Keywords
safety-critical systems, Hamilton-Jacobi reachability, computational challenges, unmanned airspace, system decomposition

Abstract
Autonomous systems are becoming pervasive in everyday life, and many of these systems are complex and safety-critical. Formal verification is important for providing performance and safety guarantees for these systems. In particular, Hamilton-Jacobi (HJ) reachability is a formal verification tool for nonlinear and hybrid systems. However, it is computationally intractable for analyzing complex systems. Computational burden is in general a difficult challenge in formal verification.

In this review, we begin by briefly presenting background on reachability analysis with an emphasis on the HJ formulation. We then present recent work showing how high-dimensional reachability verification can be made more tractable by focusing on two areas of development: system decomposition for general nonlinear systems, and traffic protocols for unmanned airspace management. By tackling the curse of dimensionality, tractable verification of practical systems is becoming a reality, paving the way towards more pervasive and safer automation.
1. Introduction

Autonomous systems have become increasingly pervasive in everyday life. These systems include unmanned aerial systems, self-driving cars, and many other types of robots. By now, it goes without saying that these systems have many potential applications, limited only by our imagination. In recent years, a tremendous amount of progress has been made in autonomous systems research, for example in sub-areas such as modeling, planning, sensing and perception. In addition, the availability of computing power and hardware platforms today have also helped bridge the gap between theory and practical implementation.

Despite the recent success in automation, our use of robots and interactions with robots are still quite limited. For example, one of the current uses of unmanned aerial vehicles (UAVs) is surveying areas with very few people and no other air traffic. In general, robotic operations are restricted to controlled environments, and involve a single robot or a few robots. These robots also have limited interactions with other robotic agents as well as humans. There are likely many reasons for this, and one reason is simple: If we put many robots close to each other and to humans, we would not know for sure whether they would harm each other or harm humans.

The perspective of safety is crucial for enabling more effective use of autonomous systems, many of which are safety-critical systems. Safety-critical systems are systems in which failure is extremely costly, or even fatal. Formal safety analysis will allow autonomous systems to become provably robust to changes in the environment and to other agents, as well as operate in much denser configurations. This would mean, for example, that thousands of UAVs could fly in an urban area. Safety analysis is also essential for allowing autonomous systems to interact closely and physically with humans.

1.1. Safety-Critical Autonomous Systems

On an intuitive level, maintaining safety could be simply avoiding an obstacle, such as a tree. Sometimes the obstacle may be an agent that can also control the way it moves, like an aircraft. On a broader level, maintaining safety means keeping within a set of safe operating conditions. Staying away from obstacles is one specific example, but this concept
Reachability analysis quantifies when the bike needs to steer away to avoid the tree represented by the interior of the red circular region (left), or in general when the system needs to take action to avoid dangerous configurations represented by the outside of the red rectangular region (right).

is quite general. For example, safe operating conditions can be defined in terms of not only position, but also any other variables of interest such as velocity, angle, or even voltage, concentration of chemicals, human comfort, and degree of trust in automation.

Verification of systems is challenging for many reasons. First, all possible system behaviors need to be accounted for. This makes most simulation-based approaches insufficient, and this is where formal verification methods are needed. Many practical systems operate in complex environments. These systems are affected by disturbances such as weather conditions. The environments can be unpredictable, and may even contain adversarial agents. In addition, the systems evolve in continuous time with complex, nonlinear dynamics.

Perhaps the most difficult challenge of all is that these systems often have high-dimensional configuration spaces. High-dimensionality means that many variables are needed to describe the state of a system. This could occur if the system of interest is very complex, or if there are many agents in the system, or both.

1.2. Hamilton-Jacobi Reachability as a Safety Analysis Tool

The focus of this review is Hamilton-Jacobi reachability analysis, one of the most powerful formal verification tools for guaranteeing performance and safety properties of systems. The idea is quite simple: Imagine riding a bike, and suppose that there is a tree in front. Obviously, we do not want to run into the tree. Figure 1 (left) shows a simplified diagram in which we have the bike, and a circular area that represents the tree. The way we can avoid hitting the tree is to make sure to change our direction of travel early enough, while taking into account variables such as momentum and steering capabilities of the bike, and any disturbances like rough terrain that might affect steering.

Reachability analysis quantifies exactly what it means to steer away early enough. This is done by computing the backward reachable tube or in some cases a backward reachable set, a region that we must stay out of in order to be able to avoid the obstacle. In a more generalized setting, where we would like to keep our system within safe operating conditions, reachability analysis tells us the distance (with respect to a suitable metric) from the unsafe conditions the system needs to maintain.

Besides the HJ formulation, there are many other methods related to reachability anal-
ysis. In general, none of the current methods, including the HJ formulation, simultaneously address all of the challenges that need to be overcome. For example, (1, 2) excel in determining whether system trajectories from a small set of initial conditions could potentially enter a set of unsafe states, but do not provide the backward reachable set (BRS) or backward reachable tube (BRT) – the set of all initial states from which entering some target set is inevitable. Due to the challenges of computing BRSs and BRTs, the state-of-the-art methods need to make trade-offs on different axes of considerations such as computational scalability, generality of system dynamics, existence of control and/or disturbance variables, and flexibility in representation of sets.

For example, the methods presented in (3, 4, 5, 6, 7) have had success in analyzing relatively high-dimensional affine systems using sets of pre-specified shapes, such as polytopes or hyperplanes. Other potentially less scalable methods are able to handle systems with the more complex dynamics (4, 8, 9, 10, 11). Computational scalability varies among these different methods, with the most scalable methods requiring that the system dynamics do not involve control and disturbance variables. The work in (12) accounts for both control and disturbances, but is only applicable to linear systems. Methods that can account for general nonlinear systems such as (13) also sometimes represent sets using simple shapes such as polytopes, potentially sacrificing representation fidelity in favor of the other aspects mentioned earlier. Hamilton-Jacobi (HJ) formulations (14, 15, 16, 17) excel in handling general nonlinear dynamics, control and disturbance variables, and flexible set representations via a grid-based approach; however, these methods are the least computationally scalable. Still other methods make a variety of other assumptions to make desirable trade-offs (18, 19, 20). In addition, under some special scenarios, it may be possible to obtain small computational benefits while minimizing trade-offs in other axes of consideration by exploiting system structure (21, 22, 23, 24, 25, 26).

This review will focus on recent developments of the HJ formulation of reachability. It is applicable to general controlled nonlinear systems that involve disturbances or adversarial behaviors, and despite this, computes the exact reachable set rather than approximations. The trade-off here is that HJ reachability is the most computationally expensive method. As with every other formal verification method, computation burden makes HJ reachability intractable for high dimensional systems. This review presents recently developed methods to alleviate this challenge, which is referred to as the “curse of dimensionality”.

In the case of HJ reachability, the computational complexity is exponential with respect to the number of system dimensions. This is depicted by Figure 2. Using HJ reachability, 1D and 2D reachable sets can be computed very quickly, and do not use much RAM. 3D reachable sets can take minutes to hours to compute, and require hundreds of megabytes of memory. 4D reachable sets typically take many hours to days to compute, and require many gigabytes of memory. Due to computation time and memory limitations, reachable sets of 5 or more dimensions have been considered intractable to compute via the HJ formulation prior to the work on system decomposition presented in this review.

Therefore, despite its recent success, reachability methods are in general intractable for high-dimensional systems. Unfortunately, these are the systems for which performance and safety guarantees are the most urgently needed, given the recent developments in automation and systems modeling. In this review, we present progress towards tractable formal verification of complex, high-dimensional systems via reachability analysis. The solutions presented involve two broad, complementary approaches:

1. **Structural solutions.** The behavior of multi-agent systems can be non-intuitive and
Figure 2: Illustration of computational complexity of HJ reachability.

difficult to monitor. In these cases, imposing various structural assumptions, such as having air highways, on the system can significantly reduce problem complexity while allowing intuitive human participation.

2. **System decomposition.** For general high-dimensional systems, this review presents recently developed techniques to decompose a full dynamical system into multiple subsystems, reducing computation cost by many orders of magnitude and enabling previously intractable analyses.

Before diving into the specifics of the recent work, we first summarize some research done in the last decade in HJ reachability to provide the background on which the more recent works build.

### 2. Background

HJ reachability analysis falls under the umbrella of optimal control problems and differential games, which are important and powerful theoretical tools for analyzing a wide variety of systems, particularly in safety-critical scenarios. They have been extensively studied in the past several decades (14, 15, 22, 27, 28, 29, 30), and have been successfully applied to practical problems such as pairwise collision avoidance (15), aircraft in-flight refueling (31), vehicle platooning (32), and many others (33, 34). With the recent growing interest in using safety-critical autonomous systems such as autonomous cars and unmanned aerial vehicles for civil purposes (35, 36, 37, 38, 39), the importance and necessity of having flexible tools that can provide safety guarantees have substantially increased.

Intuitively, in an optimal control problem, one seeks to find the cheapest way a system described by an ordinary differential equation (ODE) model can perform a certain task. In a differential game, a system is controlled by two adversarial agents competing to respectively minimize and maximize a joint cost function. HJ reachability is a common and effective way to analyze both optimal control problems and differential games because of the guarantees
that it provides and its flexibility with respect to the system dynamics.

In a reachability problem, one is given some system dynamics described by an ODE, and a target set which describes the set of final conditions under consideration. Depending on the application, the target set can represent either a set of desired or undesired states. The goal in reachability analysis is to compute various definitions of the backward reachable tube (BRT), backward reachable set (BRS), forward reachable tube (FRT), or forward reachable set (FRS). This review mostly focuses on backward reachability. When the target set is a set of desired states, the BRT or BRS represents the set of states from which the system can be guaranteed to be driven to the target set, despite the worst case disturbance. In contrast, when the target set is a set of undesired states, the BRT or BRS represents the set of states from which the system may be driven into the target set under some disturbance, despite its best control efforts to remain outside. Because of the theoretical guarantees that reachability analysis provides, it is ideal for analyzing the newest problems involving autonomous systems. Several frequently used formal definitions of BRSs and backward reachable tubes (BRTs) will be defined in Section 2.2.1.

In addition, HJ reachability is a powerful tool because BRTs and BRSs can be used for synthesizing both controllers that steer the system away from a set of unsafe states (“safety controllers”) to guarantee safety, and controllers that steer the system into a set of goal states (“goal satisfaction controllers”) to guarantee goal satisfaction. Unlike many formulations of reachability, the HJ formulations are flexible in terms of system dynamics, enabling the analysis of controlled nonlinear systems under disturbances. Furthermore, HJ reachability analysis is complemented by many numerical tools readily available to solve the associated HJ partial differential equation (PDE) (40, 41, 42). However, the computation is typically done on a grid, making the problem complexity scale exponentially with the number of states, and therefore with the number of vehicles. Consequently, HJ reachability computations are intractable for large numbers of vehicles.

We now formalize the above notions and specialize in the HJ formulation although much of the content in the following sections are agnostic to the reachability formulation.

2.1. System Dynamics

Let \( s \in [\infty, 0] \) be the time, \( z \in \mathbb{R}^n \) be the system state, which evolves according to the ordinary differential equation (ODE)

\[
\frac{dz(s)}{ds} = \dot{z}(s) = f(z(s), u(s), d(s)), u(s) \in \mathcal{U}, d(s) \in \mathcal{D}
\]

(1)

In general, the theory we present is applicable when some states are periodic dimensions (such as angles), but for simplicity we will consider \( \mathbb{R}^n \). The control and disturbance are respectively denoted by \( u(s) \) and \( d(s) \), with the control function \( u(\cdot) \) and disturbance function \( d(\cdot) \) being respectively drawn from the set of measurable functions:

\[
u(\cdot) \in \mathcal{U}(t) = \{ \phi : [t, 0] \rightarrow \mathcal{U} : \phi(\cdot) \text{ is measurable} \}
\]

\[
d(\cdot) \in \mathcal{D}(t) = \{ \phi : [t, 0] \rightarrow \mathcal{D} : \phi(\cdot) \text{ is measurable} \}
\]

where \( \mathcal{U} \subset \mathbb{R}^{n_u} \) and \( \mathcal{D} \subset \mathbb{R}^{n_d} \) are compact, and \( t < 0 \). The system dynamics, or flow field, \( f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \rightarrow \mathbb{R}^n \) is assumed to be uniformly continuous, bounded, and Lipschitz continuous in \( z \) for fixed \( u \) and \( d \). Therefore, given \( u(\cdot) \in \mathcal{U}, d(\cdot) \in \mathcal{D} \), there exists a unique

---

The notation \((s)\) from variables such as \( z \) and \( u \) referring to function values will be omitted.
trajectory solving (1) (43). We will denote solutions, or trajectories of (1) starting from state \( z \) at time \( t \) under control \( u(\cdot) \) and disturbance \( d(\cdot) \) as \( \zeta(s; z, t, u(\cdot), d(\cdot)) : [t, 0] \to \mathbb{R}^n \). \( \zeta \) satisfies (1) with an initial condition almost everywhere:

\[
\frac{d}{ds} \zeta(s; z, t, u(\cdot), d(\cdot)) = f(\zeta(s; z, t, u(\cdot), d(\cdot)), u(s), d(s))
\]

\( \zeta(t; z, t, u(\cdot), d(\cdot)) = z \) (2)

For time-invariant system dynamics, the time variables in trajectories can be shifted by any constant \( \tau \):

\[
\zeta(s; z, t, u(\cdot), d(\cdot)) = \zeta(s + \tau; z, t + \tau, u(\cdot), d(\cdot)), \forall z \in \mathbb{R}^n
\] (3)

The interaction between disturbance and control is modeled using a differential game, as in (15). We define a strategy for the disturbance as the mapping \( \gamma : \mathcal{U} \to \mathcal{D} \) that determines a disturbance signal that reacts to the control signal based on the state. The mapping \( \gamma \) is drawn from only non-anticipative strategies \( \gamma \in \Gamma(t) \), and write \( d(\cdot) = \gamma[u(\cdot)] \).

Non-anticipative strategies are defined as follows:

\[
\gamma \in \Gamma(t) := \{ K : \mathcal{U}(t) \to \mathcal{D}(t), u(r) = \hat{u}(r) \text{ for almost every } r \in [t, s] \}
\]

\( \Rightarrow K[u](r) = K[\hat{u}](r) \text{ for almost every } r \in [t, s] \) (4)

Roughly speaking, this means that the disturbance may only react to current and past control signals. A detailed discussion of this information pattern can be found in (15).

2.2. Hamilton-Jacobi Reachability Analysis

In HJ reachability, we begin with the system dynamics given by an ordinary differential equation (ODE), and a target set which represents the set of unsafe states. We then solve the HJ equation to obtain various desired forms of reachable sets or tubes, which could represent states leading to danger. To avoid danger, the system may apply any control until it reaches the boundary of a reachable set. At the boundary, applying the optimal safety controller would guarantee avoidance. We present the most commonly used definitions in this section, and more specialized definitions in their respective sections.

2.2.1. Backward Reachable Sets and Tubes. We consider two different definitions of the BRS and two different definitions of the BRT.

Intuitively, a BRS represents the set of states \( z \in \mathbb{R}^n \) from which the system can be driven into some set \( T \subseteq \mathbb{R}^n \) at the end of a time horizon of duration \( |t| \). We call \( T \) the “target set”. First we define the “Maximal BRS”; in this case the system seeks to enter \( T \) using some control function. We can think of \( T \) as a set of goal states. The Maximal BRS represents the set of states from which the system is guaranteed to reach \( T \). The second definition is for the “Minimal BRS”; in this case the BRS is the set of states that will lead to \( T \) for all possible controls. Here we often consider \( T \) to be an unsafe set such as an obstacle. The Minimal BRS represents the set of states that leads to violation of safety requirements. Formally, the two definitions of BRSs are below:

---

2 Sometimes in the literature, the argument of \( \mathcal{R}, \mathcal{A}, \bar{\mathcal{R}}, \) or \( \bar{\mathcal{A}} \) is some non-negative number \( \tau = -t \); however, for simplicity we will use the non-positive number \( t \) to refer to the time horizon of the BRS and BRT.
Figure 3: The difference between a BRS and a BRT. The state $z_1$ is in the BRT, but not in the BRS; $z_2$ is in both the BRS and the BRT.

**Definition 1 Maximal BRS.**

$$R(t) = \{ z : \forall \gamma \in \Gamma(t), \exists u(\cdot) \in U, \zeta(0; z, t, u(\cdot), \gamma[\cdot]) \in T \}$$

**Definition 2 Minimal BRS.**

$$A(t) = \{ z : \exists \gamma \in \Gamma(t), \forall u(\cdot) \in U, \zeta(0; z, t, u(\cdot), \gamma[\cdot]) \in T \}$$

While BRSs indicate whether a system can be driven into $T$ at the end of a time horizon, BRTs indicate whether a system can be driven into $T$ at any time during the time horizon of duration $|t|$. Figure 3 demonstrates the difference. BRTs are very important notions especially in safety-critical applications, in which we are interested in determining the “Minimal BRT”: the set of states that could lead to danger at any time within a specified time horizon. Formally, the two definitions of BRTs are as follows:

**Definition 3 Maximal BRT.**

$$\bar{R}(t) = \{ z : \forall \gamma \in \Gamma(t), \exists u(\cdot) \in U, \exists s \in [t, 0], \zeta(s; z, t, u(\cdot), \gamma[\cdot]) \in T \}$$

**Definition 4 Minimal BRT.**

$$\bar{A}(t) = \{ z : \exists \gamma \in \Gamma(t), \forall u(\cdot) \in U, \exists s \in [t, 0], \zeta(s; z, t, u(\cdot), \gamma[\cdot]) \in T \}$$

The terms “maximal” and “minimal” refer to the role of the optimal control (44). In the maximal (or minimal) case, the control causes the BRS or BRT to contain as many (or few) states as possible – to have maximal (or minimal) size.

2.2.2. Computing Reachable Sets and Tubes. HJ formulations such as (14, 15, 17, 27) cast the reachability problem as an optimal control or differential game problem and directly compute BRSs and BRTs in the full state space of the system. The numerical methods (such as (44)) for obtaining the optimal solution all involve solving an HJ PDE on a grid that represents a discretization of the state space. For the time-invariant case, we now summarize necessary details related to the HJ PDEs and what their solutions represent in terms of the cost function of the corresponding optimization problem. A recent time-varying formulation can be found in (22).
Let the target set $T \subseteq \mathbb{R}^n$ be represented by the implicit surface function $V_0(z)$ as $T = \{z : V_0(z) \leq 0\}$. Such a function always exists since we can choose $V_0(\cdot)$ to be the signed distance function from $T$. Consider the optimization problem

$$V_R(t, z) = \sup_{\gamma[u]\in \Gamma(t)} \inf_{u\in U} V_0(\zeta(0; z, t, u(\cdot), \gamma[u(\cdot)])) \quad \text{subject to (2)}$$

with the optimal control being given by

$$u^*_R(\cdot) = \arg \sup_{\gamma[u]\in \Gamma(t)} \inf_{u\in U} V_0(\zeta(0; z, t, u(\cdot), \gamma[u(\cdot)]))$$

The value function $V_R(t, z)$ is the implicit surface function representing the maximal BRS: $R(t) = \{z : V_R(t, z) \leq 0\}$.

Similarly, consider the optimization problem

$$V_A(t, z) = \inf_{\gamma[u]\in \Gamma(t)} \sup_{u\in U} V_0(\zeta(0; z, t, u(\cdot), \gamma[u(\cdot)])) \quad \text{subject to (2)}$$

with optimal control

$$u^*_A(\cdot) = \arg \inf_{\gamma[u]\in \Gamma(t)} \sup_{u\in U} V_0(\zeta(0; z, t, u(\cdot), \gamma[u(\cdot)]))$$

Analogously, we also have $A(t) = \{z : V_A(t, z) \leq 0\}$.

Fig. 4 provides a simple 2D example demonstrating the relationship between the target set, implicit surface function, BRS, and value function.

The value functions $V_R(t, z)$ and $V_A(t, z)$ are the viscosity solution (45, 46) of the HJ PDE

$$D_s V(s, z) + H(z, \nabla V(s, z)) = 0, \quad s \in [t, 0]$$

$$V(0, z) = V_0(z)$$

Figure 4: An illustration of HJ reachability for a 2D state space.
The Hamiltonian $H(z, \lambda)$ depends on the system dynamics and on the optimal control problem. For example, for the optimal control problem (5), the Hamiltonian is given by

$$H(z, \lambda) = \min_{u \in U} \max_{d \in D} \lambda \cdot f(z, u, d)$$  \hspace{1cm} (10)

Once the value function $V_R$ is computed, the optimal control (6) can be obtained by the expression

$$u^*_R(s) = \arg \min_{u \in U} \max_{d \in D} \nabla V_R(s, z) \cdot f(z, u, d)$$  \hspace{1cm} (11)

Similarly, for the optimal control problem (7), the Hamiltonian is given by

$$H(s, z, \lambda) = \max_{u \in U} \min_{d \in D} \lambda \cdot f(z, u, d)$$  \hspace{1cm} (12)

and the optimal control is given by

$$u^*_A(s) = \arg \max_{u \in U} \min_{d \in D} \nabla V_A(s, z) \cdot f(z, u, d)$$  \hspace{1cm} (13)

Furthermore, by the dynamic programming principle, the optimal value on optimal trajectories must be constant:

$$V_R(s, \zeta(s)) = V_R(t, z) \forall \tau, s \in [t, 0]$$
$$V_A(s, \zeta(s)) = V_A(t, z) \forall \tau, s \in [t, 0]$$  \hspace{1cm} (14)

For the BRT, several equivalent formulations have been proposed. For example, one can modify the value function to keep track of the minimum value of the function $V_0(\cdot)$ that the system trajectory achieves over some time horizon, so that (5) and (7) respectively become

$$\bar{V}_R(t, z) = \sup_{\gamma \in \Gamma(t)} \inf_{u(\cdot) \in U} \min_{s \in [t, 0]} V_0(\zeta(0; s, t, u(\cdot), \gamma(\cdot)))$$
$$\bar{V}_A(t, z) = \inf_{\gamma \in \Gamma(t)} \sup_{u(\cdot) \in U} \min_{s \in [t, 0]} V_0(\zeta(0; s, t, u(\cdot), \gamma(\cdot)))$$  \hspace{1cm} (15)

The reader is encouraged to read the details of this formulation in (22) which contains a very general time-varying reach-avoid framework, or other formulations such as (14, 15, 16, 17). However, for this review, it suffices to note that $\bar{V}_R$ and $\bar{V}_A$ are the viscosity solution of the following HJ variational inequality:

$$\min\{D_s \bar{V}(s, z) + H(z, \nabla \bar{V}(s, z)), V_0(z) - \bar{V}(s, z)\} = 0, \hspace{0.5cm} s \in [t, 0]$$
$$\bar{V}(0, z) = V_0(z)$$  \hspace{1cm} (16)

where $H(z, \lambda)$ is given by (10), (12) for $\bar{V}_R, \bar{V}_A$ respectively.

3. System Decomposition

There are several drawbacks to using reachability analysis on large systems, whether one is using the HJ or other formulations. In this section, we briefly introduce several methods that alleviate computational burden in the HJ context by exploiting system structure. In particular, we introduce in detail a new method for decomposing a system into smaller subsystems that involves computing a BRS or BRT in lower-dimensional subspaces, and then reconstructing the full-dimensional BRS or BRT. This method, presented in Section
3.1 is based on the new concept of “self-contained subsystems” commonly found in vehicle dynamics and mechanical systems; this method substantially reduces computational requirements without incurring any additional approximation errors, as long as the system dynamics are of a compatible form. Section 3.2 briefly outlines various other methods that exploit system structure.

3.1. Decomposition via Self-Contained Subsystems

In this section, we present a system decomposition method for computing BRSs and BRTs of a class of nonlinear systems. The method is applicable when “self-contained subsystems”, defined in Equation (17), can be defined. The method drastically reduces dimensionality without making any other trade-offs by first computing BRSs for lower-dimensional subsystems, and then reconstructing the full-dimensional BRS. When reconstructing the minimal BRS \( \mathcal{A} \) by taking the intersection of lower-dimensional minimal BRSs \( \mathcal{A}_i \), and when reconstructing the maximal BRS \( \mathcal{R} \) by taking the union of lower-dimensional maximal BRSs \( \mathcal{R}_i \), any approximation errors present arise only from the lower-dimensional computations; no additional approximation errors are incurred. Crucially, the subsystems can be coupled through common states, controls, and disturbances. The treatment of this coupling distinguishes this work from others which consider completely decoupled subsystems, potentially obtained through transformations (47, 48).

The theory summarized in this section is compatible with any methods that compute BRSs and BRTs, such as (8, 9, 4, 21, 24). In addition, when different decomposition methods are combined together, even more dimensionality reduction can be achieved. A more detailed account of the material presented in this section can be found in (49).

3.1.1. Summary of Formulation and Definitions. In this section, the authors seek to obtain the BRSs and BRTs in Definitions 1 to 4 in Section 2.2.1 via computations in lower-dimensional subspaces under the assumption that the system (1) can be decomposed into self-contained subsystems (SCS) (17). Such a decomposition is common, since many systems involve components that are loosely coupled. In particular, the evolution of position variables in vehicle dynamics is often weakly coupled though other variables such as heading.

Let the state \( z \in \mathbb{R}^n \) be partitioned as \( z = (z_1, z_2, z_c) \), with \( z_1 \in \mathbb{R}^{n_1}, z_2 \in \mathbb{R}^{n_2}, z_c \in \mathbb{R}^{n_c}, n_1, n_2 > 0, n_c \geq 0, n_1 + n_2 + n_c = n \). Note that \( n_c \) could be zero.

These states are grouped into subsystems by defining the SCS states \( x_1 = (z_1, z_c) \in \mathbb{R}^{n_1+n_c} \) and \( x_2 = (z_2, z_c) \in \mathbb{R}^{n_2+n_c} \), where \( x_1 \) and \( x_2 \) in general share the “common” states in \( z_c \).

**Definition 5** Self-contained subsystem. Consider the following form of system dynamics:

\[
\dot{z}_1 = f_1(z_1, z_c, u) \\
\dot{z}_2 = f_2(z_2, z_c, u) \\
\dot{z}_c = f_c(z_c, u)
\]  

(17)

Each of the subsystems with states defined as \( x_i = (z_i, z_c) \) is called a “self-contained subsystem” (SCS).
\[ \dot{z}_1 = f_1(z_1, z_c, u) \quad \dot{z}_2 = f_2(z_2, z_c, u) \]
\[
\dot{z}_c = f_c(z_c, u) \quad \dot{z}_c = f_c(z_c, u)
\]
\[\text{(Subsystem 1)} \quad \text{(Subsystem 2)}\]

Note that the two subsystems are coupled through the common state partition \( z_c \) and control \( u \). When the subsystems are coupled through \( u \), the subsystems have “shared control”. An example of a system that can be decomposed into SCSs is the Dubins Car with constant speed \( v \):

\[
\begin{bmatrix}
\dot{p}_x \\
\dot{p}_y \\
\dot{\theta}
\end{bmatrix} =
\begin{bmatrix}
v \cos \theta \\
v \sin \theta \\
\omega
\end{bmatrix}, \quad \omega \in \mathcal{U}
\]

with state \( z = (p_x, p_y, \theta) \) representing the \( x \) position, \( y \) position, and heading, and control \( u = \omega \) representing the turn rate.

\[
\begin{bmatrix}
\dot{z}_1 \\
\dot{z}_2 \\
\dot{z}_c
\end{bmatrix} =
\begin{bmatrix}
\dot{p}_x \\
\dot{p}_y \\
\dot{\theta}
\end{bmatrix} =
\begin{bmatrix}
v \cos \theta \\
v \sin \theta \\
\omega
\end{bmatrix}, \quad \omega \in \mathcal{U}
\]

where the overlapping state is \( \theta \), and the subsystem controls and their shared component is the control \( u \) itself.

**Projection Operators**: Projection operations are defined for a state and for a set. The projection of a state \( z = (z_1, z_2, z_c) \) onto a subsystem state space \( \mathbb{R}^{n_i+n_c} \) is given by

\[
\text{proj}_i(z) = x_i = (z_i, z_c)
\]

The back-projection operator applied to a point or a set is defined to be

\[
\text{proj}^{-1}(x_i) = \{ z \in \mathbb{R}^n : (z_i, z_c) = x_i \}
\]
\[
\text{proj}^{-1}(\mathcal{S}_i) = \{ z \in \mathbb{R}^n : \exists x_i \in \mathcal{S}_i, (z_i, z_c) = x_i \}
\]

where \( \mathcal{S}_i \) is some set in \( \mathbb{R}^{n_i+n_c} \).

The authors in (49) assume that the full system target set \( T \) is representable in terms of the subsystem target sets \( T_1 \subseteq \mathbb{R}^{n_1+n_c}, T_2 \subseteq \mathbb{R}^{n_2+n_c} \) in one of the following ways:

\[
T = \text{proj}^{-1}(T_1) \cap \text{proj}^{-1}(T_2)
\]
\[
T = \text{proj}^{-1}(T_1) \cup \text{proj}^{-1}(T_2)
\]

Though more restrictive than a purely grid-based representation in the full-dimensional space, this decomposition of sets can still yield relatively complex shapes, for example those in Fig. 5 and 8.

Next, we define the subsystem BRSs \( R_i, A_i \) the same way as in Definitions 1 and 2, but with the subsystems and subsystem target sets \( T_i, i = 1, 2 \), respectively.
\[ R_i(t) = \{ x_i : \exists u(\cdot), \xi_i(0; x_i, t, u(\cdot)) \in T_i \} \]
\[ A_i(t) = \{ x_i : \forall u(\cdot), \xi_i(0; x_i, t, u(\cdot)) \in T_i \} \]

Given the above definitions, this section achieves the following:

- **Decomposition of BRSs.** Full-dimensional BRSs are efficiently computed by performing computations in lower-dimensional subspaces. Specifically, subsystem BRSs \( R_i(t) \) or \( A_i(t) \) are computed and then the full system BRS \( R(t) \) or \( A(t) \) is reconstructed by taking the union or intersection of back-projections of subsystem BRSs. This process greatly reduces computation burden by decomposing the full system into two lower-dimensional subsystems.

- **Decomposition of BRTs.** BRTs are useful since they provide guarantees over a time horizon as opposed to at a particular time. However, often BRTs cannot be decomposed the same way as BRSs; instead, BRTs can be reconstructed from BRSs.

- **Treatment of disturbances.** The theoretical framework is modified to incorporate the presence of disturbances. Slightly conservative BRSs and BRTs can still be obtained in this case.

### 3.1.2. Numerical Examples.

#### 3.1.2.1. Dubins Car.

The Dubins Car is a well-known system whose dynamics are given by (18). This system is only 3D, and its BRS can be tractably computed in the full-dimensional space, so it is used to compare the full formulation with our decomposition method. The Dubins Car dynamics can be decomposed according to Equation (19). For this example, the authors of (49) computed the BRS from the target set representing positions near the origin in both the \( p_x \) and \( p_y \) dimensions:

\[ T = \{ (p_x, p_y, \theta) : |p_x|, |p_y| \leq 0.5 \} \]

Such a target set \( T \) can be used to model an obstacle that the vehicle must avoid. Given \( T \), the interpretation of the BRS \( A(t) \) is the set of states from which a collision with the obstacle may occur after a duration of \( |t| \). From \( T \), the BRS \( A(t) \) at \( t = -0.5 \) is computed. The resulting full formulation BRS is shown in Fig. 5 as the red surface which appears in the bottom subplots. To compute the BRS using the decomposition method, the unsafe set \( T \) is written as

\[ T_1 = \{ (p_x, \theta) : |p_x| \leq 0.5 \}, T_2 = \{ (p_y, \theta) : |p_y| \leq 0.5 \} \]

\[ T = \text{proj}^{-1}(T_1) \cap \text{proj}^{-1}(T_2) \]  

(25)

From \( T_1 \) and \( T_2 \), the lower-dimensional BRSs \( A_1(t) \) and \( A_2(t) \) are computed, and then used to reconstruct the full-dimensional BRS \( A(t) = \text{proj}^{-1}(A_1(t)) \cap \text{proj}^{-1}(A_2(t)) \). The subsystem BRSs and their back-projections are shown in magenta and green in the top left subplot of Fig. 5. The reconstructed BRS is shown in the top left, top right, and bottom left subplots of Fig. 5 (black mesh).

The computation benefits of using the decomposition method can be seen from Fig. 6. One can see that the direct computation of the BRS in 3D becomes very time-consuming as the number of grid points per dimension is increased, while the computation via decomposition hardly takes any time in comparison. Directly computing the BRS with 251 grid
Figure 5: Comparison of the Dubins Car BRS $A(t = -0.5)$ computed using the full formulation and via decomposition. Top left: BRSs in the lower-dimensional subspaces (green and magenta) and how they are combined to form the full-dimensional BRS (black). Top right: BRS computed via decomposition, shown by itself. Bottom left: BRSs computed using decomposition (black) and full formulation (red), superimposed, showing that they are indistinguishable. Bottom right: BRS computed using the full formulation, shown by itself. Figure taken from (49).

Figure 6: Computation times of the two methods in log scale for the Dubins Car. The time of the direct computation in 3D increases rapidly with the number of grid points per dimension. In contrast, computation times in 2D with decomposition are negligible in comparison. Figure taken from (49).
Figure 7: Minimal BRTs computed directly in 3D and via decomposition in 2D for the Dubins Car under disturbances with shared components. The reconstructed BRT is an overapproximation of the true BRT. The over-approximated regions of the reconstruction are indicated by the arrows. An over-approximation in this case is a conservative approximation; the outside of the BRT represents the set of safe states. Figure taken from (49).

(points per dimension in 3D took approximately 80 minutes, while computing the BRS via decomposition in 2D only took approximately 30 seconds.

Fig. 7 compares the BRT \( \bar{A}(t), t = -0.5 \) computed directly from the target set in (24), and using the decomposition technique from the subsystem target sets in (25). For this computation, there is a large disturbance applied to all three components of the system dynamics. Thus, the BRT computed using the decomposition technique becomes an over-approximation of the true BRT. One can see the over-approximation by noting that the black set is not flush against the red set, as marked by the arrows in Fig. 7.

3.1.3. The 6D Acrobatic Quadrotor. The real utility of decomposition methods in general is to make previously intractable BRS and BRT computations tractable. Here, BRTs for the previously intractable 6D Acrobatic Quadrotor (50) are computed using the HJ formulation for the first time in (49). The Quadrotor has state \( z = (p_x, v_x, p_y, v_y, \phi, \omega) \), and dynamics

\[
\begin{bmatrix}
\dot{p}_x \\
\dot{v}_x \\
\dot{p}_y \\
\dot{v}_y \\
\dot{\phi} \\
\dot{\omega}
\end{bmatrix} =
\begin{bmatrix}
\frac{v_x}{m}C_D v_x - \frac{T_1}{m} \sin \phi - \frac{T_2}{m} \cos \phi \\
\frac{v_y}{m} \left( mg + C_D v_y \right) + \frac{T_1}{m} \cos \phi + \frac{T_2}{m} \cos \phi \\
-\frac{T_1}{l_{xy}} C_D \omega - \frac{T_1}{l_{xy}} T_1 + \frac{T_2}{l_{xy}} T_2
\end{bmatrix}
\]

(26)

where \( x, y, \) and \( \phi \) represent the quadrotor’s horizontal, vertical, and rotational positions, respectively. Their time derivatives represent the velocity with respect to each state. The control inputs \( T_1 \) and \( T_2 \) represent the thrust exerted on either end of the quadrotor, and the constant system parameters are \( m \) for mass, \( C_D \) for translational drag, \( C_D^r \) for rotational drag, \( g \) for acceleration due to gravity, \( l \) for the length from the quadrotor’s center to an edge, and \( I_{yy} \) for moment of inertia.

The system can be decomposed into two 4D subsystems:

\[
x_1 = (p_x, v_x, \phi, \omega), \quad x_2 = (p_y, v_y, \phi, \omega)
\]

(27)
For this example the authors in (49) compute $A(t)$ and $\bar{A}(t)$, which describe the set of initial conditions from which the system may enter the target set despite the best possible control to avoid the target. The target set is defined to be a square of length 2 centered at $(p_x, p_y) = (0, 0)$ described by $T = \{(p_x, v_x, p_y, v_y, \phi, \omega) : |p_x|, |p_y| \leq 1\}$. This can be interpreted as a positional box centered at the origin that must be avoided for all angles and velocities. From the target set, we define $V_0(z)$ such that $V_0(z) \leq 0 \Leftrightarrow z \in T$. This target set is then decomposed as follows:

$$
T_1 = \{(p_x, v_x, \phi, \omega) : |p_x| \leq 1\}
$$

$$
T_2 = \{(p_y, v_y, \phi, \omega) : |p_y| \leq 1\}
$$

The BRS of each 4D subsystem is computed and then recombined into the 6D BRS. To visually depict the 6D BRS, 3D slices of the BRS along the positional and velocity axes were computed. The left image in Fig. 8 shows a 3D slice in $(p_x, p_y, \phi)$ space at $v_x = v_y = 1, \omega = 0$. The yellow set represents the target set $T$, with the BRS in other colors. Shown on the right in Fig. 8 are 3D slices in $(v_x, v_y, \omega)$ space at $p_x, p_y = 1.5, \phi = 1.5$ through different points in time. The sets grow darker as time propagates backward. The union of the BRSs is the BRT, shown as the gray surface.

### 3.2. Other decomposition techniques

The previous decomposition technique highlighted in this review is applicable to general nonlinear systems. In the context of HJ reachability, decomposition techniques for other specific forms of system dynamics also exist. In (51), the authors proposed a Schur-based decomposition technique for computing reachable sets and synthesizing safety-preserving controllers for linear time-invariant systems. Similar to the work on self-contained subsystems, lower-dimensional reachable sets of subsystems are back-projected and intersected to construct an overapproximation of the reachable set. In (26), a similar approach for linear time-invariant systems based on a modified Riccati transformation is used. Here, decentralized computations are done in transformed coordinates of subspaces. The result is an approximation of the viability kernel, which is the complement of the minimal reachable set. Figure 9 shows the conservative approximations obtained from these decomposition techniques.
techniques.

For systems of general nonlinear dynamics, approximate methods tend to be more conservative in comparison to linear systems. For example, the approximate system decoupling technique in (24) can be used to simplify system dynamics by treating key state couplings as virtual disturbances. Conservative guarantees on system performance can still be guaranteed; in addition, the idea of “disturbance splitting” allows a trade off between the amount of computation and the degree of conservatism. In a similar fashion, the projection-based technique in (23) reduces dimensionality by using virtual disturbances to directly compute projections of reachable sets. Computation results from references (23) and (24) are shown in Figure 10.

4. Unmanned Aerial Systems Traffic Management

Unmanned aerial vehicles (UAVs) have in the past been mainly used for military operations (52, 53); however, recently there has been an immense surge of interest in using UAVs for civil applications. In the future, the use of UAVs is likely to become more and more prevalent. As a result, government agencies such as the Federal Aviation Administration (FAA) and National Aeronautics and Space Administration (NASA) are also investigating unmanned aerial systems (UAS) traffic management (UTM) in order to prevent collisions among potentially numerous UAVs (39, 35). In this section, we present recently developed HJ-based approaches for managing the airspace. We will first focus on the concept of air highways and unmanned aerial platoons, and then briefly summarize several other approaches for addressing the complexity of multi-agent systems.

4.1. Air Highways and Unmanned Aerial Platoons

In order to accommodate potentially thousands of vehicles simultaneously flying in the air, additional structure is needed to allow for tractable analysis and intuitive monitoring by
human beings. An air highway system on which platoons of vehicles travel accomplishes both goals. In the first part of this section, we propose a flexible and computationally efficient method based on (42) to perform optimal air highway placement given an arbitrary cost map that captures the desirability of having UAVs fly over any geographical location. We demonstrate our method using the San Francisco Bay Area as an example. Once air highways are in place, platoons of UAVs can then fly in fixed formations along the highway to get from origin to destination. The air highway structure greatly simplifies safety analysis, while at the same time allows intuitive human participation in unmanned airspace management. A more detailed account of the material in presented this section can be found in (54).

4.1.1. Air Highways. In (54), the authors proposed the concept of air highways, virtual highways in the airspace on which a number of UAV platoons may be present. UAVs seek to arrive at some desired destination starting from their origin by traveling along a sequence of air highways. Air highways are intended to be the common pathways for many UAV platoons, whose members may have different origins and destinations. By routing platoons of UAVs onto a few common pathways, the airspace becomes more tractable to analyze and intuitive to monitor. The authors also proposed the concept of UAV platooning through a hybrid systems approach.

Air highways must account for potential costs such as people, assets on the ground, and manned aviation, entities to which UAVs pose the biggest risks (39). Thus, given an origin-destination pair (e.g., two cities), air highways must connect the two points while potentially satisfying other criteria. In addition, optimal air highway locations should ideally be able to be recomputed in real-time when necessary in order to update airspace constraints on-the-fly, in case, for example, airport configurations change or certain airspaces have to be closed (39). With this in mind, the authors of (54) define the air highway placement problem, and propose a simple and fast way to approximate its solution that allows for real-time recomputation. The solution is based on solving the Eikonal equation, which is a specific
instance of an HJ equation. The entire air highway placement process can be thought of as converting a cost map over a geographic area in continuous space into a discrete graph whose nodes are waypoints joined by edges which are the air highways. Using the San Francisco Bay Area as an example, the authors of (54) classified each point on the map into four different regions with descending cost: “regions around airports”, “highly populated cities”, “water”, and “other”. The associated cost of each region reflects the desirability of flying a vehicle over an area in the region. In general, these costs can be arbitrary and determined by government regulation agencies.

Figure 11 shows the San Francisco Bay Area geographic map, cost map, cost-minimizing paths, and the resulting air highways. The region enclosed by the black boundary represents “region around airports”, which have the highest cost. The dark blue, yellow, and light blue regions represent the “cities”, the “water”, and the “other” categories, respectively. The origin city is assumed to be “Concord”, and a number of other major cities are chosen as destinations.

The cost-minimizing paths to the various destinations in general overlap, and only split up when they are very close to entering their destination cities. This intuitive placement of highways mimics highway network designed by humans. In addition, since the computation is done on a 2D domain, placement of air highways can be done in real-time if the cost map changes at a particular time.

4.1.2. Unmanned Aerial Platoons. Air highways exhibiting trunk routes that separate near destinations motivate the use of platoons which fly on these highways. The air highway structure along with the UAV platooning concept together enable the use of reachability to analyze safety and goal satisfaction properties. The structure reduces the likelihood of multiple-way conflicts, and makes pairwise analysis more indicative of the joint safety of all UAVs. In addition to reducing complexity, the proposed structure is intuitive, and allows human participation in the monitoring and management of the unmanned airspace.

Organizing UAVs into platoons implies that the UAVs cannot fly in an unstructured way, and must have a restricted set of controllers or maneuvers depending on the UAV’s role in the airspace. To model UAVs flying in platoons on air highways, we propose a hybrid system whose modes of operations describe a UAV’s role in the highway structure. For the hybrid system model, reachability analysis is used to enable successful and safe operation and mode transitions.

In general, the problem of collision avoidance among $N$ vehicles cannot be tractably solved using traditional dynamic programming approaches such as HJ reachability. Instead,
the structure imposed by air highways and platooning enables analysis of the safety and goal satisfaction properties of the vehicles in a tractable manner. From the perspective of each vehicle, the allowable maneuvers become restricted; the authors of (54) use a hybrid system model to capture this concept. The available maneuvers and associated mode transitions are summarized in Figure 12.

Given the above modeling assumptions, the authors of (54) provide control strategies to guarantee the success and safety of all the mode transitions. The theoretical tool used to provide the safety and goal satisfaction guarantees is HJ reachability. The BRTs computed allow each vehicle to perform complex actions such as

- merge onto a highway to form a platoon
- join a new platoon
- leave a platoon to create a new one
- react to malfunctioning or intruder vehicles

Several basic controllers to perform other simpler actions such as

- follow the highway at constant altitude at a specified speed
- maintain a constant relative position and velocity with respect to the leader of a platoon

In general, the control strategy of each vehicle has a safety component, which specifies a set of states that it must avoid, and a goal satisfaction component, which specifies a set of states that the vehicle aims to reach. Together, the safety and goal satisfaction controllers guarantee the safety and success of a vehicle in the airspace making any desired mode transition. By combining HJ reachability with a hybrid system structure, the multi-UAV system is able to perform joint maneuvers essential to maintaining structure in the airspace.
4.1.2.1. Reachability-Based Controllers. Reachability analysis is useful for constructing controllers in a large variety of situations. In order to construct different controllers, an appropriate target set needs to be defined depending on the goal of the controller. If one defines the target set to be a set of desired states, the BRS would represent the states that a system needs to first arrive at in order to reach the desired states. On the other hand, if the target set represents a set of undesirable states, then the BRS would indicate the region of the state space that the system needs to avoid. In addition, the system dynamics with which the BRS is computed provide additional flexibility when using reachability to construct controllers.

Using a number of different target sets and dynamics, several different reachability-based controllers used for vehicle mode transitions are as follows.

Getting to a Target State: The controller used by a vehicle to reach a target state is important in two situations in the platooning context. First, a vehicle in the “Free” mode can use the controller to merge onto a highway, forming a platoon and changing modes to a “Leader” vehicle. Second, a vehicle in either the “Leader” mode or the “Follower” mode can use this controller to change to a different highway, becoming a “Leader” vehicle.

Getting to a State Relative to Another Vehicle: In the platooning context, being able to go to a state relative to another moving vehicle is important for the purpose of forming and joining platoons. For example, a “Free” vehicle may join an existing platoon that is on a highway and change modes to become a “Follower”. Also, a “Leader” or “Follower” may join another platoon and afterwards go into the “Follower” mode.

Avoiding Collisions: A vehicle can use a goal satisfaction controller described in the previous sections when it is not in any danger of collision with other vehicles. If the vehicle could potentially be involved in a collision within the next short period of time, it must switch to a safety controller. The safety controller is available in every mode, and executing the safety controller to perform an avoidance maneuver does not change a vehicle’s mode.

In the context of the platooning concept, an unsafe configuration can be defined as follows: a vehicle is either within a minimum separation distance to a reference vehicle in both the x and y directions, or is traveling with a speed above the speed limit in either of the x and y directions. From this specification, a minimal BRT can be computed to provide a guaranteed-safe controller.

4.1.2.2. Other Controllers. HJ reachability is used for the relatively complex maneuvers that require safety and goal satisfaction guarantees. For the simpler maneuvers of traveling along a highway and following a platoon, many other well-known methods such as proportional-integral-derivative control or model-predictive control would suffice.

4.1.2.3. Numerical Simulations. The authors of (54) consider several situations that vehicles in a platoon on an air highway may commonly encounter, and show via simulations the behaviors that emerge from the proposed controllers. Figure 13 shows the results. The left plot illustrates the scenario in which Free vehicles merge onto an initially unoccupied highway. Relevant BRTs for both goal-satisfaction and safety are shown. All 5 vehicles eventually form a single platoon and travel along the highway together. As with the first two vehicles, the goal satisfaction controllers allow the remaining vehicles to optimally and smoothly join the platoon, while the safety controllers prevent collisions from occurring.
(a) The purple vehicle is joining the platoon while avoiding collisions.
(b) Safety controllers cause vehicles to respond to the intruder in yellow.

Figure 13: Simulations showing emerging behavior of the proposed hybrid system framework. Figures taken from (54).

The right plot illustrates a platoon’s automatic response to an intruder. To avoid collision, each vehicle checks for safety with respect to the intruder and any vehicles in front and behind of it in the platoon. When the intruder in yellow comes in close proximity, the other vehicles spread out to avoid collision. After danger has passed, the vehicles in the platoon resume normal operation according to the hybrid system model.

4.2. Hamilton-Jacobi-Based Multi-Agent Analysis

An air highway network alone is likely not sufficient for UAVs to travel to the final postal address. In (55, 56), a sequential trajectory planning scheme is proposed. Although HJ reachability is well-suited for the robustness requirements needed for the airspace, simultaneous analysis of all vehicles is intractable. So in (55), the authors assign a strict priority ordering, with lower-priority vehicles treating higher-priority vehicles as moving obstacles. This allows reservation of “space-time” in the airspace for each vehicle. The space-time reservation is dynamically feasible to track when the vehicle experiences disturbances (56) and even in the presence of a single adversarial intruder vehicle under certain assumptions (57). A simulation study was done in (57) to demonstrate space-time reservation variants with different assumptions; a simulation involving one adversarial intruder is shown in Figure 14.

An air highway structure and robust routing of UAVs is useful as a first level of safety. Additional levels of safety can be provided by last-resort collision avoidance. The authors in (15) demonstrate guaranteed-safe pairwise collision avoidance, and in (58, 59), the scalability limitations of HJ reachability is alleviated by a mixed integer program that exploits the properties of pair-wise HJ solutions to provide higher-level control logic. Safety guarantees for three-vehicle collision avoidance is proved in (58) — a previously intractable task for HJ reachability — without incurring significant additional computation cost. The collision avoidance protocol can also be applied to systems involving more than three vehicles, although no theoretical guarantees can be made. Figure 15a shows an 8-vehicle collision
(a) Green and red vehicles deviate from their nominal positions significantly to avoid the intruder (black vehicle).

Figure 14: Robust space-time reservations allow multiple UAVs to arrive at their destinations while avoiding an intruder. Figures taken from (57).

(b) Actual and nominal trajectories of the red ($Q_2$) and green ($Q_3$) vehicles.

Rogue UAVs which are unregistered or unauthorized will potentially need to be monitored and captured; such a multi-vehicle capture-avoid problem is formulated and solved in (60) for vehicles of holonomic dynamics. The authors’ voronoi cell based method is illustrated in Figure 15b. To incorporate nonlinear dynamics, a reach-avoid game between a team of attackers and a team of defenders can be solved using HJ reachability combined with maximum matching (59). An example of this “man-to-man defense” solution is shown in Figure 15c.

(a) Multi-vehicle collision avoidance simulation. Figure taken from (58).

(b) Simulation of voronoi-based rogue vehicle capture. Figure taken from (60).

(c) The maximum matching process for rogue UAV interception. Figure taken from (59).

Figure 15: Multi-vehicle analysis using HJ reachability and higher-level logic.

5. Conclusion

Autonomous systems research has been tremendously successful recently, and now the perspective of safety is becoming very important, despite the difficulties of safety analysis. With the recent progress in high-dimensional verification via HJ reachability, and a com-
bination of low-dimensional verification and higher-level logic, we have made a good start on the path towards more pervasive and verified automation. If large-scale safety analysis could be combined with previous successes in the field in a modular way, we could have safe system design, planning, sensing, and learning, safe large-scale autonomous systems, and safe human-automation interaction in the near future.

ACKNOWLEDGMENTS

This review was supported by NSF under the CPS Frontiers VehiCal project (1545126) and CPS:ActionWebs (CNS-931843), by the UC-Philippine-California Advanced Research Institute under project IIID-2016-005, by the ONR MURI Embedded Humans (N00014-16-1-2206), and by NASA under grants NNX12AR18A and UCSCMCA-14-022 (UARC).

We would also like to thank Sylvia Herbert, Mahesh Vashishtha, and Somil Bansal for their collaboration on the paper (49) related to system decomposition; and Qie Hu, Jaime Fisac, Kene Akametalu, and Casey Mackin for their collaboration on the paper (54) related to UTM. Both papers, for which they are co-authors, are featured in the review.

LITERATURE CITED


26 Mo Chen, Claire Tomlin
   *Airpower Journal* 41–55


   Reachability-Based Safety and Goal Satisfaction of Unmanned Aerial Platoons on Air Highways. 
   *AIAA Journal of Guidance, Control, and Dynamics* 40:1–14

   systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality. In 

   and Imperfect Information. In *Proceedings of the American Control Conference*. Seattle: 
   IEEE. pp. 5550–5555

57. Chen M, Bansal S, Fisac JF, Tomlin C.J. ????. Robust Sequential Path Planning Under Disturbances 
   and Adversarial Intruder. *IEEE Transactions on Control Systems Technology* (to appear)

   Reachability and Mixed Integer Programming. In *Proceedings of the Conference on Decision 
   and Control*. Las Vegas: IEEE. pp. 1695–1700

   *IEEE Transactions on Automatic Control* 62:1451–1457

   Multiple Evaders With Multiple Pursuers. *IEEE Robotics and Automation Letters* 2:530–537