On Decidable Fragments of d$\mathcal{L}$

David M Kahn
davidkah@cs.cmu.edu

Siva Somayyajula
ssomayya@cs.cmu.edu

1 Abstract

Decidable fragments of differential dynamic logic (d$\mathcal{L}$) are of interest to its proof theory, but also to the practice of verifying cyberphysical systems; for example, integration into KeYmaera X could improve usability by discharging or refuting "boilerplate" goals automatically. Here we do three things to contribute to the knowledge of the decidability of d$\mathcal{L}$.

For one, we work bottom-up and identify a few decidable fragments and describe their decision procedures. These include the strictly-polynomial star-free (i.e. asterate-free) fragment and certain simultaneously box-and-diamond fragments. We provide an implementation of a decision procedure for the former in OCaml (see our final project deliverables).

For another, we work top-down and identify many undecidable fragments of d$\mathcal{L}$. Many different approaches are considered, including restricting quantifiers, limiting the number of variables, bounding the domain of computation, and restricting arithmetical operators. This contributes in much the way Edison showed many ways to not make a lightbulb.

Finally, we connect pieces in the middle to show inter-decidability of a few fragments. Should further decidability results come in the future, using these could immediately extend them.

2 Introduction

Differential dynamic logic (d$\mathcal{L}$) [15] is an extension of dynamic logic (DL), a logic with modalities (i.e. box and diamond) to reason about program execution (where programs may nondeterministically make use of assignment and test actions). In particular, it is an extension that makes use of continuous evolution according to differential equations. This allows for reasoning about continuous physical phenomena alongside programs, i.e. cyberphysical systems.

d$\mathcal{L}$ has found success in this domain. d$\mathcal{L}$ proofs have successfully been used to verify the safety of various cyberphysical systems, ranging from roller coasters[2] to aircraft controllers [13]. There is even a proof assistant, KeYmaera X, set up to automate the checking of these proofs.

However, formally proving systems safe is still rather niche; the vast majority of systems (at the time of this paper) still operate with only hand-wavy proofs of safety, or even no proofs at all! It is a common occurence today to find bugs in software from all sorts of

industry leaders. When that software is responsible, not for just the virtual world, but for physical as well, the cost of bugs can range from property damage to human lives. This makes cyberphysical systems some of the most important to formally verify.

The question then is why formal verification is so often ignored. One reason is that proofs are non-critical to pushing out a final product that appears to work (they are only critical to being sure that the product *always* works). This allows them to fall to wayside as an "extra", something to add in if there are enough leftover resources. However, at the same time, proofs can be complicated, work-intensive, and highly dependent on the mathematical skill of the human prover, so their resource cost can be quite high.

One way to encourage the formal verification of cyberphysical systems then is to reduce its cost by automating the proof process. This has already been done to some extent by KeYmaera X, which *checks* proofs, but usually cannot *create* them. Indeed, automatically creating proofs or refutations of statements is hard, and often even impossible (as discussed in the *Related Work* section below). But, if automated proof/refutation creation (i.e. a *decision procedure*) can be found for entire classes of useful statements (i.e., if that class is *decidable*), then a significant burden can be removed from the proof process. If the statement that a system satisfies certain safety properties belongs to a decidable class, then the proof or disproof of the safety for that system can be obtained simply by "plugging and chugging". Even if such a statement does not belong to a decidable class, it is likely that, while trying to prove it, decidable statements will come up in its subgoals. Thus, identifying decidable fragments of $\mathrm{d}\mathcal{L}$ is a very useful pursuit in the bigger picture of verifying cyberphysical systems.

In the *Bottom-Up* section of this paper, we identify a few modest fragments of $\mathrm{d}\mathcal{L}$ that are decidable. We also provide a *Top-Down* section with many fragments that are *not* decidable, to show a collection of non-starters for future decidability exploration. This builds the known undecidable subclasses of $\mathrm{d}\mathcal{L}$ from the top-down, while also building the the decidable from the bottom-up, so that future work can focus on the unknown space between them. To aide in looking at that space, we also provide a *Middle* section that shows translations for interdecidability of the proof theories various fragments.

3 Related Work

The decidability of DL has been thoroughly studied in prior literature. For example, Fischer and Ladner [6] present a NEXPTIME algorithm for deciding propositional dynamic logic (PDL), the fragment of DL that abstracts away the meaning of atomic actions and propositions. Alternatively, when working in a finite domain of computation, DL can be cast as a fragment of the decidable equational theory of Kleene algebra with tests (KAT) using atomic assignments and tests [10]. The variant we are interested here however, $\mathrm{d}\mathcal{L}$, not only has concrete atomic actions and propositions, but also works in an infinite domain, so neither of these is directly applicable.

In investigating $\mathrm{d}\mathcal{L}$, related decidability problems from other fields will arise. Some background on those are given below.

$\mathrm{d}\mathcal{L}$ can express natural numbers, and the predicate language of $\mathrm{d}\mathcal{L}$ uses polynomial terms, so undecidable arithmetic problems on the naturals arise. For one thing, Matiyase-

vich built upon work by Robinson, Davis, and Putnam to show that general Diophantine (i.e. polynomial) equations have undecidable solution sets on the naturals, settling one of Hilbert's problems [14]. For another, the celebrated Gödel's incompleteness theorems yield that the first-order theory of Peano arithmetic is undecidable [8], as is any extension of the Robinson fragment [22]. However, many restrictions recover decidability. The Skolem and Presburger fragments remove addition and multiplication, respectively, from the language and are each decidable [19] [23]. Tung showed the positive (i.e. using = and not $\neq$) universal-existential fragment remains simple enough to be decidable by limiting quantification to occur over just two variables. [25]. Doing a little of both by weakening multiplication to divisibility and restricting quantifier usage may yield other decidable fragments provided by Lipshitz [12], or Bozga and Iosif [3]. If instead we remove the ability to talk about integers and work only with the first order theory of real closed fields, then Tarksi has shown that to be decidable [24]. If we further extend this real arithmetic theory with the exponential function, decidability becomes an open problem, but would be implied by the weak Schanuel's conjecture [27]. Adding any other operation that allows one to pick out integers (e.g. sine) will also reintroduce the undecidable problems of Peano arithmetic. Despite this, Ratschan et al. have shown many other useful variants of real arithmetic decidable, including those that use sine [21] [7].

Because d$\mathcal{L}$ introduces differential equations, other problems arise. In general, integrals do not have closed form solutions [11], making a general reduction to real arithmetic intractable. They also have a host of undecidable questions. For example, determining if a particular point is reached over the evolution of a set of polynomial differential equations is undecidable [9]. Even in the case where the rates of change are piecewise-constant, reachability is undecidable [1].

And of course, undecidability arises from the general ability for d$\mathcal{L}$ to directly represent programs, the analysis of which has been known undecidable since Church [4] and Turing [26].

Despite all these obstacles, the literature already contains a nontrivial decidable fragment of d$\mathcal{L}$: that of Platzer and Tan [17]. More d$\mathcal{L}$ specific proof-theory work can be found in [16].

4 Top-Down

Firstly, d$\mathcal{L}$ itself is undecidable, as is already known, but an explicit proof will be given here. We can show this by embedding the undecidable Post Correspondence Problem (PCP) [18] into d$\mathcal{L}$. The PCP asks, given a finite set of pairs of words x_i, y_i indexed by I, does there exist a nonempty sequence of indices J such that the concatenation across J of x_i is equivalent to concatenation across J of y_i? The PCP is undecidable even if the alphabet for the words is of size 2.

Theorem 1. d$\mathcal{L}$ *is undecidable.*

Proof. We will embed the PCP. Let the alphabet used for the PCP instance be $\{a, b\}$. Let a correspond to 0, b correspond to 1, and a word w correspond to the binary number corresponding to its letters as digits, prefixed by 1 (to preserve leading 0s).

We can now represent concatenating a onto an accumulator word w by $w := w * 2$. Concatenating b can likewise be represented with $w := w * 2 + 1$. To concatenate entire words, simply sequentially concatenate the letters of the words.

With this encoding, we can encode concatenating a word w_i from one of PCP pairs onto its corresponding accumulator w. Call the sequence of concatenations representing this α_i when w is x, and β_i when w is y.

Now we can encode the PCP:

$$\langle x := 1; y := 1; (\bigcup_{i \in I} \alpha_i; \beta_i); (\bigcup_{i \in I} \alpha_i; \beta_i)^* \rangle x = y$$

□

This encoding and variants thereof will be used repeatedly below.

4.1 Restricting Quantifiers and Modalities

One might think that one reason that d$\mathcal{L}$ is too expressive to be decidable is due to having powerful quantifiers or modalities. It might be that if we remove some of them, then the theory becomes decidable. As the *Middle* section shows, real, integer, and natural quantifiers can all be replaced by modal statements. Therefore, to make this restriction meaningful, we will consider having only both of existential quantifiers and diamond modalities, or having only their duals.

Theorem 2. *Purely existential diamond* d$\mathcal{L}$ *is undecidable.*

Proof. See the same encoding given for general d$\mathcal{L}$'s undecidability. □

This result was not too surprising, as the existential fragment of Peano arithmetic is already known undecidable via the MRDP theorem [14].

Corollary 2.1. *Purely universal box* d$\mathcal{L}$ *is undecidable.*

Proof. Perform the translation $[\alpha]P \iff \neg\langle\alpha\rangle\neg P$ on the witness for the existential diamond fragment, and recognize that deciding the negation of a statement is just as hard as deciding that statement. □

One might further think that the comparisons used might be restricted.

Lemma 2.1. *In the integers,* $<$ *and* $\leq$ *can simulate all other comparisons without using any quantifiers or connectives.*

Proof.

$$\begin{aligned} x = y &\iff (x-y)^2 \leq 0 \\ x \neq y &\iff 0 < (x-y)^2 \\ x \leq y &\iff x < y + 1 \end{aligned}$$

□

Lemma 2.2. *In the reals, $\neq$ can simulate $=$ with universal quantification and no other connectives.*

Proof. $x = y \iff \forall z.(x - y)z \neq 1$ □

Corollary 2.2. *Purely existential diamond $\mathrm{d}\mathcal{L}$ is undecidable for every set of comparisons save for possibly $\neq$ alone, and purely universal box is undecidable save for possibly $=$ alone.*

Proof. For universal box, use the comparisons and universals to encode equality on the integers. Then use the same PCP encoding that showed universal box $\mathrm{d}\mathcal{L}$ undecidable. For existential diamond, dualize it. □

This means the only way forward with removing comparisons is to consider the purely universal box fragment with only $=$ (or dually, the purely existential diamond fragment with only $\neq$). The purely universal fragment of Peano arithmetic is in fact known decidable (it is called polynomial identity testing), so arithmetic may not necessarily be a roadblock for decidability anymore either.

It is important now to consider tests. $[?P]1 = 0 \iff P \to 1 = 0 \iff \neg P$, so we re-introduce negation unless we force tests to be the negated fragment. As shown in the *Middle* section, domain constraints can simulate tests, so these must follow the same restrictions. So, we may consider the purely universal box fragment, with only equality, and with negation only and always prefixing tests and domain constraints. This, as it turns out, is decidable. See the Platzer-Tan extension in the *Bottom-Up* section.

4.2 Restricting Variable Count

One might think restricting the number of variables sufficiently simplifies the language such that decidability might occur. Indeed, the one-variable fragment of Peano arithmetic is easily decidable, and the two variable fragment is conjectured to be, so arithmetic might not be a roadblock here.

Theorem 3. *The 2-variable fragment of $\mathrm{d}\mathcal{L}$ is undecidable.*

Proof. See the same encoding given for general $\mathrm{d}\mathcal{L}$'s undecidability. □

This was perhaps unsurprising. However, the following is far more non-trivial.

Theorem 4. *The 1-variable fragment of $\mathrm{d}\mathcal{L}$ is undecidable.*

Proof. We will embed the PCP. Let the alphabet used for the PCP instance be $\{a, b\}$. Let a correspond to 0, b correspond to 1, and a word w correspond to the binary number corresponding to its letters as digits, prefixed by 1 (to preserve leading 0s). Encode a pair of words (x, y) as the single number $2^x 3^y$.

Now note $z \in \mathbb{N} \iff \langle (z := z - 1)^* \rangle w = 0$

Using this for a test, we can now represent concatenating a single letter onto the x of the accumulator word pair $z = 2^x 3^y$ by doing the following:

- After $(z := z * \frac{5}{2}; ?(z \in \mathbb{N}))^*; ?(\neg z \in \mathbb{N}); z := z * \frac{2}{5}$, we see that $z = 3^y 5^x$

- Then run $(z := z * \frac{4}{5}; ?(z \in \mathbb{N}))^*; ?(\neg z \in \mathbb{N}); z := z * \frac{5}{4}$, after which we see that $z = 2^{2x}3^y$
- If the letter is a, then stop. If the letter is b, finish with $z := 2 * z$ so that $z = 2^{2x+1}3^y$.

The encoding for concatenating a letter onto y is about the same save for the change of base from 2 to 3 in the appropriate places. Then, as before, concatenating a word is the sequential concatenation of its letters.

With this encoding, we can encode concatenating a word w_i from one of PCP pairs onto its corresponding accumulator w. Call the sequence of concatenations representing this α_i when w is x, and β_i when w is y.

Now we can encode the PCP, noting that $x = y$ whenever 2^x3^y is a power of 6:

$$\langle z := 6; (\bigcup_{i \in I} \alpha_i; \beta_i); (\bigcup_{i \in I} \alpha_i; \beta_i)^*; (z := z * \frac{1}{6})^* \rangle z = 1$$

□

4.3 Bounded

One might think that the general ability for integers to go off to infinity might be the difference between decidability and undecidability. Indeed bounding variables in integer arithmetic makes the space of numbers finite, and thus decidable. However, it is not enough here.

Theorem 5. d$\mathcal{L}$ *using only numbers in* $[0, 1]$ *is undecidable.*

Proof. We will embed the PCP. Let the alphabet used for the PCP instance be $\{a, b\}$. Let a correspond to 0, b correspond to 1, and a word w correspond to the reverse binary number corresponding to its letters as digits, prefixed by a decimal point and suffixed by a 1 (to preserve leading 0s).

We can now represent concatenating a onto an accumulator word w by $w := w * \frac{1}{2}$. Concatenating b can likewise be represented with $w := w * \frac{1}{2} + \frac{1}{2}$. Note that if $w \in [0, 1]$, then each of these operations leaves $w \in [0, 1]$ To concatenate entire words, simply sequentially concatenate the letters of the words.

With this encoding, we can encode concatenating a word w_i from one of PCP pairs onto its corresponding accumulator w. Call the sequence of concatenations representing this α_i when w is x, and β_i when w is y.

Now we can encode the PCP:

$$\langle x := \frac{1}{2}; y := \frac{1}{2}; (\bigcup_{i \in I} \alpha_i; \beta_i); (\bigcup_{i \in I} \alpha_i; \beta_i)^* \rangle x = y$$

□

4.4 Restricting Arithmetic

The Presburger and Skolem fragments of Peano arithmetic are purely additive and multiplicative, respectively, and these restrictions regain decidability [19] [23]. One might think that similar restrictions here would regain decidability. However, this is not the case.

Theorem 6. *Purely additive* d$\mathcal{L}$ *is undecidable.*

Proof. Note that $2 * w$ can rewritten as $w + w$ and re-encode the encoding given for general d$\mathcal{L}$'s undecidability. □

Theorem 7. *Purely multiplicative* d$\mathcal{L}$ *is undecidable.*

Proof. We will embed the PCP. Let the alphabet used for the PCP instance be $\{a, b\}$. Let a correspond to 0, b correspond to 1, and a word w correspond to 2 raised to the power of the binary number corresponding to its letters as digits when that binary number is prefixed by 1 (to preserve leading 0s).

We can now represent concatenating a onto an accumulator word w by $w := w * w$. Concatenating b can likewise be represented with $w := 2 * w * w$. To concatenate entire words, simply sequentially concatenate the letters of the words.

With this encoding, we can encode concatenating a word w_i from one of PCP pairs onto its corresponding accumulator w. Call the sequence of concatenations representing this α_i when w is x, and β_i when w is y.

Now we can encode the PCP:

$$\langle x := 2; y := 2; (\bigcup_{i \in I} \alpha_i; \beta_i); (\bigcup_{i \in I} \alpha_i; \beta_i)^* \rangle x = y$$

□

4.5 Removing Operations

The *Middle* section and [16] explain some results about the inter-encoding of atomic operations. In particular, it is known that the discrete fragment (only assignments and tests), the continuous fragment (only constrainted evolution), and only unconstrained evolution with tests, are each equivalent to full d$\mathcal{L}$. Thus, each are undecidable. To regain decidability, more must be removed.

Theorem 8. d$\mathcal{L}$ *with only assignments is undecidable.*

Proof. See the same encoding given for general d$\mathcal{L}$'s undecidability. □

Theorem 9. d$\mathcal{L}$ *with only constrained evolution is undecidable, even when evolution is only at constant rates.*

Proof. Recall the work of Asarin, Maler, and Pneuli concerning the undecidability of reachability for piecewise-constant rates of change [1]. The undecidable problem discussed there, the piecewise-constant derivative (PCD) reachability problem, is as follows:

divide the space $\mathbb{R}^m$ into finitely many polytopes P_i for $i \in I$ (with each boundary belonging to a particular one of its adjacent polytopes), and associate to each a constant vector field. Starting from one particular polytope and following the vector field, can another particular polytope be reached?

It turns out to be unimportant that boundaries are unshared, as Asarin et al.'s proof does not use any features affected by the change.

Let P_i be the collection of linear inequalities picking out the polytope of the same name, P_i. Let C_i be the collection of linear inequalities picking out the closure of P_i. Let $\overline{k}_i$ be the constant rate of change vector over each of the dimensions m in polytope P_i. The following d$\mathcal{L}$ formula (with syntactic sugar for n-ary operations) then encodes the PCD reachability problem from P_s to P_t:

$$\exists \overline{x}.(P_s \wedge \langle (\bigcup_{i \in I} \{\overline{x}' = \overline{k}_i \& C_i\})^* \rangle P_t)$$

or, via dualizing, the more succinct

$$P_s \rightarrow [(\bigcup_{i \in I} \{\overline{x}' = \overline{k}_i \& C_i\})^*] \neg P_t$$

□

See some positive results about having only tests in the *Bottom-Up* section.

4.6 Removing Program Connectives

Some other features that could be removed from d$\mathcal{L}$ are the program connectives: nondeterministic choice, sequential composition, or the asterate. Removing the asterate is discussed in the *Bottom-Up* section.

Theorem 10. *Choice-free* d$\mathcal{L}$ *is undecidable.*

Proof. We will embed Diophantine equations. It is undecidable whether there are any natural solutions to a multivariate polynomial equation $D(x_1, ..., x_n) = 0$. The following encodes the existence of such solutions by encoding existential quantifiers with diamond, and does not use nondeterministic choice:

$$\langle x_1 := 0; (x_1 := x_1 + 1)^*; ...; x_n := 0; (x_n := x_n + 1)^* \rangle D(x_1, ..., x_n) = 0$$

□

Theorem 11. *Sequence-free* d$\mathcal{L}$ *is undecidable.*

Proof. See the encoding for the undecidability of d$\mathcal{L}$ with only constrained-evolution. □

It should be noted that, while many of the proofs here can easily be carried over to the fragment of d$\mathcal{L}$ without evolution, this one makes use of the evolution and so cannot.

5 Middle

It is already known constructively that the proof theory of the d$\mathcal{L}$'s discrete and continuous alone is just as hard as that of all d$\mathcal{L}$ [16]. However, there are more inter-decidability reductions that could help reduce search space when trying to identify (un)decidable fragments.

5.1 Formula Reduction

Theorem 12. *Deciding general* d$\mathcal{L}$ *statements is precisely as hard as deciding statements of the form* $[\alpha]P$ *for trivial* P*. Likewise for statements of the form* $\langle\alpha\rangle P$

Proof. $Q \iff \langle ?Q\rangle true \iff [?(\neg Q)]false$ □

This means that focusing in on statements of the form $[\alpha]P$ or $\langle\alpha\rangle P$ for the decidable fragment may be reasonable. However, this translation introduces complex tests wherever Q is a non-trivial d$\mathcal{L}$ statement. It offloads the difficulty of deciding general statements onto deciding tests. However, it is often desirable for decidability that restrictions put on tests are at least as restrictive as those for general statements, as the standard semantic translations for tests, given below, bring the test statement to the top level.

$$[?Q]P \iff Q \to P \qquad\qquad \langle ?Q\rangle P \iff Q \wedge P$$

Theorem 13. *Deciding general* d$\mathcal{L}$ *statements is precisely as hard as deciding statements of the form* $[\alpha]P$ *where* P *is non-modal, and where all tests in* α *follow the same restrictions as the statements. Likewise for statements of the form* $\langle\alpha\rangle P$*.*

Proof. We prove this for $[\alpha]P$ by induction over the syntax of d$\mathcal{L}$ propositions:

- We can embed the base case of non-modal propositions with $P \iff [?true]P$
- $\neg[\alpha]P \iff [?([\alpha]P)]false$
- $[\alpha]P \vee [\beta]Q \iff [?(\neg[\alpha]P);\beta]Q$
- $[\alpha]P \wedge [\beta]Q \iff [x := 0; \alpha \cup x := 1; \beta]((x = 0 \wedge P) \vee (x = 1 \wedge Q))$ for x fresh
- $\forall x.[\alpha]P \iff [x := *; \alpha]P$
- $\exists x.[\alpha]P \iff \neg[x := *; ?([\alpha]P)]false$
- $[\alpha][\beta]P \iff [\alpha;\beta]P$
- $\langle\alpha\rangle[\beta]P \iff \neg[\alpha; ?([\beta]P)]false$

To then show the $\langle\alpha\rangle P$ case, we apply the translation above followed by

$$[\alpha]P \iff \neg\langle\alpha\rangle\neg P$$

Deciding $\neg\langle\alpha\rangle\neg P$ to be false is precisely the same as deciding $\langle\alpha\rangle\neg P$ to be true.

The reverse direction of decision difficulty is trivial, in that $[\alpha]P$ and $\langle\alpha\rangle P$ are subsets of the d$\mathcal{L}$ statements we are considering.

□

It is for this reason that the fragments discussed in the *Top-Down* section are only of the form $[\alpha]P$ or similar.

It also follows from this theorem that, if a fragment of $\mathrm{d}\mathcal{L}$ only uses connectives whose translations are "nice", the translation to statements of the form $[\alpha]P$ or $\langle\alpha\rangle P$ may maintain more restrictions from the fragment. A particular instance is below.

Corollary 13.1. *A $\mathrm{d}\mathcal{L}$ fragment without $\neg, \vee, \exists, \langle\cdot\rangle$ and with some restriction on tests can maintain the same restriction on $\neg, \exists, \langle\cdot\rangle$ and tests in the $[\alpha]P$ translation.*

Proof. The translations for this fragment do not introduce new instances of any of the restricted pieces. □

It is also of use that quantifiers can be replaced with purely box or diamond statements.

Theorem 14. *Real, natural, and integer universal quantifiers can be defined with box statements, and existential with diamond.*

Proof.

$$\begin{aligned}
\forall x \in \mathbb{R}.P &\iff [x := *]P \\
\exists x \in \mathbb{R}.P &\iff \langle x := *\rangle P \\
\forall x \in \mathbb{N}.P &\iff [x := 0; (x := x+1)^*]P \\
\exists x \in \mathbb{N}.P &\iff \langle x := 0; (x := x+1)^*\rangle P \\
\forall x \in \mathbb{Z}.P &\iff \forall x \in \mathbb{N}.[(x := x \cup x := -x)]P \\
\exists x \in \mathbb{Z}.P &\iff \exists x \in \mathbb{N}.\langle x := x \cup x := -x\rangle P
\end{aligned}$$

□

Corollary 14.1. $\mathrm{d}\mathcal{L}$ *without quantifiers is just as expressive.*

Proof. Simulate them using box and diamond. □

5.2 Operational Reduction

Many operations of $\mathrm{d}\mathcal{L}$ are inter-definable. This means that, in order for any restriction on operations to meaningful, any fragment restricting one must either restrict those that can define it as well, or have a restriction that excludes the translation from being possible.

Theorem 15. $\mathrm{d}\mathcal{L}$ *without expression assignment is just as expressive.*

Proof. The following translation works, where the same fresh variable y may be re-used across all instances:

$$x := e \equiv y := *; ?(y = e); x := *; ?(x = y)$$

The fresh variable is necessary because e might depend on x. □

Theorem 16. $\mathrm{d}\mathcal{L}$ *without random assignment is just as expressive.*

Proof. $x := * \equiv \{x' = 1\} \cup \{x' = -1\}$ □

Theorem 17. $\mathrm{d}\mathcal{L}$ *without tests is just as expressive.*

Proof. $?Q \equiv \{x' = 0 \& Q\}$ □

Corollary 17.1. $\mathrm{d}\mathcal{L}$ *with only constrained evolution is just as expressive.*

Proof. Apply the above theorems to remove all assignments and tests. □

Using a "there and back again" translation [16], we also see the following:

Theorem 18. $\mathrm{d}\mathcal{L}$ *without domain constraints is just as expressive.*

Proof. The following translation works, where the same fresh variable y may be re-used across all instances:

$$\{x' = f(x) \& Q\} \equiv y := 0; \{x' = f(x), y' = 1\}; ?([\{x' = -f(x), y' = -1\}](y \geq 0 \to Q))$$

□

Corollary 18.1. $\mathrm{d}\mathcal{L}$ *with only unconstrained evolution and tests is just as expressive.*

Proof. Apply the above theorems to remove all assignments and domain constraints. □

6 Bottom-Up

6.1 Polynomial Star-Free Fragment

Working bottom-up, we start with the first-order theory of real arithmetic/of real closed fields, which is decidable due to Tarski [24]. We aim to identify a fragment of $\mathrm{d}\mathcal{L}$ whose decidability reduces to the decidability of said theory via sound translation. The core axioms of $\mathrm{d}\mathcal{L}$, particularly the ones governing the behavior of the box modality, already prescribe such a translation, as they show equivalence with a formula with structurally fewer connectives that are in $\mathrm{d}\mathcal{L}$. However, differential equations and indefinite iteration (the asterate) pose issues. In particular, the solution axiom, which is the "natural" choice for translating the action of box on a continuous program, could introduce exponential or even trigonometric functions. As stated before, the first-order theory of real arithmetic with the addition of such functions is still not yet known to be decidable, although there has been significant work in giving decision procedures for such theories, provided formulas are numerically stable [21]. Furthermore, the axioms for the asterate either require the computation of a loop invariant, or indirectly characterize it as the solution to the recursive equation $\alpha^* = 1 + \alpha\alpha*$ (in the language of Kleene algebras).

Thus, we consider a polynomial star-free fragment, where the asterate is removed and the solutions of differential equations are required to be polynomial and thus expressible in the first-order theory of real arithmetic. Now, we define $[\![\cdot]\!]$, which sends formulae

P, Q in this fragment to formulae in the theory of real arithmetic. It is homomorphic on (in)equalities and propositional/first-order connectives and the identity on real arithmetic, so it remains to define its action on the box modality (the diamond can be derived via duality).

$$\begin{aligned}
[\![[x' = f(x) \,\&\, Q(x)]P(x)]\!] &\triangleq \forall t \geq 0 \; ((\forall 0 \leq s \leq t \;\; [\![Q(x(s))]\!]) \to [\![P(x(t))]\!]) \quad \text{where } x'(t) = f(x(t)) \\
[\![[x := e]P(x)]\!] &\triangleq [\![P(e)]\!] \\
[\![[?Q]P]\!] &\triangleq [\![Q]\!] \to [\![P]\!] \\
[\![[\alpha \cup \beta]P]\!] &\triangleq [\![[\alpha]P]\!] \wedge [\![[\beta]P]\!] \\
[\![[\alpha;\beta]P]\!] &\triangleq [\![[\alpha][\beta]P]\!]
\end{aligned}$$

As the grammar of $\mathrm{d}\mathcal{L}$ already contains only polynomial arithmetic terms, this is *almost* sufficient for the image of $[\![\cdot]\!]$ to contain only polynomial terms; the one problem is in the solutions to differential equations. A differential equation $x' = e$ may not allow self-reference, since the solution would be exponential. In general, a system of ODEs may not have a cycle of self-reference e.g. $x' = y, y' = x$, since the solution would be trigonometric, etc. This observation leads us to the following insight.

Theorem 19. *Given a system of ODEs $x_1' = e_1, \ldots, x_n' = e_n$, let G be a directed graph where there is an edge from $x_i' = e_i$ to $x_j' = e_j$ if and only if x_i occurs in the body of e_j. Then, the system has polynomial solutions if G is acyclic.*

Proof. In particular, the solution is given by polynomial integration and back-substitution into each equation in the topological order of G. As a constructive result, this theorem is the workhorse of our decision procedure (see the next section). □

This provides an easily-checkable syntactic condition that enforces strictly polynomial real arithmetic; this is a maximally weak restriction. In short, the "DAG condition" and the lack of star characterize the polynomial star-free fragment.

Corollary 19.1. *The polynomial star-free fragment of* $\mathrm{d}\mathcal{L}$ *is decidable.*

Proof. Apply $[\![\cdot]\!]$ to recast it as a sentence in the theory of real closed fields. □

This fragment is of use quite generally. Although it cannot decide statements including the asterate, the usual proof rule for dealing with the asterate (via loop invariants) breaks it down into statements with at most one less asterate. This means that, should the human prover be able to pick out the loop invariants, the rest can be automated, so long as it is strictly polynomial.

Now, define the exponential polynomial star-free fragment to be the extension with a slightly weakened DAG condition, namely allowing cycles of length one for the particular forms that have solutions expressible with polynomials and the exponential function, like $x' = x$. The syntax can further be extended to explicitly support the exponential function in tandem with polynomial terms.

Corollary 19.2. *If the weak Schanuel's conjecture is true, the exponential polynomial star-free fragment is decidable.*

Proof. If the weak Schanuel's conjecture is true, then real arithmetic extended with the exponential function is decidable [27]. Thus the translation for the polynomial star-free fragment into decidable real arithmetic will also work here. □

6.1.1 Implementation

We have implemented the polynomial star-free fragment without modal tests and evolution domians in OCaml. In short, we exploit the availability of existing decision procedures for the theory of real arithmetic by querying one with the above translation of an input formula. The remainder of this section serves to outline the software architecture/pipeline and techniques used.

- **Shallow embedding with weak HOAS**

  Instead of providing the user with a program that simply accepts a formula and outputs "valid"/"invalid", we provide a *shallow embedding* of d$\mathcal{L}$ into OCaml, such that one may use existing OCaml operators to build the abstract syntax tree of a d$\mathcal{L}$ formula. For example, `!|(["x" ^= v"y"] & tt)|!(x >= !0.)` represents the formula $[x' = y](x \geq 0)$. Thus, the user may interactively query the truth of a formula in the OCaml toplevel, and use such results directly in their programs.

  Furthermore, we recognize that the behavior of quantifiers in either d$\mathcal{L}$ or first-order logic as name binders coincides with that of lambda abstraction. Thus, we use a technique due to Ciaffaglione et al. [5] called *weak higher-order abstract syntax* or weak HOAS which represents the body of a quantifier as a function (hence *higher-order*) accepting a variable name (as opposed to an arbitrary polynomial term, hence *weak*) and returning a d$\mathcal{L}$ formula. In doing so, we get a free implementation of capture-avoiding substitution (in particular, by function application to a variable name that does not capture any free variables in the body). We enforce this invariant by restricting the shape of user-provided variable names. This representation greatly simplified the syntax juggling required to perform sound translation, especially with first-order formulas.

- **Polynomial representation and ODE solver**

  Even though d$\mathcal{L}$ operates with arbitrary multivariate polynomials with rational coefficients, it is important to have two views on polynomials for implementation purposes. First, we must represent such polynomials in the language, but it is also useful to consider univariate polynomials *whose coefficients are multivariate polynomials*, because our expectation is that solving a system of ODEs will produce such polynomials with respect to a fresh indeterminate t. Thus, a bulk of source code is devoted to reconciling both representations and their respective operations (e.g. integration of univariate polynomials with respect to the indeterminate). With this framework in place and with the "DAG condition", we solve ODEs via univariate polynomial integration, back-substitution, and conversion to the multivariate representation.

- **Z3 backend**

  Post-translation, we decide the validity of first-order formulae via Z3, an SMT solver developed by Microsoft. In particular, given a translated formula ϕ, we ask if $\neg\phi$ is satisfiable. If it is not, then we can ask for a proof as to why $\neg\phi$ is unsatisfiable (and therefore why ϕ is valid). Otherwise, if it is, then we provide Z3's model of satisfiability i.e. the countermodel to the validity of ϕ. If necessary, we also explain why Z3 cannot determine either choice. We chose Z3 due to its rich OCaml API, but there are other (perhaps more effective) options for quantifier elimination, such as Redlog.

This implementation can be extended to support modal tests and evolution domains by suitably renaming free mutated variables to avoid clashes e.g. because in $[?([x' = x]\ x > 0); x := 1]\ x = 1$, the instance of x in the test is different from the rest of the formula. Lastly, one may consider offloading queries to a more advanced decision procedure for real arithmetic that can support e.g. exponential and trigonometric functions (under certain numerical stability conditions [21][20]) to improve the expressiveness of this fragment, with an analogous modification to the allowed solutions of ODEs, etc.

6.2 Simultaneous Fragment

When proving properties of cyberphysical systems, often one tries to prove a statement of the form $[\alpha]P$. Indeed, not only does the *Middle* section show the relative completeness of this fragment, but it is also quite a natural fragment to model in. Furthermore, it is of the same form as a few of our other decidable fragments. Despite this commonality, the actual practice of proving statements of this form has an issue, namely that these statements could be vacuously satisfied. $[\alpha]P$ is true for any P where α drops all of its nondeterministic paths of computation. This is bad because any actual model of a program or physics should continue forward. Time always chugs on, and programs that loop forever rather than taking the next step are often of poor design. This means we typically don't want to just prove $[\alpha]P$, but rather $[\alpha]P$ and $\langle\alpha\rangle P$ simultaneously.

Theorem 20. $\mathrm{d}\mathcal{L}$ *statements of the form* $[\alpha]P \wedge \langle\alpha\rangle P$ *are decidable whenever the fragment containing only* $[\alpha]P$ *is, so long as false can be defined in it.*

Proof. First note that $[\alpha]P \implies (\langle\alpha\rangle P \iff \langle\alpha\rangle true)$. Then apply this translation to the statement:

$$[\alpha]P \wedge \langle\alpha\rangle P \iff [\alpha]P \wedge \langle\alpha\rangle true \iff [\alpha]P \wedge \neg[\alpha]false$$

Now decide each of $[\alpha]P$ and $[\alpha]false$. Finally, conjunct the former with the negation of the latter. □

6.3 Platzer-Tan Extension

As already presented by Platzer and Tan, the fragment of the form $[\alpha]P$ is decidable where α contains only tests and domain constraints for inequalities and where P is non-modal

and contains only equalities [17]. This is shown by reducing this fragment inductively down to universally-quantified equality between multivariate polynomials, i.e., polynomial identity testing (which is equivalent regardless of whether the quantification is over integers, naturals, or reals).

This fragment is relatively large, but could be extended. As discussed in the *Top-Down* section, the extension cannot be to include more comparisons. However, we can extend the test test and domain constraint language to be equally as expressive as the fragment itself by allowing it to use negated members of the fragment itself as tests.

Theorem 21. *Let the Extended Platzer-Tan (EPT) fragment be the Platzer-Tan fragment where the tests and domain constraints are extended to be any negated member of the EPT fragment. This fragment is decidable.*

Proof. Take any statement in the EPT fragment. If there exist only non-modal tests, then it is part of the Platzer-Tan fragment, and thus is decidable. Now we need only worry about modal tests.

Because the statement is finitely large, there must exist an inner-most nested modal test, $?\neg[\alpha]P$. Its α cannot contain a modal test, and therefore $[\alpha]P$ belongs to the Platzer-Tan fragment. It can thus be translated into a single statement of universally-quantified equality between multivariate polynomials. Performing this translation therefore removes that modal test for a non-modal one. Inductively, all modal tests can be removed, reducing the EPT fragment to merely the Platzer-Tan fragment. □

6.4 Pure Tests

Theorem 22. $\mathrm{d}\mathcal{L}$ *with only tests is decidable.*

Proof. This can be reduced to the decidable theory of real closed fields.

First turn every diamond modality into a box via dualizing. We then proceed by inducting over the structure of hybrid programs to reduce every box modal term into a non-modal one.

- As the base case, P by itself is decidable by the theory of real closed fields.
- $[?Q]P \iff Q \to P$
- $[\alpha;\beta]P \iff [\alpha][\beta]P$
- $[\alpha \cup \beta]P \iff [\alpha]P \wedge [\beta]P$
- $[\alpha^*]P \iff P$

The last equivalence is a consequence of having only tests in the hybrid program fragment. □

7 Conclusion and Future Work

d$\mathcal{L}$ is a very rich logic, and thus it is not easy to see exactly what can be cut out to leave decidability. This work now provides a large survey across various insufficient restrictions and sufficient ones. Some insufficient ones may be surprising (like the single variable restriction) and some sufficient ones may be quite useful (like using the simultaneous version of a fragment to automatically show non-vacuity).

Further work could continue exploring these boundaries of (un)decidability. Many combinations of restrictions fall out of the encodings already presented here, but others do not. For instance, is single-variable d$\mathcal{L}$ decidable once modal tests are also removed?

Future work could also see these and other yet-to-be found decidable fragments implemented in proof assistants like KeYmaera X, in order to ease the proof burden for verifying cyberphysical (or other) systems. This could have great practical impact by making the practice of formally verifying systems more accessible.

8 Miscellany

Project deliverables include the results of this paper as well as an implementation of the decision procedure for the polynomial star-free fragment (sans modal tests). Kahn spearheaded the paper and Somayyajula the decision procedure, but both had ample say in each. Thank you to André Platzer and Yong Kiam Tan for feedback in developing this paper.

References

[1] ASARIN, E., MALER, O., AND PNUELI, A. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical computer science 138*, 1 (1995), 35–66.

[2] BOHRER, B., LUO, A., CHUANG, X. A., AND PLATZER, A. Coasterx: A case study in component-driven hybrid systems proof automation. *IFAC-PapersOnLine 51*, 16 (2018), 55 – 60. 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018.

[3] BOZGA, M., AND IOSIF, R. On decidability within the arithmetic of addition and divisibility. In *International Conference on Foundations of Software Science and Computation Structures* (2005), Springer, pp. 425–439.

[4] CHURCH, A. An unsolvable problem of elementary number theory. *American journal of mathematics 58*, 2 (1936), 345–363.

[5] CIAFFAGLIONE, A., AND SCAGNETTO, I. A weak HOAS approach to the poplmark challenge. In *Proceedings Seventh Workshop on Logical and Semantic Frameworks, with Applications, LSFA 2012, Rio de Janeiro, Brazil, September 29-30, 2012.* (2012), pp. 109–124.

[6] FISCHER, M. J., AND LADNER, R. E. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences 18*, 2 (1979), 194 – 211.

[7] FRANEK, P., RATSCHAN, S., AND ZGLICZYNSKI, P. Quasi-decidability of a fragment of the first-order theory of real numbers. *Journal of Automated Reasoning 57*, 2 (Aug 2016), 157–185.

[8] GÖDEL, K. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik 38*, 1 (1931), 173–198.

[9] HAINRY, E. Decidability and Undecidability in Dynamical Systems. Research report, 2009.

[10] KOZEN, D., AND SMITH, F. Kleene algebra with tests: Completeness and decidability. In *International Workshop on Computer Science Logic* (1996), Springer, pp. 244–259.

[11] LIOUVILLE, J. *Premier mémoire sur la détermination des intégrales dont la valeur est algébrique*. Impr. Royale, 1833.

[12] LIPSHITZ, L. The diophantine problem for addition and divisibility. *Transactions of the American Mathematical Society 235* (1978), 271–283.

[13] LOOS, S. M., RENSHAW, D., AND PLATZER, A. Formal verification of distributed aircraft controllers. In *Proceedings of the 16th international conference on Hybrid systems: computation and control* (2013), ACM, pp. 125–130.

[14] MATIYASEVICH, Y. V. The diophantineness of enumerable sets. In *Doklady Akademii Nauk* (1970), vol. 191, Russian Academy of Sciences, pp. 279–282.

[15] PLATZER, A. Differential dynamic logic for hybrid systems. *Journal of Automated Reasoning 41*, 2 (2008), 143–189.

[16] PLATZER, A. The complete proof theory of hybrid systems. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science* (2012), IEEE Computer Society, pp. 541–550.

[17] PLATZER, A., AND TAN, Y. K. Differential equation axiomatization: The impressive power of differential ghosts. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (New York, NY, USA, 2018), LICS '18, ACM, pp. 819–828.

[18] POST, E. L. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society 52*, 4 (1946), 264–268.

[19] PRESBURGER, M. Uber die vollstandigkeiteines gewissen systems der arithmetik ganzer zahlen, in welchen die addition als einzige operation hervortritt. In *Comptes-Rendus du ler Congres des Mathematiciens des Pays Slavs* (1929).

[20] RATSCHAN, S. Quantified constraints under perturbation. *Journal of Symbolic Computation 33*, 4 (2002), 493 – 505.

[21] RATSCHAN, S. Efficient solving of quantified inequality constraints over the real numbers. *ACM Trans. Comput. Logic 7*, 4 (Oct. 2006), 723–748.

[22] ROBINSON, R. M. An essentially undecidable axiom system. In *Proceedings of the international Congress of Mathematics* (1950), vol. 1, pp. 729–730.

[23] SKOLEM, T. *Über einige Satzfunktionen in der Aritmetik*. J. Dybwad, 1931.

[24] TARSKI, A. A decision method for elementary algebra and geometry. *Collected Works of A. Tarski* (1953).

[25] TUNG, S.-P. Provability and decidability of arithmetical universal-existential sentences. *Bulletin of the London Mathematical Society 18*, 3 (1986), 241–247.

[26] TURING, A. M. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society 2*, 1 (1937), 230–265.

[27] WILKIE, A. J. *Schanuel's Conjecture and the Decidability of the Real Exponential Field*. Springer Netherlands, Dordrecht, 1997, pp. 223–230.