Differential-Algebraic Dynamic Logic for KeYmaeraX

Benjamin Lim (wmlim - 15624)
Yao Chong Lim (yaochonl - 15824)

December 8, 2018

Abstract

Modelling uncertainty is an important part of hybrid systems verification. In this project, we tackle this problem by implementing Differential-Algebraic Dynamic Logic, an extension of Differential Dynamic Logic to handle existentially quantified differential equations, into KeYmaeraX by constructing a uniform substitution calculus for it, and use it to demonstrate proving facts about models with built-in imprecision.

1 Introduction and Motivation

When modelling cyber-physical systems, we are often faced with the problem of accounting for irregularities, imprecision and perturbation. Indeed, it is often impossible and impractical to guarantee the accuracy of any measurement of a physical quantity, or ensure that physical devices behaves in exactly the same way every time. Due to both imprecision of measurement, as well as factors external to the abstracted model, it may be the case that a carefully constructed proof of safety for a cyber-physical system may fail if it does not somehow account for these factors. It is therefore desirable to come up with models that are robust to such inadequacies.

There are multiple approaches that can be taken to handle these issues, which we will briefly touch on the related work section. For our project, we have decided to modify the KeYmaeraX [5] hybrid systems theorem prover implementing Differential Dynamic Logic (d$\mathcal{L}$) [14] to implement an extended logic, Differential-Algebraic Dynamic Logic (dA$\mathcal{L}$) [10], which allows for explicit representation and logical manipulation of such perturbations as part of a rich class of hybrid systems.

In this paper, we introduce dA$\mathcal{L}$, a enriched version of d$\mathcal{L}$ with quantified differential evolution, and provide some illustrative examples. Following which, we describe our attempts at modernizing the dA$\mathcal{L}$ proof system to bring it in line with the presentation of more modern d$\mathcal{L}$ proof rules, and backing it with a uniform substitution calculus [12] in order to minimize

the soundness-critical base. We demonstrate the use of our proof system on the given examples. Our implementation of dA$\mathcal{L}$ is then described in detail, after which we demonstrate using KeYmaeraX to prove the examples. Finally, we discuss our takeaway from this project and possible future work.

2 Related Work

Analyzing hybrid systems in the presence of imprecision or perturbation is a well-studied problem. Stochastic hybrid systems include stochastic differential equations, where the system evolves according to stochastic processes [11]. Stochastic differential dynamic logic (Sd$\mathcal{L}$) extends d$\mathcal{L}$ to work with stochastic hybrid systems.

Hybrid programs can also be analyzed through simulation and approximation to compute reachable end states after a run of a hybrid program. For example, some approaches involve determining possible end states after evolving a differential system given uncertainty in the initial state. This can be accomplished using interval reasoning by narrowing possible output states using interval constraint propagation [3][15] and computing approximate solutions of ODEs to bound the set of reachable states after the evolution of the ODE system [1][8][9]. Bounded model checking [2] is another simulation-based method to compute reachable states for a hybrid program. In this project, we use invariant properties of differential inequalities like in d$\mathcal{L}$ to prove desired properties of the hybrid systems.

An alternative approach is to relax the requirement for total satisfiability to instead show δ-satisfiability, that is, satisfiability up to a small perturbation of the variable values. This is the approach taken by the tool dReal [7][6]. In this project, we focus only on exact satisfiability.

Hybrid games extend hybrid systems to model games between two players. However, the continuous dynamics still stay constant, like in normal hybrid systems. On the other hand, differential games model continuous interaction between the players, allowing both to control and change the dynamics of the system. Differential hybrid games are a combination of differential games and hybrid games, allowing more complex systems to be modelled. dG$\mathcal{L}$ is an extension of d$\mathcal{L}$ to deal with differential hybrid games [13]. It turns out that dA$\mathcal{L}$ is subsumed by dG$\mathcal{L}$. However, for certain systems, expressing their models in terms of dA$\mathcal{L}$ may turn out to be more natural and expressive.

3 Differential-Algebraic Dynamic Logic

dA$\mathcal{L}$, as described in [10] is a conservative extension of d$\mathcal{L}$, i.e formulas in d$\mathcal{L}$ may be (quite trivially) embedded into dA$\mathcal{L}$ without changing their semantics. In this section, we introduce its formulation and proof rules.

3.1 Syntax and Semantics

The syntax of dA$\mathcal{L}$ is more or less the same as that of d$\mathcal{L}$, with the exception of allowing for a more flexible description of differential equations.

To see how it arises, consider the differential system $x' = -y, y' = x, t' = 1 \,\&\, t \leq 10$ as expressed in d$\mathcal{L}$. This differential system may be equivalently regarded as a conjunction of differential and non-differential *constraints* on the evolution of the system that hold at every point of time, i.e. $x' = -y \wedge y' = x \wedge t' = 1 \wedge t \leq 10$. Similarly, we may also consider other constraints or combinations of constraints, such as $x' \leq -y \wedge y' \geq x \wedge t' = 1 \wedge t \leq 10$ (subject to certain restrictions).

Finally, we also permit existential quantifiers over differential constraints. For example, the differential-algebraic constraint $\exists\omega.(x' = -\omega y, y' = \omega x)$ would be read as "for every time point in the differential evolution of the system, there always exists a ω such that $x' = -\omega y$ and $y' = \omega x$". This could be interpreted as the circular motion of an object with an *indeterminate* and possibly *constantly changing* angular velocity ω. This is what allows dA$\mathcal{L}$ to easily and obviously capture the sort of imprecision that we wish to handle in our hybrid systems.

More formally, a *differential-algebraic constraint* or *differential-algebraic system* (abbreviated as *DA-constraint* or *DASystem*) is syntactically defined as:

$$\begin{aligned} \mathit{DASystem} ::= {} & \mathit{Formula} \\ & \mid \mathit{VariableRestrict}' \bowtie \mathit{Polynomial} \\ & \mid \mathit{DASystem} \wedge \mathit{DASystem} \\ & \mid \mathit{DASystem} \vee \mathit{DASystem} \\ & \mid \exists \mathit{Variable} \,.\, \mathit{DASystem} \end{aligned}$$

where *Formula* is any formula that does not contain differentials, *VariableRestrict* is any variable that is not bound by some quantifier above in the same differential-algebraic system, while *Variable* is any variable, and *Polynomial* is a polynomial that does not mention any variables primed. Finally, $\bowtie$ is one of the comparison operators $=, <, >, \leq, \geq$.

Note that in this definition, we do not allow negative DA-constraints like $x' \neq 7$. This is because such a constraint has effectively no semantic value (as all it says is that the rate of change of x' is never 7 at any point in time). It is possible to handle these cases but they are not very useful. Similarly, the universal quantifier is excluded to avoid constraints such as $\forall c.x' = c$ which are not very enlightening as it is not possible for x' at any one time point to take on all values c.

The semantics for dA$\mathcal{L}$ is likewise mostly the same as d$\mathcal{L}$, with the only difference being that instead of defining it in terms of (unique) *solutions* of the differential system as in d$\mathcal{L}$, we define it in terms of (nonunique) *state flows* $\varphi(t)$ mapping time points to states,

where φ evolves continuously component-wise on the variables that are modified by the DA-constraint (and leaves everything else fixed) in such a way that the constraint always holds throughout its (finite length) evolution. Similar to the case in d$\mathcal{L}$, there is a degenerate case of a zero-length run whose handling is a technicality.

There is a small catch to the semantics of dA$\mathcal{L}$. Consider taking the DA-constraint, pulling out all existence quantifiers to the top, and converting it to disjunctive normal form. In this definition of the semantics, it is possible for the flow to switch control between disjunctive cases infinitely often in a finite time run. This leads to Zeno-like paradoxes which become tricky to navigate and are not realistically meaningful. We therefore restrict the semantics of DASystems to only *non-Zeno flows*, which are flows which only switch between disjunctive cases finitely often.

3.2 An Equivalent Reduced Core for dA$\mathcal{L}$

For our paper, we implement an equivalent core of dA$\mathcal{L}$ that is significantly smaller but equally expressive. As indicated in the last part of the previous section, we can convert a DASystem into an equivalent form by pulling out existential quantifiers to the top, and reducing to disjunctive normal form.

As proven in [10], it is also possible to eliminate repeated differential assignments like $x' = 3 \land x' = 4$, and bring all the individual DA-constraints down to the form $\exists \bar{y}.x' = f(x, \bar{y})$ for some variable x and some quantified variables $\bar{y}$, while maintaining the semantics of the DA-constraint. For instance, the semantics of the DA-constraints $x \leq y$ and $\exists c.(x = y - c \land c \geq 0)$ are semantically equivalent.

Furthermore, because of the restriction to non-Zeno flows noted in the previous section, we may replace disjunctions of DA-constraints $\{A \lor B\}$ with $(\{A\} \cup \{B\})^*$, since only finite switchings are allowed. After repeating these transformations until no longer possible, we can finally combine all non-differential constraints in each DASystem into a single non-differential constraint.

The upshot of all of this is that we are left with only needing to consider DA-constraints in the simplified form $\exists \bar{y}.(x_1' = \theta_1 \land x_2' = \theta_2 \land \cdots \land x_n' = \theta_n \land Q)$. We may use the regular comma syntax interchangeably with $\land$. We therefore attempt to implement these constructs (and their corresponding proof rules) into KeYmaeraX, as the sugared syntax is of secondary importance.

3.3 Examples

We present some examples of modelling with dA$\mathcal{L}$ to illustrate its use.

3.3.1 Perturbed Train Braking

Consider a simple model of a train braking to a halt before an obstacle, with some non-deterministic and constantly changing perturbation. We may model it (and its corresponding safety condition) as follows:

$$\begin{aligned} \phi &:= v^2 \leq 2(b-u)(m-z) \wedge b > u \wedge u \geq 0 \wedge l \geq 0 \\ dyn &:= \{\exists d.(z' = v, v' = -b + d \,\&\, -l \leq d \,\&\, d \leq u \,\&\, v \geq 0)\} \\ safe &:= z \leq m \\ train &:= \phi \rightarrow [dyn]\, safe \end{aligned}$$

The motion of the train located at point z and its velocity v is modeled with a DASystem that indicates a braking deceleration of $-b$ that is perturbed by an amount d between lower and upper bounds $-l$ and u. ϕ is the invariant that we will eventually prove and implies safety, and $safe$ is the safety condition that states that the train's position z is always before the obstacle position m. Then, as long as we can show that the invariant ϕ is initially true, we can show that the train halts before reaching the obstacle.

3.3.2 Disturbed Circular Motion

Consider another simple scenario where an object is in circular motion (in two dimensions) about the origin, but the x velocity of the object is always perturbed towards the y-axis, but the perturbation is unbounded. Still, we can prove that the object never leaves the confines of the circle, modelling it as follows:

$$x^2 + y^2 = 1 \rightarrow [\{\exists e.(x' = -y + e, y' = x \,\&\, xe \leq 0)\}]\, x^2 + y^2 \leq 1$$

Here x and y are the coordinates of the object and e is the disturbance. The interesting thing about this example is that, unlike the previous one, it is not immediately obvious (at least to us) how a similar property would be proven in plain d$\mathcal{L}$.

3.4 Proof System

The proof rules of dA$\mathcal{L}$ for handling DASystems are reminiscent of those for d$\mathcal{L}$. We present the critical proof rule for differential induction in dA$\mathcal{L}$, as seen in [10], below:

$$\frac{\vdash \forall^{\alpha}.\forall \bar{y}.(Q \rightarrow (P)'^{\,\theta_1}_{x'_1} \cdots ^{\theta_n}_{x'_n})}{[\{\exists \bar{y}.Q\}P] \vdash [\{\exists \bar{y}.(x'_1 = \theta_1, x'_2 = \theta_2, \cdots, x'_n = \theta_n \,\&\, Q)\}]\, P}\, DI_{old}$$

Here Q is a differential-free constraint (i.e. a domain constraint).

This rule says that if we wish to prove some invariant P of a DASystem, we may prove it if for any selection of $\bar{y}$, P holds true at the initial state if Q holds true (indicated by $[\{\exists \bar{y}.Q\}P]$, and for all variables bound (modified) by the DASystem ($\forall^\alpha$), for all choices of $\bar{y}$, if the domain constraint Q holds then the primed invariant holds. This is very similar to the regular differential induction proof rule, except that we also have to show safety regardless of what the variables $\bar{y}$ might be, since they could be anything at any point during the evolution of the DASystem.

4 Modernizing dA$\mathcal{L}$

As can be seen, the presentation of the dA$\mathcal{L}$ proof system is quite different from the modern presentation of the d$\mathcal{L}$ proof system, which can be seen in [14]. Apart from the differences in the syntactic appearance of the proof rules, the modern presentation (and actual implementation in KeYmaeraX) of d$\mathcal{L}$ is as a set of axioms forming a uniform substitution calculus (for details on uniform substitution, see [12]). Implementing proof systems as uniform calculi (instead of directly as proof rules) significantly reduces the soundness-critical surface of the prover, and so we attempted to modernize dA$\mathcal{L}$ by presenting in terms of a uniform substitution calculus.

Note: In the following sections, we use the differential program constant c in axioms and rules to denote arbitrary numbers of differential constraints.

4.1 Trials of Uniformity

The approach that we took was to look at the current uniform axiom base for d$\mathcal{L}$, and attempt to transfer over the same ideas to dA$\mathcal{L}$. We identified the critical axioms that we needed to create in order to prove anything interesting (without too much hassle), namely a differential weakening axiom, a differential cut axiom, and a differential invariance axiom.

As it turns out, analogous sound uniform substitution axioms for DASystems were not always straightforward to come up with. In the process of developing the axioms, we were met with several increasingly subtle false attempts, some of which we document below, as well as arguments as to why they are incorrect.

4.1.1 Cantrip of Minor False Weakening

While searching for a analogous weakening axiom, we tried the following rule:

$$[\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftrightarrow [\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, (Q(x,\bar{y}) \rightarrow P(x))$$

Counterexample:

$$[\{\exists y.(x' = 1 \,\&\, y \geq 0)\}]\, \mathit{false} \leftrightarrow [\{\exists y.(x' = 1 \,\&\, y \geq 0)\}]\, (y \geq 0 \rightarrow \mathit{false})$$

The left-hand-side is always false no matter the initial state, since there always exists some value of y that satisfies the domain constraint $y \geq 0$, so the zero-length evolution is always permitted. The right-hand-side is true in states where y is initially less than zero, because the value of y after the evolution of the DASystem has not changed, as the y within the DASystem is local to the DASystem.

The error is fairly straightforward to spot, as it is due to the fact that the $\bar{y}$ that was once bound inside becomes free outside. The axiom can be fixed by simply quantifying over the free $\bar{y}$.

4.1.2 Curse of False Induction

We also required a differential induction axiom. Generally, we try to leave the differential system around, as it plays the role of the $\forall^\alpha$ by binding the bound variables in the ODE as well as allowing the use of the differential effect axiom to extract out the substitutions for the primed variables. Hence the following was a natural try:

$$\begin{aligned}([\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x) &\leftrightarrow \forall \bar{y}.\, [?Q(x,\bar{y})]\, P(x)) \\ &\leftarrow \forall \bar{y}.\, [\{c \,\&\, Q(x,\bar{y})\}]\, (P(x))'\end{aligned}$$

Counterexample:

$$\begin{aligned}([\{\exists y.(x' = y, z' = -1 \,\&\, y \geq z)\}]\, x \geq 0 &\leftrightarrow \forall y\, [?y \geq z]\, x \geq 0) \\ &\leftarrow \forall y.\, [\{x' = y, z' = -1 \,\&\, y \geq z\}]\, (x \geq 0)'\end{aligned}$$

Intuitively, the problem with the axiom attempt above is that the premise only assures that the differential invariant holds while the initial value of the quantified variable y satisfies the constraints at the initial state. In particular, the range of the quantified variables y is restricted to the range allowed by the initial constraints, no matter how long the differential system evolves. However, in a DASystem, the value of the quantified variables can change continuously, and the constraints can change as well. This means that the range of permissible values for the quantified variables that satisfy the constraints can change over time, which affects the truth of the premise.

This is shown in the counterexample. The premise is true in states where $z \geq 0$, since it ensures that $x' = y \geq 0$. However, in the DASystem, z eventually becomes a negative number, allowing y to take on negative values as well, decreasing x and breaking the postcondition.

4.1.3 Summon Greater False Effect

An alternative way to look at the issue with the previous induction axiom is that in order to prove the conclusion of DI_{old} (in section 3.4), we need to prove the induction invariant for at least all states reachable by the DASystem, but by reducing it to a universally quantified ODE the set of reachable states change.

Hence, any axiom which changes DASystem into an ODE is likely to be wrong. Therefore, the clear follow up to the previous induction axiom would be the following:

$$\begin{aligned}([\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftrightarrow \forall \bar{y}.\, [?Q(x,\bar{y})]\, P(x)) \\ \leftarrow [\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, (P(x))'\end{aligned}$$

Assuming we have a weakening axiom, we are now left with the problem of getting out the substitutions from the DASystem, as we are no longer able to piggyback on the differential effect axioms of regular ODEs, which is what our previous formulation was capable of doing. Naturally the first idea that comes to mind would be to implement some sort of differential effect axiom. A differential effect axiom would look something like the following:

$$[\{\exists \bar{y}.(x' = f(x,\bar{y}), c \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftrightarrow \forall \bar{y}.\, [\{\exists \bar{y}.(x' = f(x,\bar{y}), c \,\&\, Q(x,\bar{y}))\}]\, [\{x' := f(x,\bar{y})\}]\, P(x)$$

Coupled with a commutativity axiom for the constraints in a DASystem, this would let us extract all the differential effects from the DASystem, which would eventually let us substitute them into the postcondition. This is also analogous to the way in which it is implemented for regular ODE systems.

The curious thing about this axiom is that it is, in fact, true, but useless!

The reason why this is so is because it is too conservative. If your DASystem modifies multiple variables, then extracting all the effects will result in multiple quantifiers over the DASystem, each binding a single differential assignment. Then, the state space in which we would have to prove P true would be larger than what is actually needed, since, for example, the same variable y could be assigned to different values in different assignments. A differential effect-style axiom could still work, but in a rather messy way, by introducing a special marked quantifier for each DASystem. This would be very asthetically unappealing and make the codebase quite a bit larger.

4.2 A Uniform Substitution Calculus for dA$\mathcal{L}$

Instead, in order to finally solve this issue, we introduce a new axiom which we call 'Differential Algebraic Stutter'. We list our axioms below ($\bar{x}$ refers to the variables bound in c):

`DACbase differential cut`:

$$([\{\exists \bar{y}.(c \,\&\, Q(\bar{x},\bar{y}))\}]\, P(\bar{x},\bar{y}) \leftrightarrow [\{\exists \bar{y}.(c \,\&\, Q(\bar{x},\bar{y}) \wedge R(\bar{x}))\}]\, P(\bar{x},\bar{y})) \leftarrow [\{\exists \bar{y}.(c \,\&\, Q(\bar{x},\bar{y}))\}]\, R(\bar{x})$$

`DAIbase differential invariance`:

$$\begin{aligned}([\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftrightarrow \forall \bar{y}.\, [?Q(x,\bar{y})]\, P(x)) \\ \leftarrow [\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, (P(x))'\end{aligned}$$

`DAS differential stutter`:

$$[\{\exists \bar{y}.(c \,\&\, Q(\bar{x},\bar{y}))\}]\, P(\bar{x}) \leftrightarrow \forall \bar{y}.\, [\{\exists \bar{y}.(c \,\&\, Q(\bar{x},\bar{y}))\}]\, [\{c \,\&\, Q(\bar{x},\bar{y})\}]\, P(\bar{x})$$

Theorem. *The uniform substitution axioms above for dA$\mathcal{L}$ are sound.*

Proof. The cut axiom follows more or less immediately from similar reasoning as the cut proof rule in [10]. Note that since R cannot mention $\bar{y}$ so there is no danger of a name clash.

Suppose that the premise of the implication in the invariance axiom is true. Suppose that we know the LHS of the equivalence in the axiom. The right side of the equivalence immediately follows as either the differential equation has no runs (vacuously true), in which case the test will also be true for all $\bar{y}$. The proof of the other direction follows that in [10], as we only really need to prove it for all the reachable states of the DASystem and not all of them in general.

As for the stutter axiom, suppose that the DASystem has no runs. Then both sides are vacuously true, and we are done. Hence, assume that the DASystem does indeed have some runs.

Assume that the RHS is true. We can always pick a $\bar{y}$ such that the ODE has a run, as the DASystem runs for some closed interval of time $[0, r]$, so we can choose the last value of $\bar{y}$ in the flow taken by the DASystem. Hence we can choose the 0 length run for the ODE, which is equivalent to just running the DASystem.

Assume that the LHS is true. Pick some arbitrary $\bar{y}$ for the RHS. Suppose that for that choice of $\bar{y}$ the ODE has no runs. Then the RHS always evaluates to true as the DASystem has runs by assumption. Suppose instead that for the choice of $\bar{y}$ the ODE has a run. Then joining the end of the flow of the DASystem to the beginning of the flow of the ODE gives a valid DASystem flow since Q must hold throughout. Hence we are done. □

The addition of the stutter axioms joins all the pieces together. Stutter allows us to safely extract out the underlying ODE from the DASystem, and extract out the domain condition and differential effects by taking advantage of the ODE differential weakening and effect rules, adding as little as possible to the soundness-critical core. Examples on the usage of these axioms (and proof rules derived from them) will appear in following sections.

5 Implementation Details

In this section, we describe our implementation choices and some of its details.

5.1 Core Data Structures

Internally, we represent DASystems as a thin wrapper over a DExists construct, which in turn wraps the existing ODE systems in KeYmaeraX. ODE systems contain a set of differential equations and a set of constraints to be maintained during the evolution of the differential equations. A DExists construct adds a list of quantified variables to the ODE system, which can then appear anywhere in the system, but not as a differential. As shown in Section 3.2, this is sufficient to represent the new constructs in dA$\mathcal{L}$, including differential inequalities.

The reason why DASystem wraps a DExists object instead of wrapping the ODE directly is for uniform substitution and renaming reasons. See section 5.3 for more details.

5.2 Concrete Syntax and Parsing

The parser in KeYmaeraXis a relatively straightforward shift-reduce parser. We add a new terminal, `\dexists` to the grammar to indicate the start of the DASystem. The `\dexists` terminal must be followed by a comma-separated list of variables wrapped in curly braces indicating the quantified variables of the DASystem, followed by an ODE system wrapped in curly braces, that does not contain differentials of the quantified variables. An example of the concrete syntax can be seen below:

```
\dexists{x,y,z}{v'=x,a'=y+z & x>=y & y<=a-z & v>=0}
```

5.3 Uniform Substitution and Uniform Substitution Axioms

As seen in section 4.2, we introduce a total of three new uniform substitution axioms to the trusted axiom base. Unfortunately, as we ran out of time to implement uniform substitution with vectorial variables (which would require a fair bit of re-engineering of KeYmaeraX) the uniform substitution axioms only apply to DASystems with one quantified variable.

We implemented the extra cases in the unification, matching and uniform substitution code. To note is the implementation of DASystem as just wrapping DExists. This is done because on one hand, the quantified variables need to be bound in the DASystem as they get modified within, but on the other hand, they should be free, as they do not actually modify the variables they shadow. The regular quantifiers do not face this issue as formulas never modify the state, and there is no notion of 'after' for a formula, but there is for programs.

By making the DASystem into a wrapper around a DExists, we can have the DASystem have the quantified variables free 'externally' while having the DExists bind the quantified variables 'internally'. This avoids the semantic issues that arise from tactics like Bound-Renaming.

5.4 Derived Axioms

For convenience, we derive the following axioms for differential invariance and differential cuts from our base axioms. Proofs of the derived axioms are presented in Appendix A. In this and future sections $\bar{x}$ refers to the variables bound by the differential program c.

DAI

$$[\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftarrow \\ (\forall \bar{y}.(Q(x,\bar{y}) \rightarrow P(x)) \land [\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, (P(x))')$$

DAC

$$[\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, P(x,\bar{y}) \leftarrow \\ ([\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}) \land R(x))\}]\, P(x,\bar{y}) \land [\{\exists \bar{y}.(c \,\&\, Q(x,\bar{y}))\}]\, R(x))$$

5.5 Derived Tactics

We present proof rules for weakening, invariance and cuts for quantified differential equations below, which we implemented as tactics in KeYmaera X. We show applications and derivations of the tactics for DASystems with one equation $x' = f(x,\bar{y})$. Proofs are left to Appendix B.

dAW:

$$\frac{Q(x,\bar{y}) \vdash P(x)}{\Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, P(x), \Delta}\,\texttt{dAW}$$

dAI:

$$\frac{Q(x,\bar{y}) \vdash [x' := f(x,\bar{y})]\, (P(x))'}{P(x) \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, P(x)}\,\texttt{dAI}$$

dAC:
The proof rule follows from the DAC derived axiom:

$$\frac{\Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}) \land R(x))\}]\, P(x,\bar{y}), \Delta \qquad \Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, R(x), \Delta}{\Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, P(x,\bar{y}), \Delta}\,\texttt{DAC}$$

5.6 Examples

Using the tactics shown above, as well as rules in d$\mathcal{L}$, we prove our modelling examples from Section 3.3 below. We also provide the corresponding Bellerophon proofs [4] for the examples.

5.6.1 Perturbed Train Braking

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{-l \leq d \leq u \wedge v \geq 0 \wedge v^2 \leq 2(b-u)(m-z) \wedge b > u \wedge u \geq 0 \wedge l \geq 0 \vdash z \leq m}\,\mathbb{R}}{\phi \vdash [\{\exists d.(z' = v, v' = -b + d \,\&\, -l \leq d \leq u \wedge v \geq 0 \wedge \phi)\}]\, safe}\,\texttt{dAW}}{\phi \vdash [dyn]\, safe}\,\texttt{DAC}}{\vdash \phi \rightarrow [dyn]\, safe}\,\rightarrow\mathrm{R}$$

①:

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{-l \leq d \leq u \wedge v \geq 0 \vdash 2vd \leq 2vu}\,\mathbb{R}}{-l \leq d \leq u \wedge v \geq 0 \vdash 2v(-b+d) \leq 2v(b-u)}}{-l \leq d \leq u \wedge v \geq 0 \vdash [z' := v; v' := -b + d]\,(\phi)'}\,['], [;], [':=]}{\phi \vdash [dyn]\,\phi}\,\texttt{dAI}$$

Full Bellerophon Proof (in Scala):

```
implyR(1) & andL(-1) &
DACut("v^2 <= 2*(b-u)*(m-z) & b > u & u >= 0 & l >= 0".asFormula)(1) <(
  DAWeaken(1) & QE & done,
  Dconstify(DAInvariant(1) <(
    QE & done,
    Dassignb(1)*2 & QE & done
  ))(1)
)
```

5.6.2 Disturbed Circular Motion

$$\dfrac{\dfrac{\dfrac{\dfrac{*}{x\cdot e\leq 0\vdash 2x(-y+e)+2yx\leq 0}\mathbb{R}}{x\cdot e\leq 0\vdash [x':=-y+e,y':=x]\,(x^2+y^2\leq 1)'}[']\,[;]\,[':=]}{x^2+y^2=1\vdash [\{\exists e.(x'=-y+e,y'=x\,\&\,x\cdot e\leq 0)\}]\,x^2+y^2\leq 1}\texttt{dAI}}{\vdash x^2+y^2=1\to [\{\exists e.(x'=-y+e,y'=x\,\&\,x\cdot e\leq 0)\}]\,x^2+y^2\leq 1}\to\mathrm{R}$$

Full Bellerophon Proof (in Scala):

```
implyR(1) & Dconstify(DAInvariant(1) <(
        QE & done,
        Dassignb(1)*2 & QE & done
))(1)
```

6 Conclusion and Future Work

To conclude, we have managed to successfully implement the desugared core of dA$\mathcal{L}$ in KeymaeraX for differential-algebraic systems with a single existentially quantified variable. In order to do this, we modernized the proof system for dA$\mathcal{L}$ and created a uniform substitution calculus for it. Future extensions of this project would include implementing support for multiple quantifiers, more convenient sugared syntax, and 'porting' over more of the proof rules, such as differential-algebraic variants. Another possibility would be to implement proof rules concerning hybrid games with differential-algebraic components.

References

[1] Olivier Bouissou, Eric Goubault, Sylvie Putot, Karim Tekkal, and Franck Vedrine. HybridFluctuat: A Static Analyzer of Numerical Programs within a Continuous Environment. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, pages 620–626, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] Laurent Doyen, Goran Frehse, George J. Pappas, and André Platzer. Verification of Hybrid Systems. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 1047–1110. Springer, 2018.

[3] Andreas Eggers, Martin Fränzle, and Christian Herde. SAT Modulo ODE: A Direct SAT Approach to Hybrid Systems. In Sungdeok (Steve) Cha, Jin-Young Choi, Moonzoo Kim, Insup Lee, and Mahesh Viswanathan, editors, *Automated Technology for Verification and Analysis*, pages 171–185, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[4] Nathan Fulton, Stefan Mitsch, Brandon Bohrer, and André Platzer. Bellerophon: Tactical Theorem Proving for Hybrid Systems. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *ITP*, volume 10499 of *LNCS*, pages 207–224. Springer, 2017.

[5] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An Axiomatic Tactical Theorem Prover for Hybrid Systems. In Amy P. Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 527–538. Springer, 2015.

[6] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. -Complete Decision Procedures for Satisfiability over the Reals. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, pages 286–300, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[7] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT Solver for Nonlinear Theories over the Reals. In Maria Paola Bonacina, editor, *Automated Deduction – CADE-24*, pages 208–214, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[8] Fabian Immler. Formally Verified Computation of Enclosures of Solutions of Ordinary Differential Equations. In Julia M. Badger and Kristin Yvonne Rozier, editors, *NASA Formal Methods*, pages 113–127, Cham, 2014. Springer International Publishing.

[9] Fabian Immler. Verified Reachability Analysis of Continuous Systems. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems - Volume 9035*, pages 37–51, Berlin, Heidelberg, 2015. Springer-Verlag.

[10] André Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[11] André Platzer. Stochastic Differential Dynamic Logic for Stochastic Hybrid Programs. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *CADE*, volume 6803 of *LNCS*, pages 446–460. Springer, 2011.

[12] André Platzer. A Uniform Substitution Calculus for Differential Dynamic Logic. In Amy P. Felty and Aart Middeldorp, editors, *CADE*, volume 9195 of *LNCS*, pages 467–481. Springer, 2015.

[13] André Platzer. Differential Hybrid Games. *ACM Trans. Comput. Log.*, 18(3):19:1–19:44, 2017.

[14] André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Switzerland, 2018.

[15] Nacim Ramdani and Nedialko S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149 – 162, 2011.

A Proofs of Derived Axioms

A.1 DAI

$$[\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, P(x) \leftarrow$$
$$(\forall \bar{y}.(Q(x,\bar{y}) \rightarrow P(x)) \wedge [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, (P(x))')$$

For readability, we abbreviate $P(x)$ as P and $Q(x,\bar{y})$ as Q. We further make the following abbreviations:

$$\begin{aligned}
old &\equiv [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, P \\
inv &\equiv [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, (P)' \\
base &\equiv ([\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, P \leftrightarrow \forall \bar{y}\, [?Q]\, P) \leftarrow [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, (P)' \\
&\equiv (old \leftrightarrow \forall \bar{y}\, [?Q]\, P) \leftarrow inv \\
\texttt{DAI} &\equiv old \leftarrow (\forall \bar{y}.(Q \rightarrow P) \wedge inv)
\end{aligned}$$

Proof:

$$\dfrac{\dfrac{\dfrac{*}{\forall \bar{y}.\, [?Q]\, P, inv \vdash old, base}\,\texttt{DAIbase} \quad \dfrac{\dfrac{*}{\forall \bar{y}.\, [?Q]\, P, inv \vdash old, inv}\,\texttt{id}}{\forall \bar{y}.\, [?Q]\, P, inv, base \vdash old}\,①\rightarrow\mathrm{L}}{\forall \bar{y}.\, [?Q]\, P, inv \vdash old}\,\texttt{cut}}{\dfrac{\forall \bar{y}.\, [?Q]\, P, inv \vdash old}{\forall \bar{y}.(Q \rightarrow P), inv \vdash old}\,[?]}{\vdash (\forall \bar{y}.(Q \rightarrow P) \wedge inv) \rightarrow old}\,\rightarrow\mathrm{R},\wedge\mathrm{L}$$

①:

$$\dfrac{\dfrac{*}{\forall \bar{y}.\, [?Q]\, P, inv, old \wedge \forall \bar{y}\, [?Q]\, P \vdash old}\,\wedge\mathrm{L},\,\texttt{id} \quad \dfrac{*}{\forall \bar{y}.\, [?Q]\, P, inv, \neg old \wedge \neg \forall \bar{y}\, [?Q]\, P \vdash old}\,\wedge\mathrm{L},\,\neg\mathrm{L},\,\texttt{id}}{\forall \bar{y}.\, [?Q]\, P, inv, old \leftrightarrow \forall \bar{y}\, [?Q]\, P \vdash old}\,\leftrightarrow\mathrm{L}$$

A.2 DAC

$$[\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, P(x,\bar{y}) \leftarrow$$
$$([\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}) \wedge R(x))\}]\, P(x,\bar{y}) \wedge [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q(x,\bar{y}))\}]\, R(x))$$

To prove this derived axiom, we cut in the base differential cut axiom. For readability, we abbreviate $P(x,\bar{y})$ as P, $Q(x,\bar{y})$. We further make the following abbreviations:

$$
\begin{aligned}
\mathit{old} &\equiv [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, P \\
\mathit{inv} &\equiv [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, R \\
\mathit{new} &\equiv [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q \wedge R)\}]\, P \\
\mathit{base} &\equiv ([\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, P \leftrightarrow [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q \wedge R)\}]\, P) \leftarrow [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, R \\
&\equiv (\mathit{old} \leftrightarrow \mathit{new}) \leftarrow \mathit{inv} \\
\texttt{DAC} &\equiv \mathit{old} \leftarrow (\mathit{new} \wedge \mathit{inv})
\end{aligned}
$$

Proof:

$$
\dfrac{\dfrac{}{\mathit{new}, \mathit{inv} \vdash \mathit{old}, \mathit{base}}\,* \texttt{DACbase} \quad \dfrac{\dfrac{\dfrac{}{\mathit{new}, \mathit{inv} \vdash \mathit{old}, \mathit{inv}}\,* \texttt{id} \quad ①}{\mathit{new}, \mathit{inv}, \mathit{base} \vdash \mathit{old}} \rightarrow\text{L}}{\mathit{new},\ \mathit{inv} \vdash \mathit{old}}\,\text{cut}}{\vdash (\mathit{new} \wedge \mathit{inv}) \rightarrow \mathit{old}}\,\rightarrow\text{R},\ \wedge\text{L}
$$

①:

$$
\dfrac{\dfrac{}{\mathit{new}, \mathit{inv}, \mathit{old} \wedge \mathit{new} \vdash \mathit{old}}\,* \wedge\text{L},\ \texttt{id} \quad \dfrac{\dfrac{}{\mathit{new}, \mathit{inv}, \neg \mathit{old} \vdash \mathit{old}, \mathit{new}}\,* \texttt{id}}{\mathit{new}, \mathit{inv}, \neg \mathit{old} \wedge \neg \mathit{new} \vdash \mathit{old}} \wedge\text{L},\ \neg\text{L}}{\mathit{new}, \mathit{inv}, \mathit{old} \leftrightarrow \mathit{new} \vdash \mathit{old}} \leftrightarrow\text{L}
$$

B Proofs of Derived Tactics

B.1 dAW

We abbreviate $P(x)$ with P and $Q(x,\bar{y})$ with Q:

$$
\dfrac{\dfrac{\dfrac{\dfrac{Q \vdash P}{\vdash [x' = f(x,\bar{y}) \,\&\, Q]\, P}\,\texttt{dW}}{\Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, [x' = f(x,\bar{y}) \,\&\, Q]\, P, \Delta}\,\texttt{G}}{\Gamma \vdash \forall \bar{y}\, [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, [x' = f(x,\bar{y}) \,\&\, Q]\, P, \Delta}\,\forall\text{R}}{\Gamma \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y}) \,\&\, Q)\}]\, P, \Delta}\,\texttt{DAS}
$$

B.2 dAI

We abbreviate $P(x)$ with P and $Q(x,\bar{y})$ with Q:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Q \vdash [x' := f(x,\bar{y})]\,(P)'}{\vdash [x' = f(x,\bar{y})\,\&\,Q]\,(P)'}\,\texttt{dW}}{P \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y})\,\&\,Q)\}]\,[x' = f(x,\bar{y})\,\&\,Q]\,(P)'}\,\texttt{G}}{P \vdash \forall \bar{y}\,[\{\exists \bar{y}.(x' = f(x,\bar{y})\,\&\,Q)\}]\,[x' = f(x,\bar{y})\,\&\,Q]\,(P)'}\,\forall\text{R}}{\textcircled{1} \qquad P \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y})\,\&\,Q)\}]\,(P)'}\,\texttt{DAS}}{P \vdash [\{\exists \bar{y}.(x' = f(x,\bar{y})\,\&\,Q)\}]\,P}\,\texttt{DAI}$$

①:

$$\dfrac{\dfrac{*}{P, Q \vdash P}\,\texttt{id}}{P \vdash \forall \bar{y}(Q \to P)}\,\forall\text{R},{\to}\text{R}$$