial Interconnection Structure: For this example, we assume that there is no power flow between primary power distribution G_1 and secondary power distribution G_2 (i.e., p_x is set to a constant $p_x = 0$). The rest of the variables are decoupled as follows: $E_1 = \{T_{ext}, w, H_1, \bar{P}_1\}$, $P_1 = \{h, p_f, r_f, p_d, a, b\}$ for G_1 and $E_2 = \{T_{ext}, h, H_2, \bar{P}_2\}$, $P_2 = \{p_c, c, p_T, T_{in}\}$ for G_2. We form two local specifications $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ using the list of specification in section IV. These local specifications are consistent with the separation of variables and the conditions given in Proposition 1. For instance, among the specifications, safety requirements 2.(ii) and 2.(iv) and performance requirement 3.(i) regarding h are encoded in φ_{s_1}. Due to serial interconnection, specifications 2.(ii) and 3.(i) are environment behavior for G_2, so, included in φ_{e_2}. Since specification 2.(iv) depends on a, which is not available to G_2, control protocol in G_2 cannot rely on any assumptions involving a. Therefore, specification 2.(iv) is not included in φ_{e_2}. For this serial interconnection structure, we observe that local specifications are realizable if and only if $\bar{P}_1 \geq 4$ and $\bar{P}_2 \geq 4$ for all times.

Serial Interconnection Structure with Interface Refinement: Given the local specifications in the Serial Interconnection Structure, in this example we use interface refinement to reduce the peak power requirement. As noted above, $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ becomes infeasible if $\bar{P}_2 = 3$. By invoking Proposition 2, we add the interface refinement formula $\phi \doteq \phi' \doteq \Box\Diamond(h = 1)$ to the local synthesis problems. Note that ϕ is not required in the global specification but it does not contradict with it either since it renders the refined local specifications realizable. For this case, the peak power requirements reduce to $\bar{P}_1 \geq 4$ and $\bar{P}_2 \geq 3$.

Feedback Interconnection Structure with Interface Refinement: For this example, we allow power flow from G_2 to G_1. In particular, p_x is a variable in this example and G_2 can send its excess power p_x to the battery located on primary side. We also allow more information exchange, i.e., the systems know each others health status. The variables are separated as follows: $E_1 = \{T_{ext}, w, H_1, \bar{P}_1, H_2, p_x\}$, $P_1 = \{h, p_f, r_f, p_d, a, b\}$ for G_1 and $E_2 = \{T_{ext}, h, H_2, \bar{P}_2, H_1\}$, $P_2 = \{p_c, c, p_T, T_{in}, p_x\}$ for G_2. We define the following refinement formulas:

$$\phi_1 = \phi_1' = \Box\Diamond(h = 1),$$

$$\phi_2 = \phi_2' = \Box[(\neg H_1 \rightarrow (p_x = 1)) \wedge (H_1 \rightarrow (p_x = 0))].$$

Note that both φ_{e_1} and φ_{e_2} include $\Box(H_1 \vee H_2)$ from the environment assumption listed as specification 5.(iv). In this case, peak power requirements are $\bar{P}_1 \geq 4$ and $\bar{P}_2 \geq 3$; or $\bar{P}_1 \geq 3$ and $\bar{P}_2 \geq 4$. So, if both generators have capacity of 4 units and degraded power rating of 3 units, global specification can be met unless both generators degrade simultaneously (which is against the environment assumption on health status).

Centralized Controller: For comparison, we also consider a centralized architecture where a central controller collects all the data, determines the pressure altitude and allocates power to all loads while meeting the global specification. In this architecture, we allow two-way power flow between G_1 and G_2. Note that it is nontrivial to allow two-way power flow in the distributed setting in which case both controllers would need to simultaneously decide on the value of p_x. For the centralized control, the global specification is realizable if and only if $\bar{P}_1 + \bar{P}_2 \geq 6$ for all times. Although the power requirement slightly reduces in centralized setting; the number of states for the centralized system is significantly higher than the number of states of each system in distributed

setting, substantially increasing the computational complexity of synthesis.

All the examples presented in this section were implemented using the TuLiP toolbox [25], which provides an interface to JTLV [18].

VII. Conclusions and Future Work

In this paper, we considered the problem of synthesizing distributed control protocols, for vehicle management systems, that cooperatively allocate electric power while meeting certain higher level goals and requirements, and dynamically reacting to the changes in the internal system state and external environment. We used linear temporal logic as a specification language. We extended our earlier work on control protocol synthesis for vehicle management systems to a distributed setting. Starting with a global specification, we proposed methods to decompose it into local specifications. By construction, if there exist local control protocols that satisfy the local specifications, implementing these protocols simultaneously guarantees that the global specification is met. We further presented interface refinement rules in order to reduce the conservatism of distributed synthesis by imposing cooperation between subsystems. Several case studies demonstrated the effectiveness of distributed dynamic power allocation. Our examples also show that in the distributed setting, reductions in peak power requirements and improvements in robustness can be achieved by refining the interface rules between subsystems.

We focused on vehicle management systems in an avionics context. Similar issues arise in a number of application domains, including energy management in plug-in electric hybrid vehicles which dynamically allocate the power from multiple resources to multiple loads of different characteristics [26], [27] and vehicle management for spacecraft [2]. These different applications are subject to future research. Another direction for future research is to automate the refinement procedure, for instance using ideas from [28] where the aim is to find the least restrictive environment assumptions required for realizability.

References

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

[2] M. D. Watson and S. B. Johnson, "A theory of vehicle management systems," in *IEEE Aerospace Conference*, 2007.

[3] M. D. Natale and A. L. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.

[4] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to integrated modular avionics," in *Proceedings of the IEEE /AIAA Digital Avionics Systems Conference*, 2007.

[5] C. B. Watkins, "Integrated modular avionics: managing the allocation of shared intersystem resources," in *Proceedings of the IEEE/AIAA Digital Avionics Systems Conference*, 2006.

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

[7] N. Ozay, U. Topcu, T. Wongpiromsarn, and R. M. Murray, "Distributed synthesis of control protocols for smart camera networks," in *ACM/IEEE Second International Conference on Cyber-Physical Systems*, 2011.

[8] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, 2010, submitted.

[9] L. Benvenuti, A. Ferrari, E. Mazzi, and A. L. Vincentelli, "Contract-based design for computation and verification of a closed-loop hybrid system," in *Proceedings of the Hybrid Systems: Computation and Control*, 2008, pp. 58–71.

[10] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," in *Proceedings of the IEEE*, 2000, pp. 971–984.

[11] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[12] A. Pneuli and R. Rosner, "Distributed reactive systems are hard to synthesize," in *SFCS '90: Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1990, pp. 746–757 vol.2.

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

[14] E. Filiot, N. Jin, and J.-F. Raskin, "Compositional algorithms for ltl synthesis," in *Automated Technology for Verification and Analysis*, 2010, pp. 112–127.

[15] B. Finkbeiner and S. Schewe, "Uniform distributed synthesis," *Logic in Computer Science, Symposium on*, vol. 0, pp. 321–330, 2005.

[16] P. Madhusudan and P. Thiagarajan, "Distributed controller synthesis for local specifications," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, F. Orejas, P. Spirakis, and J. van Leeuwen, Eds. Springer Berlin, 2001, vol. 2076, pp. 396–407.

[17] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[18] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking and Abstract Interpretation*, ser. Lecture Notes in Computer Science, vol. 3855. Springer-Verlag, 2006, pp. 364 – 380, software available at http://jtlv.sourceforge.net/.

[19] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," 2010, preprint.

[20] P. J. Perkins and W. J. Rieke, "Tailplane icing and aircraft performance degradation," *Flight Safety Digest*, pp. 177–182, June-September 1997.

[21] N. Czernkovich, "Understanding in-flight icing," Tech. Rep., November 2004, transport Canada Aviation Safety Seminar. [Online]. Available: aerosafety.ca/sources/aicraft_icing_paper.pdf

[22] J. Misra and K. Chandy, "Proofs of networks of processes," *Software Engineering, IEEE Transactions on*, vol. SE-7, no. 4, pp. 417 – 426, 1981.

[23] M. Abadi and L. Lamport, "Conjoining specifications," *ACM Trans. Program. Lang. Syst.*, vol. 17, no. 3, pp. 507–534, 1995.

[24] B. Jonsson and Y.-K. Tsay, "Assumption/guarantee specifications in linear-time temporal logic," *Theor. Comput. Sci.*, vol. 167, no. 1&2, pp. 47–72, 1996.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *HSCC*, 2011, software available at http://tulip-control.sf.net/.

[26] C. L. Jun-Mo, J. Kang, J. W. Grizzle, and H. Peng, "Energy management strategy for a parallel hybrid electric truck," in *Proceedings of the 2001 American Control Conference*, 2001, pp. 2878–2883.

[27] V. H. Johnson, K. B. Wipke, and D. J. Rausen, "Hev control strategy for real-time optimization of fuel economy and emissions," 2000.

[28] K. Chatterjee, T. A. Henzinger, and B. Jobstmann, "Environment assumptions for synthesis," in *CONCUR*, 2008, pp. 147–161.