Computing Differential Invariants of Hybrid Systems as Fixedpoints

André Platzer[1] **Edmund M. Clarke**[2]

February 2008
CMU-CS-08-103

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]Department of Computing Science, University of Oldenburg, Germany
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA

This research was sponsored by the German Research Council (DFG) under grant SFB/TR 14 AVACS, by General Motors and the Carnegie Mellon University-General Motors Collaborative Research Laboratory under grant no. GM9100096UMA, the National Science Foundation (NSF) under grant no. CCR-0411152, by the Air Force Research Office (AFRO) under contract no. FA9550-06-1-0312, by the Office of Naval Research (ONR), the Naval Research Laboratory (NRL), and the Army Research Office (ARO). The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution or government.

Keywords: verification of hybrid systems, differential invariants, verification logic, fixedpoint engine

Abstract

We introduce a fixedpoint algorithm for verifying safety properties of hybrid systems with differential equations that have right-hand sides that are polynomials in the state variables. In order to verify non-trivial systems without solving their differential equations and without numerical errors, we use a continuous generalization of induction, for which our algorithm computes the required *differential invariants*. As a means for combining local differential invariants into global system invariants in a sound way, our fixedpoint algorithm works with a compositional verification logic for hybrid systems. To improve the verification power, we further introduce a *saturation procedure* that refines the system dynamics successively with differential invariants until safety becomes provable. By complementing our symbolic verification algorithm with a robust version of numerical falsification, we obtain a fast and sound verification procedure. We verify roundabout maneuvers in air traffic management and collision avoidance in train control.

1 Introduction

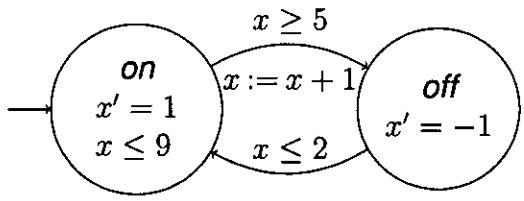
Reachability questions for systems with complex continuous dynamics are among the most challenging problems in verifying embedded systems. Hybrid systems [16, 13, 8, 1] are models for these systems with interacting discrete and continuous transitions, with the latter being governed by differential equations. For simple systems whose differential equations have solutions that are polynomials in the state variables, quantifier elimination [6] can be used for verification [13, 24, 2, 26]. Unfortunately, this symbolic approach does not scale to systems with complicated differential equations whose solutions do not support quantifier elimination (e.g., when they are transcendental functions) or cannot be given in closed form.

Numerical or approximation approaches [3, 18, 28] can deal with more general dynamics. However, numerical or approximation errors need to be handled carefully as they easily cause unsoundness. More specifically, we have shown previously that even single image computations of fairly restricted classes of hybrid systems are undecidable by numerical computation [28]. Thus, numerical approaches can be used for falsification [18, 9] but not (ultimately) for verification.

In this paper, we present an approach that combines the soundness of symbolic approaches [13, 2, 26, 27] with support for nontrivial dynamics, which is otherwise dominant in numerical approaches [3, 18, 28, 9]. During continuous transitions, the system follows a solution of its differential equation. But for nontrivial dynamics, these solutions are much more complicated than the original equations. Solutions quickly become transcendental even if the differential equations are linear. To overcome this, we handle continuous transitions based on their local dynamics, which is described by their differential equations. We use *differential induction* [25], a continuous generalization of induction that works with the differential equations themselves instead of their solutions. For the induction step, we give a condition that can be checked easily. It uses *differential invariants*, i.e., properties whose derivative holds true in the direction of the vector field of the differential equation. The derivative is a directional derivative in the direction of the (vector field generated by the) differential equation, and we generalize derivatives from functions to formulas appropriately. For this to work in practice, the most crucial steps are to find sufficiently strong local differential invariants for differential equations and compatible global invariants for the hybrid system.

To this end, we introduce a *sound* verification algorithm for hybrid systems that computes the differential invariants and system invariants in a fixedpoint loop. We follow the invariants as fixedpoints paradigm [5] using a verification logic that is generalized to hybrid systems accordingly [26, 27]. For combining multiple local differential invariants into a global invariant in a sound way, we exploit the closure properties of the underlying verification logic [26, 27] by forming appropriate logical combinations of multiple safety statements. In addition, we introduce a *differential saturation process* that refines the hybrid dynamics successively with auxiliary differential invariants until the safety statement becomes an invariant of the refined system. Finally, each fixedpoint iteration of our algorithm can be combined with numerical falsification to accelerate the overall symbolic verification in a sound way. We validate our algorithm by verifying *aircraft roundabout maneuvers* [34, 28] and train control applications [29].

In other approaches [33, 32] invariants only work for systems without inequalities [33, 32] or can only be generated for linear systems [32]. The approach of Prajna et al. [30] requires global optimization over the set of all proof attempts for the whole system at once, which is infeasi-



$q := on$; /* *initial location is on* */
$\big($ $(?q = on;\ x' = 1 \wedge x \leq 9)$
$\cup\ (?q = on \wedge x \geq 5;\ x := x + 1;\ q := off)$
$\cup\ (?q = off;\ x' = -1)$
$\cup\ (?q = off \wedge x \leq 2;\ q := on;\ ?x \leq 9)\big)^*$

Figure 1: Natural hybrid program rendition of hybrid automaton (simple water tank)

ble. Unfortunately, even for low-degree invariants, this requires solving optimization problems in several thousand dimensions for aircraft maneuvers [34, 28] and train control case studies [29].

2 Hybrid Programs and Differential Dynamic Logic

As operational models for hybrid systems, we use *hybrid programs* (HP), a program notation for hybrid automata (HA) [16]. HP can be decomposed syntactically into *fragments*: subprograms which correspond to partial executions of only a part of the full HP (programs, are easier to split structurally into parts than graphs, because handling dangling edges between graph fragments is complicated). This is important as our verification algorithm recursively decomposes an HP into fragments $\alpha_1, \ldots, \alpha_n$ (e.g., to find local invariants for each α_i) and recombines corresponding correctness statements about these fragments α_i later.

Hybrid Programs. In order to represent HA [16] textually as an HP, we represent each discrete and continuous transition as a sequence of statements, with a nondeterministic choice ($\cup$) between these transitions. For instance, the second line in Fig. 1 represents a continuous transition. It tests (denoted by $?q = on$) if the current location q is *on*, and then follows a differential equation restricted to invariant region $x \leq 9$ (i.e., the conjunction $x' = 1 \wedge x \leq 9$). The third line tests the guard $x \geq 5$ when in state *on*, resets x by a discrete assignment, and then changes location q to *off*. The * at the end indicates that the transitions of a HA repeat indefinitely. Alternatively, the resulting HP in Fig. 1 can be considered as the essential part of a program exported from Stateflow/Simulink enriched with differential equations for the continuous dynamics. Every safety property that this HP satisfies is fulfilled for *all* deterministic implementation refinements.

Formally, let V be a set of state variables of the system including auxiliary variables. As *terms* we allow polynomials over variables in V with rational constants. To make a structural decomposition of HP into fragments possible, each operation of a HP only has a single effect. There are separate classes of program statements with purely discrete effect, purely continuous effect, and statements purely for regulating their interaction. *Hybrid programs (HP)* are built with the statements depicted in Tab. 1. The effect of $x := \theta$ is an instantaneous discrete jump assigning θ to x. Instead, $x := *$ *random* randomly assigns *any* real value to x by a nondeterministic choice. During a continuous evolution $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge H$, all conjuncts need to hold. Its effect is a continuous transition controlled by the differential equation $x_1' = \theta_1, \ldots, x_n' = \theta_n$ that always satisfies the arithmetic constraint H (thus remains in the region described by H). This

directly corresponds to a continuous evolution mode of a HA. The effect of a state check or assert statement $?H$ is a *skip* (i.e., no change) if H is true in the current state and that of *abort*, otherwise. The non-deterministic choice $\alpha \cup \beta$ expresses alternatives in the behavior of the hybrid system. The sequential composition $\alpha; \beta$ expresses a behavior in which β starts after α finishes (as usual, β never starts if α continues indefinitely). In a non-deterministic repetition α^*, the HP α repeats an arbitrary number of times, possibly zero. All other discrete control structures are definable from the primitives in Tab. 1 [15].

Formulas of dℒ. Our verification algorithm repeatedly decomposes and recombines HP. As a logical framework where these operations are sound, we use a logic in which simultaneous correctness properties about multiple subsystems are expressible. The *differential dynamic logic* dℒ [26, 27] is an extension of first-order logic over the reals with modal formulas like $[\alpha]\phi$, which is true iff all states reachable by following the transitions of HP α satisfy property ϕ (*safety*).

Definition 1 (dℒ formulas) *The* formulas of dℒ *are defined by the following grammar (where θ_1 and θ_2 are terms, $\sim \in \{=, \leq, <, \geq, >\}$, ϕ, ψ are formulas, $x \in V$, and α is an HP built from the statements in Tab. 1):*

$$\text{Formulas} \ ::= \ \theta_1 \sim \theta_2 \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \rightarrow \psi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \ .$$

A Hoare-triple $\{\psi\}\alpha\{\phi\}$ can be expressed as $\psi \rightarrow [\alpha]\phi$, which is true iff all states reachable by HP α satisfy ϕ when starting from an initial state that satisfies ψ. Unlike Hoare-logics, dynamic logics are closed under logical connectives [31]. Hence, we can express simultaneous correctness statements about multiple fragments α_i using conjuncts $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$. With this, a proof for $[\alpha]\phi$ can be decomposed soundly into $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$, when $[\alpha]\phi$ and $[\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2$ are equivalent for appropriate fragments α_i of α and subproperties ϕ_i of ϕ. In turn, if $\tilde{\psi}_i$ is the result of recursively applying the verification algorithm to $[\alpha_i]\phi_i$, then these can be recombined soundly to the verification result $\tilde{\phi}_1 \wedge \tilde{\phi}_2$ for $[\alpha]\phi$. By the semantics of dℒ, this process gives a *sound* way of combining local invariants required in the respective subgoals $[\alpha_i]\phi_i$ to a global system invariant. Finally, dℒ and its proof techniques are closed under quantification, which we use to quantify over parameter choices of local invariants. For instance, $\exists p\, ([\alpha_1]\phi_1 \wedge [\alpha_2]\phi_2)$ can be used to determine if there is

Table 1: Statements and (informal) effects of hybrid programs (HP)

notation	statement	effect
$x := \theta$	discrete assignment	assigns term θ to variable $x \in V$
$x := *$	nondet. assignment	assigns any real value to $x \in V$
$x_1' = \theta_1 \wedge .. \wedge x_n' = \theta_n \wedge H$	continuous evolution	diff. equations for $x_i \in V$ and terms θ_i, with arithmetic constraint H (domain)
$?H$	state check	test formula H 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	repeating HP α n-times for any $n \in \mathbb{N}$

a common choice for parameter p that makes both subgoals $[\alpha_i]\phi_i$ true. Liveness properties are expressible using $\langle\alpha\rangle\phi \equiv \neg[\alpha]\neg\phi$, which is true iff there is a reachable state satisfying ϕ.

The *semantics* of dL and HP is a Kripke semantics and given in appendix A.

3 Inductive Verification by Combining Local Fixedpoints

For verifying safety properties of hybrid systems without having to solve their differential equations, we use a continuous form of induction. In the induction step, we use a condition on directional derivatives in the direction of the vector field generated by the differential equation. The resulting properties are invariants of the differential equation (whence called *differential invariants* [25]). The crucial step for verifying discrete systems by induction is to find sufficiently strong invariants (e.g., for loops α^*). Similarly, the crucial step for verifying dynamical systems (which correspond to a single continuous mode of a hybrid system) by induction is to find sufficiently strong invariant properties of the differential equation. Consequently, for verifying hybrid systems inductively, local invariants need to be found for each differential equation and a global system invariant needs to be found that is compatible with all local invariants.

To compute the required invariants and differential invariants, we combine the invariants as fixedpoints approach from [5] with the lifting of verification logics to hybrid systems from [26, 27]. We introduce a verification algorithm that computes invariants of a system as fixedpoints of safety constraints on subsystems. We exploit the fact that HP can be decomposed into subsystems and that dL can combine safety statements about multiple subsystems simultaneously.

A *safety statement* corresponds to a dL formula $\psi \rightarrow [\alpha]\phi$ with an HP α, a safety property ϕ about its reachable states, and an arithmetic formula ψ that symbolically characterizes the set of initial states. *Validity* of formula $\psi \rightarrow [\alpha]\phi$ (i.e., truth in all states) corresponds to ϕ being true in all states reachable by HP α from initial states that satisfy ψ. Our verification algorithm defines the function $prove(\psi \rightarrow [\alpha]\phi)$ for verifying this safety statement recursively.

3.1 Verification by Symbolic Decomposition

The cases of *prove* where dL immediately enables us to verify a property of an HP by decomposing it into a property of its parts are shown in Fig. 2. In the interest of a concise presentation, the case in line 1 introduces an auxiliary variable $\hat{x}$ to handle discrete assignments by substituting $\hat{x}$ for x. For instance, $x \geq 2 \rightarrow [x := x - 1]x \geq 0$ is shown by proving $x \geq 2 \wedge \hat{x} = x - 1 \rightarrow \hat{x} \geq 0$. The actual implementation of our algorithm uses optimizations to avoid these auxiliary variables [27]. State checks $?H$ are shown by assuming the test succeeds, i.e., H holds true (line 3), nondeterministic choices split into their alternatives (line 5), sequential compositions are proven using nested modalities (line 7), and random assignments are handled by universal quantification (line 9).

The base case in line 11, where ϕ is a formula of first-order real arithmetic, can be proven by real quantifier elimination [6] or semidefinite programming [23]. Despite the complexity of real arithmetic, this is feasible, because the formulas resulting from our algorithm do not depend on the solutions of differential equations but only their right-hand sides. Using a temporary form

of Skolemization together with Deskolemization, quantifier elimination can be lifted to eliminate quantifiers from $\mathsf{d}\mathcal{L}$ formulas. We refer to previous work [27, 26] for details.

The algorithm in Fig. 2 recursively reduces safety of HP to properties of continuous evolutions or of repetitions, which we verify in the next sections.

```
1  function prove(ψ → [x := θ]φ):
2    return prove(ψ ∧ x̂ = θ → φ_x^x̂) where x̂ is a new auxiliary variable
3  function prove(ψ → [?H]φ):
4    return prove(ψ ∧ H → φ)
5  function prove(ψ → [α ∪ β]φ):
6    return prove(ψ → [α]φ) and prove(ψ → [β]φ)  /* thus ψ → [α]φ ∧ [β]φ */
7  function prove(ψ → [α; β]φ):
8    return prove(ψ → [α][β]φ)
9  function prove(ψ → [x := random]φ):
10   return prove(ψ → ∀x φ)
11 function prove(ψ → φ) where isFirstOrder(φ):
12   return QuantifierElimination(ψ → φ)
```

Figure 2: $\mathsf{d}\mathcal{L}$-based verification by symbolic decomposition

3.2 Discrete and Differential Induction, Differential Invariants

In the sequel, we present algorithms for verifying loops by discrete induction and continuous evolutions by differential induction, which is a continuous form of induction. In either case, we prove that an invariant F holds initially (in the states characterized symbolically by ψ, thus $\psi \rightarrow F$ is valid) and finally entails the postcondition ϕ (i.e., $F \rightarrow \phi$). The cases differ in their induction step.

Definition 2 (Discrete induction) *Formula F is a* (discrete) invariant *of $\psi \rightarrow [\alpha^*]\phi$ iff the following formulas are valid:*

1. *$\psi \rightarrow F$ (induction start), and*
2. *$F \rightarrow [\alpha]F$ (induction step).*

An invariant is sufficiently strong *if $F \rightarrow \phi$ is valid.*

Definition 3 (Continuous invariants) *Let $\mathcal{D}$ be a differential equation. Formula F is a* continuous invariant *of $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ iff the following formulas are valid:*

1. *$\psi \wedge H \rightarrow F$ (induction start), and*
2. *$F \rightarrow [\mathcal{D} \wedge H]F$ (induction step).*

Again, a continuous invariant is sufficiently strong *if $F \rightarrow \phi$ is valid.*

To prove that F is a continuous invariant, it is sufficient to check a condition on the directional derivatives of all terms of the formula, which expresses that no atomic subformula of F changes its truth-value along the dynamics of the differential equation. This condition is much easier to check than a reachability property ($F \rightarrow [\mathcal{D} \wedge H]F$) of a differential equation. Applications like aircraft maneuvers need invariants with mixed equations and inequalities. Thus, we generalize directional derivatives from functions to logical formulas.

Definition 4 (Differential induction) *Let $\mathcal{D}$ be the differential equation system*

$$x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \ .$$

Formula F is a differential invariant *of $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ iff the following formulas are valid:*

1. $\psi \wedge H \rightarrow F$ *and*
2. $H \rightarrow \nabla_{\mathcal{D}} F$,

where $\nabla_{\mathcal{D}} F$ is defined as the conjunction of all directional derivatives of atomic formulas in F in the direction of the vector field of $\mathcal{D}$ (the partial derivative of b by x_i is $\frac{\partial b}{\partial x_i}$):

$$\nabla_{\mathcal{D}} F \equiv \bigwedge_{(b \sim c) \in F} \left(\left(\sum_{i=1}^{n} \frac{\partial b}{\partial x_i} \theta_i \right) \sim \left(\sum_{i=1}^{n} \frac{\partial c}{\partial x_i} \theta_i \right) \right) \quad \text{for} \sim \in \{=, \geq, >, \leq, <\}.$$

Proposition 1 (Principle of differential induction) *All differential invariants are continuous invariants (the proof is in appendix B.1).*

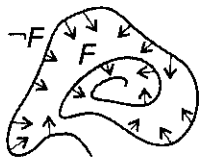


Figure 3: Differential invariant F

The region corresponding to a differential invariant F is illustrated in Fig. 3. Formula $\nabla_{\mathcal{D}} F$ is a directional derivative of F in the direction of the dynamics of $\mathcal{D}$. Intuitively, formula $\nabla_{\mathcal{D}} F$ is true if the gradient arrows are pointing inside the (possibly unbounded) region consisting of the points where F is true. In Sections 3.4–3.6, we present algorithms for finding differential invariants for differential equations, and for finding global invariants for repetitions.

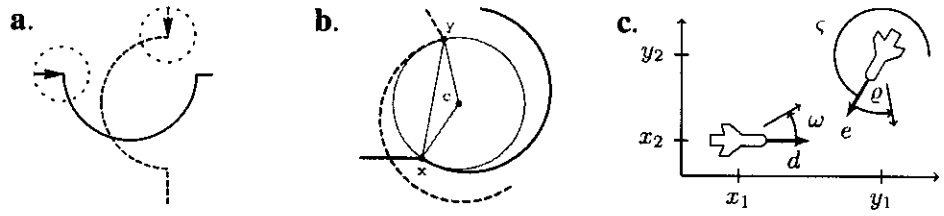


Figure 4: Roundabout maneuvers for aircraft collision avoidance.

3.3 Example: Flight Dynamics in Air Traffic Collision Avoidance

Aircraft collision avoidance maneuvers resolve conflicting flight paths, e.g., by roundabout maneuvers [34], see Fig. 4a–b. Their non-trivial dynamics makes safe separation of aircraft difficult to verify [34, 21, 10, 7, 28, 14, 17]. The parameters of two aircraft at the respective planar positions $x = (x_1, x_2) \in \mathbb{R}^2$ and $y = (y_1, y_2)$ with angular orientation ϑ and ς are illustrated in Fig. 4c (with $\vartheta = 0$). Their dynamics is determined by their linear speeds $v \in \mathbb{R}$ and $u \in \mathbb{R}$ and by their angular speeds $\omega \in \mathbb{R}$ and $\varrho \in \mathbb{R}$, see, e.g., [34] for details:

$$\begin{array}{lll} x_1' = v\cos\vartheta & x_2' = v\sin\vartheta & \vartheta' = \omega \\ y_1' = u\cos\varsigma & y_2' = u\sin\varsigma & \varsigma' = \varrho \end{array} \quad . \tag{1}$$

In safe flight configurations, aircraft are separated by distance $\geq p$:

$$(x_1 - y_1)^2 + (x_2 - y_2)^2 \geq p^2 \quad . \tag{2}$$

To handle the transcendental functions in (1), we axiomatize sin and cos by differential equations and reparametrize the system using a linear velocity vector $d = (d_1, d_2) := (v\cos\vartheta, v\sin\vartheta) \in \mathbb{R}^2$, which describes both the linear velocity $\|d\| := \sqrt{d_1^2 + d_2^2} = v$ and orientation of the aircraft in space, see Fig. 4c:

$$\left[\begin{array}{lllll} x_1' = d_1 & x_2' = d_2 & d_1' = -\omega d_2 & d_2' = \omega d_1 & t' = 1 \\ y_1' = e_1 & y_2' = e_2 & e_1' = -\varrho e_2 & e_2' = \varrho e_1 & s' = 1 \end{array}\right] \tag{$\mathcal{F}$}$$

Equations ($\mathcal{F}$) and (1) are equivalent up to reparameterization. Variables t and s are additional clocks to coordinate collision avoidance maneuvers.

We can show, e.g., that $d_1^2 + d_2^2 \geq a^2$ is a differential invariant of ($\mathcal{F}$):

$$\begin{aligned} &\nabla_{\mathcal{F}}(d_1^2 + d_2^2 \geq a^2) \equiv \nabla_{(d_1' = -\omega d_2 \wedge d_2' = \omega d_1)}(d_1^2 + d_2^2 \geq a^2) \\ \equiv\ & \frac{\partial(d_1^2 + d_2^2)}{\partial d_1}(-\omega d_2) + \frac{\partial(d_1^2 + d_2^2)}{\partial d_2}\omega d_1 \geq \frac{\partial a^2}{\partial d_1}(-\omega d_2) + \frac{\partial a^2}{\partial d_2}\omega d_1 \\ \equiv\ & 2d_1(-\omega d_2) + 2d_2\omega d_1 \geq 0 \ . \end{aligned}$$

3.4 Local Fixedpoint Computation for Differential Invariants

Fig. 5 depicts the fixedpoint algorithm for constructing differential invariants for each continuous evolution $\mathcal{D} \wedge H$ with a differential equation system $\mathcal{D}$. The algorithm in Fig. 5 (called *Differential*

```
1 function prove(ψ → [D ∧ H]φ):
2   if prove(∀cl(H → φ)) then
3     return true          /* safety property proven */
4   for each F ∈Candidates(ψ → [D ∧ H]φ, H) do
5     if prove(ψ ∧ H → F) and prove(∀cl(H → ∇D F)) then
6       H := H ∧ F         /* refine by differential invariant */
7       goto 2;            /* repeat fixedpoint loop */
8   end for
9   return "not provable using candidates"
```

Figure 5: Fixedpoint algorithm for differential invariants (*differential saturation*)

Saturation) successively refines the domain H by differential invariants until saturation, i.e., H accumulates enough information to become a strong invariant that implies postcondition ϕ (line 2). If domain H already entails ϕ, then $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ is proven (line 2). Otherwise, the algorithm considers candidates F for augmenting H (line 4). If F is a differential invariant (line 5), then H can soundly be refined to $H \wedge F$ (line 6) without affecting the states reachable by $\mathcal{D} \wedge H$ (Proposition 2 below). Then, the fixedpoint loop repeats (line 7). At each iteration of this fixedpoint loop, the previous invariant H can be used to prove the next level of refinement $H \wedge F$ (line 5). The refinement of the dynamics at line 6 is correct by the following proposition, using that the conditions in line 5 imply that F is a differential invariant and, thus, a continuous invariant by Proposition 1. (A proof is in appendix B.2.)

Proposition 2 (Differential saturation) *If F is a continuous invariant of $\psi \rightarrow [\mathcal{D} \wedge H]\phi$, then $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ and $\psi \rightarrow [\mathcal{D} \wedge H \wedge F]\phi$ are equivalent.*

This progressive differential saturation turns out to be crucial in practice. For instance, the aircraft separation property (2) cannot be proven until ($\mathcal{F}$) has been refined by invariants for d and e, because these determine x' and y'.

The function *Candidates* determines candidates for induction (line 4) depending on transitive differential dependencies, as will be explained in Section 3.5. When these are insufficient for proving $\psi \rightarrow [\mathcal{D} \wedge H]\phi$, the algorithm fails (line 9, with improvements in subsequent sections). Finally, $\forall_{cl}\phi$ denotes the *universal closure* of ϕ. It is required in lines 2 and 5, because the respective formulas need to hold in *all* states (that satisfy H), which we will improve on in Section 4.

3.5 Dependency-directed Induction Candidates

In this section, we construct likely candidates for differential induction (function *Candidates*). Later, we use the same procedure for finding global loop invariants. We construct two kinds of candidates in an order induced by differential dependencies. Our algorithm successively enriches ψ with more precise information about the symbolic prestate as obtained by the symbolic decompositions and proof steps in Fig. 2 and 5. We look for invariant symbolic state information in ψ and ϕ

by selecting subformulas that are not yet contained in H. In practice, this gives particularly good candidates for highly parametric hybrid systems.

We generate additional parametric invariants. Let $V = \{x_1, \ldots, x_n\}$ be a set of variables. We choose fresh names $a^{(l)}_{i_1,\ldots,i_n}$ for *formal parameters* of the invariant candidates and build polynomials $p_1, \ldots, p_k$ of degree d with variables V using the formal parameters as symbolic coefficients:

$$p_l := \sum_{i_1+\cdots+i_n \leq d} a^{(l)}_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} \quad (\text{for } 1 \leq l \leq k) \ .$$

We define the set of *parametric candidates* (operator $\vee$ is accordingly):

$$\mathit{ParaForm}(k, d, V) := \left\{ \bigwedge_{l=1}^{i} p_l \geq 0 \wedge \bigwedge_{l=i+1}^{k} p_l = 0 \ \mid \ 0 \leq i \leq k \right\} \ .$$

For instance, the parametric candidate $a_{0,0} + a_{1,0}d_1 + a_{0,1}x_2 = 0$ yields a differential invariant of $(\mathcal{F})$ for the choice $a_{0,0} = 0$, $a_{1,0} = 1$, $a_{0,1} = \omega$. By simple combinatorics, *ParaForm* contains $k + 1$ candidates with $k\binom{n+d}{d}$ formal parameters $a^{(l)}_{i_1,\ldots,i_n}$, which are existentially quantified. Existence of a common satisfying instantiation for these parameters can be expressed by adding $\exists a^{(l)}_{i_1,\ldots,i_n}$ to the resulting $\mathsf{d}\mathcal{L}$ formulas. For this to be feasible, the number of parameters is crucial, which we minimize by respecting (differential) dependencies.

To accelerate the differential saturation process in Section 3.4, it is crucial to explore candidates in a promising order from simple to complex, because the algorithm in Fig. 5 uses successful differential invariants to refine the dynamics, thereby simplifying subsequent proofs. For instance, (2) is only provable after the dynamics has been refined with invariants for d and e. We construct candidates in a natural order based on variable occurrence that is consistent with the differential dependencies of the differential equations. For a differential equation $\mathcal{D}$, variable x depends on variable y according to the differential equation system $\mathcal{D}$ if y occurs on the right-hand side for x' (or transitively so). The resulting set of dependencies is the transitive closure of:

$$\mathit{depend}(\mathcal{D}) := \{(x, y) \mid (x' = \theta) \in \mathcal{D} \text{ and } y \text{ occurs in } \theta\} \ .$$

For the differential equation system $(\mathcal{F})$, we determine the differential dependencies indicated as arrows (pointing to the dependent variables x) in Fig. 6.

From these dependencies we determine an order on candidates. The idea is that, as the value of x_1 depends on that of d_1, it makes sense to look for invariant expressions of d_1 first, because refinements with these help differential saturation in proving invariant expressions involving also x_1. We order variables by differential dependencies, which resembles the back substitution order in Gaussian elimination (if, in triangular form, x_1 depends on d_1 then equations for d_1 must be solved first). Now we call a set V of variables a *cluster* of the differential equation $\mathcal{D}$ iff V is closed with respect to $\mathit{depend}(\mathcal{D})$, i.e., variables of V only depend on variables in V. The resulting variable clusters for system $(\mathcal{F})$ are marked as triangular shapes in Fig. 6. Finally, we choose candidates from ψ and $\mathit{ParaForm}(k, d, V)$ starting with candidates F whose variables lie in small clusters V. Thus, the differential invariant $d_1^2 + d_2^2 \geq a^2$ of Section 3.3 within cluster $\{d_2, d_1, \omega\}$ will be discovered before invariants like $d_1 = -\omega x_2$ that involve x_2, because x_2 depends on d_2. Consequently, line 6 of Fig. 5 makes $d_1^2 + d_2^2 \geq a^2$ available for subsequent checks of invariants involving x_2.

3.6 Global Fixedpoint Computation for Loop Invariants

With the uniform setup of $\mathsf{d}\mathcal{L}$, we can adapt the algorithm in Fig. 5 easily to obtain a fixedpoint algorithm for loops ($\psi \to [\alpha^*]\phi$) in place of continuous evolutions ($\psi \to [\mathcal{D} \wedge H]\phi$). In line 5 of Fig. 5, the induction step from Def. 4 just needs to be replaced by the step for loops (Def. 2). As an optimization, invariants H of previous iterations can be exploited as refinements of the hybrid system dynamics:

Proposition 3 (Loop saturation) *If H is a discrete invariant of $\psi \to [\alpha^*]\phi$, then $H \wedge F$ is a discrete invariant iff $\psi \to F$ and $H \wedge F \to [\alpha](H \to F)$ are valid.*

The proof is in appendix B.3. The induction step from Proposition 3 can generally be proven faster, because it is a weaker property than that of Def. 2. For sake of completeness, the resulting algorithm is given in appendix D.

To adapt our approach from Section 3.5 to loops, we use discrete data-flow and control-flow dependencies of α. Dependencies can be determined immediately from the syntax of HP. There is a direct *data-flow dependency* with the value of x depending on y, if $x := \theta$ or $x' = \theta$ occurs in α with a term θ that contains y. Accordingly, there is a direct *control-flow dependency*, if, for any term θ, $x := \theta$ or $x' = \theta$ occurs in α after a $?H$ containing y.

3.7 Interplay of Local and Global Fixedpoint Loops

The local and global fixedpoint algorithms jointly verify correctness properties of HP. Their interplay needs to be coordinated with fairness. If the local fixedpoint algorithm in Fig. 5 does not converge, stronger invariants may need to be found by the global fixedpoint algorithm which result in stronger preconditions ψ for the local algorithm. Thus, the local fixedpoint algorithm should stop when it cannot prove its postcondition, either because of a counterexample or because it runs out of candidates for differential invariants. As in the work of Prajna [30], the degrees of parametric invariants, therefore, need to be bounded and increased iteratively. As in [30], there is no natural measure for how these degrees should be increased. Instead, we exploit the fact that the candidates of *Candidates* are independent and we explore them in parallel with fair time interleaving.

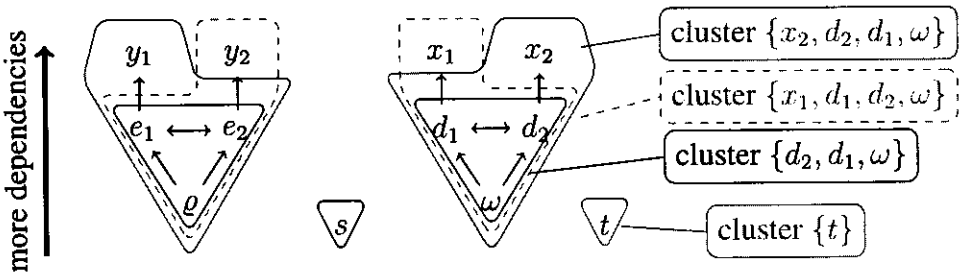


Figure 6: Differential dependencies (arrows) and (triangular) variable clusters of ($\mathcal{F}$)

3.8 Soundness

Theorem 1 (Soundness) *The verification algorithm in Section 3 is sound, i.e., whenever the algorithm* `prove`$(\psi \rightarrow [\alpha]\phi)$ *returns* "`true`", *the* $\mathsf{d}\mathcal{L}$ *formula* $\psi \rightarrow [\alpha]\phi$ *is true in all states, i.e., all states reachable by* α *from states satisfying* ψ *satisfy* ϕ.

A proof is in appendix B.4. Since reachability of hybrid systems is undecidable, our algorithm must be incomplete. It can fail to converge when the required invariants are not expressible in first-order logic. The existence of a fixedpoint in $\mathsf{d}\mathcal{L}$ can be shown, but fixedpoints are not always expressible in real arithmetic [27].

4 Optimizations

4.1 Sound Interleaving of Numerical Simulation of Hybrid Systems

During fixedpoint computations, wrong choices of candidates are time consuming. Thus, in practice, it is important to discover futile attempts quickly. For this, we use non-exhaustive numerical simulation to look for a counterexample for each candidate. To prevent rejecting good candidates due to numerical errors, we discard fragile counterexamples. We consider counterexamples with distance $<\varepsilon$ to safe states as fragile, because small numerical perturbations could make it safe (right x marks in Fig. 7). The left mark in 7, instead, is robust. Robust counterexamples can be ensured by replacing, e.g., $a \geq b$ by $a \geq b + \varepsilon$ in the formulas given to numerical reachability simulation for some estimate $\varepsilon \geq 0$ of the numerical error. Unlike in other approaches [3, 18, 24, 30, 28], numerical errors are *not* critical for soundness, because safety is exclusively established by sound symbolic verification.

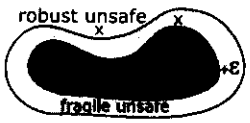


Figure 7: Robustness

We can further exploit the symbolic decomposition performed by our algorithm in Section 3 and prefix recursive calls to *prove*$(\psi \rightarrow [\alpha]\phi)$ with a partial simulation of α. Using cylindric algebraic decomposition [6], we find good samples of states satisfying ψ to start the simulation of α.

4.2 Optimizations for the Verification Algorithm

Formulas with variables that do not change in a fragment of a HP are trivial invariants, as their truth-value is unaffected. For instance, $\omega = \varrho$ is a trivial invariant of system $(\mathcal{F})$. Hence, it can be used as an invariant without proof. A formula like $\omega^2(d_1^2 + d_2^2) > r^2$ in ψ, instead, is not trivially invariant, because d_i changes during $(\mathcal{F})$. Still, it has invariant consequences like $\omega \neq 0$. To

make use of these direct and indirect trivial invariants from ψ , we (soundly) weaken all universal closures of the form $\forall_{cl}\phi$ in lines 2 and 5 of Fig. 5 by $\psi \rightarrow \forall_{cl}\phi$.

5 Experimental Results: Aircraft Roundabout Maneuver

As an example with non-trivial dynamics, we analyze aircraft roundabout maneuvers [34]. Curved flight as in roundabouts is challenging for verification, because of its transcendental solutions. The maneuver in Fig. 4a from [34] and the maneuver in Fig. 4b from [28, 25] are not flyable, because they still involve a few instant turns. A flyable roundabout maneuver without instant turns is depicted in Fig. 8. We verify safety properties for most (but not yet all) phases of Fig. 8 and provide verification results in Tab. 2. We present details in appendix C. Note that the required invariants for the roundabout maneuver cannot even be found from characteristic sets of Differential Gröbner Bases [20].

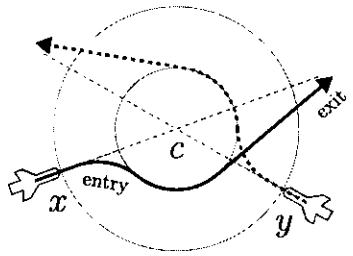


Figure 8: Flyable aircraft roundabout

Verification results for roundabout aircraft maneuvers [34, 7, 28, 25] and the European Train Control System (ETCS) [29] are in Tab. 2. Results are from a 2.6GHz AMD Opteron with 4GB memory. Memory consumption of quantifier elimination is shown in Tab. 2, excluding the graphical front-end. The results are only slightly worse on a 2 year old laptop with 1.7GHz and 1GB. We handle all variables symbolically. The dimension of the continuous state space is indicated.

6 Related Work

Other authors [33, 32, 30] have already argued that invariant techniques scale to more general dynamics than explicit reach-set computations or techniques that require solutions for differential equations [13, 24, 26]. However, they [33, 32] cannot handle hybrid systems with inequalities in initial sets or switching surfaces, which occur in most real applications like aircraft maneuvers. Barrier certificates [30] only work for inequalities, but invariants of roundabout maneuvers require mixed equations and inequalities. Prajna et al. construct barrier certificates of a fixed degree by global optimization over all certificates and modes [30]. This global approach, however, is infeasible for larger examples. Even with degree bound 2, it already requires solving a 5848-dimensional optimization problem for train control [29] and a 10005-dimensional problem for roundabouts with 5 aircraft.

Table 2: Experimental results

Case study	Time(s)	Memory(MB)	Proof steps	Dimension
tangential roundabout (2 aircraft)	14	8	117	13
tangential roundabout (3 aircraft)	387	42	182	18
tangential roundabout (4 aircraft)	730	39	234	23
tangential roundabout (5 aircraft)	1964	88	317	28
bounded speed entry	20	34	28	12
flyable roundabout entry (simplified)	6	10	98	8
ETCS-kernel safety	41	28	53	9
ETCS safety (simplified)	56	27	147	15
ETCS safety	183	87	169	15
ETCS train controllability	1	6	17	5
RBC controllability	1	7	45	16

Tomlin et al. [34] derive saddle solutions for competitive aircraft maneuvers game-theoretically using Hamilton-Jacobi-Isaacs partial differential equations and propose roundabout maneuvers. Their exponential state space discretizations for PDEs, however, do not scale to larger dimensions (they consider dimension 3). Differential invariants, instead, work for 28-dimensional systems.

Straight-line aircraft maneuvers have been analyzed by geometrical meta-level reasoning [10, 14, 17]. They did not consider curved flight paths nor prove that their maneuvers are safe with respect to actual hybrid flight dynamics. In contrast, our approach works directly for the hybrid flight dynamics, and we verify curved roundabout maneuvers instead of straight-line maneuvers with non-flyable instant turns. A few approaches [21, 7] have been undertaken to Model Check if there are *orthogonal* collisions in discretizations of roundabout maneuvers. However, the counterexamples found by our model checker in previous work [28] show that *non-orthogonal* collisions can happen in these maneuvers.

7 Conclusions and Future Work

We have presented a *sound* algorithm for verifying hybrid systems with non-trivial dynamics. It handles differential equations using differential invariants instead of requiring solutions of the differential equations, because the latter quickly yield undecidable arithmetic. We compute differential invariants as fixedpoints using a verification logic for hybrid systems. In the logic we can soundly decompose the system for computing local invariants and we obtain sound recombinations into global invariants. Moreover, we introduce a differential saturation procedure that verifies more complicated properties by refining the system dynamics in a sound way. We validate our algorithm on *roundabout collision avoidance maneuvers* for aircraft and on collision avoidance protocols for trains.

Our algorithm works particularly good for fully parametric hybrid systems, because their parameter constraints can be combined faster to find invariants than systems with a single initial state, where simulation is more appropriate. We want to validate this in further experiments. Dif-

ferential induction and the logic dℒ generalize to liveness properties and to systems with disturbances [27, 25]. In future work, we want to generalize the synthesis of corresponding differential (in)variants. Other invariant constructions for differential equations, e.g., [32] can be added and lifted to hybrid systems using our uniform algorithm.

Acknowledgments. We thank Silke Wagner and Alex Donzé for their helpful proofreading remarks.

References

[1] Rajeev Alur and George J. Pappas, editors. *HSCC*, volume 2993 of *LNCS*. Springer, 2004.

[2] Hirokazu Anai and Volker Weispfenning. Reach set computations using real quantifier elimination. In Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli, editors, *HSCC*, volume 2034 of *LNCS*, pages 63–76. Springer, 2001.

[3] Eugene Asarin, Thao Dang, and Antoine Girard. Reachability analysis of nonlinear systems using conservative approximation. In Oded Maler and Amir Pnueli, editors, *HSCC*, volume 2623 of *LNCS*, pages 20–35. Springer, 2003.

[4] Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors. *HSCC*, volume 4416 of *LNCS*. Springer, 2007.

[5] Edmund M. Clarke. Program invariants as fixedpoints. *Computing*, 21(4):273–294, 1979.

[6] George E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *J. Symb. Comput.*, 12(3):299–328, 1991.

[7] Werner Damm, Guilherme Pinto, and Stefan Ratschan. Guaranteed termination in the verification of LTL properties of non-linear robust discrete time hybrid systems. In Doron Peled and Yih-Kuen Tsay, editors, *ATVA*, volume 3707 of *LNCS*, pages 99–113. Springer, 2005.

[8] Jennifer Mary Davoren and Anil Nerode. Logics for hybrid systems. *Proc. IEEE*, 88(7), July 2000.

[9] Alexandre Donzé and Oded Maler. Systematic simulation using sensitivity analysis. In Bemporad et al. [4], pages 174–189.

[10] Gilles Dowek, César Muñoz, and Víctor A. Carreño. Provably safe coordinated strategy for distributed conflict resolution. In *AIAA Conference Proc. AIAA-2005-6047*, 2005.

[11] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, New York, second edition, 1996.

[12] Melvin Fitting and Richard L. Mendelsohn. *First-Order Modal Logic*. Kluwer Academic Publishers, Norwell, MA, USA, 1999.

[13] Martin Fränzle. Analysis of hybrid systems. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *CSL*, volume 1683 of *LNCS*, pages 126–140. Springer, 1999.

[14] André L. Galdino, César Muñoz, and Mauricio Ayala-Rincón. Formal verification of an optimal air traffic conflict resolution and recovery algorithm. In Daniel Leivant and Ruy de Queiroz, editors, *WoLLIC*, volume 4576 of *LNCS*, pages 177–188. Springer, 2007.

[15] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.

[16] Thomas A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292. IEEE, 1996.

[17] Inseok Hwang, Jegyom Kim, and Claire Tomlin. Protocol-based conflict resolution for air traffic control. *Air Traffic Control Quarterly*, 15(1), 2007.

[18] Ruggero Lanotte and Simone Tini. Taylor approximation for hybrid systems. In Morari and Thiele [22], pages 402–416.

[19] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. *Proc. IEEE - Special Issue on Hybrid Systems: Theory & Applications*, 88(7):926–947, 2000.

[20] Elizabeth L. Mansfield. *Differential Gröbner Bases*. PhD thesis, University Sydney, 1991.

[21] Mieke Massink and Nicoletta De Francesco. Modelling free flight with collision avoidance. In *ICECCS*, pages 270–280. IEEE Computer Society, 2001.

[22] Manfred Morari and Lothar Thiele, editors. *HSCC*, volume 3414 of *LNCS*. Springer, 2005.

[23] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96(2):293–320, May 2003.

[24] Carla Piazza, Marco Antoniotti, Venkatesh Mysore, Alberto Policriti, Franz Winkler, and Bud Mishra. Algorithmic algebraic model checking I: Challenges from systems biology. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *LNCS*, pages 5–19. Springer, 2005.

[25] André Platzer. Differential algebraic dynamic logic for differential algebraic programs. Submitted, `http://www.symbic.net/pub/DAL.pdf`, 2007.

[26] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. In Nicola Olivetti, editor, *TABLEAUX*, volume 4548 of *LNCS*, pages 216–232. Springer, 2007.

[27] André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reasoning*, 2008. Accepted for publication. `http://www.symbic.net/pub/freedL.pdf`.

[28] André Platzer and Edmund M. Clarke. The image computation problem in hybrid systems model checking. In Bemporad et al. [4], pages 473–486.

[29] André Platzer and Jan-David Quesel. Logical verification and systematic parametric analysis in train control. In Magnus Egerstedt and Bud Mishra, editors, *HSCC*, LNCS. Springer, 2008. Long report at `http://www.symbic.net/pub/dvpatc.pdf`.

[30] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE T. Automat. Contr.*, 52(8), 2007.

[31] Vaughan R. Pratt. Semantical considerations on Floyd-Hoare logic. In *FOCS*, 1976.

[32] Enric Rodríguez-Carbonell and Ashish Tiwari. Generating polynomial invariants for hybrid systems. In Morari and Thiele [22], pages 590–605.

[33] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. In Alur and Pappas [1], pages 539–554.

[34] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: a study in multi-agent hybrid systems. *IEEE T. Automat. Contr.*, 43(4), 1998.

A Semantics of Hybrid Programs and Differential Dynamic Logic

The semantics of $\mathsf{d}\mathcal{L}$ is a Kripke semantics in which states of the Kripke model are states of the hybrid system. A state is a map $\nu : V \to \mathbb{R}$; the set of all states is denoted by State. We write $\nu \models \phi$ if formula ϕ is true at state ν (Def. 6). Likewise, $[\![\theta]\!]_\nu$ denotes the real value of term θ at state ν. The semantics of HP α is captured by the state transitions that are possible by running α. For continuous evolutions, the transition relation holds for pairs of states that can be interconnected by a continuous flow respecting the differential equation and invariant region. That is, there is a continuous transition along $x' = \theta \wedge H$ from state ν to state ω, if there is a solution of the differential equation $x' = \theta$ that starts in state ν and ends in ω and that always remains within the region H during its evolution. As in [16, 8], we assume non-zeno behavior, for simplicity.

Definition 5 (Transition system of hybrid programs) *The* transition relation, $\rho(\alpha)$, of a hybrid program α, *specifies which state ω is reachable from a state ν by operations of α and is defined as follows*

1. $(\nu, \omega) \in \rho(x := \theta)$ *iff the state ω is identical to ν except that $\omega(x) = [\![\theta]\!]_\nu$.*
2. $(\nu, \omega) \in \rho(x := \textit{random})$ *iff the state ω agrees with ν except for the value of x, which is an arbitrary real value.*
3. $(\nu, \omega) \in \rho(x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge H)$ *iff for some $r \geq 0$, there is a* (flow) *function $\varphi{:}[0, r] \to$ State with $\varphi(0) = \nu, \varphi(r) = \omega$, such that,*
   - *The differential equation holds, i.e., for each x_i and each time $\zeta \in [0, r]$,*
   $$\frac{\mathsf{d}\,[\![x_i]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) = [\![\theta_i]\!]_{\varphi(\zeta)} \ .$$
   - *For other variables $y \notin \{x_1, \ldots, x_n\}$ and $\zeta \in [0, r]$, the value remains constant, i.e., $[\![y]\!]_{\varphi(\zeta)} = [\![y]\!]_{\varphi(0)}$.*
   - *The invariant is always respected, i.e., $\varphi(\zeta) \models H$ for each $\zeta \in [0, r]$.*
4. $\rho(\alpha \cup \beta) = \rho(\alpha) \cup \rho(\beta)$
5. $\rho(\alpha; \beta) = \{(\nu, \omega) : (\nu, z) \in \rho(\alpha), (z, \omega) \in \rho(\beta) \textit{ for a state } z\}$
6. $(\nu, \omega) \in \rho(\alpha^*)$ *iff there are an $n \in \mathbb{N}$ and $\nu = \nu_0, \ldots, \nu_n = \omega$ such that $(\nu_i, \nu_{i+1}) \in \rho(\alpha)$ for all $0 \leq i < n$.*

Definition 6 (Interpretation of $\mathsf{d}\mathcal{L}$ formulas) *The* interpretation $\models$ *of a $\mathsf{d}\mathcal{L}$ formula with respect to state ν uses the standard meaning of first-order logic:*

1. $\nu \models \theta_1 \sim \theta_2$ *iff* $[\![\theta_1]\!]_\nu \sim [\![\theta_2]\!]_\nu$ *for* $\sim \in \{=, \leq, <, \geq, >\}$

2. $\nu \models \phi \wedge \psi$ *iff* $\nu \models \phi$ *and* $\nu \models \psi$, *accordingly for* $\neg, \vee, \rightarrow, \leftrightarrow$

3. $\nu \models \forall x\, \phi$ *iff* $\omega \models \phi$ *for all* ω *that agree with* ν *except for the value of* x

4. $\nu \models \exists x\, \phi$ *iff* $\omega \models \phi$ *for some* ω *that agrees with* ν *except for the value of* x

It extends to correctness statements about a HP α *as follows*

5. $\nu \models [\alpha]\phi$ *iff* $\omega \models \phi$ *for all* ω *with* $(\nu, \omega) \in \rho(\alpha)$

B Proofs

B.1 Proof of Differential Induction

For the proof of Proposition 1, we prove a result showing that the directional derivative $\nabla_{\mathcal{D}} F$ of formula F as defined in Def. 4 is a generalization of standard function derivatives. We show that the directional derivatives of terms in $\nabla_{\mathcal{D}} F$ in the direction of the vector field of $\mathcal{D}$ agree with the standard differentiation. That is, they agree with the differentiation in the Euclidean real space of the value of these terms along a flow solving the corresponding differential equation $\mathcal{D}$. As a notation for the proof, we introduce an abbreviation for the terms occurring in Def. 4. Let $\mathcal{D}$ be the differential equation system $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n$ and c a term. We define

$$\nabla_{\mathcal{D}}(c) \;:=\; \sum_{i=1}^{n} \frac{\partial c}{\partial x_i} \theta_i \;.$$

For a term c, $\nabla_{\mathcal{D}}(c)$ is a term. For a formula F, the directional derivatives in Def. 4 can be written with this notation as

$$\nabla_{\mathcal{D}} F \;\equiv \bigwedge_{(b \sim c) \in F} (\nabla_{\mathcal{D}}(b) \sim \nabla_{\mathcal{D}}(c)) \quad \text{for } \sim \in \{=, \geq, >, \leq, <\} \;.$$

Lemma 1 *Let* $\mathcal{D} \wedge H$ *be a continuous evolution and let* $\varphi : [0, r] \rightarrow \text{State}$ *be a corresponding flow of duration* $r > 0$ *(Def. 5). Then we have for all terms* c *and all* $\zeta \in [0, r]$ *that*

$$\frac{\mathsf{d}\, [\![c]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) = [\![\nabla_{\mathcal{D}}(c)]\!]_{\varphi(\zeta)} \;.$$

In particular, $[\![c]\!]_{\varphi(t)}$ *is continuously differentiable and its derivative exists on* $[0, r]$.

Proof: The proof is by induction on term c. Let $\mathcal{D}$ be $x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n$.

- If c is variable x_j for some j (for other variables, the proof is simple because c is constant):

$$\frac{\mathsf{d}\, [\![x_j]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) = [\![\theta_j]\!]_{\varphi(\zeta)} = [\![\sum_{i=1}^{n} \frac{\partial x_j}{\partial x_i} \theta_i]\!]_{\varphi(\zeta)} \;.$$

  The last equation holds as $\frac{\partial x_j}{\partial x_j} = 1$ and $\frac{\partial x_j}{\partial x_i} = 0$ for $i \neq j$. The derivatives exist because φ is (continuously) differentiable.

- If c is of the form $a + b$, the desired result can be obtained by using the properties of derivatives and interpretations:

$$\begin{aligned}
&\frac{\mathsf{d}\,[\![a+b]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) \\
&= \frac{\mathsf{d}\,([\![a]\!]_{\varphi(t)} + [\![b]\!]_{\varphi(t)})}{\mathsf{d}t}(\zeta) && [\![\cdot]\!]_\nu \text{ homomorphism for } + \\
&= \frac{\mathsf{d}\,[\![a]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) + \frac{\mathsf{d}\,[\![b]\!]_{\varphi(t)}}{\mathsf{d}t}(\zeta) && \frac{\mathsf{d}}{\mathsf{d}t} \text{ is a linear operator} \\
&= [\![\nabla_{\mathcal{D}}(a)]\!]_{\varphi(\zeta)} + [\![\nabla_{\mathcal{D}}(b)]\!]_{\varphi(\zeta)} && \text{by induction hypothesis} \\
&= [\![\nabla_{\mathcal{D}}(a) + \nabla_{\mathcal{D}}(b)]\!]_{\varphi(\zeta)} && [\![\cdot]\!]_\nu \text{ homomorphism for } + \\
&= [\![\nabla_{\mathcal{D}}(a+b)]\!]_{\varphi(\zeta)} && \nabla \text{ is linear, because } \frac{\partial}{\partial x_i} \text{ is linear}
\end{aligned}$$

- The case if c is of the form $a \cdot b$ is accordingly, using Leibniz's product rule.

□

Proof (of Proposition 1): We have to show that $\nu \models F \rightarrow [\mathcal{D} \wedge H]F$ for all states ν. Let ν satisfy $\nu \models F$ as, otherwise, there is nothing to show. We can assume F to be in disjunctive normal form and consider any disjunct G of F that is true at ν. In order to show that F remains true during the continuous evolution, it is sufficient to show that each conjunct of G is. We can assume these conjuncts to be of the form $c \geq 0$ (or $c > 0$ where the proof is accordingly). Finally, using vectorial notation, we write $x' = \theta$ for the differential equation system. Now let $\varphi : [0, r] \rightarrow \text{State}$ be any flow of $x' = \theta \wedge H$ beginning in $\varphi(0) = \nu$ according to Def. 5. If the duration of φ is $r = 0$, we have $\varphi(0) \models c \geq 0$, because $\nu \models c \geq 0$. For duration $r > 0$, we show that $c \geq 0$ holds all along the flow φ, i.e., $\varphi(\zeta) \models c \geq 0$ for all $\zeta \in [0, r]$.

Suppose there was a $\zeta \in [0, r]$ with $\varphi(\zeta) \models c < 0$, which will lead to a contradiction. The function $h : [0, r] \rightarrow \mathbb{R}$ defined as $h(t) = [\![c]\!]_{\varphi(t)}$ satisfies the relation $h(0) \geq 0 > h(\zeta)$, because $h(0) = [\![c]\!]_{\varphi(0)} = [\![c]\!]_\nu$ and $\nu \models c \geq 0$ by assumption (induction start of Def. 4). By Lemma 1, h is continuous on $[0, r]$ and differentiable at every $\xi \in (0, r)$. The mean value theorem implies that there is a $\xi \in (0, \zeta)$ such that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) \cdot (\zeta - 0) = h(\zeta) - h(0) < 0$. In particular, since $\zeta \geq 0$, we can conclude that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) < 0$. Now Lemma 1 implies that $\frac{\mathsf{d}h(t)}{\mathsf{d}t}(\xi) = [\![\nabla_{\mathcal{D}}(c)]\!]_{\varphi(\xi)} < 0$. This, however, is a contradiction, because the induction step of Def. 4 implies that $H \rightarrow \nabla_{\mathcal{D}}(c \geq 0)$ is true in all states (due to the universal closure $\forall_{cl}$), including $\varphi(\xi) \models H \rightarrow \nabla_{\mathcal{D}}(c \geq 0)$. In particular, as φ is a flow for $\mathcal{D} \wedge H$, we know that $\varphi(\xi) \models H$ holds, and we have $\varphi(\xi) \models \nabla_{\mathcal{D}}(c \geq 0)$, which contradicts $[\![\nabla_{\mathcal{D}}(c)]\!]_{\varphi(\xi)} < 0$. □

B.2 Proof of Differential Saturation

Proof (of Proposition 2): Let F be a continuous invariant, which implies that $\psi \rightarrow [\mathcal{D} \wedge H]F$ is valid. Let ν be a state satisfying ψ (otherwise there is nothing to show). Then, $\nu \models [\mathcal{D} \wedge H]F$. Since this means that F is true all along all flows φ of $\mathcal{D} \wedge H$ that start in ν (Def. 5), the latter differential equation and $\mathcal{D} \wedge H \wedge F$ have the same dynamics and the same reachable states

from ν, i.e., $(\nu,\omega) \in \rho(\mathcal{D} \wedge H)$ holds if and only if $(\nu,\omega) \in \rho(\mathcal{D} \wedge H \wedge F)$ (Def. 5). Thus, we can conclude that $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ and $\psi \rightarrow [\mathcal{D} \wedge H \wedge F]\phi$ are equivalent, because their semantics uses the same transition relation. □

B.3 Proof of Loop Saturation

Proof (of Proposition 3): Let H be a discrete invariant of $\psi \rightarrow [\alpha^*]\phi$. Let, further, F be a discrete invariant of $\psi \rightarrow [\alpha^*]\phi$. Then $\psi \rightarrow F$ and $F \rightarrow [\alpha]F$ are valid by Def. 2. Hence, trivially, $F \rightarrow [\alpha](H \rightarrow F)$ is valid, because all states that satisfy F also satisfy the weaker property $H \rightarrow F$. Especially, $H \wedge F \rightarrow [\alpha](H \rightarrow F)$ is valid. Finally, the validity of $\psi \rightarrow H \wedge F$ clearly entails $\psi \rightarrow F$.

Conversely, let, H be a discrete invariant. Let, further, $H \wedge F \rightarrow [\alpha](H \rightarrow F)$ and $\psi \rightarrow F$ be valid. For $H \wedge F$ to be a discrete invariant, we have to show that F satisfies the induction step of Def. 2 (the induction start $\psi \rightarrow H \wedge F$ is an immediate combination of the validity of $\psi \rightarrow H$ and $\psi \rightarrow F$). Since H is a discrete invariant, $H \rightarrow [\alpha]H$ is valid, which entails $H \wedge F \rightarrow [\alpha]H$ as a special case. Since $H \wedge F \rightarrow [\alpha](H \rightarrow F)$ is valid and $H \wedge F \rightarrow [\alpha]H$ is valid, we conclude that $H \wedge F \rightarrow [\alpha](H \wedge F)$ is valid for the following reason. Let ν be a state satisyfing the initial constraints $H \wedge F$. Then $\nu \models [\alpha]H$ and $\nu \models [\alpha](H \rightarrow F)$. Hence, all states ω reachable from ν by α satisfy $\omega \models H$ and $\omega \models H \rightarrow F$. Thus, they satisfy $\omega \models H \wedge F$, essentially by modus ponens. Consequently, we have shown that $H \wedge F \rightarrow [\alpha](H \wedge F)$ is valid. and, hence, $H \wedge F$ is a discrete invariant of $\psi \rightarrow [\alpha^*]\phi$. □

B.4 Proof of Soundness of the Verification Algorithm

Proof (of Theorem 1): The proof is by induction on the structure of the algorithm.

- In the base case (line 11 of Fig. 2), *prove* returns the result of quantifier elimination, which is a sound decision procedure [6].
- If α is of the form $x := \theta$, the algorithm in line 1 of Fig. 2 is responsible. If it returns "true", then *prove*$(\psi \wedge \hat{x} = \theta \rightarrow \phi_x^{\hat{x}})$ has returned "true". Hence, by induction hypothesis, $\psi \wedge \hat{x} = \theta \rightarrow \phi_x^{\hat{x}}$ is valid. Now, because $\hat{x}$ was a fresh variable, the substitution lemma can be used to show that $\psi \rightarrow \phi_x^{\theta}$ and $\psi \rightarrow [x := \theta]\phi$ are valid. Hence, the result of *prove* is sound.
- If α is of the form $x :=$ *random*, the algorithm in line 9 of Fig. 2 is responsible. Soundness can be proven directly using the fact that ϕ being true after *all* random assignments to x is equivalent to ϕ being true for all real values of x. Hence, $\psi \rightarrow [x :=$ *random*$]\phi$ is valid if and only if $\psi \rightarrow \forall x\, \phi$ is.
- The other cases of Fig. 2 are accordingly.
- If α is of the form $\mathcal{D} \wedge H$ for a differential equation system $\mathcal{D}$, the algorithm in Fig. 5 is responsible. If it returns "true" in line 3 in the first place, then the calls *prove* in line 2 must have

resulted in "true", hence, by induction hypothesis, H entails ϕ. Thus postcondition ϕ is true in a subregion of the evolution domain H. Thus $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ is valid, trivially, because all evolutions along $\mathcal{D} \wedge H$ always satisfy H and, hence, ϕ. If, however, H was changed in line 6 during the fixedpoint computation, then the calls to *prove* for the properties in line 5 must have returned "true". Thus, by induction hypothesis, the dL formulas $\psi \wedge H \rightarrow F$ and $\forall_{cl}(H \rightarrow \nabla_{\mathcal{D}}(F))$ are valid, hence F is a differential invariant of $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ by Def. 4. Consequently, by Proposition 1, F also is a continuous invariant (Def. 3). Thus, by Proposition 2, the dL formulas $\psi \rightarrow [\mathcal{D} \wedge H]\phi$ and $\psi \rightarrow [\mathcal{D} \wedge H \wedge F]\phi$ are equivalent, and we can (soundly) verify the former by proving the latter. Consequently, the modification of the evolution domain H to $H \wedge F$ in line 6 is sound, because the algorithm will continue proving a refined but equivalent formula for a refined but equivalent system.

- If α is a loop of the form β^*, the proof is similar to the case for differential equations, except that it uses Proposition 3 instead of Proposition 1.

□

C Case Studies

In this section, we present details on the case studies that we have verified with our verification algorithm. The verification tool[1] in which we have implemented our algorithm and the problem specification files for the case studies[2] are available online. As case studies, we verify collision avoidance properties for flight control maneuvers [34, 19, 21, 7, 10, 28, 14, 17] and train control protocols [29].

C.1 Flyable Tangential Roundabout Maneuver

As a case study, we show how safety properties of collision avoidance maneuvers in air traffic management can be verified with our verification algorithm. Aircraft maneuvers are challenging for verification [34, 19, 21, 7, 10, 28, 14, 17], because of the complicated spatial/geometrical movement of aircraft. Technically, this complexity manifests in difficulties with analyzing hybrid systems for flight equation (1) or the equivalently reparameterized differential equation system ($\mathcal{F}$).

On straight lines, i.e., where the angular velocity is $\omega = 0$, the value of $\sin\vartheta$ and $\cos\vartheta$ remain constant during continuous evolutions such that the solutions of (1) are (possibly piecewise) *linear functions*. For hybrid systems with linear evolution functions, there are well-known analysis techniques [16]. Pure straight line maneuvers [34, 21, 10, 14, 17] are aircraft maneuvers with piecewise linear evolutions, see, e.g., Fig. 9. They assume instant turns for heading changes of the aircraft between multiple straight line segments. Instant turns, however, are impossible in midflight, because they are *not flyable*: Aircraft cannot suddenly change their flight direction from 0 to 45 degrees discontinuously but need to follow a smooth curve instead, in which they slowly gear towards the desired direction.

[1] Verification tool KeYmaera available at `http://www.symbic.net/info/KeYmaera.html`

[2] All case studies are available at `http://www.symbic.net/pub/fpdi-examples.zip`

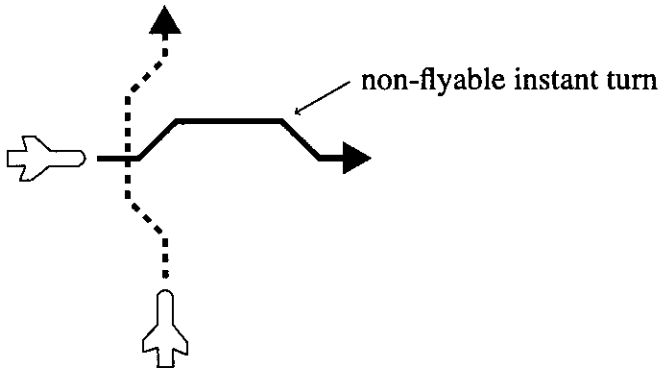


Figure 9: Non-Flyable straight line maneuver with instant turns

During curves, the angular velocity ω is non-zero, which causes the trigonometric expressions in (1) to have a permanent non-constant effect on the dynamics of the system. Accordingly, for $\omega \neq 0$, the equivalent differential equation system ($\mathcal{F}$) has transcendental solutions, such that reachability problems along these solutions fall into undecidable classes of arithmetics. Consequently, maneuvers with curves like in Fig. 8 are much more challenging for verification than straight line maneuvers like Fig. 9, because the flight equations (1) and ($\mathcal{F}$) become highly non-trivial. To verify roundabout maneuvers with curves like in Fig. 4, our algorithm works with differential invariants (Def. 4) instead of solutions of differential equations.

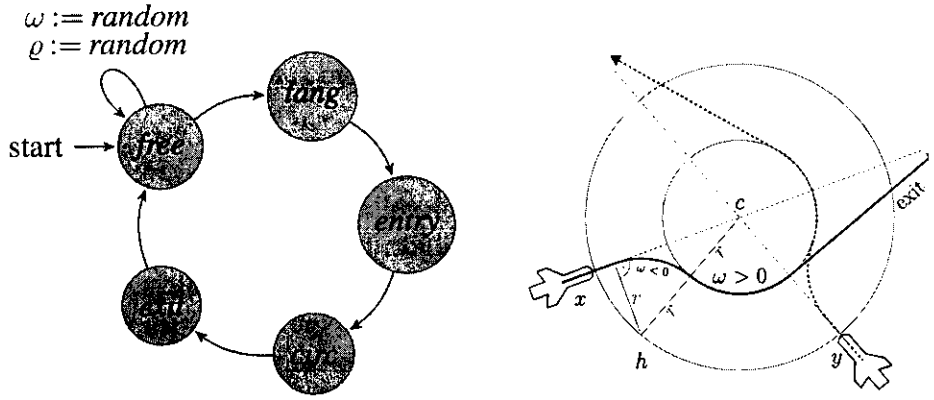


Figure 10: Protocol cycle and construction of flyable roundabout maneuver

A fully flyable roundabout maneuver is depicted in Fig. 8. It does not contain instant turns, but all of its curves are sufficiently smooth. The flyable roundabout maneuver consists of the phases in Fig. 10 which correspond to the flight phases marked in Fig. 8.

During free flight, the aircraft move without restrictions by repeatedly choosing arbitrary new angular velocities ω and ϱ (in phase *free*). When they come closer, the aircraft agree on a roundabout maneuver by negotiating a common roundabout center c (in the coordination phase *tang*). Next, the aircraft approach the roundabout circle by a right curve with $\omega < 0$ (*entry* mode). During the *circ* mode, the aircraft follow the circular roundabout maneuver around the agreed center c with

a left curve of common angular velocity ω. Finally, the aircraft leave the circular roundabout in cruise mode ($\omega = 0$) into their original direction (*exit*) and enter free flight again when they have sufficient distance. The maneuver is symmetric when exchanging left and right curves.

C.1.1 Verification Overview.

We pursue the following overall verification plan by verifying, subsequently:

1. *Tangential roundabout maneuver:* Prove that the protected zones of aircraft are safely separated at all times during the whole maneuver when using a simplified *entry* operation.
2. *Bounded entry speed*: Prove that linear speeds are bounded for the overall maneuver.
3. *Flyable entry procedure*: Prove that the simplified *entry* procedure can be replaced by a flyable curve.
4. *Entry separation*: Prove that the protected zone is respected during flyable entry procedure.

We present details on these verification tasks in the sequel. Informally, the property in case 4 is a consequence of the bounded speed and bounded duration of the flyable entry procedure when initiating the negotiation phase *tang* with sufficient distance. For the time being, we did not yet verify case 4 formally.

C.1.2 Tangential Roundabout Maneuver.

First, we prove that the tangential roundabout maneuver safely avoids collisions, i.e., the aircraft always maintain a safe distance $\geq p$ during the curved flight in the roundabout circle. In addition, we verify that arbitrary repetitions of the protocol cycle are safe at all times for a simplified choice of the entry maneuver.

The flight equations for aircraft x are denoted by $\mathcal{F}(\omega)$, i.e., the upper equations of $(\mathcal{F})$. We abbreviate the differential equations for aircraft y by $\mathcal{G}(\varrho)$ for the lower equations of $(\mathcal{F})$.

The model and specification for this tangential roundabout are given in Fig. 11. There, safety property ψ for collision avoidance maneuvers expresses that protected zones are respected during the flight (specified by separation property ϕ). The flight controller (*trm**) performs collision avoidance maneuvers by tangential roundabouts and repeats these maneuvers any number of times as needed. During each *trm* phase, the aircraft first perform free flight (*free*) by (repeatedly) independently adjusting their angular velocities ω and ϱ while they are safely separated, which is expressed by conjunct ϕ of the differential equation. Due to invariant region ϕ of *free*, the tangential roundabout maneuver must be initiated (by a tangential initiation controller *tang*) before the flight paths become unsafe. Then, the tangential roundabout maneuver itself is carried out by the differential equation $\mathcal{F}(\omega) \wedge \mathcal{G}(\omega)$ according to some common angular velocity ω determined by *tang*. Finally, the collision avoidance roundabouts can be left again by repeating the loop *trm** and entering arbitrary free flight at any time. When further conflicts occur during free flight, the controller in Fig. 11 again enters roundabout conflict resolution maneuvers.

$$
\begin{aligned}
\psi &\equiv \phi \rightarrow [\mathit{trm}^*]\phi \\
\phi &\equiv \|x-y\|^2 \geq p^2 \equiv (x_1-y_1)^2+(x_2-y_2)^2 \geq p^2 \\
\mathit{trm} &\equiv \mathit{free};\ \mathit{tang};\ \mathcal{F}(\omega) \wedge \mathcal{G}(\omega) \\
\mathit{free} &\equiv (\omega := \mathit{random};\ \varrho := \mathit{random};\ \mathcal{F}(\omega) \wedge \mathcal{G}(\varrho) \wedge \phi)^* \\
\mathit{tang} &\equiv \omega := \mathit{random};\ c := \mathit{random}; \\
&\quad d_1 := -\omega(x_2-c_2);\ d_2 := \omega(x_1-c_1); \\
&\quad e_1 := -\omega(y_1-c_1);\ e_2 := \omega(y_2-c_2)
\end{aligned}
$$

Figure 11: Flight control with tangential roundabout collision avoidance maneuvers

In summary, property ψ of Fig. 11 expresses that the aircraft remain safe during the flight, especially during evasive roundabout maneuvers. Our verification results for this property are indicated in row 1 of Tab. 2. The next rows in Tab. 2 prove a corresponding property for up to 5 aircraft, which jointly participate in the roundabout maneuver. There, the safety property is mutual collision avoidance, i.e., each of the aircraft has a safe distance $\geq p$ to all the other aircraft. For instance, Fig. 12 contains the system and separation property specification for the roundabout maneuver with 4 aircraft. There, property ψ expresses that the 4 aircraft at x, y, z and u, respectively, keep mutual distance $\geq p$, which gives a quadratic number of constraints. This quadratic increase in the property that actually needs to proven for a safe roundabout of n aircraft causes the increased verification times for more aircraft in Tab. 2.

C.1.3 Bounded Speed.

The tangential roundabout maneuver in Fig. 11 maintains collision avoidance for all its chocies of center c and angular velocity ω in *tang*. Next, we show that there always is a choice respecting external requirements on linear speed (aircraft can neither fly too slow nor too fast). Hence, for all external choices of the linear speed v, there is a choice for the options in *tang* such that the velocity is v:

$$\forall v\,(\phi \rightarrow \langle \mathit{tang} \rangle (\phi \wedge d_1^2 + d_2^2 = v^2))\ .$$

The verification results for this property are indicated in line 5 of Tab. 2.

C.1.4 Flyable Entry Procedure.

In order to generalize the verification results about the tangential roundabout maneuver with simplified entry procedures to the fully flyable tangential roundabout maneuver, we analyze a flyable entry procedure, which replaces our simple choice of *entry* in Fig. 11 and Fig. 12.

A flyable entry maneuver that follows the smooth entry curve from Fig. 8 is depicted in Fig. 13. Its construction uses the anchor point h indicated in Fig. 10. Anchor h is positioned relative to the roundabout center c and the x position at the start of the entry curve (i.e., with x at the right angle indicated in Fig. 10). The property in Fig. 13 specifies that the tangential configuration of the simple choice for *tang* in Fig. 11 can be reached by a flyable curve when waiting until x and center c

$$
\begin{aligned}
\psi &\equiv \phi \rightarrow [\mathit{trm}^*]\phi \\
\phi &\equiv (x_1-y_1)^2+(x_2-y_2)^2 \geq p^2 \wedge (y_1-z_1)^2+(y_2-z_2)^2 \geq p^2 \\
&\quad \wedge (x_1-z_1)^2+(x_2-z_2)^2 \geq p^2 \wedge (x_1-u_1)^2+(x_2-u_2)^2 \geq p^2 \\
&\quad \wedge (y_1-u_1)^2+(y_2-u_2)^2 \geq p^2 \wedge (z_1-u_1)^2+(z_2-u_2)^2 \geq p^2 \\
\mathit{trm} &\equiv \mathit{free};\ \mathit{tang}; \\
&\quad x_1'=d_1 \wedge x_2'=d_2 \wedge d_1'=-\omega_x d_2 \wedge d_2'=\omega_x d_1 \\
&\quad \wedge y_1'=e_1 \wedge y_2'=e_2 \wedge e_1'=-\omega_y e_2 \wedge e_2'=\omega_y e_1 \\
&\quad \wedge z_1'=f_1 \wedge z_2'=f_2 \wedge f_1'=-\omega_z f_2 \wedge f_2'=\omega_z f_1 \\
&\quad \wedge u_1'=g_1 \wedge u_2'=g_2 \wedge g_1'=-\omega_u g_2 \wedge g_2'=\omega_u g_1 \\
\mathit{free} &\equiv (\omega_x := \mathit{random};\ \omega_y := \mathit{random};\ \omega_z := \mathit{random};\ \omega_u := \mathit{random}; \\
&\quad x_1'=d_1 \wedge x_2'=d_2 \wedge d_1'=-\omega_x d_2 \wedge d_2'=\omega_x d_1 \\
&\quad \wedge y_1'=e_1 \wedge y_2'=e_2 \wedge e_1'=-\omega_y e_2 \wedge e_2'=\omega_y e_1 \\
&\quad \wedge z_1'=f_1 \wedge z_2'=f_2 \wedge f_1'=-\omega_z f_2 \wedge f_2'=\omega_z f_1 \\
&\quad \wedge u_1'=g_1 \wedge u_2'=g_2 \wedge g_1'=-\omega_u g_2 \wedge g_2'=\omega_u g_1 \wedge \phi)^* \\
\mathit{tang} &\equiv \omega := \mathit{random};\ c := \mathit{random}; \\
&\quad d_1 := -\omega(x_2-c_2);\ d_2 := \omega(x_1-c_1); \\
&\quad e_1 := -\omega(y_1-c_1);\ e_2 := \omega(y_2-c_2); \\
&\quad f_1 := -\omega(z_1-c_1);\ f_2 := \omega(z_2-c_2); \\
&\quad g_1 := -\omega(u_1-c_1);\ g_2 := \omega(u_2-c_2)
\end{aligned}
$$

Figure 12: Tangential roundabout collision avoidance maneuver (4 aircraft)

have distance r. The existence of a choice for the anchor point h satisfying the requirements in Fig. 13 can be shown by proving the following dL diamond formula:

$$
\begin{aligned}
&\langle h := \mathit{random}; \\
&\quad ?(d_1=\omega(x_2-h_2) \wedge d_2=-\omega(x_1-h_1)); \\
&\quad ?((h_1-c_1)^2+(h_2-c_2)^2=(2r)^2); \\
&\quad ?(r^2=(x_1-h_1)^2+(x_2-h_2)^2); \\
&\rangle \mathit{true}
\end{aligned}
$$

This property can be verified together with that in Fig. 13 in a simplified version. To overcome the complexity of real quantifier elimination [6], which is doubly exponential in the number of quantifier alternations, we use symmetry reduction to simplify the property in Fig. 13.

Without loss of generality, we can recenter the coordinate system to have c at position 0. Further, we can assume aircraft x to come from the left by changing the orientation of the coordinate system. Finally, we can assume, without loss of generality, the linear speed to be 1 (by rescaling units appropriately). Observe that we cannot fix a value for both the linear speed and the angular

$$
\begin{array}{l}
[h := \mathit{random}; \\
\quad ?(d_1 = \omega(x_2 - h_2) \wedge d_2 = -\omega(x_1 - h_1)); \\
\quad ?((h_1 - c_1)^2 + (h_2 - c_2)^2 = (2r)^2); \\
\quad ?(r^2 = (x_1 - h_1)^2 + (x_2 - h_2)^2); \\
\quad x_1' = d_1 \wedge x_2' = d_2 \wedge d_1' = \omega d_2 \wedge d_2' = -\omega d_1 \wedge ((x_1 - c_1)^2 + (x_2 - c_2)^2 \geq r^2) \\
](\\
\qquad (x_1 - c_1)^2 + (x_2 - c_2)^2 > r^2 \\
\qquad \vee (d_1 = -\omega(x_2 - c_2) \wedge d_2 = \omega(x_1 - c_1)) \\
\quad)
\end{array}
$$

Figure 13: Flyable entry procedure

velocity, because their units are interdependent. In other words, if we fix the linear speed, we need to consider all angular velocities to verify all possible curve radii r for the roundabout maneuver. The x position resulting from these symmetry reductions can be determined as follows, see Fig. 10:

$$x = (0, 2r\cos\frac{\pi}{6}) = (0, \sqrt{(2r)^2 - r^2}) = (0, \sqrt{3}r) \ .$$

To express the square root functoin, we can easily use a random assignment for x_2 with a test condition $x_1^2 = (\sqrt{3}r)^2 = 3r^2$. Consequently, we simplify Fig. 13 by specializing to the following situation:

$$
\begin{array}{l}
d_1 := 1;\ d_2 := 0;\ \ c_1 := 0;\ c_2 := 0; \\
x_2 := 0; \\
r := \mathit{random};\ ?r > 0;\ \omega := 1/r; \\
x_1 := \mathit{random};\ ?x_1^2 = 3r^2 \wedge x_1 \leq 0;
\end{array}
$$

Verification results for the resulting entry procedure, including the proof of existence of a corresponding anchor point h are shown in Tab. 2.

C.2 European Train Control System (ETCS)

The European Train Control System (ETCS) is a standard to assure safe operation of trains and high throughput of high speed trains. ETCS level 3 follows the *moving block principle*, i.e., movement authorities are not known beforehand but determined based on the current track situation by a *Radio Block Controller* (RBC). Trains are only allowed to move within their current movement authority block (denoted by $\mathfrak{m}$), which can be updated by the RBC using wireless communication. Hence the train controller needs to regulate the movement of a train locally such that it always remains within $\mathfrak{m}$, see Fig. 14.

The properties in Tab. 2 prove safety and controllability properties of the parametric ETCS protocol. The ETCS system model is given in Fig. 15. We refer to [29] for details on this case study. The properties in Tab. 2 correspond to the respective propositions in previous work [29]. Note that the algorithm that we introduced in this paper computes the invariants for ETCS automatically, which had to be provided manually in [29].

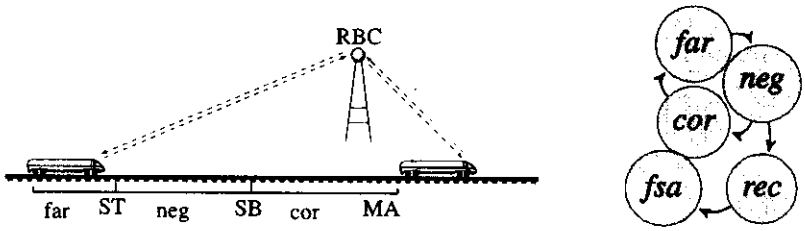


Figure 14: ETCS train coordination protocol

D Additional Algorithms

The algorithm in Fig. 16 verifies loops. It is a direct adaption of that in Fig. 5, except that it uses Proposition 3 as an induction step for loops. The algorithm in Fig. 16 performs a fixedpoint computation for loops and recursively combines the local differential invariants obtained by differential saturation to form a global invariant. It recursively uses *prove* for verifying its subtasks, which handle the discrete switching behavior according to Fig. 2 and infer local differential invariants according to differential saturation by the fixedpoint algorithm in Fig. 5.

E Proof Rules of the dℒ Sequent Calculus

In this section, we briefly summarize the sequent calculus for dℒ [26].

A *sequent* is of the form $\Gamma \vdash \Delta$, where the *antecedent* Γ and *succedent* Δ are finite sets of formulas. Its semantics is that of the formula $\bigwedge_{\phi\in\Gamma}\phi \rightarrow \bigvee_{\psi\in\Delta}\psi$. Sequents will be treated as an abbreviation.

The dℒ calculus uses substitutions. The result of applying to ϕ the substitution that replaces x_i by θ is defined as usual; it is denoted by ϕ^{θ}_{x}. In the dℒ calculus, only admissible substitutions are applicable, which is crucial for soundness. We assume α-conversion for renaming as needed.

Definition 7 (Admissible substitution) *An application of a substitution σ is* admissible *if no replaced term t occurs in the scope of a quantifier or modality binding a variable of σt or t. A modality binds x if it contains an assignment $x := \theta$ or a differential equation containing x'.*

As usual in sequent calculus—although the direction of entailment is from *premisses* (above rule bar) to *conclusion* (below)—the order of reasoning and reading is *goal-directed* in practice: Rules are applied in tableau-style, that is, starting from the desired conclusion at the bottom (goal) to

$$
\begin{array}{lll}
\text{spec} & : & \tau.v^2 - \mathrm{m}.d^2 \leq 2b(\mathrm{m}.e - \tau.p) \wedge \tau.v \geq 0 \wedge \mathrm{m}.d \geq 0 \wedge b > 0 \\
 & & \rightarrow\ [\text{ETCS}](\tau.p \geq \mathrm{m}.e \rightarrow \tau.v \leq \mathrm{m}.d) \\
\text{ETCS} & : & (\text{train} \cup \text{rbc})^* \\
\text{train} & : & \text{spd; atp; move} \\
\text{spd} & : & (?\tau.v \leq \mathrm{m}.r;\ \tau.a := *;\ ? - b \leq \tau.a \leq A) \\
 & & \cup(?\tau.v \geq \mathrm{m}.r;\ \tau.a := *;\ ?0 > \tau.a \geq -b) \\
\text{atp} & : & SB := \frac{\tau.v^2 - \mathrm{m}.d^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon\,\tau.v\right); \\
 & & (?(\mathrm{m}.e - \tau.p \leq SB \vee rbc.message = emergency);\ \tau.a := -b) \\
 & & \cup(?\mathrm{m}.e - \tau.p \geq SB \wedge rbc.message \neq emergency) \\
\text{move} & : & t := 0;\ (\tau.p' = \tau.v, \tau.v' = \tau.a, t' = 1 \wedge \tau.v \geq 0 \wedge t \leq \varepsilon) \\
\text{rbc} & : & (rbc.message := emergency) \\
 & & \cup\ (\mathrm{m}_0 := \mathrm{m}; \mathrm{m} := *; \\
 & & \quad ?\mathrm{m}.r \geq 0 \wedge \mathrm{m}.d \geq 0 \wedge \mathrm{m}_0.d^2 - \mathrm{m}.d^2 \leq 2b(\mathrm{m}.e - \mathrm{m}_0.e))
\end{array}
$$

Figure 15: Formal model of parametric ETCS cooperation protocol

```
1  function prove(ψ → [α*]φ):
2    H := true    /* currently known invariant of ψ → [α*]φ */
3    if prove(∀cl(H → φ)) then
4      return true          /* correctness property proven */
5    for each F ∈IndCandidates(ψ → [α*]φ, H) do
6      if prove(ψ ∧ H → F) and prove(∀cl(H ∧ F → [α](H → F))) then
7        H := H ∧ F         /* refine by discrete invariant */
8        goto 3;            /* repeat fixedpoint loop */
9    end for
10   return "not provable using candidates"
```

Figure 16: Fixedpoint algorithm for discrete loop invariants (loop saturation)

the resulting premisses (sub-goals). The proof rules of the $\mathsf{d}\mathcal{L}$ calculus are depicted in Fig. 17. The calculus consists of propositional rules (P-rules), first-order quantifier rules (F-rules), rules for dynamic modalities (D-rules), and global rules (G-rules).

For propositional logic, standard P-rules are listed in Fig. 17. Unlike in uninterpreted first-order logic [11, 12], quantifiers are dealt with using quantifier elimination (QE) over the reals [6]. Compatibility with dynamic modalities is established using side deductions for the F-rules.

The D-rules handle HP by successively transforming them into logical formulas. State checks like $?H$ are shown by assuming the test suceeds, i.e., H holds true (D4), nondeterministic choices split into their alternatives (D1), $\alpha; \beta$ is proven using nested modalities (D2), and random assignments are handled by universal quantification (D5). D3 uses substitutions for handling discrete change. Rules D6–D7 are weakening and strengthening rules for DAF, respectively.

The G-rules are global rules. They depend on the truth of their premisses in all states, which is ensured by the universal closure with respect to all free variables. If $x_1, \ldots, x_n$ are the free variables of Φ, then $\forall x_1 \ldots \forall x_n \; \Phi$ is its *universal closure*. The G-rules are given in a form that best displays their underlying logical principles. The general pattern for applying G-rules to prove that the succedent of their conclusion holds is to prove that both the antecedent of their conclusion and their premiss holds. Formally such derived rules can be obtained using a cut (R10). Cuts are not needed in practice.

G1 is a generalisation rule. G2 is a discrete induction schema for repetitions with inductive invariant F. G2 says that F holds after any number of repetitions of α, if it holds initially and is sustained after each execution of α.

G3 is a rule for *differential induction*, which is a continuous form of induction along differential constraints. The induction rules G2 and G3 differ in the way the invariant remains true once it is true initially. G2 uses the inductive nature of repetition. G3, instead, uses continuity of evolution and the differential equation for a continuous induction step with the *differential invariant* F: If F holds initially (antecedent of conclusion) and its gradient $\nabla F \cdot \mathcal{D}$ (see Def. 4 on page 6) satisfies the same relations when taking into account the differential constraints (premiss), then F itself is sustained differentially (succedent of conclusion). Finally, G-rules can be combined with generalisation (G1) to strengthen postconditions as needed.

Definition 8 (Provability) *A formula ψ is* provable *from a set Φ of formulas, denoted by $\Phi \vdash_{\mathsf{d}\mathcal{L}} \psi$ iff there is a finite set $\Phi_0 \subseteq \Phi$ for which the sequent $\Phi_0 \vdash \psi$ is derivable. In turn, a sequent $\Phi \vdash \Psi$ is* derivable *iff there is an inference rule of the $\mathsf{d}\mathcal{L}$ calculus (Fig. 17) with conclusion $\Phi \vdash \Psi$ such that all premisses of the rule are derivable.*

$$\text{(P1)}\ \frac{\Gamma, \phi \vdash}{\Gamma \vdash \neg\phi} \quad \text{(P3)}\ \frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi} \quad \text{(P5)}\ \frac{\Gamma \vdash \phi, \psi}{\Gamma \vdash \phi \vee \psi} \quad \text{(P7)}\ \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \rightarrow \psi}$$

$$\text{(P2)}\ \frac{\Gamma \vdash \phi}{\Gamma, \neg\phi \vdash} \quad \text{(P4)}\ \frac{\Gamma, \phi, \psi \vdash}{\Gamma, \phi \wedge \psi \vdash} \quad \text{(P6)}\ \frac{\Gamma, \phi \vdash \quad \Gamma, \psi \vdash}{\Gamma, \phi \vee \psi \vdash} \quad \text{(P8)}\ \frac{\Gamma \vdash \phi \quad \Gamma, \psi \vdash}{\Gamma, \phi \rightarrow \psi \vdash}$$

$$\text{(R9)}\ \frac{}{\Gamma, \phi \vdash \phi} \quad \text{(R10)}\ \frac{\Gamma \vdash \phi \quad \Gamma, \phi \vdash}{\Gamma \vdash}$$

$$\text{(F1)}\ \frac{\text{QE}(\forall x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma \vdash \Delta, \forall x\, \phi} \qquad \text{(F3)}\ \frac{\text{QE}(\exists x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma \vdash \Delta, \exists x\, \phi}$$

$$\text{(F2)}\ \frac{\text{QE}(\exists x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma, \forall x\, \phi \vdash \Delta} \qquad \text{(F4)}\ \frac{\text{QE}(\forall x\ \bigwedge_i(\Gamma_i \vdash \Delta_i))}{\Gamma, \exists x\, \phi \vdash \Delta}$$

$$\text{(D1)}\ \frac{\Gamma \vdash [\alpha]\phi \wedge [\beta]\phi}{\Gamma \vdash [\alpha \cup \beta]\phi} \quad \text{(D2)}\ \frac{\Gamma \vdash [\alpha][\beta]\phi}{\Gamma \vdash [\alpha;\beta]\phi} \quad \text{(D3)}\ \frac{\Gamma \vdash \phi_x^\theta}{\Gamma \vdash [x := \theta]\phi} \quad \text{(D4)}\ \frac{\Gamma, H \vdash \theta}{\Gamma \vdash [?H]\phi}$$

$$\text{(D5)}\ \frac{\Gamma \vdash \forall x\, \theta}{\Gamma \vdash [x := \mathit{random}]\phi}$$

$$\text{(D6)}\ \frac{\Gamma \vdash [?\chi]\phi}{\Gamma \vdash [\mathcal{D} \wedge \chi]\phi} \qquad \text{(D7)}\ \frac{\Gamma \vdash [\mathcal{D}]\chi \quad \Gamma \vdash [\mathcal{D} \wedge \chi]\phi}{\Gamma \vdash [\mathcal{D}]\phi}$$

$$\text{(G1)}\ \frac{\Gamma \vdash \forall_{cl}(\phi \rightarrow \psi)}{\Gamma, [\alpha]\phi \vdash [\alpha]\psi} \qquad \text{(G2)}\ \frac{\Gamma \vdash \forall_{cl}(F \rightarrow [\alpha]F)}{\Gamma, F \vdash [\alpha^*]F}$$

$$\text{(G3)}\ \frac{\Gamma \vdash \forall_{cl}(\chi \rightarrow \nabla F \cdot (x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n))}{\Gamma, (\chi \rightarrow F) \vdash [x_1' = \theta_1 \wedge \cdots \wedge x_n' = \theta_n \wedge \chi]F}$$

In all rule schemata, *all* substitutions need to be admissible. In D6–D7, $\mathcal{D}$ is a differential equation and χ an arithmetic formula. In G3, F is first-order without negative equalities. For F-rules, the $\Gamma_i \vdash \Delta_i$ are obtained from the resulting sub-goals of a side deduction, see $(\star)$ in Fig. 18. The side deduction is started from the goal $\Gamma \vdash \Delta, \phi$ at the bottom (or $\Gamma, \phi \vdash \Delta$ for F2 and F4), where x is assumed not to occur in Γ, Δ using renaming. In the resulting sub-goals $\Gamma_i \vdash \Delta_i$, variable x is assumed to occur in first-order formulas only, as quantifier elimination (QE) is then applicable.

Figure 17: Rule schemata of the $\mathsf{d}\mathcal{L}$ calculus.

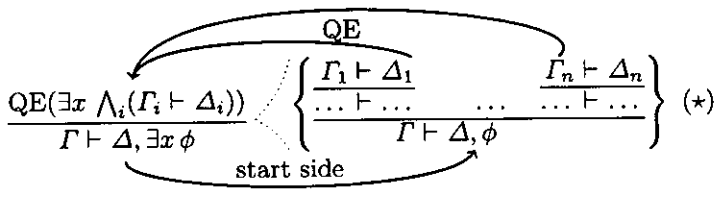


Figure 18: Side deduction for quantifier elimination rules.