Formal Synthesis of Embedded Control Software: Application to Vehicle Management Systems

T. Wongpiromsarn, U. Topcu, and R. M. Murray

Control and Dynamical Systems, California Institute of Technology, CA, 91125, USA.

Motivated by the transition from federated to integrated architectures in aerial vehicles, we propose an automated methodology for the synthesis of correct-by-construction control protocols for vehicle management systems. We use linear temporal logic as the specification language for precisely describing correct behaviors of the system as well as the admissible dynamic behavior of the environment due to, for example, wind gusts and changes in the flight conditions. We apply the method in the context of dynamic power allocation among a number of subsystems of varying flight-criticality. The resulting power management protocol is guaranteed to be correct, with respect to the overall system specification, for all admissible environment profiles. This approach also enables reasoning about design tradeoffs such as between efficiency (imposed through formal specifications) and system weight (characterized by the amount of required power generation and energy storage). We present our preliminary results in a simple setting and discuss extensions of the methodology to capture more realistic system and environment models and specifications.

I. Introduction

Vehicle management systems (VMS) control and coordinate a number of subsystems of aerial vehicles including the flight controllers, electrical systems, various power systems, fuel management, environmental control systems, deicing units, and landing gear.[1,2] They also interface with additional aircraft subsystems such as sensor pointing, data acquisition, and pilot and ground interfaces. See Fig. 1 for a schematic view of a vehicle management system. Traditional VMS are typically based on federated architectures in which integrated hardware and software components realize independent or loosely interconnected functions.[3] These components are self-contained units (e.g., line-replacable units) and are connected with point-to-point wired interfaces. The VMS regulate the basic functions of subsystems, either automatically or on requests from the pilot, monitor, display, and log system status, and perform fault detection and recovery.

Next generation VMS are expected to become significantly more sophisticated than currently deployed systems, with distributed computation, integration of more advanced networking and computing architectures, and increased levels of automatic operations and electric power requirements. Additionally, the move to autonomous flight will require the VMS to be interactive in dynamically changing environments and reconfigurable. In order to deal with the resulting system complexity, integrated modular avionics (IMA) architectures provide an alternative to federated architectures. The IMA architectures utilize high-integrity, partitioned platforms that host multiple avionics functionalities of different criticalities. Unlike the federated architectures, where there are dedicated computation and communication resources and power is allocated for each functionality, the IMA architecture is based on highly-integrated resource management among the functionalities that share the existing resources.[4,5] The transition to IMA architectures leads to two competing trends: possibilities for system-level optimization by dynamically allocating spare resources and reduction in the weight and power consumption come at the expense of extra layers of integration complexities.

Due to the increasing complexity of VMS functionality, certification of safety and performance properties will require the use of formal specifications and systematic methods for verifying those specifications, combined with additional validation experiments and tests. Next generation VMS must also be at least partly designed for verification, since it will not be possible to analyze systems of this complexity without structuring the design to allow verification tools to be applied. In this paper, building on our recent work,[6] we take an initial step toward formal, automated synthesis of control protocols that enable dynamic configuration

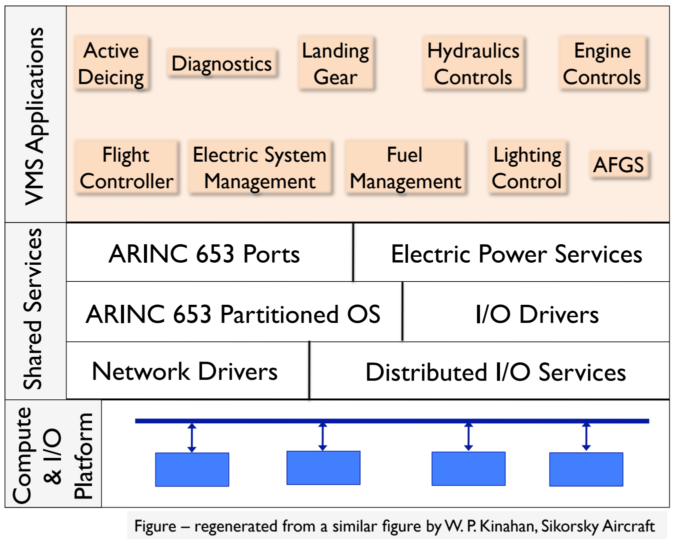

Figure 1. A schematic view of a vehicle management system including subsystem functionalities, communication interfaces, and distributed, networked computing platform.

for integrated power management in VMS. In this methodology, the specifications are expressed in the so-called linear temporal logic (LTL)[7,8] and a combination of tools from controls and computer science formal methods domains (discussed in Section III in more detail) are utilized for the automatic synthesis of control protocols. The use of formal analysis and synthesis methods here follows their successful integration in the verification of hardware and software systems in computer science and engineering[9–14] and robotics.[15–17]

The rest of the paper is organized as follows: We discuss the motivation of the current work in the following section followed by the introduction to formal specification and synthesis in Section III. Section IV is dedicated to the problem formulation and an overview of modeling aspects included in the study. The application of the synthesis procedure to vehicle management systems in Section V is followed by a discussion of the results of the current paper and possible directions for extending the current work.

II. Motivation

The transition to "more-electric" technologies—such as electric main engine start, electric flight control actuators, and active deicing—on a number of commercial and military aerial vehicles offers increases the efficiency in power use. For example, a Boeing 787 extracts as much as 35 percent less power from the engines than traditional pneumatic systems on previous generation airplanes.[a] This transition leads to increases in electric power demands as well as new challenges for example dynamic reconfiguration and scheduling of power allocation among different types of electric loads of different levels of flight-criticality.

Consider, for instance, power management among a collection of the subsystems, namely flight controllers, landing gear, deicing units, and environmental control. The main design considerations include:

- Real-time reconfiguration in a dynamic environment: The subsystems interact with their environment (both the external factors, e.g., due to outside temperature variations and changes in flight conditions, and the rest of the VMS and other systems of the airplane); hence, they need to react to the changes in their environment in real time.

- 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.

- Resource constraints: With the increase in the electric loads and introduction of integrated architectures, the subsystems share the limited electric power resources. A further important issue is improving

[a]See http://www.boeing.com/commercial/787family/programfacts.html. Retrieved September 27, 2010.

the efficiency of the vehicle-level energy use to reduce the volume and weight of the hardware for electric generation, distribution, and storage.

- 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, control protocols for power management need 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.

Furthermore, analysis and design of power management protocols are made challenging due to the interleaving between the high-level constraints and requirements and low-level dynamics. The low-level dynamics capture, for example, the dynamics of each of the subsystems. This complexity arising from the heterogeneity of subsystems and requirements requires a formal specification languages that are capable of unambiguously and concisely stating system requirements. Such requirements include the constraints (e.g., resource constraints) on the system behavior, safety and performance requirements, and the assumptions on the behavior of the environment. Additionally, there is a need for methods and tools to systematically reason about the formal specifications and automate the design of control protocols that ensure that the system satisfies its specifications.

In the rest of the paper, we utilize linear temporal logic (LTL) as the formal specification language and expand our previous work on the synthesis of control protocols for embedded control systems[6] to the design of protocols for dynamic configuration of integrated power management. The output of the synthesis procedure is a hierarchical control protocol: (i) a discrete planner, represented as a finite state automaton whose states are pairs of discretized values of the system and environment states, creates a high-level plan so that the system satisfies the specification; (ii) a continuous control implements the discrete plan at the lower-level. The behavior of the system under the resulting control structure can be considered as real-time allocation of the power resources to loads in a dynamically changing environment. The notions of "system" and "environment" and the distinction between them will be made explicit in the following sections.

We here investigate design of control protocols for vehicle management systems in an avionics context. Similar issues arise in a number of application domains. Examples include energy management systems in plug-in electric hybrid vehicles which dynamically allocate the power from multiple resources to multiple loads of different criticality[18, 19] and vehicle management for spacecraft.[2] Similarly, in the envisioned smart grid applications, energy distribution management systems are supposed to reconfigure the allocation of power resources reacting to the changes in supply (due to the integration of intermittent renewable energy resources), demand and system health in real time.[20]

III. Preliminaries on formal specification and synthesis

Formal methods are mathematically-based techniques for ensuring system correctness. These approaches rely on constructing a mathematical representation of a system and its specification (i.e., desired properties). Examples of such mathematical objects typically used in modeling systems include finite state machines, differential equations, timed automata and hybrid automata.[21, 22] ω-regular languages and temporal logics are widely used to precisely describe system specifications.[10] With their expressive power, a wide class of properties including deadlocks, livelocks, correctness of system invariants, safety, stability and non-progress execution cycles can be specified.

In this section, we first describe linear temporal logic, which is used throughout the paper as a specification language. Then, we provide a brief summary of automatic synthesis of digital designs that satisfy a large class of properties expressed in linear temporal logic even in the presence of an adversary (typically arising from changes in the environments).[11] Finally, we describe our recent work, which integrates digital design synthesis and hybrid system theory to allow automatic synthesis of provably correct embedded control software for continuous systems.

A. Linear temporal logic

Temporal logic is a branch of logic that incorporates temporal aspects and can be used to reason about a time line.[7, 8, 10, 23] Its use as a specification language was introduced by Pnueli.[24] Since then, temporal logic has been demonstrated to be an appropriate specification formalism for reasoning about various kinds of systems, especially those of concurrent programs. It has been utilized to formally specify and verify behavioral properties in various applications.[12–14, 25, 26]

In this paper, we consider a version of temporal logic, namely linear temporal logic (LTL), which is particularly suitable for describing properties of software systems. Before describing LTL, we need to define an atomic proposition, which is LTL's main building block. An atomic proposition can be defined based on a variable structure of the system as follows.

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 $v \in dom(V)$.*

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 $v \nVdash p$.*

In this language, an *execution* of a system is described by an infinite sequence of its states. Specifically, for a discrete-time system whose state is only evaluated at time $t \in \{0, 1, \ldots\}$, its execution σ can be written as $\sigma = v_0v_1v_2\ldots$ where for each $t \geq 0$, $v_t \in dom(V)$ is the state of the system at time t.

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* ($\Longrightarrow$). The temporal modal operators include *next* ($\bigcirc$), *always* ($\square$), *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: (a) $\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi)$, (b) $\varphi \Longrightarrow \psi \triangleq \neg\varphi \vee \psi$, (c) $\diamond\varphi \triangleq \mathit{True} \, \mathcal{U} \, \varphi$, and (d) $\square\varphi \triangleq \neg\diamond\neg\varphi$.

A *propositional formula* is one that does not include temporal operators. Given a set of LTL formulas $\varphi_1, \ldots, \varphi_n$, their *Boolean combination* is an LTL formula formed by joining $\varphi_1, \ldots, \varphi_n$ with logical connectives.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution $\sigma = v_0v_1v_2\ldots$ 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:

1. For an atomic proposition p, $v_i \models p$ iff $v_i \Vdash p$;
2. $v_i \models \neg\varphi$ iff $v_i \not\models \varphi$;
3. $v_i \models \varphi \vee \psi$ iff $v_i \models \varphi$ or $v_i \models \psi$;
4. $v_i \models \bigcirc\varphi$ iff $v_{i+1} \models \varphi$; and
5. $v_i \models \varphi \, \mathcal{U} \, \psi$ 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}, $\square\varphi$ 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 3 *An execution $\sigma = v_0v_1v_2\ldots$* satisfies *φ, denoted by $\sigma \models \varphi$, if $v_0 \models \varphi$.*

Definition 4 *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, that is, $(\Sigma \models \varphi)$ iff $\big(\forall\sigma, (\sigma \in \Sigma) \implies (\sigma \models \varphi)\big)$.*

Examples of LTL formulas: Given propositional formulas p and q, important and widely used properties can be defined in terms of their corresponding LTL formulas as follows.

1. **Safety** (invariance): A safety formula is of the form $\square p$, which asserts that the property p remains invariantly true throughout an execution. Typically, a safety property ensures that nothing bad happens and that the system maintains safe operating conditions.
2. **Guarantee** (reachability): A guarantee formula is of the form $\diamond p$, which guarantees that the property p becomes true at least once in an execution. Reaching a goal state is an example of a guarantee property.
3. **Progress** (recurrence): A progress formula is of the form $\square\diamond p$, which essentially states that the property p holds infinitely often in an execution. As the name suggests, a progress property typically ensures that the system makes progress throughout an execution.
4. **Response**: A response formula is of the form $\square(p \implies \diamond q)$, which states that following any point in an execution where the property p is true, there exists a point where the property q is true. A response property can be used, for example, to describe how the system should react to changes in the operating conditions.
5. **Stability** (persistence): A stability formula is of the form $\diamond\square p$, which asserts that there is a point in an execution where the property p becomes invariantly true for the remainder of the execution. This definition corresponds to the definition of stability in the controls domain since it ensures that eventually, the system converges to a desired operating point and remains there for the remainder of the execution.

Remark 1 *Properties typically studied in the control and hybrid systems domains are safety (usually in the form of constraints on the system state) and stability (i.e., convergence to an equilibrium or a desired state). LTL thus offers extensions to properties that can be expressed. Not only can it express a more general class of properties, but it also allows more general safety and stability properties than constraints on the system state or convergence to an equilibrium since p in $\square p$ and $\diamond\square p$ can be any propositional formula.*

B. Synthesis of a digital design: 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, whether an aerial vehicle exhibits the correct behavior may depend on the weather condition, the behaviors of other vehicles in its vicinity, software and hardware faults and failures, etc. In this section, we informally describe the work of Piterman, et al.[11] on automatic synthesis of a finite state automaton from its specification. We refer the reader to[11] and references therein for the detailed discussion.

From Definition 4, for a system to be correct, its specification φ must be satisfied in all of its executions regardless of the behavior of the environment in which it operates. Thus, the environment can be treated as an adversary and the synthesis problem can be viewed as a two-player game between the system and the environment: the environment attempts to falsify φ while the system attempts to satisfy φ. We say that φ is *realizable* if the system can satisfy φ no matter what the environment does.

For a specification of the form

$$(\bigwedge_{i\in I} \square\diamond\varphi_i) \implies (\bigwedge_{j\in J} \square\diamond\psi_j),$$

known as *Generalized Reactivity(1)*, Piterman, et al. shows that checking its realizability and synthesizing the corresponding automaton can be performed in polynomial time. In particular, we are interested in a specification of the form

$$\varphi = (\varphi_e \implies \varphi_s)$$

where roughly speaking, φ_e characterizes the assumptions on the environment and φ_s describes the correct behavior of the system, including the valid transitions the system can make. We refer the reader to[11] for precise definitions of φ_e and φ_s. Note that since $\varphi_e \implies \varphi_s$ is satisfied whenever φ_e is *False*, if the assumptions on the environment φ_e are violated, then the correct behavior φ_s of the system is not ensured, even though the specification φ is satisfied.

If the specification is realizable, the digital design synthesis tool such as JTLV[11] generates a finite state automaton that represents a set of transitions the system should follow in order to satisfy φ. 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.

If the specification is not realizable, the synthesis tool provides an initial state of the system starting from which there exists a set of moves of the environment such that the system cannot satisfy φ. The knowledge of the nonrealizability of the specification is useful since it provides information about the conditions under which the system will fail to satisfy its desired properties.

The main limitation of the synthesis of finite state automata is the state explosion problem. In the worst case, the resulting automaton may contain all the possible states of the system. For example, if the system has N variables, each can take any value in $\{1, \ldots, M\}$, then there may be as many as M^N nodes in the automaton.

C. Synthesis of embedded control software

In our recent work,[6,27,28] a correct-by-construction approach has been applied to systems that comprise the physical component, which we refer to as the plant, and the (potentially dynamic and not a priori known) environment in which the plant operates.

Consider a system model $\mathbb{S}$ with a set $V = S \cup E$ of variables where S and E are disjoint sets that represent, respectively, the set of plant variables that are regulated by the control protocol and the set of environment variables whose values may change arbitrarily throughout an execution. The domain of V is given by $dom(V) = dom(S) \times dom(E)$ and a state of the system can be written as $v = (s, e)$ where $s \in dom(S) \subseteq \mathbb{R}^n$ and $e \in dom(E)$. In this paper, we call s the *controlled* state and e the *environment* state.

Assume that the controlled state evolves according to the following discrete-time linear time-invariant state space model: for $t \in \{0, 1, 2, \ldots\}$,

$$\begin{array}{rcl} s[t+1] & = & As[t] + Bu[t] + Ed[t], \\ u[t] & \in & U, \\ d[t] & \in & D, \\ s[0] & \in & dom(S), \end{array} \tag{1}$$

where $U \subseteq \mathbb{R}^m$ is the set of admissible control inputs, $D \subseteq \mathbb{R}^p$ is the set of exogenous disturbances and $s[t]$, $u[t]$ and $d[t]$ are the controlled state, the control signal and the exogenous disturbance, respectively, at time t.

Given a model $\mathbb{S}$ of a physical system and its specification φ expressed in linear temporal logic, we proposed a methodology for automatic synthesis of embedded control software that provides a formal guarantee of system correctness with respect to φ. Our approach, as illustrated in Fig. 2, relies on constructing a finite transition system $\mathbb{D}$ that serves as an abstract model of $\mathbb{S}$ (which typically has infinitely many states). A digital design synthesis tool such as JTLV can then be used to synthesize a strategy, represented by a finite state automaton, satisfying the specification φ based on the abstract model $\mathbb{D}$. This leads to a hierarchical, two-layer design (see Fig. 3) with a discrete planner/scheduler computing a discrete plan based on the abstract model $\mathbb{D}$ and a continuous controller computing a sequence of control inputs based on the physical model $\mathbb{S}$ to continuously implement the discrete plan. Simulations/bisimulations provide the proof that the continuous execution preserves the desired properties.[29] The correctness of the system is guaranteed even in the presence of an adversary (typically arising from changes in the environments), disturbances and modeling errors.

For systems with a certain structure, the computational complexity of the planner synthesis can be alleviated by solving the planning problems in a receding horizon fashion, i.e., compute the plan or strategy over a "shorter" horizon, starting from the current state, implement the initial portion of the plan, move the horizon one step ahead, and recompute.[6,30] This approach essentially reduces the planner synthesis problem into a set of smaller problems while preserving the desired system-level temporal properties. We illustrated the application of this receding horizon temporal logic planning approach on the autonomous driving examples.

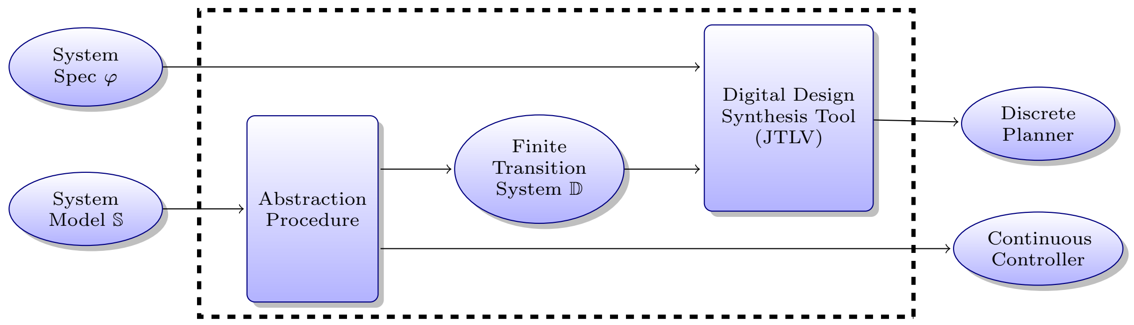
System Spec φ
System Model $\mathbb{S}$
Abstraction Procedure
Finite Transition System $\mathbb{D}$
Digital Design Synthesis Tool (JTLV)
Discrete Planner
Continuous Controller

Figure 2. Synthesis of embedded control software.

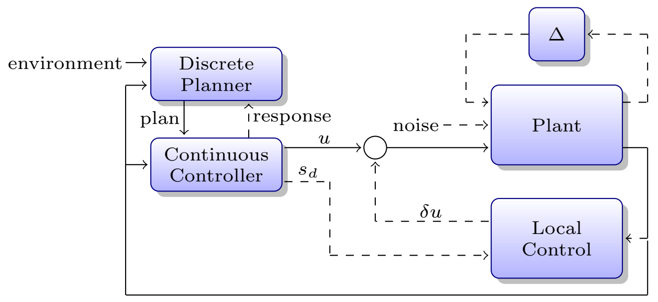

Figure 3. The planner-controller subsystem. In addition to the components discussed in this section, Δ, which captures uncertainties in the plant model, may be added to make the model more realistic.

IV. Problem formulation

We consider a vehicle management system that involves flight controllers, environmental control, and active deicing. The aim is to automatically synthesize a scheduling protocol that manages the dynamic power allocation among these subcomponents taking the underlying dynamics and certain high-level goals and requirements into account, and reacting to the changes in the environment. To this end, 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. As a prelude to the problem formulation, we summarize the factors considered in modeling and specifying the constraints and desired requirements.

A. Overview of modeling aspects

In-flight icing is a complex phenomenon that affects the aircraft by changing aerodynamic properties in multiple ways including decreased lift, increased drag, decreased stall angle, and reduced controllability.[31] Table 1 shows the effects of different levels of icing on the airspeed, required power increase to regain airspeed, and reductions in climb rate and control authority.[32] 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).[31] 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.[33] Fig. 4 shows the empirical relation between the concentration of freezing nuclei and the temperature.[34] The accumulation of ice is faster in low-altitude cumulus-type clouds compared to higher-altitude stratiform clouds. The collection efficiency is a function of the airspeed, size of water droplets, and size and shape of the moving surface: it is highest for high airspeeds, large droplets, and small objects.

In the following, we use simple characterizations of power requests from flight controllers, deicing subsystem, and environmental control as functions of the pressure altitude, level of icing, severity of wind gusts, and outside temperature. In general, the environmental control unit has multiple functions, including humidity control, ram air cooling, bleed flow and temperature control.[1] In this paper, we only consider cabin

level	airspeed reduction	power increase to regain airspeed	climb-rate reduction	reduction in control authority
trace	< 10 knots	$< 10\%$	$< 10\%$	no effect
light	$10 - 19$ knots	$10 - 19\%$	$10 - 19\%$	no effect
moderate	$20 - 39$ knots	$20 - 39\%$	$\geq 20\%$	slow or overly sensitive response
severe	≥ 40 knots	unable	unable	limited or no response

Table 1. Effects of icing on airspeed, power increase to regain airspeed, climb-rate reduction, and control authority.

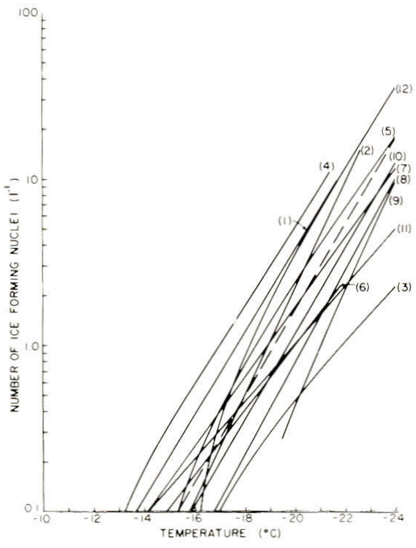

Figure 4. Concentration of freezing nuclei versus temperature.[33,34] Different curves represent results from various researchers.

pressurization. Based on the above discussion, we model the power requests from these three subsystems to capture the following trends.

- The power request from the flight controller increases with increasing levels of wind gusts, pressure altitude, and icing.
- The power request from the deicing subsystems increases with decreasing outside temperature and pressure altitude.
- The power request from the environmental control subsystem (for the regulation of cabin pressurization) increases with increasing pressure altitute and decreasing outside temperature.

B. Problem setup

Let H denote the set of admissible pressure altitudes of the aircraft and let P_f, P_d, and P_e denote sets of admissible amount of power supplied to the flight actuators, deicing and environmental control operations, respectively. We also consider an energy storage unit (a battery) on board with capacity B. Let $0 \leq b \leq B$ be the amount of energy stored in the battery. Consider that the power generation is limited by $\bar{P}$.

At each time instant, the control protocol determines the pressure altitude $h \in H$ and assigns (allocates) power $p_f \in P_f$, $p_d \in P_d$ and $p_e \in P_e$ to the three 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 deicing and environmental control operations. That is, the flight actuators always get the power they request for. The deicing and environmental control operations share the leftover amount of power. The amount of power that is not supplied to these three operations will be stored in the battery (subject to its constraints).

Let $r_f \in P_f$, $r_d \in P_d$ and $r_e \in P_e$ be the amount of power requested by the flight actuators, deicing and environmental control operations, respectively. As discussed in the previous section, we assume that r_f is a function of the amount a of ice accumulation, the severity w of the wind gusts and the pressure altitude h whereas r_d and r_e only depend on the outside temperature T and the pressure altitude h. Note the difference between the sets of variables p_f, p_d ,p_e and r_f, r_d, r_e: variables p_f, p_d, p_e are controlled, i.e., determined by the control protocol while variables r_f, r_d, r_e are dependent on the states of the system and the environment. For example, if the aircraft is subject to severe wind gusts, then r_f will be high so that the flight actuators act to the effects of the wind gusts.

Based on the description above, we define the independent and dependent variables needed to specify the problem. As an initial step, we use a crude discretization for the values that these variables can take. These discretized values essentially model different levels (i.e., low vs high) of these variables. The computational procedure introduced in Section III.C is applicable (through a more sophisticated discretization process) for the case where these variables evolve in continuous spaces.

1. Independent variables

Independent variables can be classified as environment[b] 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) based on their previous values and the current values of the environment variables. The values of the environment variables may change arbitrarily over an execution, subject to the assumptions they satisfy (discussed later).

Environment variables The environment variables in the above description are the outside temperature T and the severity w of the wind gust (leading to deviations from the planned flight path). We use a four-level quantization $T \in \{\text{low, medium-low, medium-high, high}\}$ as a crude discretization of the temperature range between $-22°$F and $32°$F (similar to the discrete values used in reference [35] and Fig. 4). For wind gust, we use three crude levels $w \in \{\text{low, moderate, severe}\}$.

Controlled variables The variables whose values are directly determined by the control protocol are the pressure altitude $h \in H$ and the amount $p_f \in P_f$, $p_d \in P_d$ and $p_e \in P_e$ of power supplied to the flight controllers, deicing and environmental control operations, respectively. Based on the charts in reference [35] , we use a five-level quantization $H = \{\text{low, medium-low, medium, medium-high, high}\}$ as a crude discretization of the typical pressure altitude range of 10,000–30,000 ft. For the admissible amount of power to the three operations, we use the following crude discretization: $P_f = P_d = \{\text{low, medium-low, medium-high, high}\}$ and $P_e = \{\text{low, medium, high}\}$.

2. Dependent variables and finite state model

In addition to the controlled and environmental variables, there are variables that are dependent on the controlled and environmental variables. For example, as discussed in the previous section, the power request r_f of the flight actuators is a function of the amount a of ice accumulation (because icing degrades control authority[36]), the severity w of wind gusts and the pressure altitude h. Table 2 provides the values of r_f for different levels of a, w and h. The values of r_d and r_e for different levels of T and h are given in Table 3 and Table 4, respectively.

Our problem setup also includes variables whose evolution is dependent on the controlled and environmental variables. At time $t + 1$, the amount $a[t + 1]$ of ice accumulation is determined by its value $a[t]$ at the previous time t and the amount $r_d[t]$ and $p_d[t]$ of power requested and supplied to the deicing unit. Since r_d depends on the outside temperature $T[t]$, which is a controlled state, and the pressure altitude $h[t]$,

[b]Throughout the text, the word "environment" refers to two different concepts. The use in "environmental control" refers to the regulation of the cabin conditions (e.g., air supply, thermal conditions, etc.) of an airplane. The use here refers to the factors over which the system has no control. The distinction is to be understood from the context.

Table 2. The value of r_f as a function of a, h and w. Here, L denotes low, ML denotes medium-low, MH denotes medium-high, and H denotes high. Each entry contains three values of r_f. The first value is for $w =$ low, the second for $w =$ moderate, and the last for $w =$ severe.

r_f		h				
		low	medium-low	medium	medium-high	high
a	none	L, L, ML	L, ML, ML	L, ML, MH	ML, ML, MH	ML, MH, MH
	trace	L, ML, ML	L, ML, MH	ML, ML, MH	ML, MH, MH	ML, MH, H
	light	L, ML, MH	ML, ML, MH	ML, MH, MH	ML, MH, H	MH, MH, H
	moderate	ML, ML, MH	ML, MH, MH	ML, MH, H	MH, MH, H	MH, H, H
	severe	ML, MH, MH	ML, MH, H	MH, MH, H	MH, H, H	MH, H, H

Table 3. The value of r_d as a function of T and h.

r_d		h				
		low	medium-low	medium	medium-high	high
T	low	high	high	high	medium-high	medium-high
	medium-low	high	high	medium-high	medium-high	medium-low
	medium-high	medium-high	medium-high	medium-low	medium-low	low
	high	medium-high	medium-low	medium-low	low	low

which is a controlled state, the evolution of the ice accumulation depends on both the controlled and the environment states. Similarly, the cabin pressure $c[t+1]$ is determined by its value $c[t]$ at the previous time t and the amount $r_e[t]$ and $p_e[t]$ of power requested and supplied to the environmental control unit. The environmental control subsystem regulates the cabin pressure to be below pressure level at 8000 ft.[1] Hence, we discretize the values of the cabin pressure c into 8 discrete states: $C_0, \dots, C_7$. For $i \in \{0, \dots, 6\}$, the state C_i represents the cabin pressure range between $8000i/7$ and $8000(i+1)/7$. C_7 is the state in which the cabin pressure is more than 8000 ft. Fig. 5 provides the finite transition systems that model the evolution of a and c. Finally, the amount $b[t+1]$ of energy stored in the onboard battery is determined by its value $b[t]$ at the previous time t, the amount $p_f[t]$, $p_d[t]$ and $p_e[t]$ of power supplied to the flight actuators, deicing and environmental control operations and the amount $\bar{P}$ of power generated by the power generator.[c] For convenience and the ease of presentation, we use the numbers, 0, 1, 2 and 3 to represent the levels, low, medium-low, medium-high and high, respectively, in the crude discretization of p_f and p_d. Similarly, for p_d, we use the numbers, 0, 1 and 2 to represent its three levels, low, medium and high. The total power generated by the power generator and the energy stored in the battery can be discretized and represented by a finite set of discrete numbers in a similar manner. Then, we assume that the evolution of the energy storage is governed by the difference equation

$$b[t+1] = \min(B, b[t] + \bar{P} - p_f[t] - p_d[t] - p_e[t]). \tag{2}$$

Note that the use of numbers to represent these different levels in the crude discretization is only for convenience and visualization purpose. More sophisticated model for the evolution of b can also be specified using a finite state system model as in Figure 5 for a and c.

3. System Specifications

System specifications include physical resource constraints and safety and performance requirements for the system as discussed earlier. The following (non-exhaustive) list contains a sample of specifications of interest expressed in LTL.

[c]With abuse of notation, $\bar{P}$, $p_f[t]$, $p_d[t]$, and $p_e[t]$ denote the total energy generated and energy supplied to the flight controllers, deicing units, and environmental control, respectively, over the time period $[t, t+1]$.

Table 4. The value of r_e as a function of T and h.

r_e		h				
		low	medium-low	medium	medium-high	high
T	low	medium	medium	high	high	high
	medium-low	low	medium	high	high	high
	medium-high	low	medium	medium	high	high
	high	low	low	medium	medium	high

Resource Constraints Limit on the total power imposes the constraint on the amount of power that can be allocated to each component. Using a set of discrete numbers to represent the levels in the crude discretization of p_f, p_d, p_e, b and $\bar{P}$ as previously done in modeling the evolution of b, the resource constraint on the amount of power can be expressed in LTL as $\Box(p_f + p_d + p_e \leq \bar{P} + b)$, i.e, the sum of power supplied to each subsystem is always less than or equal to the total available power. More sophisticated constraints can also be expressed. For example, one may enumerate all the admissible combinations of the amount of power supplied to each unit, e.g., if p_f is high and p_d is high, then p_e has to be low: $\Box(p_f = \text{high} \wedge p_d = \text{high} \implies p_e = \text{low})$, and so on.

Safety Requirements The safety requirements capture the conditions that must be maintained in order for the system to operate safely. Examples of such safety requirements include prioritization of loads and requirements on the level of icing.

- Prioritization: Flight controller has the highest priority (i.e., it always gets the power it requests): $\Box(p_f \geq r_f)$, where r_f is considered to be a function of the level of ice accumulation, wind gust and pressure altitude as discussed in Section IV.B.
- Requirements on the altitude change: the pressure altitude cannot change more than 2 levels between any two consecutive time instances. For example, if $h[t]$ is low, then $h[t+1]$ cannot be medium-high or high: $\Box\big(h = \text{low} \implies (\bigcirc h \neq \text{medium-high} \wedge \bigcirc h \neq \text{high})\big)$. In addition, if the level of ice accumulation is moderate, then the pressure altitude cannot change more than 1 level between any two consecutive time instances. If the level of ice accumulation is severe, the pressure altitude cannot change at all: $\Box(a = \text{severe} \implies \bigcirc h = h)$.
- The amount of ice accumulation cannot be severe: $\Box(a \neq \text{severe})$.

Performance Requirements The performance requirements specify the desired operating conditions of the system. For example, the environmental control unit needs to ensure that the cabin is pressurized so that a cabin altitude of about 8,000 ft is never exceeded.[1] In addition, it is more desirable to fly at a higher altitude. Such requirements can be specified as follows.

- The cabin pressure altitude does not exceed 8000 ft: $\Box(c \neq C_7)$.
- Requirements on the altitude: Throughout the flight, altitude variations from the desirable flying altitude range $h = \text{high}$ are allowed. It is required that altitude $h = \text{high}$ is acquired infinitely often, stated as $\Box\Diamond(h = \text{high})$.

Assumptions Assumptions on the behavior of the environment variables are included to restrict the environment behavior into its admissible range as well as to make sure that the desired properties are achievable. These assumptions must be explicitly stated as part of the system specification. For example, if the flight actuators always request power and the generation level $\bar{P}$ is not high enough to supply power to all subsystems at all times then the requirement on the amount of ice accumulation and the cabin pressure cannot be realized. Additionally, flight conditions in which the flight actuators always requests high levels of power may not be realistic and a design that accounts for the behavior of the environment variables that lead to such steady request may be overly conservative. Therefore, such behavior of the environment should not be accounted for in the synthesis of control protocols. The following assumption imposes restrictions on the environment variable w on which r_f depends and the changes in the outside temperature.

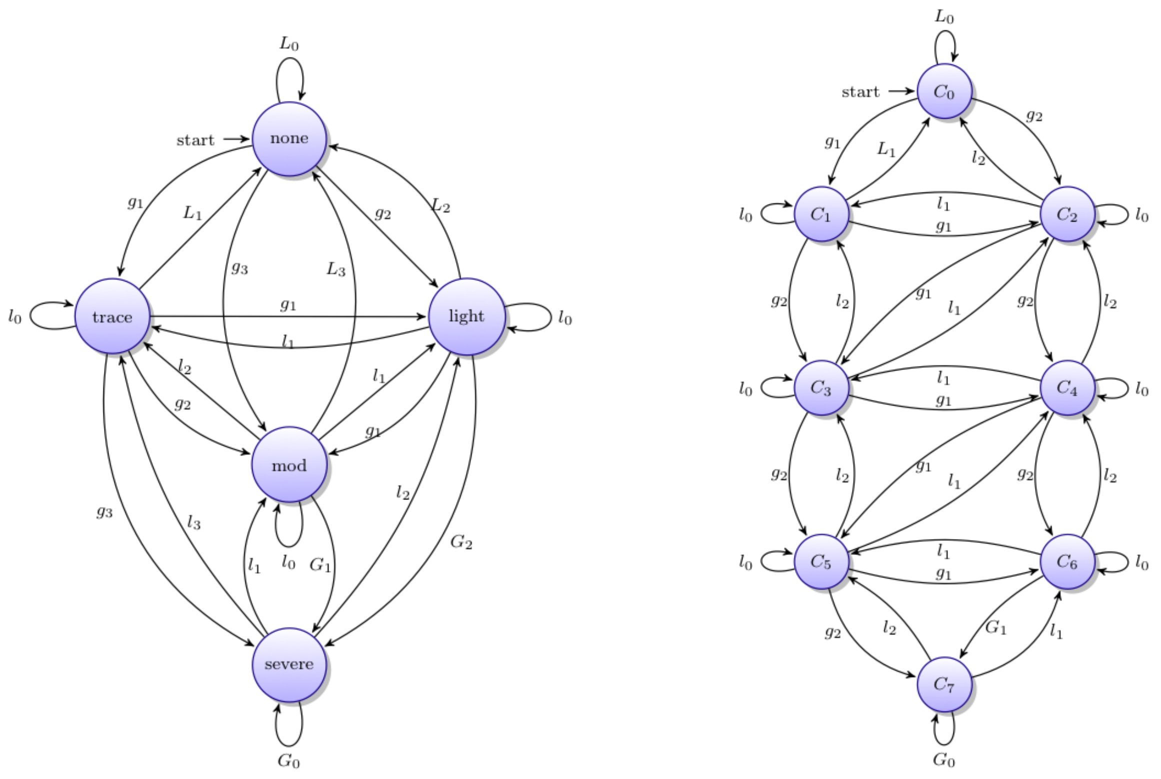

Figure 5. The finite state automaton on the left represents the evolution of the amount of ice accumulation as a function of p_d and r_d (which is dependent on T and h). The one on the right represents the evolution of the cabin pressure as a function of p_e and r_e (which is also dependent on T and h). For each $i \in \{0, \ldots, 3\}$, label l_i (and g_i, respectively) in the left figure represents the condition that r_d is i levels smaller (greater) than p_d. For example, label l_1 indicates that if p_d is high, then r_d is medium-high. Label L_i (and G_i, respectively) represents the condition that r_d is i or more levels smaller (greater) than p_d. For example, label L_1 indicates that if p_d is high, then r_d is either medium-high, medium-low or low. In the right figure, the interpretation of the transition labels l_i, L_i, g_i and G_i where $i \in \{0, 1, 2\}$ for the pairs of r_e and p_e is similar to their interpretation for the pairs of r_d and p_d in the left figure.

- The level w wind gust cannot be severe for more than N_w consecutive time steps. Let n_w be the number of consecutive time steps for which the wind gust is severe. Then, this assumption can be written as $\square(n_w \geq N_w \implies \bigcirc(w \neq \text{severe}))$.
- No abrupt change in temperature, i.e., the temperature can only change one level between any two consecutive time instances. For example, if the current temperature is medium-low, then in the next time instance, the temperature cannot be high: $\square(T = \text{medium-low} \implies \bigcirc T \neq \text{high})$.

More sophisticated assumptions and requirements, such as conditions on the speed that imply certain timing constraints, can be imposed using LTL. These extensions along with an investigation of the suitability of other formal specification languages for the analysis and design of control protocols for VMS are subject to future work.

V. Synthesis of Correct-by-Construction Vehicle Management Systems

A. Problem statement

Given the assumptions on the environment variables and the system, we are interested in specifications of the form

$$\varphi_e \implies \varphi_s, \tag{3}$$

where φ_e is the conjunction of all the assumptions and φ_s is the conjunction of all the resource constraints, safety requirements and performance requirements listed in the previous section. The specification in (3) essentially requires that whenever the environment variables satisfy their assumptions, then the system meets its requirements.[d] Then, the control (scheduling) protocol synthesis problem is formally stated as

Synthesis Problem: Synthesize a scheduling protocol such that $\varphi_e \implies \varphi_s$ holds.

The system requirements are functions of several design parameters that appear in the safety and performance requirements, including $\bar{P}$, B and N_w. Therefore, an interesting design consideration is solving the above synthesis question while identifying the "optimal" (or desirable) values of these parameters, for example, the optimal values of the generation capacity and storage capacity so that the analysis and synthesis questions are solvable. The optimality of these design variables can for example be interpreted as the values that minimize the weight of the aircraft.

B. Approach

We apply the approach presented in Section III.C. As discussed in Section IV.B, we start with a simple coarse discretization of the state space and construct the finite transition systems in Fig. 5. The finite state model $\mathbb{D}$ can be considered as the composition of these finite transition systems. Properly constructing $\mathbb{D}$ from $\mathbb{S}$ based on simulation relations as discussed in Section III.C is possible, provided that the difference equations in the form of (1) can be obtained.

The specification φ is as stated in (3). The output of the synthesis procedure includes a (high-level) discrete planner which is represented as a finite state automaton whose states are pairs of the system states (e.g., h, p_f, p_d, p_e, b, a, and c) and the environment states (e.g., T and w). As long as the system states follow the transitions in this automaton, the system satisfies its requirements under every allowable actions of the environment (i.e., specified in the environment assumptions).

C. Preliminary results

We now present some preliminary results in a simple case with the coarse discretizations of the variables as discussed in the previous section. The results of simulation runs where $N_w = 2$ and the levels[e] of $\bar{P}$ and B are 5 and 3, respectively, for different wind gust and temperature profiles are shown in Figures 6 and 7. The synthesis was performed on a MacBook with a 2 GHz Intel Core 2 Duo processor. The computation time was approximately 5 minutes. The resulting automaton contains 1896 states. Observe that since $r_f, r_d \in \{0,1,2,3\}$ and $r_e \in \{0,1,2\}$, in general, $\bar{P}$ should not be below 8 to ensure that there is enough power to supply to the three subsystems.

By using the formal synthesis procedure discussed in Section III.C, we show that due to the complicated relationship between different variables and the specification of the system, only $\bar{P} = 5$ is sufficient to ensure system correctness for all the behaviors of the environment (subject to the assumptions specified in the system specification). Note that due to the infinite number of admissible environment profiles, it is not obvious in this case how one could manually design a control protocol in order to verify such a property. In our case, since the automaton obtained from the synthesis procedure is guaranteed to be correct by construction, verification is not necessary. Of course, one should validate that the simple system model used in the synthesis is a useful approximation of the actual system. As discussed in Section III, the synthesis procedure utilized in Section V is capable of integrating more sophisticated models and specifications. The simple setup of sections IV and V is chosen to provide a preliminary demonstration and an initial step toward automated correct-by-construction design of control protocols for vehicle management systems. Validation of the models and verification and synthesis with models of multiple scales and fidelity levels are important topics subject to current research yet beyond the scope of the current paper.

[d]Note that the specification in (3) is satisfied when the assumption on the environment is violated. Hence, in that case, the requirement is not necessarily imposed.

[e]See the discussion about the levels of power in Section IV.B.

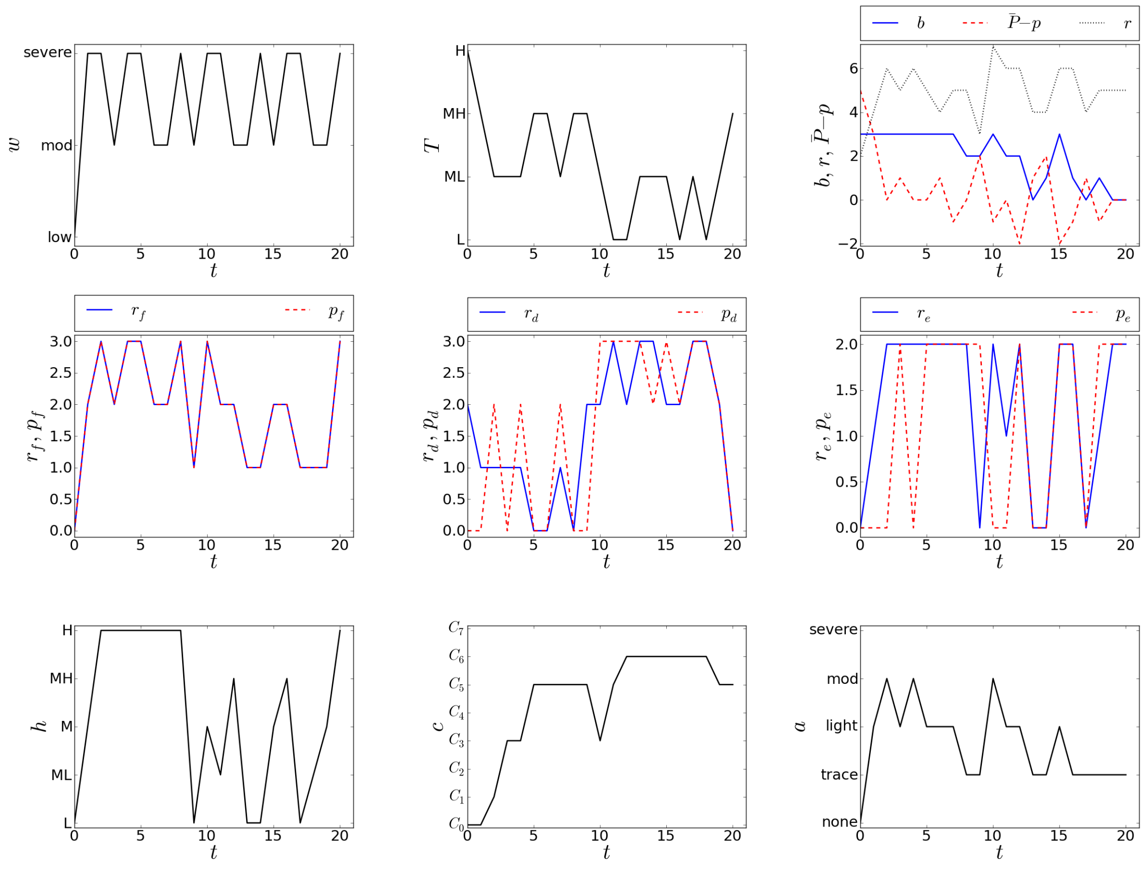

Figure 6. Simulation results (1). The horizontal axis is the time. The two leftmost columns on the top row show the profiles of the environment variables: the wind gust profile (left) and the temperature profile (middle). We use L, ML, MH and H in the temperature profile to denote the temperature levels low, medium-low, medium-high and high, respectively. The right column shows the battery storage profile, the total request for power and the difference between generation and the total request for power. We denote $r_f + r_d + r_e$ by r and $p_f + p_d + p_e$ by p. Middle row shows the the power requests from and power supplied to the flight control (left), deicing (middle), and environmental control (right) subsystems. Bottom row, left column shows the altitude profile. L, ML, M, MH and H denote the altitude levels low, medium-low, medium, medium-high and high, respectively. The remaining two figures show that the requirements on a and c are satisfied.

VI. Conclusions and future work

We proposed an automated synthesis procedure for the correct-by-construction design of control protocols for vehicle management systems. In this framework, correct behavior of the system and admissible dynamic behavior of its environment are specified in linear temporal logic. We applied the method in the context of dynamic power allocation among a number of subsystems of varying flight-criticality. The resulting power management protocol is guaranteed to be correct, with respect to the overall system specification, for all the admissible environment profiles. This formal approach enables systematic reasoning about design tradeoffs for example between the efficiency of the power system and its weight. We presented our preliminary results in a simple setting.

The work reported in this paper is an initial step toward correct-by-construction synthesis of control protocols for vehicle management systems. There are a number of potential and promising directions for both practical and theoretical future research. We conclude the paper with a non-exhaustive list.

- The synthesis procedure outputs a hierarchical control structure with a low-level continuous controller and higher-level discrete planner (scheduler). In this paper, we restricted the study to the design of a discrete planner by using models based on finite state automaton. More realistic system models with variables on continuous spaces governed by differential equations can readily be handled by the

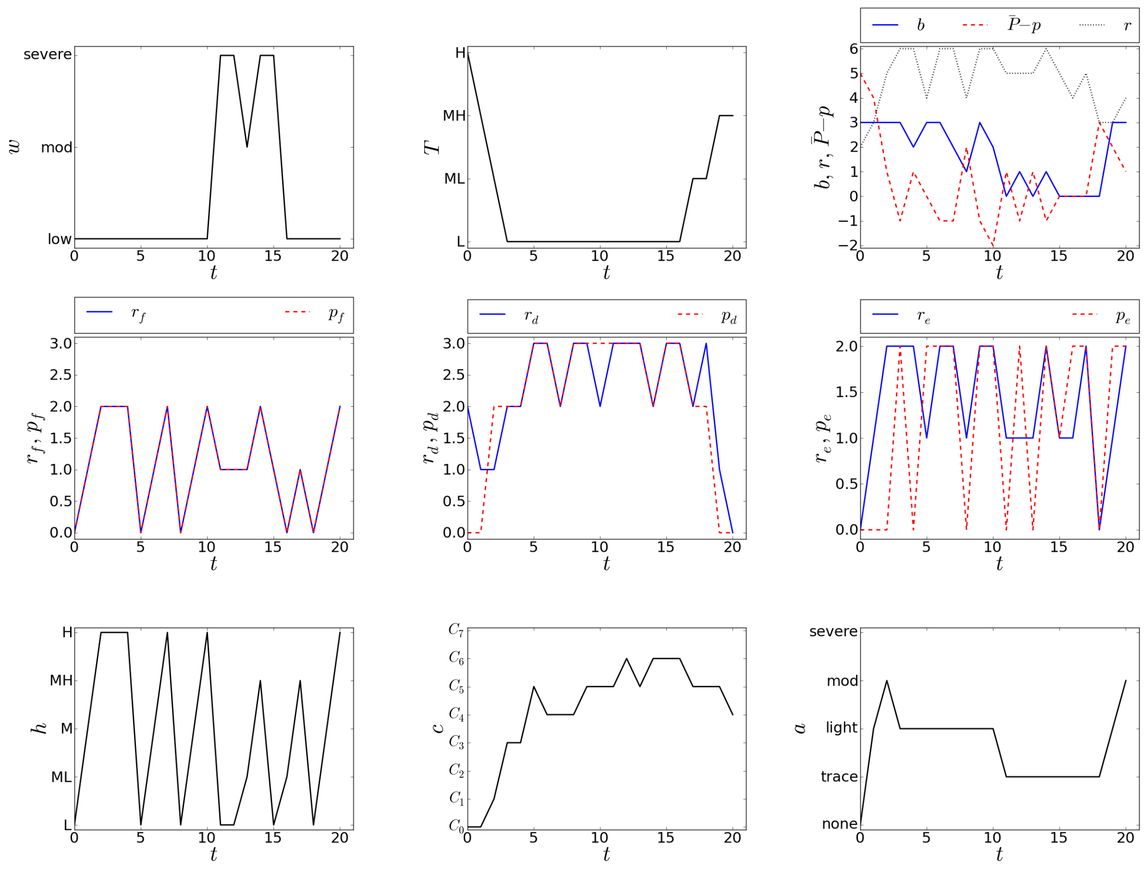

Figure 7. Simulation results (2). See the caption of Figure 6 for an explanation of the plots.

synthesis procedure.

- The hierarchical architecture can be extended to incorporate models of multi-fidelity and requirements and objectives of multiple scales. For example, low fidelity models may be used at the vehicle level with increasing fidelity for subsystems. Vehicle-level energy optimization and subsystem level requirements can be coordinated in this hierarchical architecture. See[30] for the development of such a hierarchical architecture in the context of autonomous driving. Distribution of the computation in the synthesis procedure and implementation of the synthesized controllers will be a key enabler for the scaling of hierarchical control architectures.

- We restricted the attention to power management and three subsystems, flight controller, active de-icing, and cabin pressurization. The procedure can be extended to include other functionalities of vehicle management systems, e.g., landing gear, functionality of the environmental control system (in addition to cabin pressurization), fuel management. Moreover, the central constrained resource in the current study is electric power. The procedure can be extended to capture other constrained resources such computation or communication whose allocation becomes complicated with the introduction of integrated modular avionics architectures as discussed in Section I.

- Certain failures and faults can be specified in the LTL formalism (potentially by introducing extra variables). For example, failures in generation units (i.e., drops in the generation capacity $\bar{P}$) can be captured by treating $\bar{P}$ as an environment variables and restricting its behavior by appropriate assumptions as in Section IV.B.3. Moreover, faults, for example those in the sensors, can be modeled as boolean-valued environment variables.

- The receding horizon temporal logic planning framework discussed in Section III.C may be utilized in order to handle larger problems (e.g. incorporating path planning, fuel management, etc into the power allocation problem). The applicability of this approach, however, is restricted to systems with a certain partial order structure.[6,30] Verifying whether VMS satisfy this partial order condition is subject future work.

VII. Acknowledgments

The authors gratefully acknowledge Michel Ingham, Scott Livingston, Necmiye Ozay and Mumu Xu and for the helpful discussions and comments on the paper. This work is partially supported by AFOSR and the Boeing Corporation.

References

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

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

[3]Natale, M. D. and Sangiovanni-Vincentelli, A. L., "Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools," *Proceedings of the IEEE*, Vol. 98, No. 4, 2010, pp. 603–620.

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

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

[6]Wongpiromsarn, T., Topcu, U., and Murray, R. M., "Receding horizon control for temporal logic specifications," *HSCC*, edited by K. H. Johansson and W. Yi, ACM ACM, 2010, pp. 101–110.

[7]Manna, Z. and Pnueli, A., *The temporal logic of reactive and concurrent systems*, Springer-Verlag, 1992.

[8]Emerson, E. A., "Temporal and modal logic," *Handbook of theoretical computer science (vol. B): formal models and semantics*, MIT Press, Cambridge, MA, USA, 1990, pp. 995–1072.

[9]Clarke, E. M., Grumberg, O., and Peled, D. A., *Model Checking*, MIT Press, 1999.

[10]Baier, C. and Katoen, J.-P., *Principles of Model Checking*, MIT Press, 2008.

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

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

[13]Galton, A., editor, *Temporal Logics and Their Applications*, Academic Press Professional, Inc., San Diego, CA, 1987.

[14]Holzmann, G., "The Theory and Practice of A Formal Method: NewCoRe," *Proc. of the IFIP World Computer Congress*, Vol. 1, North-Holland Publ., 1994, pp. 35–44.

[15]Kloetzer, M. and Belta, C., "A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications," *IEEE Transactions on Automatic Control*, Vol. 53, No. 1, 2008, pp. 287–297.

[16]Kress-Gazit, H., Fainekos, G., and Pappas, G., "Where's Waldo? Sensor-Based Temporal Logic Motion Planning," *Proc. of IEEE International Conference on Robotics and Automation*, April 2007, pp. 3116–3121.

[17]Girard, A. and Pappas, G. J., "Hierarchical control system design using approximate simulation," *Automatica*, Vol. 45, No. 2, 2009, pp. 566–571.

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

[19]Johnson, V. H., Wipke, K. B., and Rausen, D. J., "HEV Control Strategy for Real-Time Optimization of Fuel Economy and Emissions," 2000.

[20]"The smart grid: an introduction," Tech. rep., 2008, U.S. Department of Energy, Office of Electricity Delivery and Energy Reliability.

[21]Alur, R. and Dill, D. L., "A Theory of Timed Automata," *Theoretical Computer Science*, Vol. 126, 1994, pp. 183–235.

[22]Henzinger, T. A., "The theory of hybrid automata," *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 278–.

[23]Huth, M. and Ryan, M., *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press, 2nd ed., 2004.

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

[25]Gabbay, D. M., Hogger, C. J., and Robinson, J. A., *Handbook of Logic in Artificial Intelligence and Logic Programming (Vol. 4): Epistemic and Temporal Reasoning*, Oxford University Press, Oxford, UK, 1995.

[26]Cerrito, S. and Mayer, M. C., "Using Linear Temporal Logic to Model and Solve Planning Problems," *AIMSA*, 1998, pp. 141–152.

[27]Wongpiromsarn, T., Topcu, U., and Murray, R. M., "Receding Horizon Temporal Logic Planning for Dynamical Systems," *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2009.

[28]Wongpiromsarn, T., Topcu, U., and Murray, R. M., "Automatic Synthesis of Robust Embedded Control Software," *AAAI Spring Symposium on Embedded Reasoning: Intelligence in Embedded Systems*, 2010, pp. 104–111.

[29]Alur, R., Henzinger, T. A., Lafferriere, G., and Pappas, G. J., "Discrete Abstractions of Hybrid Systems," *Proceedings of the IEEE*, 2000, pp. 971–984.

[30]Wongpiromsarn, T., Topcu, U., and Murray, R. M., "Receding Horizon Temporal Logic Planning," *IEEE Transactions on Automatic Control*, submitted.

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

[32]J. P. Dow, S., "Understanding the stall-recovery procedure for turboprop airplanes in icing conditions," *Flight Safety Digest*, April 2005, pp. 1–17.

[33]Czernkovich, N., "Understanding in-flight icing," Tech. rep., November 2004, Transport Canada Aviation Safety Seminar.

[34]Pruppacher, H. R. and Klett, J. D., *Microphysics of Clouds and Precipication*, D. Riedel Publishing Company, 1978.

[35]"Code of Federal Regulations. Part 25. Airworthiness standards: transport category airplanes. Appendix C." Tech. rep., Federal Aviation Agency.

[36]Ranaudo, R. J., Mikkelsen, K. L., Mcknight, R. C., Ide, R. F., Reehorst, A. L., Jordan, J. L., Schinstock, W. C., and Platz, S. J., "The measurement of aircraft performance and stability and control after flight through natural icing conditions," Tech. Rep. NASA-TM-87265, 1986.