Formal Verification of Distributed Aircraft Controllers*

Sarah M. Loos
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA, USA
sloos@cs.cmu.edu

David Renshaw
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA, USA
dwrensha@cs.cmu.edu

André Platzer
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA, USA
aplatzer@cs.cmu.edu

ABSTRACT

As airspace becomes ever more crowded, air traffic management must reduce both space and time between aircraft to increase throughput, making on-board collision avoidance systems ever more important. These safety-critical systems must be extremely reliable, and as such, many resources are invested into ensuring that the protocols they implement are accurate. Still, it is challenging to guarantee that such a controller works properly under every circumstance. In tough scenarios where a large number of aircraft must execute a collision avoidance maneuver, a human pilot under stress is not necessarily able to understand the complexity of the distributed system and may not take the right course, especially if actions must be taken quickly. We consider a class of distributed collision avoidance controllers designed to work even in environments with arbitrarily many aircraft or UAVs. We prove that the controllers never allow the aircraft to get too close to one another, even when new planes approach an in-progress avoidance maneuver that the new plane may not be aware of. Because these safety guarantees always hold, the aircraft are protected against unexpected emergent behavior which simulation and testing may miss. This is an important step in formally verified, flyable, and distributed air traffic control.

Categories and Subject Descriptors

F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

Keywords

formal verification; distributed aircraft controllers

1. INTRODUCTION

Verification of air traffic control is particularly challenging because it lies in the intersection of many fields which already give tough verification problems when examined independently. It is a distributed system, with a large number of aircraft interacting over an unbounded time horizon. Each aircraft has nonlinear continuous dynamics combined with complex discrete controllers. And finally, every protocol must be flyable (i.e. not cause the aircraft to enter a stall, bank too sharply, or require it to turn on sharp corners).

In this paper, we investigate the safety of collision-avoidance controllers for aircraft systems. We want to prove safety not just for a single aircraft or a pair of aircraft, but for all aircraft operating simultaneously in the sky. Because this system is composed of multiple independent computational agents that interact with the physical world, it is called a *distributed hybrid system*. It is this combination of continuous flight dynamics, discrete flight control decisions, and distributed communication that causes verification of aircraft control protocols to be extremely challenging.

Aircraft control systems are safety-critical, so they must be designed with a high assurance of correctness. When the costs of failure are high, system designers must be able to guarantee ahead of time that their systems work as intended. Many methods, such as testing and simulation, are used in combination to improve reliability. While testing and simulation may reveal software bugs and increase safety assurance, they are not able to prove safety guarantees over the continuous and infinite state-spaces characteristic of hybrid systems like flight control, where the aircraft move continuously through space and time. The complexity of curved flight dynamics has been difficult for many analysis techniques [1–8], which often resort to unflyable approximations of flight trajectories that require aircraft to turn on corners. However, the formal verification techniques described in this paper are able to provide guarantees for flyable maneuvers over the entirety of this continuous state-space and therefore over all evolutions of all aircraft movement.

These strong guarantees are especially important in a distributed system with a large number of interacting participants. As in [1, 4, 7, 9, 10], many previous approaches to aircraft control have looked into a relatively small number of agents. But with thousands of aircraft flying through commercial airspace daily, this system is already far too complex for humans to predict every scenario by looking at interactions of only a few aircraft. And this challenge increases when we examine controllers for Unmanned Aerial Vehicles (UAVs), which are becoming increasingly autonomous and fly even closer together, with less direct supervision by humans. As a result, we must provide a good argument for why a controller will always take the right action, even in extremely crowded airspace.

In this paper we specify and verify two control policies for planar aircraft avoidance maneuvers using automated theorem prover KeYmaeraD to produce a proof of safety for each of them. We design these policies such that all aircraft adhere to a simple and easy-to-implement separation principle: associated with each aircraft is

*This material is based upon work supported by the National Science Foundation under NSF CAREER Award CNS-1054246, NSF EXPEDITION CNS-0926181, grant no. CNS-0931985, and grant no. CNS-1035800. Sarah M. Loos was supported by a DOE Computational Science Graduate Fellowship.

HSCC'13, April 8–11, 2013, Philadelphia, Pennsylvania, USA.

a disc, within which the aircraft must remain. In this way, the problem reduces to proving that i) sufficient separation is maintained between pairs of discs, and ii) individual aircraft always remain inside their associated disc. We model 2D flight dynamics since they are the relevant dynamics for planar maneuvers, but investigating 3D maneuvers and dynamics may make interesting future work.

The complexities which arise from the curved flight trajectories of an arbitrary number of aircraft interacting in a distributed manner, along with the tight coupling of discrete control and continuous dynamics presently make KeYmaeraD the only verification tool capable of proving safety for this system. Our contributions are:

- We provide the *first formally* verified distributed system of aircraft with *curved flight* dynamics.
- Our controller requires only *flyable* aircraft trajectories with no corners or instantaneous changes of ground speed.
- We prove our controller is safe for an *arbitrarily large number of aircraft*. This guarantee is necessary for high-traffic applications such as crowded commercial airspace, unmanned aerial vehicle maneuvers, and robotic swarms.
- Other aircraft may enter an avoidance maneuver already *in progress* and safety for all aircraft is guaranteed still.
- We use *Arithmetic coding* to reduce proof complexity and branching.
- We prove that even when the interactions of many aircraft cause unexpected *emergent behaviors*, all resulting control choices are still safe.
- We present *hierarchical and compositional* techniques to reduce a very complex system into smaller, provable pieces.

2. RELATED WORK

Many methods for ensuring correctness have been researched, each having different strengths in dealing with the various challenges posed by air traffic control. Pallottino et al. [11] proposed a distributed collision avoidance policy that is closely related to the systems we examine here. They provide a thorough empirical description of the system's behavior, emphasizing simulation and physical experiment. They formulate a liveness property and give probabilistic evidence for it using Monte Carlo methods. They also provide an informal proof of safety that is similar in high-level ideas to our proofs, but does not consider a model for flight dynamics. However, since we provide *formal* proofs of safety based directly on the control protocols and working with a continuous model of flight dynamics, we provide a higher degree of assurance and a clearer avenue to safely extend the systems.

Verification methods for systems with an arbitrary number of agents behaving under distributed control fall primarily into one of two categories: theorem proving and parameterized verification. Johnson and Mitra [8] use parameterized verification to guarantee that a distributed air traffic landing protocol (SATS) is collision free. Using backward reachability, they prove safety in the SATS landing maneuver given a bound on the number of aircraft that can be engaged in the landing maneuver. The protocol divides the airspace into regions and models the aircraft flight trajectory within each region by a clock. We consider the complementary problem of free flight instead of airport landing traffic, and we show that in free space, *arbitrarily* many aircraft can join our maneuver, and we model aircraft movement using *flyable*, curved flight dynamics.

Other provably safe systems with a specific (usually small) number of agents are presented in [1, 4, 7, 9]. The work by Umeno and Lynch [4, 7] is complementary to ours; however while they consider real-time properties of airport protocols using Timed I/O Automata, we prove local properties of the actual hybrid system flight dynamics. Duperret et al. [9] verify a roundabout maneuver with three vehicles. Each vehicle is constrained to a specific, pre-defined path, so physical dynamics are simplified to one dimension. Tomlin et al. [1] analyze competitive aircraft maneuvers game theoretically using numerical approximations of partial differential equations. As a solution, they propose roundabout maneuvers and give bounded-time verification results for up to four aircraft using straight-line approximations of flight dynamics.

Flyability is identified as a major challenge in Košecká et al. [12], where planning based on superposition of potential fields is used to resolve air traffic conflicts. This planning does not guarantee flyability but, rather, defaults to classical vertical altitude changes whenever a nonflyable path is detected. The resulting maneuver has not yet been verified. The planning approach has been pursued by Bicchi and Pallottino [13] with numerical simulations.

Numerical simulation algorithms approximating discrete-time Markov Chain approximations of aircraft behavior have been proposed by Hu et al. [2]. They approximate bounded-time probabilistic reachable sets for one initial state. We consider hybrid systems combining discrete control choices and continuous dynamics instead of uncontrolled, probabilistic continuous dynamics. Hwang et al. [6] have presented a straight-line aircraft conflict avoidance maneuver involving optimization over complicated trigonometric computations, and validate it using random numerical simulation and informal arguments. The work of Dowek et al. [3] and Galdino et al. [5] shares many goals with ours. They consider unflyable, straight-line maneuvers and formalize geometrical proofs in PVS.

Our approach has a different focus from complementary work:

- Our maneuver directly involves curved flight unlike [1–8]. This makes our maneuver more realistic since it is *flyable*, but much more difficult to analyze.
- Unlike [2,6,12], we do not give results for a finite (sometimes small) number of initial flight positions (as in simulation). Instead, we verify uncountably many initial states and give unbounded-time horizon verification results.
- Unlike [1, 2, 6, 12–14], we use symbolic computation so that numerical and floating point errors can not violate soundness.
- Unlike [2–8, 13, 15], we analyze hybrid system dynamics directly, not approximations like clocks.
- Unlike [1,2,6,11–13,15] we produce *formal*, deductive proofs.
- In [3–7, 11], it is not proved that the hybrid dynamics and flight equations follow the geometrical thoughts. In contrast, our approach directly works for the hybrid flight dynamics.
- Unlike [1–3,5,6,9,10,12,13], we verify the case of *arbitrarily many* aircraft, which is crucial for dense airspace.
- Unlike [13, 14], we do not guarantee optimality of the resulting maneuver.

3. PRELIMINARIES

Quantified Hybrid Programs.

QHPs [16, 17] are defined by the following grammar (α, β are QHPs, θ terms, i a variable of sort C, f a function symbol, s a term with sort compatible to f, and H is a formula of first-order logic):

$$\alpha, \beta \ ::= \ \forall i : C\, \mathcal{A} \mid \forall i : C\, \{\mathcal{D}\ \&\ H\} \mid ?H \mid \alpha \cup \beta \mid \alpha;\beta \mid \alpha^*$$

where $\mathcal{A}$ is a list of assignments of the form $f(s) := \theta$ and nondeterministic assignments of the form $f(s) := *_C$, and $\mathcal{D}$ is a list of differential equations of the form $f(s)' = \theta$. When an assignment list does not depend on the quantified variable i, we may elide the quantification for clarity.

The effect of *assignment* $f(s) := \theta$ is a discrete jump assigning θ to $f(s)$. The effect of *nondeterministic assignment* $f(s) := *_C$ is

a discrete jump assigning *any value* in C to $f(s)$. The effect of *quantified assignment* $\forall i:C\ \mathcal{A}$ is the simultaneous effect of all assignments in $\mathcal{A}$ for all objects i of sort C. For example, the QHP $\forall i:C\ \omega(i) := \omega(i) + 1$ expresses that all aircraft i of sort C simultaneously increase their angular velocity. The effect of *quantified differential equation* $\forall i:C\{\mathcal{D}\&H\}$ is a continuous evolution where, for all objects i of sort C, all differential equations in $\mathcal{D}$ hold and formula H holds throughout the evolution (i.e. the state remains in the region described by *evolution domain constraint* H). The dynamics of QHPs changes the interpretation of terms over time: for an $\mathbb{R}$-valued function symbol f, $f(\vec{s})'$ denotes the derivative of the interpretation of the term $f(\vec{s})$ over time during continuous evolution, not the derivative of $f(\vec{s})$ by its argument $\vec{s}$. For this paper, f does not occur in $\vec{s}$. In most cases, $\vec{s}$ is just i. For instance, the following QHP expresses that all aircraft i of sort C fly by $\forall i:C\ x(i)' = d(i), d(i)' = \omega(i)d(i)$ such that their position $x(i)$ changes continuously according to their respective direction $d(i)$ and their direction changes according to their angular velocity $\omega(i)$.

The effect of *test* $?H$ is a *skip* (i.e., no change) if formula H is true in the current state and *abort* (blocking the system run by a failed assertion), otherwise. *Nondeterministic choice* $\alpha \cup \beta$ is for alternatives in the behavior of the distributed hybrid system. In the *sequential composition* $\alpha;\beta$, QHP β starts after α finishes (β never starts if α continues indefinitely). *Nondeterministic repetition* α^* repeats α an arbitrary number of times, possibly zero times.

Quantified Differential Dynamic Logic.

The formulas of QdL [16, 17] are defined as in first-order dynamic logic plus many-sorted first-order logic by the following grammar (ϕ, ψ are formulas, θ_1, θ_2 are terms of the same sort, i is a variable of sort C, and α is a QHP):

$$\phi, \psi \ ::= \theta_1 = \theta_2 \mid \theta_1 \geq \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi$$
$$\mid \forall i:C\ \phi \mid \exists i:C\ \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi$$

We use standard abbreviations to define $\leq, >, <, \rightarrow$. The real numbers $\mathbb{R}$ form a distinguished sort, upon which are defined the rigid functions + and $\times$. Sorts $C \neq \mathbb{R}$ have no ordering and hence $\theta_1 = \theta_2$ is the only relation allowed on them. For sort $\mathbb{R}$, we abbreviate $\forall x:\mathbb{R}\ \phi$ by $\forall x\, \phi$. In the following, all formulas and terms have to be well-typed. QdL formula $[\alpha]\phi$ expresses that *all states* reachable by QHP α satisfy formula ϕ. Likewise, $\langle\alpha\rangle\phi$ expresses that *there is at least one state* reachable by α for which ϕ holds.

Proof Calculus and Prover.

The QdL proof calculus [16, 17] consists of *proof rules* that operate on *sequents*, which are syntactic objects of the form $\Gamma \Rightarrow \Delta$ where Γ and Δ are finite sets of QdL formulas. Loos et al. [18] use QdL to verify adaptive cruise control for arbitrarily many cars on a highway, with simple continuous dynamics on a straight lane.

KeYmearaD is a theorem prover which mechanizes the use of the QdL proof calculus. It has previously been used to verify a simple car control system [19]. KeYmaeraD further implements quantified differential invariants [20]. KeYmaeraD constructs proofs by following a user-created *tactic script*. When KeYmaeraD runs a tactic script, it applies proof rules to the sequent based on tactics specified in the script which will ultimately reduce the sequent to several problems in first-order real arithmetic. These simpler problems are then sent to a backend decision procedure in Mathematica.

4. CASE STUDY SYSTEMS

In this section, we present two classes of aircraft controllers, each of which maintains a guaranteed minimum distance p between all aircraft. We then prove that both of these controller classes are safe, (i.e. that the minimum distance p between aircraft is never violated). Each aircraft has a disc-shaped zone large enough to fly a circle within and which no other aircraft will be allowed to enter.

In the first controller class (Section 4.1), each aircraft maintains a larger buffer disc with the aircraft at its center. This disc allows pilots some freedom during an avoidance maneuver, including the choice of circling direction. We imagine this controller will be useful when passenger comfort is a factor, as in commercial airlines. The second class of controllers (Section 4.2) uses smaller buffer discs centered to the left or right of the aircraft. The smaller discs allow the aircraft to fly closer together, but there may be little choice in how a maneuver is executed. This is well suited to UAVs which may fly very close together and are concerned only with flyability, not passenger comfort. Additionally, since many UAVs may be monitored and managed remotely by a small group of people, it may be more desirable to have a specific collision avoidance maneuver with little freedom and high predictability. Because the first controller class requires a larger disc than the second, we call it *Big Disc*, and appropriately we name the second class *Small Discs*.

We use two levels of abstraction to analyze the controllers. At the higher abstraction level we model the buffer discs, which can freeze instantaneously when they get within p distance of each other. At the lower level, we model the movement of aircraft within their discs, ensuring they always stay within the buffer zone while following flyable trajectories. In the proof, these two levels of abstraction are joined so that safety is assured for the system as a whole.

We model airspace as $\mathbb{R}^2$ and aircraft as points moving in this space. Each aircraft i steers by adjusting its angular velocity $\omega(i)$. When $\omega(i)$ is zero, the plane flies in a straight line. As angular velocity increases, the plane flies in a tighter and tighter circle, so we put an upper bound on the angular velocity $\Omega(i)$ based on the smallest circle that aircraft i can fly while maintaining constant linear speed $v(i)$. We keep linear speed $v(i)$ constant for each aircraft. We can determine the radius of each aircraft's smallest flyable circle by the equation $minr(i) = v(i)/\Omega(i)$. This model is known as the *Dubins vehicle* [21] and has been used previously for aircraft verification [1]. We allow an arbitrarily large number of aircraft to be present in airspace, so long as there is enough space to pack their discs. To our knowledge, no other method has been able to verify a protocol or controller safe for an arbitrary number of aircraft using a continuous model of their flight dynamics. This is not surprising, since safety must be guaranteed even for unpredictable emergent behaviors and in crowded, worst-case scenarios. Models written as QHPs inherently have a compositional and hierarchical structure which makes them easier to decompose into smaller, provable pieces by using sound proof rules. We also use nondeterminism in the model of the controller, which means that our proof is robust to variations in implementation on individual aircraft.

4.1 Big Disc

During normal free flight (i.e. whenever the aircraft is not engaged in a collision avoidance maneuver), the buffer zone for an aircraft i is a disc of radius $2minr(i)$ centered around the aircraft, which is at planar position $x(i) = (x_1(i), x_2(i))$. So long as the aircraft does not enter a collision avoidance maneuver, its buffer disc remains centered on the aircraft. However, should aircraft i come too close to another plane, it will enter collision avoidance mode and begin circling at radius $minr(i)$ to either the left or the right. The disc allows just enough room for this maneuver; however, the disc is big in the sense that it allows a considerable amount of freedom once the aircraft has gone halfway around this initial circle. The beginning of one possible trajectory of a collision avoidance

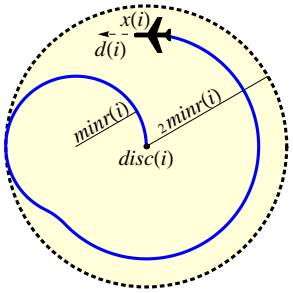

Figure 1: A possible collision avoidance trajectory of `BigDisc`.

maneuver is illustrated in Figure 1. The current direction of flight of aircraft i is given by the indexed variable $d(i)$ as a 2D unit vector. The variable $disc(i)$ stores the position of the center of i's buffer disc. The aircraft need not always turn at the maximum angular velocity $\Omega(i)$; we require only that the aircraft remain within the disc by circling in its original direction. (Note: while it is possible for the aircraft to change its circling direction while staying within the disc by flying a figure eight or an 'S' shape, we disallow this behavior since it would increase the complexity of the controller.)

4.1.1 Formal Model

The Big Disc policy is presented formally in QHP 1 and we describe it in this section. The variable $ca(i)$, indicates whether aircraft i is in a collision avoidance maneuver. If $ca(i) = 0$ then i is in free flight; if $ca(i) = 1$ then i is in an avoidance maneuver and is circling within its disc. Each aircraft has the ability to enter collision avoidance independently and asynchronously. This simplifies collision avoidance maneuvers with more than two aircraft and improves reliability, since no perfect synchronization is required, which would be difficult to implement in a distributed system.

The variable $side(i)$ indicates i's circling direction when it enters an avoidance maneuver. If $side(i) = 1$, i will circle counter-clockwise; if $side(i) = -1$, i circles clockwise. We use $\|y\|$ to denote the Euclidean norm and we use $y^\perp$ to denote the vector obtained by rotating y ninety degrees counter-clockwise, $y^\perp := (-y_2, y_1)$.

The quantified hybrid program `BigDisc` is a loop, which is represented in line 1 by *, the nondeterministic repetition operator. Each iteration is either a control action as represented by `Control` or an evolution of physics as represented by `Plant`. This loop may repeat arbitrarily many times. In the `Control` branch, the program nondeterministically selects an aircraft k ($k := *_{\mathbb{A}}$ assigns an arbitrary aircraft into k) and then allows k to perform some action. The allowed actions depend on whether k is in a collision avoidance maneuver. If it is (case `CA`), then k may either adjust its angular velocity in the `Steer` branch, or exit the maneuver with the `Exit` branch. The new angular velocity in the `Steer` branch is arbitrary ($\omega(k) := *_{\mathbb{R}}$, where $*_{\mathbb{R}}$ is an arbitrary real number) but bounded by $-\Omega(k)$ and $\Omega(k)$ due to the subsequent test. The aircraft may only `Exit` the collision avoidance circling maneuver when $x(k) = disc(k)$, i.e., the aircraft must return to the center of the disc before exiting the maneuver. If k is not in a collision avoidance maneuver (case `NotCA`), then it may once again `Steer`, or it may switch its circling direction with the `Flip` branch, or it may enter collision avoidance with the `Enter` branch. In the `Enter` branch, the aircraft sets its angular velocity so that it will circle with radius $minr(k)$, thereby entering a collision avoidance. It also sets the $ca(k)$ flag to indicate internally that it has entered this maneuver.

The other branch in `BigDisc`'s main loop is `Plant`. The position of the aircraft, $x(i)$, changes according to its direction, $d(i)$, which in turn changes according to its angular velocity, $\omega(i)$. This

Quantified Hybrid Program 1 Big Disc

$$\texttt{BigDisc} \equiv (\texttt{Control} \cup \texttt{Plant})^* \quad (1)$$
$$\texttt{Control} \equiv k := *_{\mathbb{A}};\ (\texttt{CA} \cup \texttt{NotCA}) \quad (2)$$
$$\texttt{CA} \equiv ?(ca(k) = 1);\ (\texttt{Steer} \cup \texttt{Exit}) \quad (3)$$
$$\texttt{NotCA} \equiv ?(ca(k) = 0);\ (\texttt{Steer} \cup \texttt{Flip} \cup \texttt{Enter}) \quad (4)$$
$$\texttt{Steer} \equiv \omega(k) := *_{\mathbb{R}};\ ?(-\Omega(k) \le \omega(k) \le \Omega(k)) \quad (5)$$
$$\texttt{Exit} \equiv ?(disc(k) = x(k));\ ca(k) := 0 \quad (6)$$
$$\texttt{Enter} \equiv \omega(k) := side(k) \cdot \Omega(k);\ ca(k) := 1 \quad (7)$$
$$\texttt{Flip} \equiv side(k) := -side(k) \quad (8)$$
$$\texttt{Plant} \equiv \forall i : \mathbb{A} \left(x(i)' = v(i) \cdot d(i),\ d(i)' = \omega(i) \cdot d(i)^\perp, \right. \quad (9)$$
$$\left. disc(i)' = (1 - ca(i)) \cdot v(i) \cdot d(i)\ \&\ \texttt{EvDom} \right) \quad (10)$$
$$\texttt{EvDom} \equiv \forall j : \mathbb{A} \quad (11)$$
$$((j \neq i \wedge (ca(i) = 0 \vee ca(j) = 0)) \rightarrow \mathsf{Sep}(i, j) \quad (12)$$
$$\wedge \|disc(i) - (x(i) + minr(i) \cdot side(i) \cdot d(i)^\perp)\|$$
$$\le minr(i)) \quad (13)$$
$$\mathsf{Sep}(i, j) \equiv \|disc(i) - disc(j)\| \ge 2minr(i) + 2minr(j) + p \quad (14)$$

makes $\omega(i)$ our primary control variable. These physical dynamics are modeled by the differential equation in line 9. The center of the disc, $disc(i)$, is stationary during a collision avoidance maneuver, but otherwise it is equal to the aircraft position. This case distinction is achieved by using arithmetic coding, whereby we multiply by $(1 - ca(i))$, in line 10. This reduces a branching of the system which would be incurred if we were to use traditional if-else coding style, and thereby reduces the complexity of the safety proof. When the aircraft is in free flight, $ca(i) = 0$, which causes $disc(i)'$ to equal $x(i)'$. But, when the aircraft is in a collision avoidance maneuver, $ca(i) = 1$, causing $disc(i)' = 0$, so the disc is stationary. The evolution domain, `EvDom`, has two purposes. First, it monitors the disc positions of other aircraft (line 12). Recall that in order for the system to be considered safe, no aircraft can pass closer than distance p to another. So, if the aircraft's disc comes within p of another disc (as quantified in line 14), it forces both aircraft to enter collision avoidance. Second, while the aircraft has a great amount of freedom in how it maneuvers during collision avoidance, it must always be able to flyably remain within its buffer disc. We use the inequality in line 13 to quantify this condition. It states that if the aircraft turns in a tight circle with radius $minr(i)$, the origin of that tight circle is no more than $minr(i)$ away from the point $disc(i)$.

Our model allows for a huge amount of nondeterminism, both in the discrete dynamics of the controller (e.g. which aircraft are controlled ($k := *_{\mathbb{A}}$, line 2), how the aircraft steer ($\omega(k) := *_{\mathbb{R}}$, line 5), and whether to enter a collision avoidance maneuver), and in the continuous dynamics of the plant (e.g. how long to wait between control choices). This nondeterminism is a beneficial property of our collision avoidance protocol, since it allows for each aircraft to implement slightly different control algorithms without violating the proof of safety for the entire system. As a result, we have verified a class of controllers, rather than one specific implementation.

4.1.2 Theorem Statement

In order to guarantee safety, we must prove that for all pairs of distinct aircraft i, j, the distance between i and j is greater than or equal to p. We can express this condition formally as

$$\mathsf{Safe} \equiv \forall\, i, j : \mathbb{A}\ \ (i \neq j \rightarrow \|x(i) - x(j)\| \ge p).$$

We must show that Safe holds during all executions of BigDisc. The QdL formula expressing this property is [BigDisc]Safe. We must also ensure that the aircraft begin in a controllable state. This means each aircraft must have a buffer disc (InitA), which is empty (InitB), and within which it may flyably maneuver (InitC). For aircraft i that are not in an avoidance maneuver, we ensure that i's disc is at the same point as i. Since $ca(i) = 1$ for aircraft in a maneuver and $ca(i) = 0$ otherwise, we may write this property as

$$\texttt{InitA} \equiv \forall i : \mathbb{A}\ (1 - ca(i)) \cdot disc(i) = (1 - ca(i)) \cdot x(i).$$

By again using this arithmetic coding style rather than an if-else statement, we eliminate a significant branching factor in the resulting proof. We then show that the discs are empty by ensuring sufficient separation between the discs as defined in QHP 1 line 14.

$$\texttt{InitB} \equiv \forall i, j : \mathbb{A}\ \ (i \neq j \rightarrow \mathsf{Sep}(i, j))$$

Finally, we ensure that aircraft in collision avoidance maneuvers are able to flyably remain within their discs. We do this by proving that the following formula is an invariant of our system. It states that if i tightly circles in its current circling direction, the center of this tight circle will be within distance $minr(i)$ of $disc(i)$.

$$\texttt{InitC} \equiv \forall i : \mathbb{A}\ \ \|disc(i) - (x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp})\| \leq minr(i)$$

Note that these initial conditions all hold trivially if the aircraft begin far enough apart that none is in a collision avoidance maneuver.

Theorem 1 (Safety of BigDisc). *If the aircraft are initially in a controllable state, then no two aircraft will come closer than distance p while all aircraft follow* Control*; therefore safety of the* BigDisc *controller is expressed by the provable* QdL *formula:*

$$(\texttt{InitA} \wedge \texttt{InitB} \wedge \texttt{InitC}) \rightarrow [\texttt{BigDisc}]\mathsf{Safe}$$

We proved Theorem 1 for all parameter values by showing that InitA, B and C are maintained as invariants. This proof was generated using KeYmaeraD from a 330 line user-generated tactic script. The tactic file and the KeYmaeraD theorem prover are available online [22]. A discussion of the critical techniques needed to complete this proof is presented in [23, Appendix A.1].

4.2 Small Discs

One drawback of the Big Disc policy is that it may trigger collision avoidance maneuvers that are not strictly necessary; the buffer zones are larger than the required circling space of the aircraft. Our second policy, *Small Discs*, aims to decrease the size of the disc. The only way to do this is to abandon the assumption that the disc must be centered on the aircraft during free flight. Instead, the buffer zone, a disc of radius $minr(i)$, is centered at a point with distance $minr(i)$ away from $x(i)$, in a direction perpendicular to i's motion either to the left or the right. Thus the aircraft is always on the edge of its disc, and during collision avoidance, the aircraft follows the circumference of its disc. As with BigDisc, an aircraft may flip its circling direction during free flight. This now makes the disc jump to the other side of the aircraft, and before an aircraft can flip its circling direction, it must check that it may do so safely.

Fig. 2 illustrates a situation where flipping the disc to the opposite side prevents an unnecessary collision avoidance maneuver. Each aircraft has an *active disc* (solid discs in Fig. 2) that it will use for collision avoidance if needed. We also illustrate the *inactive disc* (dotted discs) as an alternative choice for the disc if the aircraft decides to flip its circling direction. In Fig. 2, the active discs of aircraft i and aircraft j are on a collision course. If nothing is done, at the latest when the edges of the discs are separated by distance p, both aircraft will enter collision avoidance by circling to the right

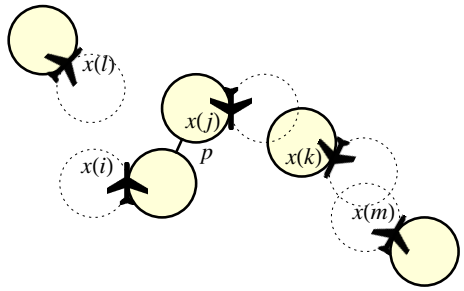

Figure 2: One possible scenario in the Small Discs policy

around the circumference of their respective discs. Notice that in this case, the aircraft may pass as close as the minimum separation distance p when aircraft i is at the top of its disc while aircraft j is at the bottom. However, since aircraft i has free space to its left, it may flip its circling disc to the opposite side and no collision avoidance maneuvers are necessary. Aircraft j is unable to make such a flip, since aircraft k's disc occupies the necessary space. Only collisions of active discs are a problem. The fact that the inactive discs of k and m overlap is immaterial, because if collision avoidance is necessary, every aircraft will follow the circumference of its *active* disc. This also illustrates that aircraft must synchronize disc flipping. If, to enable j to flip its disc, k flips its disc, but, at the same time, and unaware of this, m flips its disc, then k and m would have incompatible collision avoidance discs. Since purely discrete standard solutions exist for ensuring consistency in such discrete mode changes, our model simply uses sequentialized flipping decisions.

4.2.1 Formal Model

The Small Discs policy is presented formally as SmallDiscs in QHP 2. The overall structure is similar to that of BigDisc. One notable difference is that SmallDiscs no longer uses the state variable $disc(i)$. During free flight, the center of an aircraft's disc moves with dynamics that are not easy to express in terms of other variables; it is certainly not as easy as setting $disc(i)' = x(i)'$, as we did in BigDisc. The point $disc(i)$ moves faster or slower than $x(i)$, depending on whether the aircraft is veering away from its active disc, or towards it. As a result, $disc(i)$ has very involved continuous dynamics. Fortunately, however, the position of aircraft i's disc can be simply expressed in terms of other state variables. By using differential-algebraic equations as in [24], we equate

$$disc(i) = x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}. \tag{15}$$

In order to simplify the mathematics of the system, we directly reduce the system to ordinary differential equations, which also makes the connection to BigDisc more apparent. Thus, instead of using (15) as part of a differential-algebraic equation [24], we consider (15) as a definition and statically replace all occurrences of $disc(i)$ by the right-hand side of (15). The subsequent model should be read with this in mind. The other major change in QHP 2 is the separation condition Sep and the newly introduced separation condition FlipSep for flipping the disc.

4.2.2 Theorem Statement

Here again we want to prove that under safe initial conditions, the SmallDiscs controller is always Safe, where Safe is exactly as we defined it for BigDisc. We need to modify the initial conditions for SmallDiscs. There are two properties that we want to hold: first, the discs are separated (InitD), and second, when an aircraft is engaged in collision avoidance it is flying along the circum-

Quantified Hybrid Program 2 Small Discs

$$\texttt{SmallDiscs} \equiv (\texttt{Control} \cup \texttt{Plant})^* \tag{16}$$
$$\texttt{Control} \equiv k := *_{\mathbb{A}};\ (\texttt{CA} \cup \texttt{NotCA}) \tag{17}$$
$$\texttt{CA} \equiv ?(ca(k) = 1);\ (\texttt{Exit} \cup \texttt{Skip}) \tag{18}$$
$$\texttt{NotCA} \equiv ?(ca(k) = 0);\ (\texttt{Steer} \cup \texttt{Flip} \cup \texttt{Enter}) \tag{19}$$
$$\texttt{Skip} \equiv ?true \tag{20}$$
$$\texttt{Steer} \equiv \omega(k) := *_{\mathbb{R}};\ ?(-\Omega(k) \le \omega(k) \le \Omega(k)) \tag{21}$$
$$\texttt{Exit} \equiv ca(k) := 0 \tag{22}$$
$$\texttt{Enter} \equiv (\omega(k) := side(k) \cdot \Omega(k));\ ca(k) := 1 \tag{23}$$
$$\texttt{Flip} \equiv ?(\forall j : \mathbb{A}\ (j \neq k \rightarrow \mathsf{FlipSep}(j,k)));\ side(k) := -side(k) \tag{24, 25}$$
$$\mathsf{FlipSep}(i,j) \equiv \|(x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}) - (x(j) - minr(j) \cdot side(j) \cdot d(j)^{\perp})\| \ge minr(i) + minr(j) + p \tag{26–28}$$
$$\texttt{Plant} \equiv \forall i : \mathbb{A}\ \Big(x(i)' = v(i) \cdot d(i),\ d(i)' = \omega(i) d(i)^{\perp} \ \&\ \forall j : \mathbb{A}\ ((j \neq i \wedge (ca(i) = 0 \vee ca(j) = 0)) \rightarrow \mathsf{Sep}(i,j))\Big) \tag{29–31}$$
$$\mathsf{Sep}(i,j) \equiv \|(x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}) - (x(j) + minr(j) \cdot side(j) \cdot d(j)^{\perp})\| \ge minr(i) + minr(j) + p \tag{32–34}$$

ference of its active disc (`InitE`). `InitD` is similar to `InitB` for `BigDisc`, but it uses our new definition of Sep for `SmallDiscs`:

$$\texttt{InitD} \equiv \forall i, j : \mathbb{A}\ \ (i \neq j \rightarrow \mathsf{Sep}(i,j)).$$

If i is in a collision avoidance maneuver, then i is turning at maximal angular velocity. This implication is expressed with arithmetic coding by multiplying both sides with the indicator $ca(i)$:

$$\texttt{InitE} \equiv \forall i : \mathbb{A}\ \ (\omega(i) \cdot ca(i) = \Omega(i) \cdot side(i) \cdot ca(i)).$$

THEOREM 2 (SAFETY OF `SmallDiscs`). *If the aircraft are initially in a controllable state (i.e. where* `InitD` *and* `InitE` *hold), then no aircraft will come closer than distance p to any other aircraft so long as each aircraft follows* `Control`*; therefore safety of the* `SmallDiscs` *controller is expressed by the provable* dL *formula:*

$$(\texttt{InitD} \wedge \texttt{InitE}) \rightarrow [\texttt{SmallDiscs}]\mathsf{Safe}$$

We proved Theorem 2 in KeYmaeraD by showing `InitD` and `InitE` are invariant. The accompanying tactic script is 309 lines in length and is available online [22].

5. REFERENCES

[1] Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management. IEEE T. Automat. Contr. **43**(4) (1998) 509–521

[2] Hu, J., Prandini, M., Sastry, S.: Probabilistic safety analysis in three-dimensional aircraft flight. In: CDC. (2003)

[3] Dowek, G., Muñoz, C., Carreño, V.A.: Provably safe coordinated strategy for distributed conflict resolution. In: AIAA-2005-6047. (2005)

[4] Umeno, S., Lynch, N.A.: Proving safety properties of an aircraft landing protocol using I/O automata and the PVS theorem prover. In Misra, J., Nipkow, T., Sekerinski, E., eds.: FM. Volume 4085 of LNCS., Springer (2006) 64–80

[5] Galdino, A.L., Muñoz, C., Ayala-Rincón, M.: Formal verification of an optimal air traffic conflict resolution and recovery algorithm. In Leivant, D., de Queiroz, R., eds.: WoLLIC. Volume 4576 of LNCS., Springer (2007) 177–188

[6] Hwang, I., Kim, J., Tomlin, C.: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly **15**(1) (2007) 1–34

[7] Umeno, S., Lynch, N.A.: Safety verification of an aircraft landing protocol: A refinement approach. In Bemporad, A., Bicchi, A., Buttazzo, G., eds.: HSCC. Volume 4416 of LNCS., Springer (2007) 557–572

[8] Johnson, T., Mitra, S.: Parameterized verification of distributed cyber-physical systems:an aircraft landing protocol case study. In: ACM/IEEE ICCPS. (2012)

[9] Duperret, J.M., Hafner, M.R., Del Vecchio, D.: Formal design of a provably safe roundabout system. (In: IEEE/RSJ IROS) 2006–2011

[10] Platzer, A., Clarke, E.M.: Formal verification of curved flight collision avoidance maneuvers: A case study. In: FM. Volume 5850 of LNCS., Springer (2009) 547–562

[11] Pallottino, L., Scordio, V., Frazzoli, E., Bicchi, A.: Decentralized cooperative policy for conflict resolution in multi-vehicle systems. IEEE Trans. on Robotics **23**(6) (2007)

[12] Košecká, J., Tomlin, C., Pappas, G., Sastry, S.: 2-1/2D conflict resolution maneuvers for ATMS. In: CDC. Volume 3., Tampa, FL, USA (1998) 2650–2655

[13] Bicchi, A., Pallottino, L.: On optimal cooperative conflict resolution for air traffic management systems. IEEE Trans. ITS **1**(4) (2000) 221–231

[14] Hu, J., Prandini, M., Sastry, S.: Optimal coordinated motions of multiple agents moving on a plane. SIAM Journal on Control and Optimization **42** (2003) 637–668

[15] Massink, M., Francesco, N.D.: Modelling free flight with collision avoidance. In Andler, S.F., Offutt, J., eds.: ICECCS, Los Alamitos, IEEE (2001) 270–280

[16] Platzer, A.: Quantified differential dynamic logic for distributed hybrid systems. In Dawar, A., Veith, H., eds.: CSL. Volume 6247 of LNCS., Springer (2010) 469–483

[17] Platzer, A.: A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. Logical Methods in Computer Science **8**(4) (2012) 1–44

[18] Loos, S.M., Platzer, A., Nistor, L.: Adaptive cruise control: Hybrid, distributed, and now formally verified. In Butler, M., Schulte, W., eds.: FM. LNCS, Springer (2011) 42–56

[19] Renshaw, D.W., Loos, S.M., Platzer, A.: Distributed theorem proving for distributed hybrid systems. In Qin, S., Qiu, Z., eds.: ICFEM. Volume 6991 of LNCS., Springer (2011) 356–371

[20] Platzer, A.: Quantified differential invariants. In Frazzoli, E., Grosu, R., eds.: HSCC, ACM (2011) 63–72

[21] Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Am J Math **79**(3) (1957) pp. 497–516

[22] Electronic proofs: `www.ls.cs.cmu.edu/discworld`.

[23] David Renshaw, Sarah M. Loos, A.P.: Formal verification of distributed aircraft controllers. Technical Report CMU-CS-12-132, Carnegie Mellon (2012)

[24] Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. **20**(1) (2010) 309–352