Mechanized Safety Proofs for Disc-Constrained Aircraft

David Renshaw Sarah M. Loos André Platzer

AUGUST 2012
CMU-CS-12-132

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

As airspace becomes ever more crowded, air traffic management must reduce both space and time between aircraft to increase throughput, and on-board collision avoidance systems become ever more important. These systems and the policies that they implement must be extremely reliable. In this paper we consider implementations of distributed collision avoidance policies designed to work in environments with arbitrarily many aircraft. We formally verify that the policies are safe, even when new planes approach an in-progress avoidance maneuver. We show that the policies are flyable and that in every circumstance which may arise from a set of controllable initial conditions, the aircraft will never get too close to one another. Our approach relies on theorem proving in Quantified Differential Dynamic Logic (Qd$\mathcal{L}$) and the KeYmaeraD theorem prover for distributed hybrid systems. It represents an important step in formally verified, flyable, and distributed air traffic control.

This research was sponsored by the National Science Foundation under grant numbers CNS-1054246, CNS-1035800, CNS-0931985, CNS-0926181. Sarah M. Loos was supported by a DOE Computational Science Graduate Fellowship.

Keywords: Aircraft maneuvers, distributed hybrid systems, differential dynamic logic, theorem proving, formal verification.

1 Introduction

Safety-critical systems such as aircraft controllers must be designed with a high assurance of correctness. When the costs of failure are extremely high, system designers must strive to provide extremely reliable guarantees that their systems work as intended. Formal verification is an important tool at their disposal. The goal of *mechanized theorem proving*, our favored style of formal verification, is not merely to exhibit a proof of system correctness, but also to check by computer the validity of every step in that proof. Doing so drastically reduces the latent possibility of human errors. To trust the statement of a formally proved theorem, it suffices merely to trust the implementation of the proof checker, rather than the typically intricate details of the proof itself.

Chief among the many difficult tasks inherent in a mechanized theorem proving effort is finding the right formalisms to work within. The challenge is to reduce into concise notation and simple automatable steps the sophisticated reasoning principles that mathematicians usually take for granted. Simplifying too much can lead to incomplete formalisms that lack the power to prove interesting theorems. Simplifying too little or choosing the wrong primitives can lead to an intractable mess. The appropriate choice of formalisms depends, of course, on the problem domain of interest. In this paper, we focus on collision-avoidance policies for aircraft systems, a problem domain that poses quite a few difficult challenges. The systems that we analyze are composed of multiple independent computational agents that interact with the physical world—making them so-called *distributed hybrid systems*. The continuous flight dynamics, the discrete computation for flight control decisions, and the distributed communication aspects all contribute to the difficulty of verification in this domain. Our formalism has three components. First we have a language for specifying distributed hybrid systems and their behavior, called *Quantified Hybrid Programs*. Second, we have a logic for describing properties of systems, called *Quantified Differential Dynamic Logic* or $\mathsf{Qd}\mathcal{L}$ [1, 2]. Third, we have a proof calculus, i.e. a system of syntactic manipulations that allows us to reduce $\mathsf{Qd}\mathcal{L}$ formulas into simpler subcomponents in order to prove their validity.

Once a formalism is in place, the challenge is using it to build proofs of interesting theorems. Here is where software tool support becomes crucial. In practice, we often do not just want computerized help in *checking* our proofs, but we also want help in *constructing* proofs. In some cases it is possible to achieve full automation, but in advanced problem domains theorem proving is undecidable, so tools must be interactive. KeYmaeraD is an interactive theorem prover that both mechanizes the $\mathsf{Qd}\mathcal{L}$ formalism and facilitates proof search.

The devil is in the details. Getting all of this machinery to work together and produce interesting results is a nontrivial task. The purpose of this paper is exhibit two case studies that show a practical application of mechanized theorem proving in the domain of aircraft control and avoidance maneuvers. We specify and verify two interesting policies. These policies are designed with a simple separation principle in mind: associated with each aircraft is a disc, within which the aircraft is required always to remain. Then the problem reduces to i) maintaining sufficient separation between pairs of discs, and ii) proving that the actual controller for the flight dynamics will always stay inside its disc.

The primary contribution of this paper is this pair of formally verified policies for aircraft. The systems are distributed, allowing participation of arbitrarily many aircraft, and rippling effects are taken care of. The trajectories of the aircraft are flyable with no sharp corners or instantaneous changes of speed. Additionally, they have some nice space efficiency properties. We argue that Qd$\mathcal{L}$ is a useful formalism and that KeYmaeraD is an effective verification tool. To the best of our knowledge, we provide the first formal verification results of safe separation of controllers for flyable aircraft dynamics with arbitrarily many aircraft. It also appears to be the case that KeYmaeraD is the only verification tool with which this can be proved.

2 Related Work

Verification of air traffic control is particularly challenging because it lies in the intersection of many fields which are already tough verification problems when examined independently. It is a distributed system, with an arbitrarily 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 by flyable (i.e. not cause the aircraft to enter a stall, bank too sharply, or require it to turn on sharp corners). However, because flight is so safety critical, it is important that we get it right.

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. [3] 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 probabilistically verify it with Monte Carlo methods. They give an informal proof of safety that broadly follows similar high-level ideas as our proofs here. Our proofs, however, are formal, and our methods directly operate on the code describing the systems. We therefore provide a much higher degree of assurance and a clearer avenue to extending the systems while retaining that assurance. Also, unlike [3], our proofs take the flight dynamics into account.

Similarly, the work by Umeno and Lynch [4, 5] is complementary to ours. They consider real-time properties of airport protocols using Timed I/O Automata. We are interested in proving local properties of the actual hybrid system flight dynamics.

Verification methods for systems with an arbitrary number of agents behaving under distributed control fall primarily into one of two categories: semi-automated theorem proving and parameterized verification.

Johnson and Mitra [6] use parameterized verification to guarantee that a distributed air traffic landing protocol (SATS) is collision free. Using backward reachability, they prove that six aircraft is the maximum number that can be engaged in a SATS maneuver simultaneously. The SATS landing protocol divides the airspace into 11 regions and approximates aircraft movement by clocks within each region. We consider the complementary problem of free flight instead of airport landing traffic.

Other provably safe systems with a specific (usually small) number of agents are presented in [7, 8, 4, 5]. Duperret et al. [7] verify a roundabout with three vehicles. Each vehicle is constrained to a pre-defined path, so dynamics are flattened to one dimension.

Tomlin et al. [8] 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 straight-line approximations.

Flyability has been identified as one of the major challenges in Košecká et al. [9], where planning based on superposition of potential fields has been 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 [10] with numerical simulations.

Numerical simulation algorithms approximating discrete-time Markov Chain approximations of aircraft behavior have been proposed by Hu et al. [11]. 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. [12] have presented a straight-line aircraft conflict avoidance maneuver that involves optimization over complicated trigonometric computations, and validate it using random numerical simulation and informal arguments.

The work of Dowek et al. [13] and Galdino et al. [14] shares many goals with ours. They consider straight-line maneuvers and formalize geometrical proofs in PVS.

Our approach has a very different focus than other complementary work:

- Our maneuver directly involves curved flight unlike [8, 11, 13, 14, 12, 4, 5]. This makes our maneuver more realistic but much more difficult to analyze.
- Unlike [9, 11, 12], we do not give results for a finite (sometimes small) number of initial flight positions (simulation). Instead, we verify uncountably many initial states and give unbounded-time horizon verification results.
- Unlike [8–11, 15, 12], we use symbolic instead of numerical computation so that numerical and floating point errors cannot cause soundness problems.
- Unlike [10, 16, 11, 13, 14, 12, 4, 5], we analyze hybrid system dynamics directly.
- Unlike [9, 8, 10–12, 16, 3] we produce formal, deductive proofs.
- In [13, 14, 12, 4, 5], it remains to be proven that the hybrid dynamics and flight equations follow the geometrical thoughts. In contrast, our approach directly works for the hybrid flight dynamics. We illustrate verification results graphically to help understand them, but the figures do not prove anything.
- Unlike [10, 15], we do not guarantee optimality of the resulting maneuver.
- Unlike [7–14, 17], we verify the case of arbitrarily many aircraft.

3 Preliminaries: Quantified Differential Dynamic Logic

Quantified Hybrid Programs. QHPs [1, 2] are defined by the following grammar (α, β are QHPs, θ terms, i a variable of sort C, f is a function symbol, s is 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. The QHP $\forall i : C\ a(i) := a(i) + 1$, for example, expresses that all cars i of sort C simultaneously increase their acceleration. 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, $f(\boldsymbol{s})'$ denotes the derivative of the interpretation of the term $f(\boldsymbol{s})$ over time during continuous evolution, not the derivative of $f(\boldsymbol{s})$ by its argument $\boldsymbol{s}$. We assume that f does not occur in $\boldsymbol{s}$. In most quantified assignments/differential equations $\boldsymbol{s}$ is just i. For instance, the following QHP expresses that all cars i of sort C drive by $\forall i : C\ x(i)'' = a(i)$ such that their position $x(i)$ changes continuously according to their respective acceleration $a(i)$.

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 Qd$\mathcal{L}$ [1, 2] 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):

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

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. Qd$\mathcal{L}$ 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 Qd$\mathcal{L}$ proof calculus [1, 2] 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 Qd$\mathcal{L}$ formulas. Loos et al [18] use the proof calculus to verify lane-following behavior for arbitrarily many autonomous cars on a highway. KeYmearaD is a theorem prover that facilitates use of the Qd$\mathcal{L}$ proof calculus and has been used successfully to verify many systems [19][20]. KeYmaeraD further implements quantified differential invariants [21]. To construct a proof in KeYmaeraD is to write a *tactic script* that, when run, applies proof rules to a sequent, reducing it to problems in first-order real arithmetic, which are then sent to a backend decision procedure (Mathematica).

4 Case Study Systems

We model airspace as $\mathbb{R}^2$ and aircraft as particles moving in this space. Each aircraft i has a constant scalar speed $v(i)$ and may steer itself by adjusting its angular velocity $\omega(i)$. There is an upper bound $\Omega(i)$ on the magnitude of the angular velocity, implying a minimum turning radius $minr(i) = v(i)/\Omega(i)$. This model is known as the *Dubins vehicle* [22]. We make no assumptions about the number of aircraft present in airspace. Our goal is to design control policies that prevent aircraft from ever getting closer to each other than a (symbolic) distance p, representing the *protected zone*. We consider a policy to be *safe* if it guarantees this property. Below, we present two policies and prove that they are safe in this sense. The main idea in both policies is to maintain around each aircraft a disc-shaped buffer zone that is large enough so that when two aircraft approach each other their buffer zones contain sufficient maneuvering room to avoid a collision. The policies operate on two levels of abstraction—though each policy is a single self-contained formal artifact. At the higher abstraction level the policies deal with buffer-zone discs that float around and freeze when they get too close to each other. When an aircraft's disc is frozen, we say that the aircraft is engaged in a *collision avoidance maneuver*. At the lower level the policies specify how aircraft move within these discs, ensuring that they may always continue to stay within them in the future and that they can follow flyable trajectories without physically impossible instant turns. Even if the discs cannot move, because they come too close, the aircraft can still fly on flyable trajectories inside the disc without colliding or stalling. The sizes and relative positions of the discs differ between the two policies, representing a set of trade-offs in which both policies have strengths and weaknesses.

4.1 Big Disc

The first policy is *Big Disc*, which we now describe. During normal free flight of an aircraft i, the buffer zone for i is a disc of radius $2minr(i)$ centered at $x(i)$. Such a disc allows just enough room for i to begin to circle at radius $minr(i)$ when i enters a collision avoidance maneuver. However, the disc is big in the sense that it allows a considerable amount of freedom once i has gone halfway around this initial circle. The beginning of one possible trajectory of a collision avoidance maneuver is illustrated in Figure 1. The indexed state variable $d(i)$ is a unit vector in the direction of i's motion. The indexed state 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 $\Omega(i)$; all that we require is that the aircraft has chosen a circling direction, and that it is able to remain within the disc by circling in that direction. We will see a more formal version of this property below.

Formal Model The Big Disc policy is presented formally in Quantified Hybrid Program 1. Note that we encode boolean data in real state variables. The indexed state 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. The indexed state variable $side(i)$ indicates the direction that i will circle when i enters an avoidance maneuver. If $side(i) = 1$ then i will circle counter-clockwise. If $side(i) = -1$ then i will circle clockwise. The program also uses some new notation. If y is a vector in $\mathbb{R}^2$, then $y^\perp$ is defined as the vector obtained by rotating y ninety degrees counter-clockwise (i.e. $(y_1, y_2)^\perp = (-y_2, y_1)$), and $||y||$ is the standard Euclidean norm.

The quantified hybrid program `BigDisc` is a loop, given by the * construct, where each iteration is either a control action as represented by `Control` or an evolution of physics as represented by `Plant`. The program makes use of nondeterminism in order to allow individual aircraft to make independent choices. Thus, in the `Control` branch, the program nondeterministically selects an

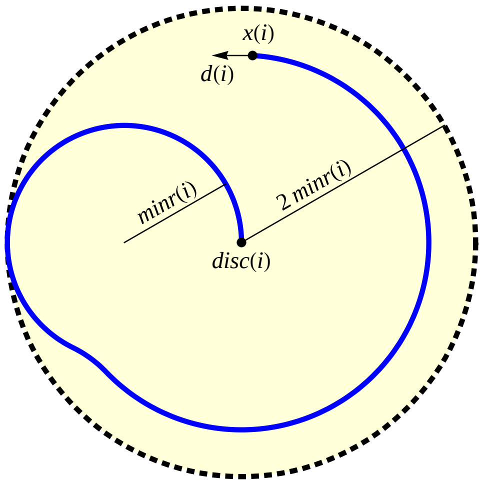

Fig. 1. A possible trajectory during collision avoidance in the Big Disc policy

aircraft k (by $k := *_{\mathbb{A}}$) 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 with its `Steer` branch, or exit the maneuver with its `Exit` branch. Note that the only allowed new angular velocities in the `Steer` branch are those between $-\Omega(k)$ and $\Omega(k)$. Note also that `Exit` is only allowed when $x(k) = disc(k)$; 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 an avoidance maneuver with the `Enter` branch. Note that k is a state variable, meaning that its value can change over the evolution of the program. In contrast, i and j are logical variables, bound by quantifiers and given meaning by substitution.

The other branch in `BigDisc`'s main loop is `Plant`. The derivative $x(i)'$ of position is of magnitude $v(i)$ in the direction $d(i)$. The derivative of $d(i)$ is of magnitude $\omega(i)$ in a direction perpendicular to $d(i)$ (line 9). The derivative of the position of the center of the buffer disc is equal to the derivative of $x(i)$ if i is in an avoidance maneuver, and zero otherwise. This case distinction is achieved by multiplication with $1 - ca(i)$ (line 10). Thus, the center of the disc is stationary in an avoidance maneuver, and it precisely tracks the aircraft otherwise. In accordance with the semantics, all other state variables are given derivative zero during the evolution. So the speed $v(i)$, for example, remains constant. The evolution constraint, i.e. the formula that appears to the right of the ampersand &, has two purposes. It ensures that aircraft that are in avoidance maneuvers may always remain within their buffer discs (line 11, and cf. the discussion of InvD below), and it forces aircraft to enter collision avoidance maneuvers if they get too close to one another (line 12).

Theorem Statement The safety condition that we would like to guarantee is as follows: 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

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

Quantified Hybrid Program 1 Big Disc Policy

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

We want to show that Safe holds after every execution of `BigDisc`. The Qd$\mathcal{L}$ formula expressing this property is [`BigDisc`]Safe. As usual, however, we need to make some assumptions about initial conditions. Thus, the safety theorem that we will prove is of the form

$$\mathsf{Init} \to [\texttt{BigDisc}]\mathsf{Safe},$$

where we have yet to define appropriate initial conditions Init. Since `BigDisc` is a loop, we will need to define a loop invariant LoopInv to prove the safety theorem; we might as well define Init to be exactly that loop invariant. We certainly need to put into LoopInv the assumptions we are making about the parameters of our system. We collect these assumptions as

$$\mathsf{InvA} \equiv$$
$$p > 0 \wedge \forall i : \mathbb{A}.\ v(i) > 0 \wedge \Omega(i) > 0 \wedge minr(i) > 0 \wedge \Omega(i) \cdot minr(i) = v(i) \wedge \|d(i)\| = 1.$$

Because *ca* and *side* are meant to encode boolean values, we also need to assume

$$\mathsf{InvB} \equiv \forall i : \mathbb{A}.\ (ca(i) = 0 \vee ca(i) = 1) \wedge (side(i) = -1 \vee side(i) = 1).$$

For aircraft *i* that are not in an avoidance maneuver, *i*'s disc is at the same point as *i*. Since $ca(i) = 0$ for aircraft not in a maneuver and $ca(i) = 1$ for aircraft in a maneuver, we may write this property as

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

We also need to ensure that aircraft in collision avoidance maneuvers remain within their discs. We do this with the following formula, which implies that if *i* immediately begins to tightly circle in its circling direction, then the center of its circle will be within distance $minr(i)$ of $disc(i)$.

$$\mathsf{InvD} \equiv \forall i : \mathbb{A}.\ \|disc(i) - (x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp})\| \le minr(i)$$

Note that this condition holds trivially for aircraft not in a collision avoidance maneuver, because for them $disc(i) = x(i)$. Finally, we need some sort of separation condition between the discs.

$$\mathsf{MainInv} \equiv \forall i, j : \mathbb{A}.\ i \neq j \rightarrow \mathsf{Separated}(i, j)$$

The full initial condition is the conjunction of these conditions.

Theorem 1 (Safety of `BigDisc`). *The following* Qd$\mathcal{L}$ *formula is valid:*

$$(\mathsf{InvA} \wedge \mathsf{InvB} \wedge \mathsf{InvC} \wedge \mathsf{InvD} \wedge \mathsf{MainInv}) \rightarrow [\mathtt{BigDisc}]\mathsf{Safe}$$

We discuss our formal, mechanized proof of Theorem 1 in Section 5.

4.2 Small Discs

One downside of the Big Disc policy is that it may trigger collision avoidance maneuvers that are not strictly necessary; the buffer zones are significantly larger than the required circling space of the aircraft. Our second policy, *Small Discs*, aims to make the buffer zone as small as possible. It does so by abandoning the (natural) 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. Thus the aircraft is always on the very edge of the disc. During a collision avoidance maneuver, the aircraft just follows the circumference of the disc. As in the Big Disc Policy, during free flight an aircraft may switch its circling direction. The difference is that now this makes the disc jump to an entirely new location; each aircraft in Small Discs has two possible discs, one for each circling direction, and before an aircraft switches its circling direction, it must first check to see whether doing so is safe.

Figure 2 illustrates a situation where making a switch prevents the need for entering a 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 alternative choice (open circles) for the disc if the aircraft decided to switch. The active discs of aircraft i and aircraft j are on a collision course. If nothing is done, both aircraft will need to enter a collision avoidance maneuver. Aircraft i, however, has free space to its left. If i flips its circling direction, its disc shifts to this free space on the left, and no collision avoidance maneuvers are necessary. Aircraft j may not make such a flip, because aircraft k's disc occupies the necessary space. Aircraft j could only flip if aircraft k flips first. Note that only collisions of active discs are a problem. The fact that the inactive alternative circles of k and m overlap is immaterial, because every aircraft promises to follow collision avoidance procedures that make it stay inside its active disc, if necessary. This also illustrates that aircraft have to synchronize on switching discs. If, to enable j to switch its disc, k switches its disc, but, at the same time, and unaware of this, m switches 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.

Formal Model The Small Discs policy is presented formally as `SmallDiscs` in Quantified Hybrid Program 2. The overall structure is similar to that of `BigDisc`. One notable difference is that `SmallDiscs` no longer uses the state variable $disc(i)$. Indeed, during free flight, the center of an

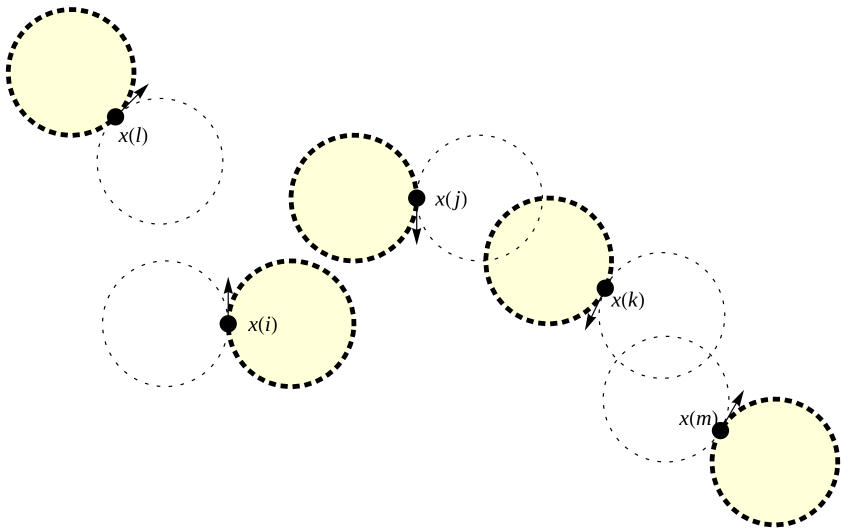

Fig. 2. One possible scenario in the Small Discs policy

aircraft's disc moves with dynamics that are not easy to express in terms of other state variables; it is certainly not as easy as setting $disc(i)' = x(i)'$, which is essentially what we did for `BigDisc`. The point $disc(i)$ moves faster or slower than $x(i)$, depending on whether the disc is on whether the aircraft is flying a left curve when the disc is on the right or on the left, and vice versa. This leads to $disc(i)$ having more involved continuous dynamics. Fortunately, however, the position, if not the velocity, of aircraft i's disc can be simply expressed in terms of other state variables. Thus, if we were to use differential-algebraic equations as in [23], we could equate

$$disc(i) = x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}. \tag{14}$$

In order to simplify the mathematics required to understand the system, we directly reduce the system to ordinary differential equations, which also makes the connection to `BigDisc` more apparent. Thus, instead of using (14) as part of a differential-algebraic equation, we consider (14) as a definition and statically replace all occurrences of $disc(i)$ by the right-hand side of (14). The subsequent model should be read with this in mind.

Now, we can write down the separation conditions of `SmallDiscs`.

$$\begin{aligned}
\text{Separated}(i, j) \equiv{} & \|(x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}) - (x(j) + minr(i) \cdot side(j) \cdot d(j)^{\perp})\| \\
& \geq minr(i) + minr(j) + p \\
\text{FlipSeparated}(i, j) \equiv{} & \|(x(i) + minr(i) \cdot side(i) \cdot d(i)^{\perp}) - (x(j) - minr(i) \cdot side(j) \cdot d(j)^{\perp})\| \\
& \geq minr(i) + minr(j) + p
\end{aligned}$$

Theorem Statement Here again we want to prove a formula of the form

$$\text{Init} \rightarrow [\texttt{SmallDiscs}]\text{Safe},$$

Quantified Hybrid Program 2 Small Discs Policy

$$
\begin{aligned}
\texttt{SmallDiscs} &\equiv (\texttt{Control} \cup \texttt{Plant})^* && (15)\\
\texttt{Control} &\equiv k := *_{\mathbb{A}};\ (\texttt{CA} \cup \texttt{NotCA}) && (16)\\
\texttt{CA} &\equiv\ ?(ca(k) = 1);\ (\texttt{Exit} \cup \texttt{Skip}) && (17)\\
\texttt{NotCA} &\equiv\ ?(ca(k) = 0);\ (\texttt{Steer} \cup \texttt{Flip} \cup \texttt{Enter}) && (18)\\
\texttt{Skip} &\equiv\ ?true && (19)\\
\texttt{Steer} &\equiv \omega(k) := *_{\mathbb{R}};\ ?(-\Omega(k) \le \omega(k) \le \Omega(k)) && (20)\\
\texttt{Exit} &\equiv ca(k) := 0 && (21)\\
\texttt{Enter} &\equiv (\omega(k) := side(k) \cdot \Omega(k));\ ca(k) := 1 && (22)\\
\texttt{Flip} &\equiv\ ?(\forall j : \mathbb{A}. j \ne k \to \ \mathsf{FlipSeparated}(j,k));\ side(k) := -side(k) && (23)\\
\texttt{Plant} &\equiv \forall i : \mathbb{A}.\{x(i)' = v(i) \cdot d(i),\ d(i)' = \omega(i) d(i)^{\perp}\ \& && (24)\\
&\qquad \forall j : \mathbb{A}.(j \ne i \wedge (ca(i) = 0 \vee ca(j) = 0)) \to \mathsf{Separated}(i,j)\} && (25)
\end{aligned}
$$

where Safe is exactly as we defined it for `BigDisc`. We need to redefine Init for `SmallDiscs`. We reuse InvA and InvB from `BigDisc`. The invariant InvC from `BigDisc` is not appropriate, because it has the aircraft at the center of the disc in free flight, instead of on the boundary. The invariant InvD is not useful, because it collapses to a triviality.

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

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

The main invariant for `SmallDiscs` has the same form as that from `BigDisc`, but it uses a different definition of Separated.

$$\mathsf{MainInv} \equiv \forall i, j : \mathbb{A}.\ i \ne j \to \mathsf{Separated}(i, j).$$

Theorem 2 (Safety of `SmallDiscs`). *The following* Qd$\mathcal{L}$ *formula is valid:*

$$(\mathsf{InvA} \wedge \mathsf{InvB} \wedge \mathsf{InvE} \wedge \mathsf{MainInv}) \to [\texttt{SmallDiscs}]\mathsf{Safe}$$

5 Case Study Proofs

We have proved Theorems 1 and 2 in KeYmaeraD . The Big Disc and Small Discs tactic scripts comprise 350 and 410 lines of Scala code, respectively. Each of them takes less than 30 seconds to run on a medium-end desktop machine. In the mechanized proofs, we do not explicitly use the abbreviations defined above. We expand everything out. In particular, all state variables become scalars, so the vectors $x(i)$, $d(i)$ and $disc(i)$ are each split into two components. All files and proofs are available online at the KeYmaeraD repository `www.github.com/keymaerad` in the `examples/aircraft` directory.

5.1 Big Disc

Loop Invariant At its outer level, `BigDisc` is a loop, so the first proof rule that we apply in order to prove Theorem 1 is the loop induction rule, which requires a loop invariant. Our loop invariant LoopInv is Init. This leaves us with two branches to prove: the induction branch, where we need to show that LoopInv $\rightarrow$ [`Control` $\cup$ `Plant`]LoopInv, and the postcondition branch, where we need to show that LoopInv $\rightarrow$ Safe.

Differential Invariants The Qd$\mathcal{L}$ proof calculus provides two ways to deal with continuous dynamics. Symbolic solutions may allow us to convert the evolution into an equivalent closed form assignment operation [1], and differential invariants may allow us to strengthen the evolution constraint [21]. The dynamics of `BigDisc` do not have closed-form solutions in a decidable class of arithmetic. Therefore we use differential invariants. Our goal in the `Plant` branch of the proof is to strengthen the evolution constraint, until it implies the loop invariant LoopInv, because then we will able to apply the following proof rule and be done:

$$\frac{\forall i{:}C.H \Rightarrow \phi}{\Gamma \Rightarrow [\forall i : C\,\{\mathcal{D}\,\&\,H\}]\phi,\, \Delta}(\nabla_{close}).$$

The rule that allows us to strengthen the evolution constraint is (∇):

$$\frac{\Gamma \Rightarrow I, \Delta \quad \forall i{:}C.H \Rightarrow \nabla_{\mathcal{D}} I \quad \Gamma \Rightarrow [\forall i : C\,\{\mathcal{D}\,\&\,H \wedge I\}]\phi, \Delta}{\Gamma \Rightarrow [\forall i : C\,\{\mathcal{D}\,\&\,H\}]\phi,\, \Delta}(\nabla).$$

We say that a formula I is a differential invariant of the differential evolution $\forall i{:}C\{\mathcal{D}\&H\}$ if the second premise of the (∇) rule is valid, i.e. if $(\forall i{:}C.H) \Rightarrow \nabla_{\mathcal{D}} I$ is valid. Thus, whether or not I is a differential invariant depends crucially on the evolution constraint H. This means that if we have some differential invariant candidates, the order in which we attempt to use them to strengthen the evolutions constraint can be important. For instance InvC is not immediately a differential invariant for `Plant`, but it is a differential invariant once the constraint has been strengthened with InvA and InvB. Therefore we use InvA, then InvB, and then InvC as differential invariants. Such *differential cuts* have been proven to be necessary in general [24]. In fact, rule ∇ is sound, because it derives from the differential invariance rule and a differential cut [21]. We do not need to try to use InvD as a differential invariant because it is already effectively included in the evolution constraint (line 11). Finally, we would like to use MainInv as a differential invariant, but it does not work. For it to be a differential invariant, all pairs of discs would have to never be allowed to move towards each other. Therefore, we use the following variant of MainInv, which, by inserting multiplications by $ca(i)$ and $ca(j)$ on both sides of the relation, only looks at pairs of aircraft in which both aircraft are in a collision avoidance maneuver:

$$\forall i, j : \mathbb{A}.\; ca(i) \cdot ca(j) \cdot \|disc(i) - disc(j)\|^2 \geq ca(i) \cdot ca(j) \cdot (2minr(i) + 2minr(j) + p)^2.$$

Because discs of aircraft in collision avoidance maneuvers are stationary, this is easily shown to be a differential invariant. It is a weaker formula than MainInv, but the evolution constraint already includes a formula (line 12) that covers the rest of the cases, i.e. when either $ca(i)$ or $ca(j)$ is 0. These cases in the evolution constraint amount to required sensor capabilities: do not miss the presence of other aircraft and the need for collision avoidance. The above formula, instead, is what we need to prove in order to ensure that the aircraft stay safe once collision avoidance happens.

Instantiation A challenge that has been previously identified [19] in this kind of proof is figuring out how to instantiate variables. In order to use a universally quantified formula that appears as an assumption, i.e. on the left in a sequent, we need to give a term with which to instantiate its bound variable. For example, all antecedents in the inductive branch contain $\mathsf{MainInv}$, and to use it we need to choose concrete terms to plug in for i and j. The situation is complicated by the fact that other rules often generate fresh names. The assignment $k := *_{\mathbb{A}}$ generates a new state variable which in our implementation might be $k\$42$, i.e., the original name k with an appended number to ensure unique names and avoid variable capture. We can always read off the base variable from such a name, but we do not know that number until the tactic script runs.

We have designed a tactic called `instantiatebyT` and show it to be extremely versatile for instantiation. It provides the ability to use the names of quantified variables as hints indicating base names of the state variables with which they are to be instantiated. More concretely, it takes as an argument a map from base variable names to lists of base variable names. When the tactic is run, it looks for universally quantified formulas on the left of the sequent. If it finds one, say $\forall i.P(i)$, binding a variable with base name i, it looks up i in its map to get a list of base variable names. All state variables in the signature whose base name is in that list then get used in instantiation of P. This tactic makes it easy to leverage the structural information encoded in variable names. To give a simple example, when we use $\mathsf{MainInv}$, it often makes sense to instantiate i with all state variables of base name i, and j with all state variables of base name j; in certain cases it makes sense to instantiate one or the other with the k-based state variables.

Triangle Inequalities In the postcondition branch of the proof, we need to prove that the loop invariant $\mathsf{LoopInv}$ implies the postcondition Safe. The first steps are straightforward enough. We instantiate appropriately to eliminate indexed variables, and we are left with a formula that can be passed to the arithmetic decision procedure. But the arithmetic is too hard. To make progress, we need to help the decision procedure along. Intuitively, the reason we expect the formula to be true has to do with the *triangle inequality*, which says that for all vectors u, v, w we have

$$||u - v|| \geq ||w - v|| - ||u - w||.$$

The discs of aircraft i and j are at least a distance $2minr(i) + 2minr(j) + p$ apart. By InvD, aircraft i is at most distance $2minr(i)$ from $disc(i)$, and aircraft j is at most distance $2minr(j)$ from $disc(j)$. Thus we may repeatedly use the triangle inequality to prove

$$\begin{aligned} ||x(i) - x(j)|| &\geq ||x(i) - disc(j)|| - 2minr(j) \\ &\geq ||disc(i) - disc(j)|| - 2minr(j) - 2minr(i) \\ &= p. \end{aligned}$$

We get the mechanized proof to succeed by using the Cut rule to break down the complicated formula into this sequence of triangle inequalities. The decision procedure is then able to then able to verify these simpler goals.

5.2 Small Discs

The proof of Theorem 2 is very similar. In fact, once we had a working tactic script for Theorem 1, finding one for Theorem 2 was mostly a matter of fitting together the pieces we already had in a slightly different way. This bodes well for the practicality of Qd$\mathcal{L}$ and KeYmaeraD and the generality of the proof approach. One difference is that we need to use the Unsubstitute proof rule (called *null* in [19]) to get the triangle inequalities to go through at the end. The expressions end up being more complicated because *disc* is now just written in terms of *x* and *d* and *side*.

References

1. 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
2. Platzer, A.: A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. Logical Methods in Computer Science (2012) Special issue for selected papers from CSL'10.
3. Pallottino, L., Scordio, V., Frazzoli, E., Bicchi, A.: Decentralized cooperative policy for conflict resolution in multi-vehicle systems. IEEE Trans. on Robotics **23** (2007) 1170–1183
4. 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
5. 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
6. Johnson, T., Mitra, S.: Parameterized verification of distributed cyber-physical systems:an aircraft landing protocol case study. In: ACM/IEEE Third International Conference on Cyber-Physical Systems, April 2012, Beijing, China. (2012)
7. Duperret, J.M., Hafner, M.R., Vecchio, D.D.: Formal design of a provably safe roundabout system. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. (2010) 2006–2011
8. Tomlin, C., Pappas, G.J., Sastry, S.: Conflict resolution for air traffic management. IEEE T. Automat. Contr. **43** (1998) 509–521
9. 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
10. Bicchi, A., Pallottino, L.: On optimal cooperative conflict resolution for air traffic management systems. IEEE Trans. ITS **1** (2000) 221–231
11. Hu, J., Prandini, M., Sastry, S.: Probabilistic safety analysis in three-dimensional aircraft flight. In: CDC. Volume 5. (2003) 5335 – 5340
12. Hwang, I., Kim, J., Tomlin, C.: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly **15** (2007) 1–34
13. Dowek, G., Muñoz, C., Carreño, V.A.: Provably safe coordinated strategy for distributed conflict resolution. In: AIAA-2005-6047. (2005)
14. 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
15. 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
16. 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
17. Platzer, A., Clarke, E.M.: Formal verification of curved flight collision avoidance maneuvers: A case study. In Cavalcanti, A., Dams, D., eds.: FM. Volume 5850 of LNCS., Springer (2009) 547–562
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)
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. Renshaw, D.W., Platzer, A.: Differntial invariants and symbolic integration for distributed hybrid systems. Technical Report CMU-CS-12-107, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2012)
21. Platzer, A.: Quantified differential invariants. In Frazzoli, E., Grosu, R., eds.: HSCC, ACM (2011)
22. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics **79** (1957) pp. 497–516

23. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. **20** (2010) 309–352
24. Platzer, A.: The structure of differential invariants and differential cut elimination. Logical Methods in Computer Science (2012) To appear.