Affine Systems of ODEs in Isabelle/HOL for Hybrid-Program Verification*

Jonathan Julián Huerta y Munive[0000−0003−3279−3685]

University of Sheffield, Western Bank, Sheffield, S10 2TN, South Yorkshire, UK
`jjhuertaymunive1@sheffield.ac.uk`

Abstract. We formalise mathematical components for solving affine and linear systems of ordinary differential equations in Isabelle/HOL. The formalisation integrates the theory stacks of linear algebra and analysis and substantially adds content to both of them. It also serves to improve extant verification components for hybrid systems by increasing proof automation, removing certification procedures, and decreasing the number of proof obligations. We showcase these advantages through examples.

Keywords: Hybrid systems · Formal verification· Proof assistants

1 Introduction

With the increased number of computers controlling physical mechanisms, also known as cyber-physical systems, proofs of their correctness become more relevant. An important approach is differential dynamic logic ($\mathsf{d}\mathcal{L}$) [21]. It is an extension of dynamic logic with inference rules to reason about flows and invariants of ordinary differential equations (ODEs). Numerous case studies apply it and its domain-specific proof assistant, KeYmaera X [16,14]. Despite other approaches to verification [1,2], we focus on $\mathsf{d}\mathcal{L}$-style deductive verification.

Our recent $\mathsf{d}\mathcal{L}$-inspired components allow the verification of hybrid programs in the general purpose proof assistant Isabelle/HOL [17,19,7]. Using a shallow embedding and Kleene algebras instead of dynamic logics, the implementation of these components makes them modular and extensible. Their modularity has been explored before in various ways, however their extensibility for the benefit of proof performance has not yet been pursued. In particular, extensions of Isabelle's mathematical components for ordinary differential equations to specific classes promises significant benefits in this regard.

Linear and affine systems of ODEs, for example, those described by linear (resp. affine) transformations, are among the simplest and most studied variants. They enjoy desirable mathematical properties like existence and uniqueness of solutions to their associated initial value problems (IVPs), and come with various methods for solving them. In particular, there is an explicit way to compute the general solution for their time-independent versions [10,22]. Although there is

* This work was funded by CONACYT's scholarship no. 440404.

much work in extending both the ODE libraries [13,12,11] and the linear algebra libraries [23,6], there are no formalisations in Isabelle connecting both theory stacks and this reverberates in the verification process with our components. For instance, formalising existence and uniqueness results for affine and linear systems reduces the proofs that users have to supply. Also, where users have to find solutions to the time-independent versions, we can provide the general one.

Thus, inspired by the deductive approach to verification of hybrid systems, our main contribution is the first formalisation of linear and affine systems of ODEs in a proof assistant by combining the theory stacks of linear algebra and ODEs. We add to this integration by extending these libraries with properties about operator norms, diagonal matrices, and derivatives involving matrix-vector multiplication. In addition, we provide evidence that the study and analysis of these systems with a proof assistant is feasible.

Moreover, we extend the Kleene algebra verification components for hybrid systems by improving their tactics for checking if a function is a derivative of another one. We use these tactics to formalise the fact that all linear and affine systems of ODEs have unique solutions, and we certify the general solution for the time-independent case. In the cases where the linear transformation has a diagonalisable representation, we also prove lemmas that include a simpler representation of the general solution. Finally, we add proof automation for operations with the list-representation of $n{\times}n$ matrices.

The Isabelle formalisation itself forms a major contribution of this paper. It adds new mathematical components to an important field of analysis and improves our verification components for hybrid systems. The formalisations are available in the reviewed Archive of Formal Proofs [18].

2 Affine Systems of ODEs

We first review the mathematical definitions and results for differential equations needed for our formalisation.

Dynamical systems describe the time dependency of points in a state space S. Formally, they are monoid actions $\varphi : T \to S \to S$ that satisfy

$$\varphi\,(t_1 + t_2) = \varphi\,t_1 \circ \varphi\,t_2 \qquad \text{and} \qquad \varphi\,0 = id,$$

where the monoid $(T, +, 0)$ represents time. A dynamical system is a *flow* or *continuous* if $T = \mathbb{R}$ or $T = \mathbb{R}_+$, the non-negative real numbers. Flows emerge from solutions to systems of ordinary differential equations as explained below.

In a system of ODEs $X'\,t = f\,(t, X\,t)$, the function $f : T \times S \to \mathbb{R}^n$ is a *vector field*; it assigns a vector to each point in $T \times S$ with $T \subseteq \mathbb{R}$ and $S \subseteq \mathbb{R}^n$, it is continuous and it suffices to describe the system [10,22]. An *initial value problem* then consists of a vector field f and an initial condition $(t_0, s) \in T \times S$, where t_0 and s represent the initial time and state of the system. Therefore, a *solution* to this system is a continuously differentiable function $X : T \to S$ that satisfies $X'\,t = f\,(t, X\,t)$ for all $t \in T$. This function also solves the IVP if it satisfies $X\,t_0 = s$. Finally, if for each $s \in S$ there is a unique solution or *trajectory*

$\varphi_s^f : T \to S$ to the IVP given by f and $(0, s)$, then the equation $\varphi\, t\, s = \varphi_s^f\, t$ defines the flow $\varphi : T \to S \to S$ of f. Geometrically, the trajectory φ_s^f is the only curve in S that passes through s and is always tangential to f.

Picard-Lindelöf's theorem guarantees local existence and uniqueness of solutions for some IVPs [10,22]. It requires the domain $T \times S$ of f to be open with $(t_0, s) \in T \times S$, and f to be locally Lipschitz continuous in S. That is, there must be $\varepsilon > 0$ and $\ell \geq 0$ such that for all $t \in \overline{B_\varepsilon(t_0)} \cap T$ and all $s_1, s_2 \in \overline{B_\varepsilon(s)} \cap S$,

$$\|f\,(t, s_1) - f\,(t, s_2)\| \leq \ell\, \|s_1 - s_2\|\,,$$

where $\|-\|$ denotes the euclidean norm in $\mathbb{R}^n$ and $\overline{B_\varepsilon(t)} = \{\tau \mid \|\tau - t\| \leq \varepsilon\}$. If these conditions are satisfied, then the theorem asserts the existence of an interval $T_s \subseteq T$ where a unique local solution $\varphi_s^f : T_s \to S$ for the IVP exists, that is $(\varphi_s^f)'\, t = f\,(t, \varphi_s^f\, t)$ and $\varphi_s^f\, t_0 = s$ for all $t \in T_s$. If $t_0 = 0$ and $T = \bigcup_{s \in S} T_s$, then the flow φ of f exists and is a monoid action [22].

An important class of vector fields with unique solutions are those representing *affine* systems of ODEs. They satisfy the equation

$$(\varphi_s^f)'\, t = A\, t \cdot \varphi_s^f\, t + B\, t,$$

for matrix-vector multiplication $\cdot$, $n \times n$ matrices $A\, t$ and vectors $B\, t$, where A and B are continuous functions on T. Equally important are the corresponding *linear* systems where $B\, t = 0$ for all $t \in T$.

Affine systems of ODEs are Lipschitz continuous with respect to the operator norm $\|M\|_{op} = \mathit{Sup}\, \{\|M \cdot s\| \mid \|s\| = 1\}$, where M is a matrix with real coefficients and Sup denotes the supremum of a set. Indeed, with Lipschitz constant $\ell = \mathit{Sup}\, \{\|A\, t\|_{op} \mid t \in \overline{B_\varepsilon(s)}\}$,

$$\|(A\, t) \cdot s_1 - (A\, t) \cdot s_2\| = \|(A\, t) \cdot (s_1 - s_2)\| \leq \|A\, t\|_{op}\, \|s_1 - s_2\| \leq \ell\, \|s_1 - s_2\|\,.$$

Constant ℓ exists by continuity of A and $\|-\|$, and compactness of $\overline{B_\varepsilon(s)}$. Picard-Lindelöf thus guarantees a unique local solution for the associated IVPs. In particular, in the time-independent or *autonomous* case where A and B are constant functions, their unique solutions are well-characterised and globally defined. That is, flows φ for autonomous affine systems exist and satisfy

$$\varphi\, t\, s = \exp\,(tA) \cdot s + \exp\,(tA) \cdot \int_0^t (\exp\,(-\tau A) \cdot B)\, d\tau,$$

where exp is the matrix exponential $\exp\, A = \sum_{i \in \mathbb{N}} \frac{1}{i!} A^i$.

Computing such exponentials may be computationally expensive due to the iteration of matrix multiplication. Exceptions are *diagonalisable matrices* A which are similar to a diagonal matrix D in the sense that there is an invertible P such that $A = P^{-1} D P$. For these matrices,

$$\exp\, A = \exp\,(P^{-1} D P) = P^{-1} (\exp\, D) P,$$

where $\exp\, D$ in the right hand side is diagonal and easy to characterise: its entries in the main diagonal are the exponential of those in D. Therefore, when working with solutions of autonomous affine (or linear) systems, it is preferable to work with those in diagonal form.

3 Isabelle Components for Affine Systems of ODEs

We describe our Isabelle formalisation of the mathematical concepts outlined in Section 2. More specifically, we explain our addendum of definitions and lemmas for an integration of the existing libraries for ODEs and linear algebra. We finish with an instance of Picard-Lindelöf's theorem for affine and linear systems.

As Isabelle only allows total functions, we formalise the type of vector fields as $\mathit{real} \Rightarrow {}'a \Rightarrow ({}'a :: \mathit{real\text{-}normed\text{-}vector})$, which by currying is isomorphic to $\mathbb{R} \times V \to V$, where V is a normed vector space over $\mathbb{R}$. We then restrict domains and codomains of solutions using our definition in [19].

```
definition ivp-sols :: (real ⇒ 'a ⇒ ('a :: real-normed-vector)) ⇒ real set ⇒ 'a set ⇒
  real ⇒ 'a ⇒ (real ⇒ 'a) set  (Sols)
  where Sols f T S t₀ s = {X |X. (D X = (λt. f t (X t)) on T) ∧ X t₀ = s ∧ X ∈ T → S}
```

The first conjunct $D\ X = (\lambda t.\ f\, t\ (X\ t))$ in the definiendum above translates to $X'\, t = f\,(t, X\, t)$, the second states that $X :: \mathit{real} \Rightarrow {}'a$ is a solution to the associated IVP, and the third that X maps elements of T into S.

We use $\mathbb{R}^n$, with $n \geq 0$, as our default vector space. It is formalised using Isabelle's type $(\mathit{real}, {}'n)\ \mathit{vec}$ (abbreviated as $\mathit{real}\hat{\ }{}'n$) of real valued vectors of dimension n. Isabelle's HOL-Library builds this type using a bijection to the type of functions ${}'n \Rightarrow \mathit{real}$ with finite ${}'n$. For $s :: \mathit{real}\hat{\ }{}'n$, the expression $s\$i$ denotes the ith coordinate of s. That is, \$ is the bijection from $\mathit{real}\hat{\ }{}'n$ to ${}'n \Rightarrow \mathit{real}$. Its inverse is written with a binder χ that replaces λ-abstraction. Thus, $\chi i.\ s\$i = s$ and $(\chi i.\ c)\$i = c$ for all $s :: \mathit{real}\hat{\ }{}'n$ and $c :: \mathit{real}$.

Matrices are then vectors of vectors—an $m \times n$ matrix A has type $\mathit{real}\hat{\ }{}'n\hat{\ }{}'m$. The product of matrix A with vector s is denoted $A *v\ s$; the scaling of vector s by real number c is written $c *_R s$. In Isabelle, a solution X to an affine system of ODEs with $A :: \mathit{real} \Rightarrow \mathit{real}\hat{\ }{}'n\hat{\ }{}'n$ and $B :: \mathit{real} \Rightarrow \mathit{real}\hat{\ }{}'n$ then satisfies the predicate $D\ X = (\lambda t.\ A\ t *v\ X\ t + B\ t)\ \mathit{on}\ T$.

We use a formalisation of Picard-Lindelöf's theorem from [19]. The locale *picard-lindeloef* groups its assumptions. If *picard-lindeloef* $f\ T\ S\ t_0$ holds, then T and S are open, $t_0 \in T$, $s \in S$, $\lambda t.\ f\, t\, s$ is continuous on T, and f is locally Lipschitz continuous. The context of the locale also contains the lemma *picard-lindeloef.unique-solution*, stating that any two functions solving an IVP

$$(D\ X = (\lambda t.\ f\ t\ (X\ t))\ \mathit{on}\ \{t_0{-}{-}t\}) \qquad X\ t_0 = s \qquad X \in \{t_0{-}{-}t\} \to S$$

are equal at $t \in T$. Here, $\{t_0{-}{-}t\}$ is Isabelle notation for the set of all numbers between t and t_0 where t can be above or below t_0. Our following lemma then yields a generic instance of *picard-lindeloef.unique-solution* for affine systems.

```
lemma picard-lindeloef-affine:
  fixes A :: real ⇒ real^'n^'n
  assumes Ahyp: matrix-continuous-on T A
      and ⋀τ ε. τ ∈ T ⟹ ε > 0 ⟹ bdd-above {‖A t‖_op |t. dist τ t ≤ ε}
      and Bhyp: continuous-on T B and open S
      and t₀ ∈ T and Thyp: open T is-interval T
    shows picard-lindeloef (λ t s. A t *v s + B t) T S t₀
  ⟨proof⟩
```

Assumptions *Ahyp* and *Bhyp* state that functions A and B are continuous. The second one requires that the image of $\overline{B_\tau(\varepsilon)}$ for $\tau \in T$ under $\lambda\, t.\ \|A\,t\|_{op}$ is bounded above. The remaining ones are direct conditions of Picard-Lindelöf's theorem. Continuity in *Ahyp* is different from that in *Bhyp* because Isabelle's default norm for matrices $A :: real\hat{}'n\hat{}'m$ is the Euclidean norm, not the operator norm from Section 2. Thus, for the lemma above, we formalise the Lipschitz continuity argument at the end of Section 2 starting with the following definition.

abbreviation *op-norm* :: $('a::real\text{-}normed\text{-}algebra\text{-}1)\hat{}'n\hat{}'m \Rightarrow real$ $(1\|\text{-}\|_{op})$
where $\|A\|_{op} \equiv onorm\ (\lambda x.\ A *v\ x)$

Function *onorm* lives in Isabelle's HOL-Analysis library and it is an alternative definition of the operator norm $onorm\,f = Sup\,\{\|f\,x\|\,/\,\|x\| \mid x \in V\}$. However, for many proofs, the definition of $\|-\|_{op}$ in Section 2 is more convenient. Hence we formalise the equivalence as shown below.

lemma *op-norm-def*: $\|A\|_{op} = Sup\ \{\|A *v\ x\| \mid x.\ \|x\| = 1\}$
⟨*proof*⟩

We omit its proof because lack of automation for suprema in Isabelle/HOL makes it an 8-line script. We also show that $\|-\|_{op}$ satisfies the norm axioms.

lemma *op-norm-ge-0*: $0 \leq \|A\|_{op}$
using *ex-norm-eq-1 norm-ge-zero norm-matrix-le-op-norm basic-trans-rules*(*23*) **by** *blast*

lemma *op-norm-zero-iff*: $(\|A\|_{op} = 0) = (A = 0)$
unfolding *onorm-eq-0*[*OF blin-matrix-vector-mult*] **using** *matrix-axis-0*[*of 1 A*] **by** *fastforce*

lemma *op-norm-triangle*: $\|A + B\|_{op} \leq (\|A\|_{op}) + (\|B\|_{op})$
using *onorm-triangle*[*OF blin-matrix-vector-mult*[*of A*] *blin-matrix-vector-mult*[*of B*]]
matrix-vector-mult-add-rdistrib[*symmetric*, *of A - B*] **by** *simp*

lemma *op-norm-scaleR*: $\|c *_R\ A\|_{op} = |c| * (\|A\|_{op})$
unfolding *onorm-scaleR*[*OF blin-matrix-vector-mult*, *symmetric*] *scaleR-vector-assoc* **..**

With this norm, we can define continuity for time-dependent matrix functions and prove Lipschitz continuity.

definition *matrix-continuous-on* :: $real\ set \Rightarrow (real \Rightarrow ('a::real\text{-}normed\text{-}algebra\text{-}1)\hat{}'n\hat{}'m) \Rightarrow bool$
where $matrix\text{-}continuous\text{-}on\ T\ A = \forall\, t{\in}T.\forall\,\varepsilon{>}0.\exists\,\delta{>}0.\forall\,\tau{\in}T.\ |\tau - t|{<}\delta \longrightarrow \|A\ \tau - A\ t\|_{op}{\leq}\varepsilon$

lemma *lipschitz-cond-affine*:
fixes $A :: real \Rightarrow real\hat{}'n\hat{}'m$ **and** $T{::}real\ set$
defines $L \equiv Sup\ \{\|A\ t\|_{op}\ |t.\ t \in T\}$
assumes $t \in T$ **and** $bdd\text{-}above\ \{\|A\ t\|_{op}\ |t.\ t \in T\}$
shows $\|A\ t *v\ x - A\ t *v\ y\| \leq L * (\|x - y\|)$
⟨*proof*⟩

Using the constant *UNIV*, the universal set for a given type, we prove the fact that solutions for the autonomous affine and linear case are globally defined. The proofs are just an instantiation of *picard-lindeloef-affine*.

lemma *picard-lindeloef* $(\lambda\ t\ s.\ A *v\ s + B)$ *UNIV UNIV 0*
using *picard-lindeloef-affine*[*of -* $\lambda t.\ A\ \lambda t.\ B$] **by** (*simp only*: *diff-self op-norm0*, *auto*)

lemma *picard-lindeloef* $(\lambda\ t.\ (*v)\ A)$ *UNIV UNIV 0*
using *picard-lindeloef-affine-constant*[*of A 0*] **by** *force*

All the lemmas and abbreviations displayed above are part of a new addition to Isabelle's Archive of Formal Proofs [18]. Our work in this section covers over 10 pages of proofs and definitions about matrix limits, norms, and operations. This is equivalent to more than 600 lines of code. It also includes various tangential concepts to affine systems of ODEs like the maximum norm for matrices and its relation to the operator norm [22].

4 Flows for Affine Systems of ODEs in Isabelle

One part is still missing from our formalisation: the general solution for autonomous affine and linear systems of ODEs. This is the focus of this section. Moreover, we prove that these solutions are proper flows in the sense that they are defined over the entire monoid $\mathbb{R}$ and state space $\mathbb{R}^n$.

The general solution for autonomous affine systems was introduced in Section 2. Similarly, the general solution for the respective autonomous linear systems of ODEs is $X\,t = (\exp\,((t - t_0)\,A)) \cdot s$ for $s \in \mathbb{R}^n$. The exponential operation $\exp\,x = \sum_{n\in\mathbb{N}} \frac{1}{n!} x^n$, however, is available in Isabelle only within the type-class *real-normed-algebra*-1 with an identity element 1 satisfying $\|1\| = 1$. As this is not true for *real^'n^'n*, because $\|(\chi\ i.\ 1)\| \neq 1$, we define a sub-type of square matrices and show that it is an instance of *real-normed-algebra*-1 and *banach*.

```
typedef 'm sq-mtx = UNIV::(real^'m^'m) set
  morphisms to-vec sq-mtx-chi by simp

instance sq-mtx :: (finite) real-normed-algebra-1
  ⟨proof⟩

instance sq-mtx :: (finite) banach
  ⟨proof⟩
```

The command *morphisms* introduces the bijection *to-vec* and its inverse *to-mtx* between *'n sq-mtx* and *real^'n^'n*. Both instantiations require proving that matrices form normed vector spaces. Beyond that, the first instantiation requires showing that they also form a ring. The second instantiation formalises the fact that every Cauchy sequence of square matrices converges.

As many properties in previous sections apply to matrices as vectors of vectors, we lift various operations from this type to our new type of square matrices.

```
lift-definition sq-mtx-ith :: 'm sq-mtx ⇒ 'm ⇒ (real^'m) (infixl $$ 90) is ($) .

lift-definition sq-mtx-vec-mult :: 'm sq-mtx ⇒ (real^'m) ⇒ (real^'m) (infixl *V 90)  is (*v) .

lift-definition sq-mtx-inv :: ('m::finite) sq-mtx ⇒ 'm sq-mtx (-⁻¹ [90]) is matrix-inv .
```

This means that we can write $$ and $*_V$ instead of $ and $*v$ respectively, and we can convert proofs between the new and the old type. We thus obtain the same results as before in the new type, including Picard-Lindelöf's theorem.

```
lemma picard-lindeloef-sq-mtx-affine:
  assumes continuous-on T A and continuous-on T B
      and t₀ ∈ T is-interval T open T and open S
    shows picard-lindeloef (λt s. A t *V s + B t) T S t₀
  ⟨proof⟩
```

Next we extend the derivative tactics in [19] that determine whether one function is a derivative of another. We extend them by adding derivative rules for $(*_V)$. This allows us to formalise the general solution for autonomous linear and affine systems of ODEs.

lemma *has-vderiv-on-sq-mtx-linear*:
$D\ (\lambda t.\ exp\ ((t - t_0) *_R A) *_V s) = (\lambda t.\ A *_V (exp\ ((t - t_0) *_R A) *_V s))\ on\ \{t_0{-}{-}t\}$
by (*rule poly-derivatives*)+ (*auto simp*: *exp-times-scaleR-commute sq-mtx-times-vec-assoc*)

lemma *has-vderiv-on-sq-mtx-affine*:
fixes t_0::*real* **and** A :: (*'a::finite*) *sq-mtx*
defines $lSol\ c\ t \equiv exp\ ((c * (t - t_0)) *_R A)$
shows $D\ (\lambda t.\ lSol\ 1\ t *_V s + lSol\ 1\ t *_V (\int_{t_0}{}^{t} (lSol\ (-1)\ \tau *_V B)\ \partial\tau)) =$
$(\lambda t.\ A *_V (lSol\ 1\ t *_V s + lSol\ 1\ t *_V (\int_{t_0}{}^{t} (lSol\ (-1)\ \tau *_V B)\ \partial\tau)) + B)\ on\ \{t_0{-}{-}t\}$
⟨*proof*⟩

As no conditions on the parameter t are given, these general solutions are flows. We formalise these results with the locale *local-flow* of [19].

lemma *local-flow-sq-mtx-affine*: *local-flow* $(\lambda s.\ A *_V s + B)$ *UNIV UNIV*
$(\lambda t\ s.\ exp\ (t *_R A) *_V s + exp\ (t *_R A) *_V (\int_0{}^{t}(exp\ (-\ \tau *_R A) *_V B)\partial\tau))$
⟨*proof*⟩

lemma *local-flow-sq-mtx-linear*:
local-flow $((*_V)\ A)$ *UNIV UNIV* $(\lambda t\ s.\ exp\ (t *_R A) *_V s)$
⟨*proof*⟩

As reasoning with general solutions is easier for diagonalisable matrices, we formalise matrix invertibility, similarity and diagonal matrices from linear algebra. We also characterise the exponential of a matrix in terms of these concepts.

lemma *mtx-invertible-def*: *mtx-invertible* $A \longleftrightarrow (\exists A'.\ A' * A = 1 \wedge A * A' = 1)$
⟨*proof*⟩

definition *similar-sq-mtx* :: (*'n::finite*) *sq-mtx* ⇒ *'n sq-mtx* ⇒ *bool* (**infixr** ∼ *25*)
where $(A \sim B) \longleftrightarrow (\exists\ P.\ mtx\text{-}invertible\ P \wedge A = P^{-1} * B * P)$

definition *diag-mat* $f = (\chi\ i\ j.\ if\ i = j\ then\ f\ i\ else\ 0)$

lemma *exp-scaleR-diagonal1*:
assumes *mtx-invertible* P **and** $A = P^{-1} * (\mathrm{diag}\ i.\ f\ i) * P$
shows $exp\ (t *_R A) = P^{-1} * (\mathrm{diag}\ i.\ exp\ (t * f\ i)) * P$
⟨*proof*⟩

The first three concepts and related properties are available for matrices of type *real^'n^'n* and *'n sq-mtx*. The exponential is only available for the latter. For example, the notation (diag $i.\ f\ i$) is the *'n sq-mtx* version of *diag-mat f*.

Our development of the type *'m sq-mtx* and the diagonalisation of square matrices is over 16 pages long. It spans over 900 lines of code or more than 200 lemmas whose proofs are long due to the various convergence arguments and instantiations. Yet, the substantial formalisation in this section is new and it allows Isabelle users to prove facts involving derivatives of matrix operations.

5 Working with Linear Systems in Isabelle

The mathematical development so far supports several ways of proving properties of affine systems of ODEs in Isabelle/HOL. In this section, we discuss various use cases of our components, including their limitations. Our classification depends on whether users know a solution to an affine system

$$X'\,t = A\,t \cdot X\,t + B\,t. \tag{1}$$

In the autonomous case $A\,t = A$, it also depends on the diagonalisability of A.

Firstly, users may want to certify that a function $\varphi^f : T \to S$ solves system (1), with $T \subseteq \mathbb{R}$ and $S \subseteq \mathbb{R}^n$. If they formalise equation (1) in Isabelle by substituting φ^f for X, then they can use tactic *poly-derivatives* to check that both sides of the equation reduce to the same expression as in lemma *has-vderiv-on-sq-mtx-linear* of Section 4.

Alternatively, users might want to relate two different characterisations φ_1^f, φ_2^f of the solution to an IVP $X'\,t = A\,t \cdot X\,t + B\,t$ with $X\,t_0 = s$. Using uniqueness, as provided by Picard-Lindelöf's theorem, they can convert easily between one characterisation to the other by firstly formalising that $\varphi_1^f\,t = \varphi_2^f\,t$ for all $t \in T \subseteq \mathbb{R}$. Our most general uniqueness lemma is *picard-lindeloef.unique-solution* of Section 3, but we have derived specific instances for the autonomous affine case $X'\,t = A \cdot X\,t + B$, linear case $X'\,t = A \cdot X\,t$, and the case when $t_0 = 0$ and φ^f is the general solution in terms of the matrix exponential.

A particular case where our uniqueness lemmas are useful is in $\mathsf{d}\mathcal{L}$-style verification of hybrid systems [21]. Its postconditions must hold for all the points in all solutions of an IVP or along all points of the flow [7,20,19]. Uniqueness therefore simplifies the verification procedure by restricting it to only one solution. We further explore this in Sections 7 and 8.

So far we have covered cases where users have a solution to system (1). Otherwise, our formalisation provides the general solution for autonomous affine systems $X'\,t = A \cdot X\,t + B$ in terms of the matrix exponential with lemmas *has-vderiv-on-sq-mtx-affine* and *has-vderiv-on-sq-mtx-linear* for the type of matrices *'n sq-mtx* of Section 4.

A formalisation of the general solution for the non-autonomous case is harder and left for future work. The difficulty resides in that one usually needs a solution to the associated linear system $X'\,t = A\,t \cdot X\,t$ [8] which can be difficult to find. With this solution, one can use the variation of parameters method to generate the corresponding solution to the affine system (1) [10]. An alternative approach consists in finding the solution to an equivalent linear system of ODEs in one more dimension. Indeed, the system

$$\begin{pmatrix} X'\,t \\ 1' \end{pmatrix} = \begin{pmatrix} (A\,t) & (B\,t) \\ 0^\top & 0 \end{pmatrix} \cdot \begin{pmatrix} X\,t \\ 1 \end{pmatrix} = \begin{pmatrix} A\,t \cdot X\,t + B\,t \\ 0 \end{pmatrix}$$

subsumes system (1) in its first entry. Here, $0^\top$ is a transposed zero-vector with the same length as $B\,t$ and $0, 1 \in \mathbb{R}$. Thus, if the solution to any of those two linear systems is known, our components are the basis for solving the affine system of interest.

Yet, working with the general solution might require the simplifications at the end of Section 2 via the diagonalisation $A = P^{-1}DP$ provided that the associated diagonal D and change of basis P matrices are known. In Section 4, Lemma *exp-scaleR-diagonal1* includes the simplification based on this proviso.

If matrices are non-diagonalisable, the general solution is left for future work as we need Jordan Normal forms to formalise this result [22]. However, many non-diagonalisable matrices can still be tackled with our components. An example of this are nilpotent matrices that satisfy the condition $A^k = 0$ for some $k \geq 0$. If k is small, the exponential is still easy to characterise as $\exp A = \sum_{i=0}^{k} \frac{1}{i!} A^i$. We exemplify how to use our components in such cases in Sections 6 and 8.

Finally, users may want to work with the flow $\varphi\, t\, s$ of the autonomous affine system of ODEs $X'\, t = A \cdot (X\, t) + B$ with initial condition $X\, 0 = s$. For this, they must use the type *'n sq-mtx* and the locale *local-flow* applied to the general solution. With this locale, users can change between characterisations of the flow via uniqueness theorems or use the lemma that formalises the monoid-action behaviour of the flow over $\mathbb{R}$.

6 Examples

In this section, we analyse two systems of ODEs and characterise their flows with our Isabelle formalisation. In the first example, we diagonalise the associated matrix and use this to describe the general solution more conveniently. For the second one, we construct the general solution by computing the associated matrix exponential directly.

Example 1 (Diagonalizable matrix). An ubiquitous second order ODE in physics and engineering is

$$x''\, t = a(x\, t) + b(x'\, t).$$

Fixing $a = -\frac{k}{m}$ and $b = -\frac{d}{m}$ yields the damped harmonic oscillator equation of a mass m attached to a spring with constant k sliding along a horizontal track with damping factor d. Alternatively, with $a = \frac{1}{CL}$ and $b = \frac{R}{L}$, we obtain an ODE for modelling the current of a closed circuit where a resistor (R), inductor (L) and capacitor (C) are in series with a source of constant voltage [10].

Introducing variable y such that $x'\, t = y\, t$ yields the linear system

$$\begin{pmatrix} x'\, t \\ y'\, t \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ a & b \end{pmatrix} \cdot \begin{pmatrix} x\, t \\ y\, t \end{pmatrix}.$$

In Isabelle, we use our function *mtx* that turns lists into the type *'n sq-mtx*.

```
abbreviation mtx-hOsc :: real ⇒ real ⇒ 2 sq-mtx (A)
  where A a b ≡ mtx
   ([0, 1] #
    [a, b] # [])
```

We use WolframAlpha® to diagonalise it: generate its eigenvalues ι_1, ι_2 and its change of basis matrix P. Then we formalise these entities and certify the diagonalisation as follows.

```
abbreviation mtx-chB-hOsc :: real ⇒ real ⇒ 2 sq-mtx (P)
  where P a b ≡ mtx
   ([a, b] #
    [1, 1] # [])

lemma mtx-hOsc-diagonalizable:
  defines ι1 ≡ (b − sqrt (b^2+4∗a))/2 and ι2 ≡ (b + sqrt (b^2+4∗a))/2
  assumes b^2 + a ∗ 4 > 0 and a ≠ 0
  shows A a b = P (−ι2/a) (−ι1/a) ∗ (diag i. if i = 1 then ι1 else ι2) ∗ (P (−ι2/a) (−ι1/a))^−1
  ⟨proof⟩
```

Integrating a computer algebra system directly into Isabelle, so that inputs and certification are done automatically, is beyond our research goals in this article. Although we omit the proof, it is a simple 4-line script thanks to the addition of our lemmas about standard matrix operations to Isabelle's simplifier.

Finally, we use this diagonalisation to compute the general solution of the ODEs generated by $A\ a\ b$ and instantiate *local-flow-sq-mtx-linear* to this result.

```
lemma mtx-hOsc-solution-eq:
  defines ι1 ≡ (b − sqrt (b^2+4∗a))/2 and ι2 ≡ (b + sqrt (b^2+4∗a))/2
  defines Φ t ≡ mtx (
   [ι2∗exp(t∗ι1) − ι1∗exp(t∗ι2),      exp(t∗ι2)−exp(t∗ι1)]#
   [a∗exp(t∗ι2) − a∗exp(t∗ι1), ι2∗exp(t∗ι2)−ι1∗exp(t∗ι1)]#[])
  assumes b^2 + a ∗ 4 > 0 and a ≠ 0
  shows P (−ι2/a) (−ι1/a) ∗ (diag i. exp (t ∗ (if i=1 then ι1 else ι2))) ∗ (P (−ι2/a) (−ι1/a))^−1
= (1/sqrt (b^2 + a ∗ 4)) ∗R (Φ t)
  ⟨proof⟩

lemma local-flow-mtx-hOsc:
  defines ι1 ≡ (b − sqrt (b^2+4∗a))/2 and ι2 ≡ (b + sqrt (b^2+4∗a))/2
  defines Φ t ≡ mtx (
   [ι2∗exp(t∗ι1) − ι1∗exp(t∗ι2),      exp(t∗ι2)−exp(t∗ι1)]#
   [a∗exp(t∗ι2) − a∗exp(t∗ι1), ι2∗exp(t∗ι2)−ι1∗exp(t∗ι1)]#[])
  assumes b^2 + a ∗ 4 > 0 and a ≠ 0
  shows local-flow ((∗V) (A a b)) UNIV UNIV (λt. (∗V) ((1/sqrt (b^2 + a ∗ 4)) ∗R Φ t))
  ⟨proof⟩
```

Our matrix operation lemmas make the proof of both results easy to tackle for the experimented Isabelle user. The last lemma yields an automated certification of the uniqueness and the monoid-action behavior of this flow. These results will be useful later in the verification of a simple hybrid program.

Example 2 (Non-diagonalizable matrix). To derive the equations for constantly accelerated motion in one dimension, we start with the ODE $x'''\,t = 0$. This is equivalent to the linear system

$$\begin{pmatrix} x'\,t \\ v'\,t \\ a'\,t \end{pmatrix} = \begin{pmatrix} 0\,1\,0 \\ 0\,0\,1 \\ 0\,0\,0 \end{pmatrix} \cdot \begin{pmatrix} x\,t \\ v\,t \\ a\,t \end{pmatrix},$$

where x, v and a represent the position, velocity and acceleration of the motion. Although the matrix in this system is non-diagonalisable, it is nilpotent as formalised below.

```
abbreviation mtx-cnst-acc :: 3 sq-mtx (K)
  where K ≡ mtx (
  [0,1,0] #
  [0,0,1] #
  [0,0,0] # [])

lemma powN-scaleR-mtx-cnst-acc: n > 2 ⟹ (t *R K)^n = 0
  ⟨proof⟩
```

We can use this fact to obtain the general solution and the kinematics equations for constantly accelerated motion with initial state $s = (s\$1, s\$2, s\$3)$.

```
lemma exp-mtx-cnst-acc: exp (t *R K) = ((t *R K)²/R 2) + (t *R K) + 1
  unfolding exp-def apply(subst suminf-eq-sum[of 2])
  using powN-scaleR-mtx-cnst-acc by (simp-all add: numeral-2-eq-2)

lemma exp-mtx-cnst-acc-vec-mult-eq: exp (t *R K) *V s =
  vector [s$3 * t^2/2 + s$2 * t + s$1, s$3 * t + s$2, s$3]
  ⟨proof⟩
```

Here, *vector* is a function that turns lists into vectors. From this, a simple instantiation shows that the kinematics equations describe the flow of the ODE.

```
lemma local-flow-mtx-cnst-acc:
  local-flow ((*V) K) UNIV UNIV (λt s. ((t *R K)²/R 2 + (t *R K) + 1) *V s)
  using local-flow-sq-mtx-linear[of K] unfolding exp-mtx-cnst-acc .
```

Throughout this section, formalisation and proofs are relatively simple. This is because our lemmas about matrix operations, if added to Isabelle's simplifier, improve proof automation.

7 Applications in Hybrid Program Verification

To illustrate an application of our formalisation, we use our Isabelle verification components for hybrid programs [17]. This approach starts with an algebra $(K, +, ;, 0, 1, ^*)$ for simple while-programs that also supports a boolean subalgebra $(B, +, ;, 0, 1, \neg)$ of tests such as a Kleene algebra with tests or a modal Kleene algebra [15,4]. With the interpretation of elements of K as programs, $+$ as nondeterministic choice, ; as sequential composition, * as finite iteration, and 0 and 1 as the aborting and ineffective programs respectively, the equations

$$\textbf{if } p \textbf{ then } \alpha \textbf{ else } \beta = p;\alpha + \neg p;\beta,$$
$$\textbf{while } p \textbf{ do } \alpha = (p;\alpha)^*;\neg p$$

model the behaviour of while-programs. These algebras allow us to write correctness specifications via Hoare-triples $\{-\} - \{-\}$ or weakest liberal preconditions wlp [9,3]. This means that we can derive the rules of Hoare-logic and/or those of the wlp-calculus. In Isabelle, this approach accelerates the verification process as our Kleene algebra components automatically generate domain-specific conditions by handling the program-structure without intervention from the user.

Moreover, these algebras have *state transformer* models where elements of the algebra are interpreted as functions of type $S \to \mathcal{P}\, S$ for a given set S. In this setting, Kleisli composition $(f \circ_K g)\, s = \bigcup\{g\, s' \mid s' \in f\, s\}$ interprets ;, + is pointwise union $\lambda\, s.\ f\, s \cup g\, s$, 0 is $\lambda\, s.\ \emptyset$, 1 is the Kleisli unit $\eta_S\, s = \{s\}$, and * is $f^{*_K}\, s = \bigcup\{f^n\, s \mid n \geq 0\}$, where $f^0 = \eta_S$ and $f^{n+1} = f^n \circ_K f$ [19].

Given a finite set of program variables V, the isomorphism between $\mathbb{R}^n$ and $\mathbb{R}^V$ allows us to work in the state transformer semantics of $S \subseteq \mathbb{R}^V$, effectively giving us hybrid stores. Defining $f[a \mapsto b]\, a = b$ and $f[a \mapsto b]\, t = f\, t$ if $t \neq a$, the function $\lambda\, s.\ \{s[x \mapsto e\, s]\}$ is a state transformer. It maps a store $s \in S$ to the singleton of that store with variable $x \in V$ updated to $e\, s$, for $e : S \to \mathbb{R}$. In particular, it models program assignments

$$(x := e)\, s = \{s[x \mapsto e\, s]\}.$$

Similarly, for an interval $U \subseteq T$ such that $0 \in U$, the *orbit* map $\gamma^\varphi : S \to \mathcal{P}\, S$ defined by $\gamma^\varphi\, s = \mathcal{P}\, \varphi_s^f\, U$ is a state transformer. It sends each $s \in S$ to the set of all the points in the trajectory φ_s^f for the IVP induced by f and $(0, s)$. However, for modelling boundary conditions, an alternative *G-guarded* version is better. For predicate $G : S \to \mathbb{B}$, we use the *evolution command* state transformer

$$(x' = f \,\&\, G)\, s = \{\varphi_s^f\, t \mid t \in U \wedge (\forall \tau \in \downarrow t.\ G\, (\varphi_s^f\, \tau))\},$$

where "$x' =$" is syntactic sugar to resemble ODEs, and $\downarrow t = \{\tau \in U \mid \tau \leq t\}$.

By adding assignments and evolution commands to the language of these algebras of programs, we get *hybrid programs*. In particular, we also have correctness specifications for these commands

$$\{\lambda\, s.\ Q\, (s[x \mapsto e\, s])\}\ x := e\ \{Q\},$$
$$\{\lambda\, s \in S.\ \forall t \in U.\ (\forall \tau \in \downarrow t.\ G\, (\varphi_s^f\, \tau)) \to Q\, (\varphi_s^f\, t)\}\ x' = f \,\&\, G\ \{Q\}.$$

The above definition of evolution commands requires uniqueness of the solution to the IVP $X' = f\,(t, X\, t)$ and $X\, 0 = s$. For a more general definition where this is not needed see [19]. Yet, affine and linear systems have unique solutions for specific IVPs. Thus, our formalisation of affine and linear systems is compositional with respect to the verification style described in [19,7].

8 Verification Examples

In this section, we verify two simple hybrid programs using the components of [19] and our formalisation of linear systems of ODEs. Both verifications follow directly from our results in Section 6.

Example 3 (Overdamped door-closing mechanism). We use the system of ODEs

$$\begin{pmatrix} x'\, t \\ y'\, t \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ a & b \end{pmatrix} \cdot \begin{pmatrix} x\, t \\ y\, t \end{pmatrix} = (A\, a\, b) \cdot \begin{pmatrix} x\, t \\ y\, t \end{pmatrix},$$

with $a = -\frac{k}{m}$ and $b = -\frac{d}{m}$ to model a damped harmonic oscillator as described in Example 1. The expression $b^2 + 4 \cdot a$ dictates the behaviour of the system. If

$b^2+4\cdot a < 0$, the damping factor is too big and there is no oscillation. Otherwise, the oscillation continues. Overdamping is a desired property of some oscillators inside door mechanisms where the engineer does not want the doors to slam or open on the opposite side.

We use Isabelle's *s$1* and *s$2* to formalise respectively the position (x) and velocity (y) of one of these door-oscillators. We represent a closed door with the equation $s\$1 = 0$. Hence, an open door immediately after being pushed by a person corresponds to the conjunction $s\$1 > 0 \wedge s\$2 = 0$. We can prove that once this happens, the door will never open on the opposite side, that is $s\$1 \geq 0$, if its oscillator is overdamped.

```
lemma overdamped-door:
  assumes b² + a ∗ 4 > 0 and a < 0 and b ≤ 0 and 0 ≤ t
  shows PRE (λs. s$1 = 0)
  HP (LOOP
      (λs. {s. s$1 > 0 ∧ s$2 = 0});
      (x´=(∗V) (A a b) & G on {0..t} UNIV @ 0)
    INV (λs. 0 ≤ s$1))
  POST (λs. 0 ≤ s $ 1)
  apply(rule fbox-loopI, simp-all add: le-fun-def)
  apply(subst local-flow.fbox-g-ode-ivl[OF local-flow-mtx-hOsc[OF assms(1)]])
  using assms apply(simp-all add: le-fun-def fbox-def)
  unfolding sq-mtx-scaleR-eq UNIV-2 sq-mtx-vec-mult-eq
  by (clarsimp simp: overdamped-door-arith)
```

Notation *PRE P HP X POST Q* is syntactic sugar for the Hoare triple $\{P\}X\{Q\}$, meaning that if the system starts satisfying precondition P, then after the execution of the hybrid program X, postcondition Q will hold. In the lemma above we assume $a < 0$ and $b \geq 0$ because $a = -\frac{k}{m}$, $b = -\frac{d}{m}$ and, in physics, the constants k, d and m are often positive. The condition $t \geq 0$ guarantees the verification for a positive lapse of time.

The hybrid program is the finite iteration of a discrete door-opening, modelled by the state transformer $\lambda\, s.\ \{s\$1 > 0 \wedge s\$2 = 0\}$, followed by the ODE $x'\,t = A \cdot (x\,t)$. The loop-invariant of this iteration is the same as the desired postcondition. As we do not deal with boundary conditions, we use variable G for the guard of the evolution command. The first two lines in the proof of this lemma apply the Hoare-rules for loops and evolution commands respectively. The remaining lines simplify the emerging proof obligations.

Example 4 (Automatic docking). A space ship is aligned with its docking station d and approaching it with velocity $v_0 > 0$. The ship needs to stop exactly at d and its current position is x_0, where $d > x_0$. In order to do this, the ship calculates that it needs a constant deceleration of $a = -\frac{v_0^2}{2(d-x_0)}$. Its motion follows the system of Example 2,

$$\begin{pmatrix} x'\,t \\ v'\,t \\ a'\,t \end{pmatrix} = \begin{pmatrix} 0\;1\;0 \\ 0\;0\;1 \\ 0\;0\;0 \end{pmatrix} \cdot \begin{pmatrix} x\,t \\ v\,t \\ a\,t \end{pmatrix} = K \cdot \begin{pmatrix} x\,t \\ v\,t \\ a\,t \end{pmatrix}.$$

We formalise the position, velocity and acceleration of the ship with state $s = (s\$1, s\$2, s\$3)$ and its discrete behaviour with an assignment of $s\$3$ to the value of the safe acceleration. Under these assumptions, we need to guarantee that the ship will stop ($s\$2 = 0$) if and only if its position coincides with d ($s\$1 = d$). The formalisation is shown below.

```
lemma docking-station-arith:
  assumes (d::real) > x and v > 0
  shows (v = v^2 * t / (2 * d − 2 * x)) ⟷ (v * t − v^2 * t^2 / (4 * d − 4 * x) + x = d)
  ⟨proof⟩

lemma docking-station:
  assumes d > x0 and v0 > 0
  shows PRE (λs. s$1 = x0 ∧ s$2 = v0)
  HP ((3 ::= (λs. −(v0 ^2/(2∗(d−x0))))); x´=(∗V) K & G)
  POST (λs. s$2 = 0 ⟷ s$1 = d)
apply(clarsimp simp: le-fun-def local-flow.fbox-g-ode[OF local-flow-sq-mtx-linear[of K]])
  unfolding exp-mtx-cnst-acc-vec-mult-eq using assms by (simp add: docking-station-arith)
```

In the proof of this hybrid program, as before, the first line applies the Hoare-rule for evolution commands. The second line simplifies the emerging proof obligation by calling the lemma *docking-station-arith* which we proved separately.

9 Conclusion

We have developed new mathematical components for affine and linear systems of ODEs that improve a modular semantic framework for verification of hybrid programs based on Kleene algebras [19] in Isabelle. These extend the tactics of the framework and simplify the verification procedure by eliminating uniqueness and existence requirements for solutions to these systems of ODEs.

As many systems in physics and engineering are linear, our work impacts a wide range of applications for our verification components. Furthermore, our extension showcases the advantages of using a general purpose proof assistant. It demonstrates that our components can handle exponentiation and other transcendental functions beyond first-order real arithmetic, to which traditional deductive verification of hybrid programs is confined [21].

Our work is also an extension to Isabelle's HOL-Analysis library as it adds lemmas from linear algebra and the theory of ODEs. Previous formalisations in Isabelle/HOL intersect with our components in both fields, but none of them combines them. For instance, there are two formalisations of Jordan Normal forms in Isabelle's archive of formal proofs (AFP) [23,6]. They have been combined and made executable in their exported versions to Standard ML or Haskell [5,6]. An integration of this work and our verification components to handle more than just diagonalisable matrices is a pursuable endeavour. On the other hand, there is much work in extending Isabelle's libraries for ODEs[13,12,11]. The AFP contains a definition for bounded linear operators and a proof that linear systems expressed with these have unique solutions [13]. However, the affine version of this result has not yet been formalised and it requires further work to make it compatible with the type of vectors *real^'n* and our components.

Yet, much work remains to make this approach widely-adoptable in current practice. The general solution for non-autonomous linear systems of ODEs using resolvent matrices remains to be formalised in a proof assistant. Also, our work can only certify diagonalisations and solutions, but the generation of these is left to the user. An alternative approach would automate our procedure in Example 1. That is, a computer algebra system (CAS) would obtain the solution (or diagonalisation) and another tool would generate the Isabelle theory with a certification of the solution provided. This is left for future work.

Acknowledgements. The author wishes to thank the reviewers for their insightful comments. He also thanks Georg Struth, Harsh Beohar, Rayna Dimitrova, Kirill Bogdanov and Michael Foster for discussions.

References

1. Althoff, M., Bak, S., Forets, M., Frehse, G., Kochdumper, N., Ray, R., Schilling, C., Schupp, S.: ARCH-COMP19 category report: Continuous and hybrid systems with linear continuous dynamics. In: ARCH19. pp. 14–40 (2019)
2. Alur, R.: Formal verification of hybrid systems. In: EMSOFT 2011. pp. 273–278. ACM (2011)
3. Armstrong, A., Gomes, V.B.F., Struth, G.: Building program construction and verification tools from algebraic principles. Formal Aspects of Computing **28**(2), 265–293 (2016)
4. Desharnais, J., Möller, B., Struth, G.: Algebraic notions of termination. Logical Methods in Computer Science **7**(1) (2011)
5. Divasón, J., Aransay, J.: Gauss-Jordan algorithm and its applications. Archive of Formal Proofs (2014)
6. Divasón, J., Kunčar, O., Thiemann, R., Yamada, A.: Perron-Frobenius theorem for spectral radius analysis. Archive of Formal Proofs (2016)
7. Foster, S., y Munive, J.J.H., Struth, G.: Differential Hoare logics and refinement calculi for hybrid systems with Isabelle/HOL. In: RAMiCS 2020[postponed]. pp. 169–186 (2020)
8. Friedland, B., Director, S.W.: Control Systems Design: An Introduction to State-Space Methods. McGraw-Hill Higher Education (1985)
9. Gomes, V.B.F., Struth, G.: Modal Kleene algebra applied to program correctness. In: FM 2016. LNCS, vol. 9995, pp. 310–325 (2016)
10. Hirsch, M.W., Smale, S., Devaney, R.L.: Differential equations, dynamical systems, and linear algebra. Academic Press (1974)
11. Immler, F.: Formally verified computation of enclosures of solutions of ordinary differential equations. In: NFM 2014. LNCS, vol. 8430, pp. 113–127. Springer (2014)
12. Immler, F., Hölzl, J.: Numerical analysis of ordinary differential equations in Isabelle/HOL. In: ITP 2012. LNCS, vol. 7406, pp. 377–392. Springer (2012)
13. Immler, F., Hölzl, J.: Ordinary differential equations. Archive of Formal Proofs (2012), https://www.isa-afp.org/entries/Ordinary_Differential_Equations.shtml
14. Jeannin, J., Ghorbal, K., Kouskoulas, Y., Schmidt, A., Gardner, R., Mitsch, S., Platzer, A.: A formally verified hybrid system for safe advisories in the next-generation airborne collision avoidance system. STTT **19**(6), 717–741 (2017). https://doi.org/10.1007/s10009-016-0434-1, https://doi.org/10.1007/s10009-016-0434-1
15. Kozen, D.: Kleene algebra with tests. ACM TOPLAS **19**(3), 427–443 (1997)

16. Loos, S.M., Platzer, A., Nistor, L.: Adaptive cruise control: Hybrid, distributed, and now formally verified. In: FM 2011. LNCS, vol. 6664, pp. 42–56. Springer (2011)
17. Huerta y Munive, J.J.: Verification components for hybrid systems. Archive of Formal Proofs (2019), https://www.isa-afp.org/entries/Hybrid_Systems_VCs.html
18. Huerta y Munive, J.J.: Matrices for odes. Archive of Formal Proofs (2020), https://www.isa-afp.org/entries/Matrices_for_ODEs.html
19. Huerta y Munive, J.J., Struth, G.: Predicate transformer semantics for hybrid systems: Verification components for Isabelle/HOL (2019), arXiv:1909.05618 [cs.LO]
20. Huerta y Munive, J.J., Struth, G.: Verifying hybrid systems with modal Kleene algebra. In: RAMiCS 2018. LNCS, vol. 11194, pp. 225–243. Springer (2018)
21. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer (2018)
22. Teschl, G.: Ordinary Differential Equations and Dynamical Systems. AMS (2012)
23. Thiemann, R., Yamada, A.: Matrices, Jordan normal forms, and spectral radius theory. Archive of Formal Proofs (2015)