Formal Verification of the Collision Avoidance Maneuver

Daniel H. Draper

Karlsruhe Institut für Technologie – Supervisor: Sarah Grebing

Abstract. Controlling air traffic and avoiding inflight collisions is a complex and non-trivial matter, due to the hybrid nature of combining a discrete control system with the aircraft's continuous movement. Analyzing aircraft is further complicated by constraints imposed by the aircraft's need to maintain speed and its inability to turn instantaneously. As the last instance preventing a possible collision, collision avoidance systems are critical systems and formal correctness proofs are important. To be able to verify collision avoidance maneuvers using the hybrid verification tool KeYmaera, an introduction to hybrid systems, – automata and programs is given, and the verification logic called Differential Dynamic Logic, required by KeYmaera is introduced. The Fully Flyable Roundabout Collision Avoidance Maneuver is explained and a sound verification approach for up to five aircraft is given. Finally, the verification results by KeYmaera are presented and put into context.

1 Introduction

Flying aircraft are intrinsically difficult to analyze, as their movement always follows differential equations. On top of this, instantaneous turns are impossible mid-flight, making non-trivial trigonometry always part of the analysis. The mid-air collision over Überlingen 2002 shows the importance of Collision Avoidance Systems: Crossing flightpaths of two aircrafts and contradictory instructions by the flight controller and the Traffic Collision Avoidance System (TCAS) led to the tragic death of 71. [1]

1.1 Usage of Collision Avoidance Systems Today: TCAS II

Today's aircraft implement a form of Collision Avoidance using TCAS II. It gives both Traffic Advisory (TA) (notice that another aircraft is entering the Caution Area) and Resolution Advisory (RA) (Avoidance maneuver paths calculated when the other aircraft enters the Warning Area) to the pilot when closing in on other aircraft. It performs surveillance of the surrounding area, by issuing transponder requests that other aircraft's transponders reply to, which are then decoded by the surveillance portion of TCAS software and passed on to the actual collision-avoidance algorithm. The algorithm used to establish the RA given to both aircraft is a vertical "straight-line" maneuver: One aircraft is advised to fly down and the other one to fly up. [2]

1.2 Offline Analysis

As the last resort before a mid-air crash, collision avoidance maneuvers, as the one discussed here, are critical life-saving systems, which require offline analysis to be able to guarantee their safety. In the following section, the concept of hybrid systems is introduced and modelling approaches are discussed.

2 Hybrid Systems

2.1 Definition

The term **Cyber Physical System** refers to a system, in which physical aspects, for example a moving train, a water heater or an aircraft, are being controlled by a computational software. Cyber-Physical-Systems are closely related to the concept of a **Hybrid System**, as the physical controlled aspect follows continuous/analog mechanics, while the controller has discrete/digital mechanics, which defines hybrid systems:

> "[...]*the system state evolves over time according to interacting laws of discrete and continuous dynamics*[...]" [3]

2.2 Example: ETCS

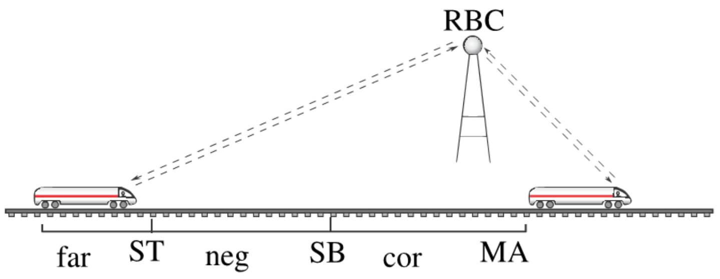


Fig. 1. A rail section as part of the ETCS. [3]

A good example for a hybrid system is the European Train Control System (ETCS). For a section of a rail a Radio Block Controller (RBC) exists, that is responsible for controlling trains that are in its section, preventing collision when they are traveling along the rail (See Fig. 1). The RBC can assign discrete values to the acceleration of the trains, thus decelerating or accelerating them, depending on the distance between the trains. In the simplified version that will be used in the next section the RBC brakes only if the chasing train passes point SB. The physical equations of linear motion concerning acceleration (a) and velocity (v), related to location (z) and time (t) (See Eq. 1) still hold true, which. when graphed, produces a discrete graph for the acceleration, while the other two variables are still continuous. (See Fig. 2).

$$\begin{aligned} v &= \frac{dz}{dt} \\ a &= \frac{d^2z}{dt^2} \end{aligned} \tag{1}$$

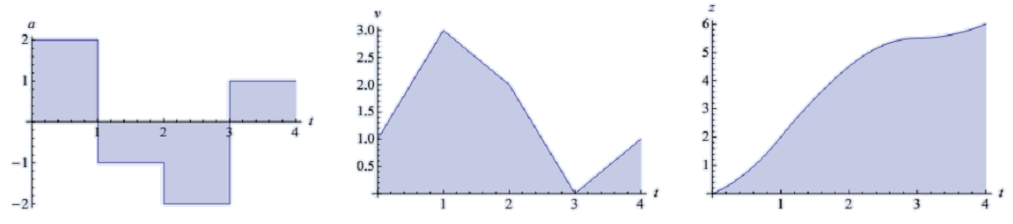


Fig. 2. Possible graphs of a hybrid system. [3]

2.3 Modeling Hybrid Systems: Hybrid Automata

As to modeling the behavior hybrid systems, the simplest approach are hybrid automata. Based on non-deterministic finite automata (NFA), they are easy to understand for humans and serve well to model the abstract behavior of hybrid systems. To understand the differences to NFAs we take a look at a simplified version of the ETCS introduced previously (See Fig. 1).

The automaton (Fig. 3) describes the behavior of the RBC controlling a train on its section. A train is therefore always either traveling freely (accelerating) or being slowed by the RBC. The biggest difference to NFAs is apparent when looking at the value assignment inside both states: They are continuous, so there is no single value assigned to the train's acceleration but rather a series of values one after each other. This means, that the single state "accelerate" can also be seen as a (possibly infinite) series of states with each one assigning a discrete value to the acceleration. The system still obviously exhibits its continuous behavior, as any change to the acceleration also brings the change of speed and location. (See Eq. 1). As the train comes closer to the train in front of it and passes the point *SB* in Fig. 1, the RBC automatically assigns a discrete negative value to the train's acceleration, decelerating it, so it slows down. In the brake state the train again follows the equations of linear motion with continuous assignment, only that the guard $v \geq 0$ makes sure the train never travels backwards (with a

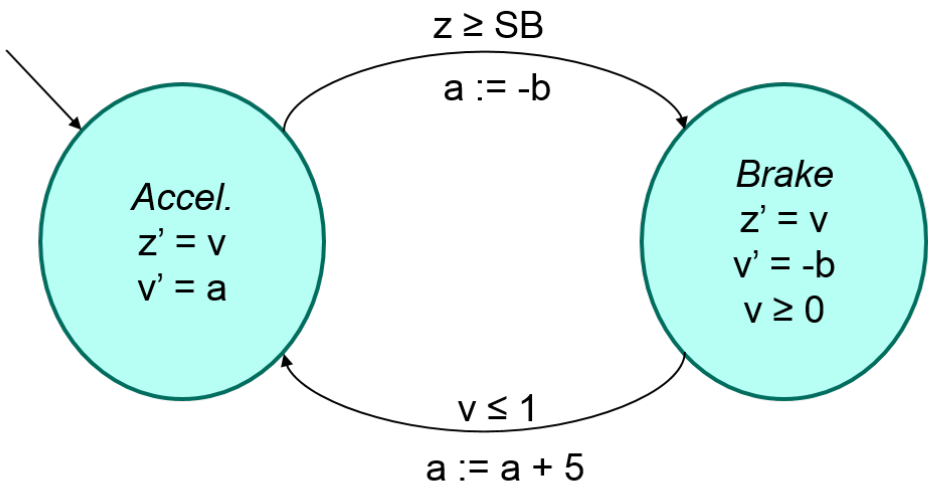


Fig. 3. Simplified Version of the ETCS behaviour. [3]

negative velocity), but rather at some point starts accelerating again according to the discrete assignment in the state change between brake and accelerate.

2.4 Verification Technique: Compositional Verification

While hybrid automata are easy to understand for humans, they are not easy for computers. As automatic verification is the main goal here, we need a better suitable model of hybrid systems. To use automatic deductive verification for proofs of safety a "divide-and-conquer style" compositional verification is used. This means, that to prove certain properties of hybrid systems we prove them for their parts, which implies their verifiability for the complete system. Unfortunately, hybrid automata as the one discussed above are not suited for composition. The problem becomes evident when trying to decompose the just discussed automaton, into two subgraphs in such a fashion, that when proving the property

$$v \leq 8 \tag{2}$$

holds true after execution for both of them, this property is also valid for the entire automaton. Dangling edges between the subgraphs, whose effects cannot be disregarded, make this decomposition non-valid. (See Fig. 4) Hybrid programs are introduced in the next section as a model that allows for decompositional and automatic verification.

2.5 Modeling Hybrid Systems: Hybrid Programs

As a model better suited for automated/compositional verification, Hybrid Programs follow the syntax depicted in table 1, whereby θ_i are terms, $x_i \in \Sigma$ are

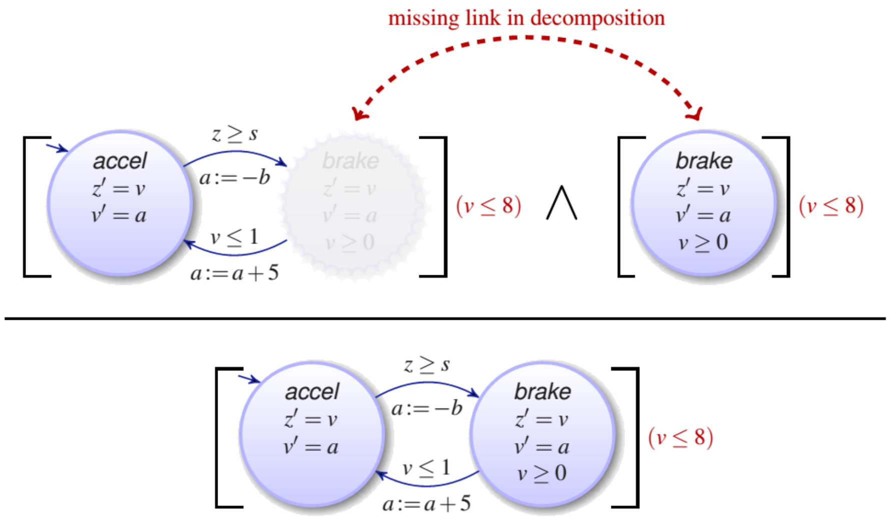


Fig. 4. Attempted Decomposition of hybrid automaton for verification purposes. [3]

state variables and χ is a formula of first-order logic. [4] As an example we take a look at the hybrid program notation of the ETCS example (See Fig. 5). As can be seen, the state variable x is first assigned the state accelerate, this corresponds to the hybrid automaton's (See Fig. 3) starting state of the system. In line 3 we can see the hybrid program syntax being an extension of first-order logic: Only if $q = accel$ **and** $z \geq SB$ are true can this case be enacted (corresponding to the state change from accel. to brake in the automaton).

notation	statement	effect
$x := \theta$	discrete assignment	assigns term θ to variable $x \in V$
$x := *$	nondet. assignment	assigns any real value to $x \in V$
$x'_1 = \theta_1 \wedge ...$	continuous evolution	diff. equations for $x_i \in V$ and terms θ_i,
$... \wedge x'_n = \theta_n \wedge \chi$		with formula χ as evolution domain
$?\chi$	state check	test formula χ at current state
$\alpha; \beta$	seq. composition	HP β starts after HP α finishes
$\alpha \cup \beta$	nondet. choice	choice between alternatives HP α or β
α^*	nondet. repetition	repeats HP α n-times for any $n \in \mathbb{N}$
do α *until* χ	evolve until	evolve HP α until χ holds

Table 1. Syntax of Hybrid Programs [4]

$q := accel;$
$\cup\ (?q = accel; z' = v, v' = a$
$\cup\ (?q = accel \wedge z \geq SB; a := b; q := brake; ?v \geq 0)$
$\cup\ (?q = brake; z' = v, v' = b \wedge v \geq 0)$
$\cup\ (?q = brake \wedge v \leq 1; a := a + 5; q := accel))^*$

Fig. 5. The simplified ETCS version as a Hybrid Program. [3]

2.6 Differential Dynamic Logic as our Verification Logic

Hybrid programs are valid models of hybrid systems that can feasibly be used for automated and compositional verification. However, the introduced syntax does not include a way to express correctness statements: That after execution of a certain hybrid program α, expressed in the syntax discussed above, a certain property ϕ, expressed in first-order logic, always holds true (*safety*) or that ϕ is reachable, meaning one concrete run exists where ϕ holds true (*liveliness*) (See Eq. 3). Differential Dynamic Logic, or d$\mathcal{L}$,

> [...]*is an extension of first-order logic over the reals with modal formulas*[...] [4].

The Hoare triple $\{\psi\}\alpha\{\phi\}$ can be expressed as $\psi \rightarrow [\alpha]\phi$.

$$\begin{aligned} Safety &: [\alpha]\phi \\ Liveliness &: \langle\alpha\rangle\phi \end{aligned} \tag{3}$$

3 KeYmaera

KeYmaera is an interactive hybrid verification tool. It was developed using the KeY discrete system verification tool and extending it to be able to verify hybrid systems. It combines deductive, real algebraic and computer algebraic prover technologies. It supports d$\mathcal{L}$ as the input language for our hybrid programs and corresponding correctness properties. KeYmaera uses a rule base, to apply rules according to a rule strategy to be able to prove properties, incorporating its algebraic solvers (e.g. Mathematica) (See Fig. 6).

As can be seen, the interactive part comes into play, when either an error (red X) or a timeout (hour glass) is the result of a rule activation: The user has to check the input or/and has to select the correct rule/help with rule usage. Using compositional verification techniques, a proof tree is created for the various subproof goals created automatically. A successful subgoal is then marked with a green check mark. A typical proof tree can be seen in the screenshot app. Fig. 11. [5]

4 Aircraft in-flight motion

To reduce its complexity for verification purposes, airplane motion is only examined in the *xy*-plane. An aircraft in our model therefore has a 2-component

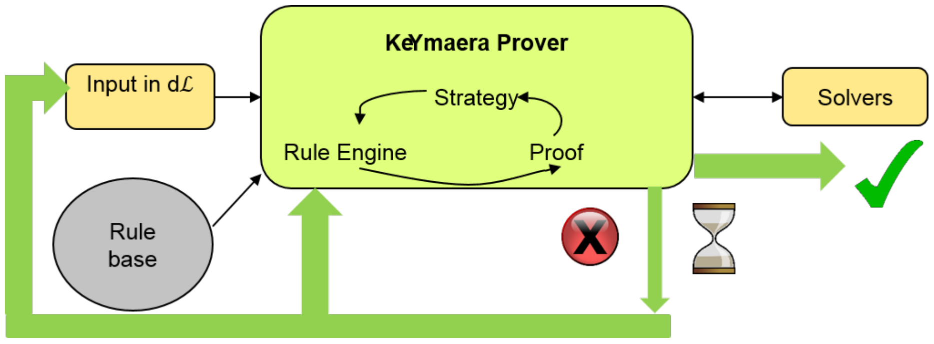


Fig. 6. KeYmaera's Architecture.

position x in the plane that changes continuously when moving. To be able to express the movement, each aircraft has a vectorial velocity and, as aircraft are not able to turn instantaneously, an angular velocity to be able to express the curves airplanes are flying (See app. Fig. 13). This means, that when the angular velocity $\omega = 0$, the airplane is flying a straight line along the direction of its velocity d , and if $\omega \neq 0$, the aircraft is flying a curve along the circle defined by d and ω. Together with the still valid physics equations for linear motion (See Eq. 1), we get the flight equation for a free moving aircraft.

4.1 Flight equation

$$\mathcal{F}(\omega) = [x' = d \; d' = \omega d^{\perp}] \tag{4}$$

When freely moving, an aircraft in our model always follows the Flight equation (See Eq. 4). This means, that according to the current angular velocity ω the aircraft moves continously according to differential equations: the change of the position is the vectorial velocity and the change of the vectorial velocity is the angular velocity $\times$ the current velocity's orthogonal complement, meaning, that the velocity d is tangential to the curve the aircraft is currently flying (See Fig. 13). [4]

4.2 Guaranteeing Safety of free-flying Aircraft: Safety Property

The goal of this verification approach is to prove that a collision cannot occur during a maneuver. Therefore we need a way to express a safe state of two freely traveling aircraft. This means, that aircraft are separated far enough as to not be entering each other's protective zones in which a collision would be unavoidable. The safe state $S(p)$ can be expressed as seen in Eq. 5: The distance between aircrafts at position x and y, vectorially calculated for both components, must be greater than or equal to the protective zones between both aircraft to guarantee non-collision. As the size of protective zones can vary between different aircrafts,

p is an arbitrary value in $\mathbb{R}$. A graphical representation of safe separation can be seen in app. Fig. 12.

$$S(p) \equiv \|x - y\|^2 \geq p^2 \equiv (x_1 - y_1)^2 + (x_2 + y_2)^2 \geq p^2, p \in \mathbb{R} \tag{5}$$

4.3 Far Separation

While $S(p)$ denotes the general requirement for safely separated aircraft, a stronger requirement for the distance is needed when multiple Collision-Avoidance maneuvers could be enacted one after each other and are to be deemed safe (e.g. during a maneuver another aircraft enters the region, so a new maneuver has to be enacted). This *far separation* is expressed as $S(p \geq f)$, with f as in Eq. 6, whereby r is the radius of the circle to be flown in the maneuver. [3]

$$f := \sqrt{2}(p + \frac{2}{3}\pi r) \tag{6}$$

5 Explaining Collision Avoidance Maneuvers

To avoid collision, so to always fulfill safety property $S(p)$ between all involved aircrafts, different approaches can be taken: Fig. 7a would be the easiest approach a straight-line vertical evasive maneuver. Maneuver b shows a horizontal approach in which both aircraft fly a large curve to avoid each other. However, c shows an error in this approach that can occur when turning one aircraft's arrival path a little. 7d then shows the improved version of the maneuver in which all aircraft fly the same circle. While all these maneuvers still are not-flyable as they incorporate instantaneous directional changes for the aircraft at the start, maneuver d is the basis for the Non-Flyable Tangential Roundabout Maneuver [NTRM], that we use as a stepping stone to prove the correctness of the Fully-Flyable Tangential Roundabout Maneuver [FTRM] introduced in the next section.

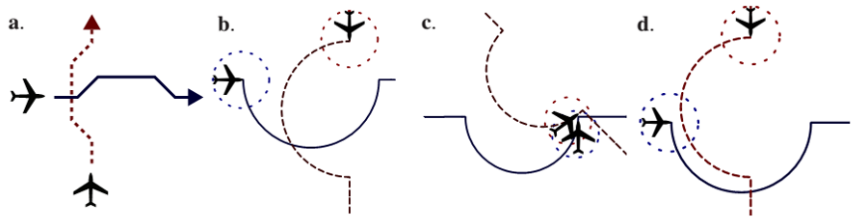


Fig. 7. Different Collision-Avoidance Maneuvers. [4]

5.1 A Collision Avoidance Maneuver: FTRM

The Fully-Flyable Tangential Roundabout Maneuver follows the finite state machine of 8a. Involved aircraft start out by flying freely in the *free* phase, following their respective flight equations and non-deterministically choosing their angular velocities continuously, until the maneuver is first invoked, due to aircrafts falling under a certain distance threshold. Then they *agree* on the maneuver's center point and the angular velocity with which to fly the circle. In the *entry* phase, the aircraft then fly a smaller circle to be able to enter the big circle without turning instantaneosly. The *circ* phase then sees the aircraft flying the circle according to the values established beforehand. In the *exit* phase, the aircraft then reach safe separation as to be able to freely fly again. An exemplary execution of FTRM for two aircrafts can be seen in 8b.

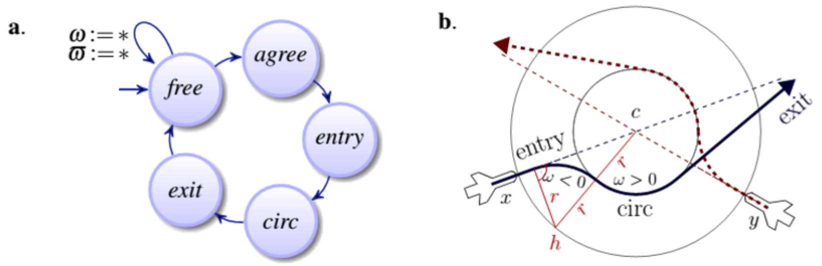


Fig. 8. The FTRM. [4]

5.2 Modularized Verification Plan of FTRM

Verification of FTRM is based on a compositional modular verification plan. To ensure correctness of FTRM we start with NTRM and prove different properties one after each other to be able to simplify the verification of FTRM's safety properties and its ability to always avoid collision. The plan follows the following scheme, where each part is then entered into KeYmaera and verified computationally.

AC1 *Tangential roundabout maneuver cycle*: Proof, that the simplified NTRM (in which the flyable *entry*-phase is replaced by a non-flyable version $entry_n$) fulfills safety property $S(p)$ at all times, even when invoking the NTRM multiple times and with multiple aircraft.

AC2 *Bounded control choices for aircraft velocities*: Proof, that linear speed remains unchanged during the whole maneuver, so that aircrafts do not stall due to low speeds.

AC3 *Flyable entry*: Proof, that NTRM's non-flyable *entry* phase can be replaced by a flyable *entry*-phase that still reaches the same configuration after its completion as the non-flyable version.
AC4 *Bounded entry duration*: Proof, that the flyable *entry* phase takes less than ∞ time to complete. Important to be able to guarantee safety when in the *entry* phase.
AC5 *Safe entry separation*: Proof, that the flyable *entry*-phase still respects the safety property $S(p)$.
AC6 *Successful negotiation*: Proof, that the FTRM's *agree* phase works for multiple aircraft at once and identification of the constraints necessary for successful negotiation.
AC7 *Safe exit separation*: Proof, that the *exit* procedure does not produce collisions and reaches the initial far separation, so FTRM can be initiated multiple times again.

Detailed Verification of NTRM As an example of how one of the proofs from the verification plan works, we take a look at the NTRM verification, **AC1** from the plan. To ensure safe-separation of aircraft during the execution of the maneuver we have to ensure it fulfills the safety property during every phase. Expressed in d$\mathcal{L}$, this would look as in Fig. 9.

$$\begin{array}{ll}
\psi & \equiv \mathcal{S}(p) \to [NTRM]\mathcal{S}(p) \\
\mathcal{S}(p) & \equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2 + (x_2-y_2)^2 \geq p^2 \\
NTRM & \equiv (free; agree; entry_n; circ)^* \\
free & \equiv (\omega := *; \varpi := *; \mathcal{F}(\omega) \land \mathcal{G}(\varphi) \land \mathcal{S}(p))^* \\
agree & \equiv \omega := *; c := * \\
entry_n & \equiv d := \omega(x-c)^{\perp}; e := \omega(y-c)^{\perp} \\
circ & \equiv \mathcal{F}(\omega) \land \mathcal{G}(\omega)
\end{array}$$

Fig. 9. AC1 expressed in d$\mathcal{L}$ [4]

The beginning expresses our proof goal: We want to show, that when entering the NTRM from a safe state (in which $S(p)$ holds), $S(p)$ still holds true after its complete execution. It then details the safety property as explained earlier, before listing the hybrid program NTRM: It consists of 4 phases that can be repeated indefinitely (see line 3). An *exit* -phase, as in FTRM, is not necessary, as no distance is required to be able to enter the NTRM again; the small entry circle in FTRM is not necessary for NTRM, as aircraft are presumed to be able to turn instantaneously. In the *free*- phase both aircraft choose their angular velocities non-deterministically, following their flight equations, while still being safely separated and non-deterministically repeating the phase. The *agree*-phase then sees both aircraft selecting the correct angular velocity for the maneuver circle as well as its center. The *entry*$_n$-phase just consists of the configuration $\mathcal{R}$

the aircraft are in on the maneuver circle: Their respective vectorial velocities are tangential to the circle, so the angular velocity of the maneuver circle times the orthogonal complement of the vector between their position and the center is their velocity. This is the same tangential configuration that is also reached by the FTRM *entry*-phase, only that in this case the aircrafts just turn instantaneously into the configuration. For a graphical representation see app. Fig. 14. At last, in the *circ*-phase, both aircraft fly the circle according to the angular velocity ω that was established in the *agree*-phase beforehand. NTRM has been verified for up to 5 aircraft by KeYmaera in [6].

6 KeYmaera's Verification Results

By following the verification plan established earlier, each phase of FTRM is deemed collision-free by KeYmaera. Fig. 10 denotes pre- and postconditions for each phase of FTRM (e.g.the phase *agree* has to fulfill: $S(f) \to [agree]S(f) \wedge C$). The introduced verification plan then still holds, when replacing any of the phases with a different one that still fulfills its pre- and postconditions.

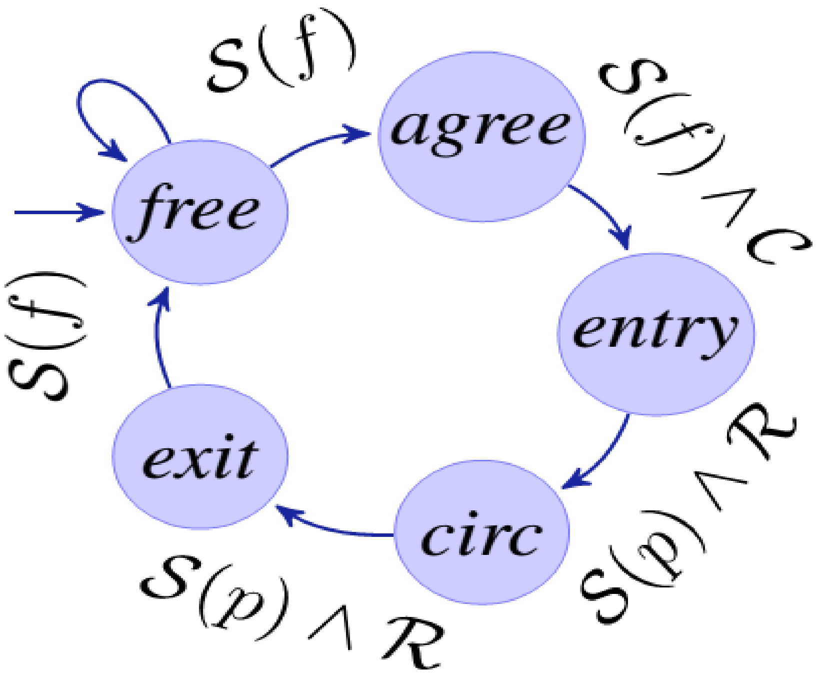


Fig. 10. Diagram showing the pre- and postconditions of the phases in FTRM. [3]

$S(f)$ denotes *far separation*, C the completion of communication between all involved aircraft and establishment of center and angular velocity of the maneuver circle, R denotes the tangential configuration of aircraft at the maneuver circle and $S(p)$ the safety property. Combining verification results from our verification plan, overall we have shown:

$$\psi \equiv \|d\| = \|e\| \wedge r > 0 \wedge S(f) \rightarrow [FTRM^{*}]S(p) \tag{7}$$

This translates to the fact, that FTRM, even when executed multiple times, always will keep all involved aircraft *safely separated*, if they are *far separated* beforehand.

> *FTRM is collision free, i.e., the collision avoidance property ψ in*[...][Eq. 7] *is valid. Even any variation of FTRM with a modified entry procedure that safely reaches tangential configuration R in some bounded time T is safe*[...] [4]

Overall, the case study on the FTRM collision-avoidance maneuver shows that even complicated hybrid systems as moving aircraft can be verified by KeYmaera.

6.1 Possible Improvements of the Proof

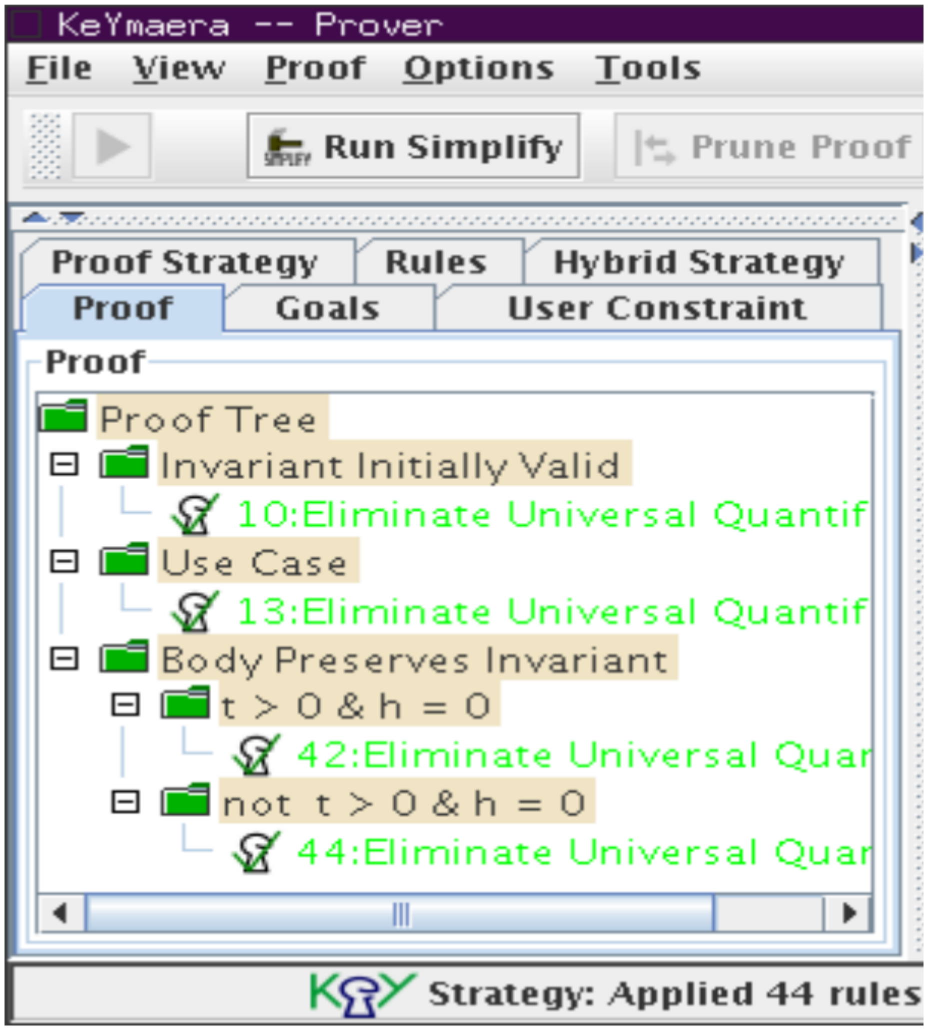
When looking at our overall collision avoidance property ψ in Eq. 7, one possible improvement is immediately obvious: Our proof only works on the premise that involved aircraft fly with the same numerical velocity ($\|d\| = \|e\|$), something that is not very realistic, but could be generalized in further work. Also, FTRM is a symmetric and synchronized maneuver, which could be generalized to a more asymmetric or asynchronous approach. As can be seen in [7], a prediction on future positions of aircraft can never be absolute, and the usage of a statistical model is necessary to incorporate wind effects into the aircraft's movement. In [8], an informal robustness study is presented, that could be carried over to a formal verification result like this one.

References

1. Schöneberg et al.: Untersuchungsbericht AX001-1-2/02. PDF available at `http://www.bfu-web.de/DE/Publikationen/Untersuchungsberichte/2002/Bericht_02_AX001-1-2.pdf?__blob=publicationFile`, Bundesstelle für Flugunfalluntersuchung, Braunschweig, Germany (2004)
2. U.S. Department of Transportation – Federal Aviation Administration: Introduction to TCAS II Version 7.1. PDF available at `http://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf/` (2011)
3. Platzer, A.: Logical Analysis of Hybrid Systems. Springer, Pittsburgh (2010)
4. André Platzer and Edmund M. Clarke: Formal verification of curved flight collision avoidance maneuvers: A case study. In Ana Cavalcanti and Dennis Dams, ed.: 16th International Symposium on Formal Methods, FM, Eindhoven, Netherlands, Proceedings. Volume 5850., Eindhoven, Netherlands, Springer (2009) 547–562

5. André Platzer and Jan-David Quesel: KeYmaera: A hybrid theorem prover for hybrid systems. In Alessandro Armando, Peter Baumgartner and Gilles Dowek, ed.: Automated Reasoning, Fourth International Joint Conference, IJCAR 2008, Sydney, Australia, Proceedings. Volume 5195., Sydney, Australia, Springer (2008) 171–181
6. Platzer, A., Clarke, E.M.: Formal verification of curved flight collision avoidance maneuvers: A case study. Technical Report CMU-CS-09-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA (2009)
7. Jianghai Hu, Maria Prandini, and Shankar Sastry: Probabilistic safety analysis in threedimensional aircraft flight. In: CDC. Volume 5. (2003) 5335–5340
8. Inseok Hwang, Jegyom Kim, and Claire Tomlin: Protocol-based conflict resolution for air traffic control. Air Traffic Control Quarterly (2007) 1–34

A More images



```
Proof Tree
⊟ Invariant Initially Valid
  └ 10:Eliminate Universal Quantif
⊟ Use Case
  └ 13:Eliminate Universal Quantif
⊟ Body Preserves Invariant
  ⊟ t > 0 & h = 0
    └ 42:Eliminate Universal Quar
  ⊟ not t > 0 & h = 0
    └ 44:Eliminate Universal Quar
```

Strategy: Applied 44 rules

Fig. 11. Screenshot of how a typical proof tree could look like in KeYmaera. [5]

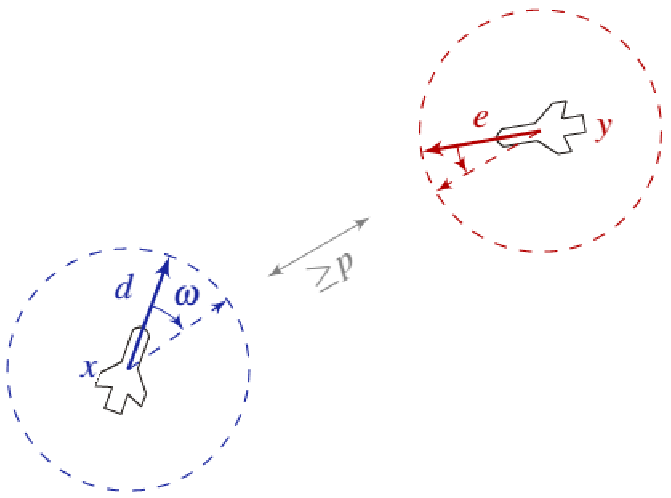


Fig. 12. Graphical representation of safe separation of two aircraft. [4]

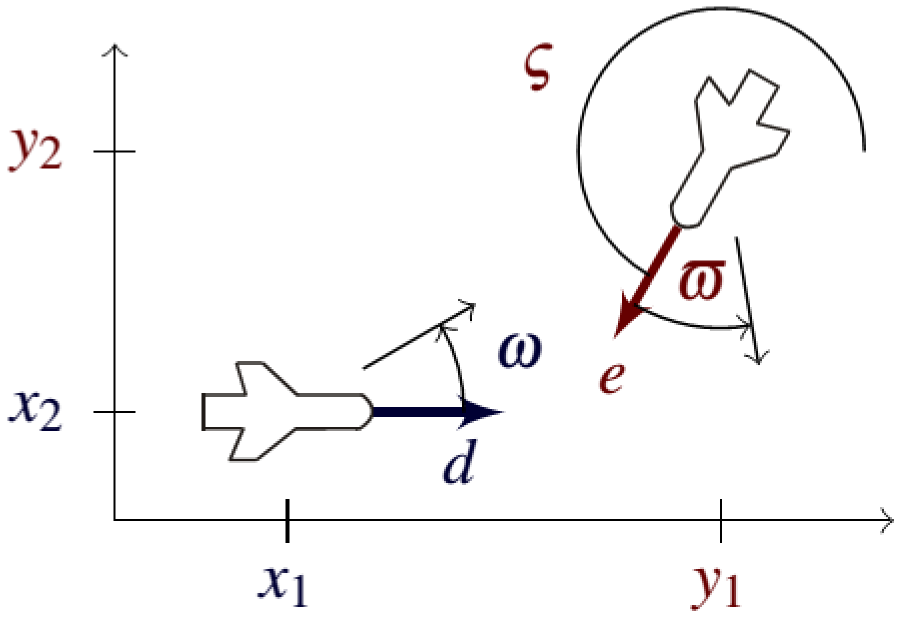


Fig. 13. Aircraft in-flight motion diagram. [4]

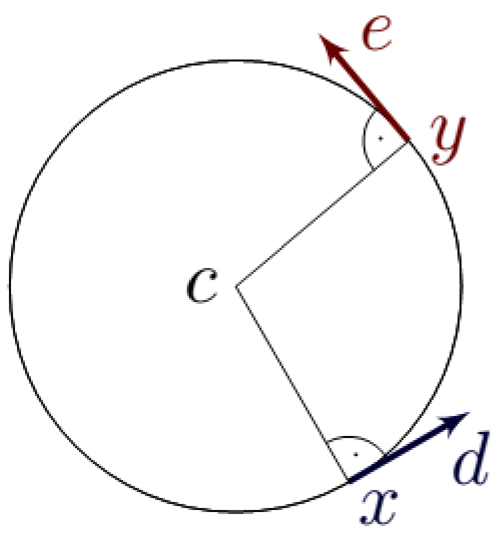


Fig. 14. The tangential configuration of aircraft during NTRM/FTRM. [6]