Implicit Definitions with Differential Equations for KeYmaera X
(System Description)

James Gallicchio, Yong Kiam Tan, Stefan Mitsch, and André Platzer

Computer Science Department, Carnegie Mellon University, Pittsburgh, USA
jgallicc@andrew.cmu.edu
{yongkiat,smitsch,aplatzer}@cs.cmu.edu

Abstract. Definition packages in theorem provers provide users with means of defining and organizing concepts of interest. This system description presents a new definition package for the hybrid systems theorem prover KeYmaera X based on differential dynamic logic (dL). The package adds KeYmaera X support for user-defined smooth functions whose graphs can be implicitly characterized by dL formulas. Notably, this makes it possible to implicitly characterize functions, such as the exponential and trigonometric functions, as solutions of differential equations and then prove properties of those functions using dL's differential equation reasoning principles. Trustworthiness of the package is achieved by minimally extending KeYmaera X's soundness-critical kernel with a single axiom scheme that expands function occurrences with their implicit characterization. Users are provided with a high-level interface for defining functions and non-soundness-critical tactics that automate low-level reasoning over implicit characterizations in hybrid system proofs.

Keywords: Definitions · differential dynamic logic · verification of hybrid systems · theorem proving

1 Introduction

KeYmaera X [7] is a theorem prover implementing differential dynamic logic dL [17,19,20,21] for specifying and verifying properties of hybrid systems mixing discrete dynamics and differential equations. Definitions enable users to express complex theorem statements in concise terms, e.g., by modularizing hybrid system models and their proofs [14]. Prior to this work, KeYmaera X had only one mechanism for definition, namely, non-recursive abbreviations via uniform substitution [14,20]. This restriction meant that common and useful functions, e.g., the trigonometric and exponential functions, could not be directly used in KeYmaera X, even though they can be uniquely characterized by dL formulas [17].

This system description introduces a new KeYmaera X definitional mechanism where functions are *implicitly defined* in dL as solutions of ordinary differential equations (ODEs). Although definition packages are available in most general-purpose proof assistants, our package is novel in tackling the question

J. Blanchette, L. Kovacs and D. Pattinson (Eds.): IJCAR 2022, LNCS 13385, pp. 723–733, 2022.
DOI: 10.1007/978-3-031-10769-6_42

of how best to support user-defined functions in the *domain-specific* setting for hybrid systems. In contrast to tools with builtin support for *some* fixed subsets of special functions [1,9,23]; or higher-order logics that can work with functions via their infinitary series expansions [4], e.g., $\exp(t) = \sum_{i=0}^{\infty} \frac{t^i}{i!}$; our package strikes a balance between practicality and generality by allowing users to define and reason about *any* function characterizable in dL as the solution of an ODE (Section 2), e.g., $\exp(t)$ solves the ODE $e' = e$ with initial value $e(0) = 1$.

Theoretically, implicit definitions strictly expand the class of ODE invariants amenable to dL's complete ODE invariance proof principles [22]; such invariants play a key role in ODE safety proofs [21] (see Proposition 3). In practice, arithmetical identities and other specifications involving user-defined functions are proved by automatically unfolding their implicit ODE characterizations and reusing existing KeYmaera X support for ODE reasoning (Section 3). The package is designed to provide seamless integration of implicit definitions in KeYmaera X and its usability is demonstrated on several hybrid system verification examples drawn from the literature that involve special functions (Section 4).

All proofs are in the supplement [8]. The definitions package is part of KeYmaera X with a usage guide at: http://keymaeraX.org/keymaeraXfunc/.

2 Interpreted Functions in Differential Dynamic Logic

This section briefly recalls differential dynamic logic (dL) [17,18,20,21] and explains how its term language is extended to support implicit function definitions.

Syntax. Terms $e, \tilde{e}$ and formulas ϕ, ψ in dL are generated by the following grammar, with variable x, rational constant c, k-ary function symbols h (for any $k \in \mathbb{N}$), comparison operator $\sim \in \{=, \neq, \geq, >, \leq, <\}$, and hybrid program α:

$$e, \tilde{e} ::= x \mid c \mid e + \tilde{e} \mid e \cdot \tilde{e} \mid h(e_1, \ldots, e_k) \tag{1}$$

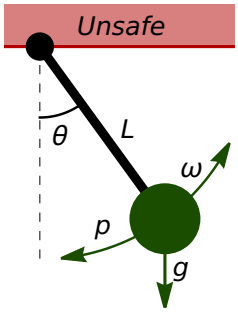
$$\phi, \psi ::= e \sim \tilde{e} \mid \phi \wedge \psi \mid \phi \vee \psi \mid \neg\phi \mid \forall x\, \phi \mid \exists x\, \phi \mid [\alpha]\phi \mid \langle\alpha\rangle\phi \tag{2}$$

The terms and formulas above extend the first-order language of real arithmetic ($\text{FOL}_{\mathbb{R}}$) with the box ($[\alpha]\phi$) and diamond ($\langle\alpha\rangle\phi$) modality formulas which express that *all* or *some* runs of hybrid program α satisfy postcondition ϕ, respectively. Table 1 gives an intuitive overview of dL's hybrid programs language for modeling systems featuring discrete and continuous dynamics and their interactions thereof. In dL's uniform substitution calculus, function symbols h are *uninterpreted*, i.e., they semantically correspond to an arbitrary (smooth) function. Such uninterpreted function symbols (along with uninterpreted predicate and program symbols) are crucially used to give a parsimonious axiomatization of dL based on uniform substitution [20] which, in turn, enables a trustworthy microkernel implementation of the logic in the theorem prover KeYmaera X [7,16].

Running Example. Adequate modeling of hybrid systems often requires the use of *interpreted* function symbols that denote specific functions of interest. As

Table 1. Syntax and informal semantics of hybrid programs

Program	Behavior
$?\phi$	Stay in the current state if ϕ is true, otherwise abort and discard run.
$x := e$	Store the value of term e in variable x.
$x := *$	Store an arbitrary real value in variable x.
$x' = f(x)\,\&\,Q$	Continuously follow ODE $x' = f(x)$ in domain Q for any duration ≥ 0.
$\texttt{if}(\phi)\,\alpha$	Run program α if ϕ is true, otherwise skip. Definable by $?\phi; \alpha \cup ?\neg\phi$.
$\alpha; \beta$	Run program α, then run program β in any resulting state(s).
$\alpha \cup \beta$	Nondeterministically run either program α or program β.
α^*	Nondeterministically repeat program α for n iterations, for any $n \in \mathbb{N}$.
$\{\alpha\}$	For readability, braces are used to group and delimit hybrid programs.



Hybrid program model (auxiliary variables s, c):

$$\hat{\alpha}_s \equiv \left(\begin{array}{l} s := *; c := *; ?\phi_{\sin}(s,\theta); ?\phi_{\cos}(c,\theta); \\ p := *; \texttt{if}\left(\frac{1}{2}(\omega - p)^2 < \frac{g}{L}c\right)\ \{\omega := \omega - p\}; \\ \{\theta' = \omega, \omega' = -\frac{g}{L}s - k\omega, s' = \omega c, c' = -\omega s\} \end{array} \right)^*$$

Hybrid program model (trigonometric functions):

$$\alpha_s \equiv \left(\begin{array}{l} p := *; \texttt{if}\left(\frac{1}{2}(\omega - p)^2 < \frac{g}{L}\cos(\theta)\right)\ \{\omega := \omega - p\}; \\ \{\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega\} \end{array} \right)^*$$

dL safety specification:

$$\phi_s \equiv g > 0 \wedge L > 0 \wedge k > 0 \wedge \theta = 0 \wedge \omega = 0 \rightarrow [\alpha_s]\,|\theta| < \frac{\pi}{2}$$

Fig. 1. Running example of a swinging pendulum driven by an external force (left), its hybrid program models and dL safety specification (right). Program α_s uses trigonometric functions directly, while program $\hat{\alpha}_s$ uses variables s, c to implicitly track the values of $\sin(\theta)$ and $\cos(\theta)$, respectively (additions in red). The implicit characterizations $\phi_{\sin}(s,\theta), \phi_{\cos}(c,\theta)$ are defined in (4), (5) and are not repeated here for brevity.

a running example, consider the swinging pendulum shown in Fig. 1. The ODEs describing its continuous motion are $\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$, where θ is the swing angle, ω is the angular velocity, and g, k, L are the gravitational constant, coefficient of friction, and length of the rigid rod suspending the pendulum, respectively. The hybrid program α_s models an external force that repeatedly pushes the pendulum and changes its angular velocity by a nondeterministically chosen value p; the guard $\texttt{if}(\ldots)$ condition is designed to ensure that the push does not cause the pendulum to swing above the horizontal as specified by ϕ_s. Importantly, the function symbols $\sin, \cos$ must denote the usual real trigonometric functions in α_s. Program $\hat{\alpha}_s$ shows the same pendulum modeled in dL *without* the use of interpreted symbols, but instead using auxiliary variables s, c.

Note that $\hat{\alpha}_s$ is cumbersome and subtle to get right: the implicit characterizations $\phi_{\sin}(s,\theta), \phi_{\cos}(c,\theta)$ from (4), (5) are lengthy and the differential equations $s' = \omega c, c' = -\omega s$ must be manually calculated and added to ensure that s, c correctly track the trigonometric functions as θ evolves continuously [18,22].

Interpreted Functions. To enable extensible use of interpreted functions in dL, the term grammar (1) is enriched with k-ary function symbols h that carry an *interpretation* annotation [5,27], $h_{\ll\phi\gg}$, where $\phi \equiv \phi(x_0, y_1, \ldots, y_k)$ is a dL formula with free variables in $x_0, y_1, \ldots, y_k$ and no uninterpreted symbols. Intuitively, ϕ is a formula that characterizes the graph of the intended interpretation for h, where $y_1, \ldots, y_k$ are inputs to the function and x_0 is the output. Since ϕ depends only on the values of its free variables, its formula semantics $[\![\phi]\!]$ can be equivalently viewed as a subset of Euclidean space $[\![\phi]\!] \subseteq \mathbb{R} \times \mathbb{R}^k$ [20,21]. The dL term semantics $\nu[\![e]\!]$ [20,21] in a state ν is extended with a case for terms $h_{\ll\phi\gg}(e_1, \ldots, e_k)$ by evaluation of the smooth C^∞ function characterized by $[\![\phi]\!]$:

$$\nu[\![h_{\ll\phi\gg}(e_1,\ldots,e_k)]\!] = \begin{cases} \hat{h}(\nu[\![e_1]\!],\ldots,\nu[\![e_k]\!]) & \text{if } [\![\phi]\!] \text{ graph of smooth } \hat{h}{:}\mathbb{R}^k{\to}\mathbb{R} \\ 0 & \text{otherwise} \end{cases}$$

This semantics says that, if the relation $[\![\phi]\!] \subseteq \mathbb{R} \times \mathbb{R}^k$ is the graph of some smooth C^∞ function $\hat{h} : \mathbb{R}^k \to \mathbb{R}$, then the annotated syntactic symbol $h_{\ll\phi\gg}$ is interpreted semantically as $\hat{h}$. Note that the graph relation uniquely defines $\hat{h}$ (if it exists). Otherwise, $h_{\ll\phi\gg}$ is interpreted as the constant zero function which ensures that the term semantics remain well-defined for all terms. An alternative is to leave the semantics of some terms (possibly) undefined, but this would require more extensive changes to the semantics of dL and extra case distinctions during proofs [2].

Axiomatics and Differentially-Defined Functions. To support reasoning for implicit definitions, annotated interpretations are reified to characterization axioms for expanding interpreted functions in the following lemma.

Lemma 1 (Function interpretation). *The FI axiom (below) for* dL *is sound where h is a k-ary function symbol and the formula semantics $[\![\phi]\!]$ is the graph of a smooth C^∞ function $\hat{h} : \mathbb{R}^k \to \mathbb{R}$.*

$$\text{FI} \quad e_0 = h_{\ll\phi\gg}(e_1,\ldots,e_k) \leftrightarrow \phi(e_0,e_1,\ldots,e_k)$$

Axiom FI enables reasoning for terms $h_{\ll\phi\gg}(e_1, \ldots, e_k)$ through their implicit interpretation ϕ, but Lemma 1 does not directly yield an implementation because it has a soundness-critical side condition that interpretation ϕ characterizes the graph of a smooth C^∞ function. It is possible to syntactically characterize this side condition [2], e.g., the formula $\forall y_1, \ldots, y_k \, \exists x_0 \, \phi(x_0, y_1, \ldots, y_k)$ expresses that the graph represented by ϕ has at least one output value x_0 for each input value $y_1, \ldots, y_k$, but this burdens users with the task of proving this

side condition in dL before working with their desired function. The KeYmaera X definition package opts for a middle ground between generality and ease-of-use by implementing FI for univariate, *differentially-defined* functions, i.e., the interpretation ϕ has the following shape, where $x = (x_0, x_1, \ldots, x_n)$ abbreviates a vector of variables, there is one input $t = y_1$, and $X = (X_0, X_1, \ldots, X_n)$, T are dL terms that do not mention any free variables, e.g., are rational constants, which have constant value in any dL state:

$$\phi(x_0,t) \equiv \langle x_1,\ldots,x_n := *; \left\{ \begin{matrix} x' = -f(x,t), t' = -1 \cup \\ x' = f(x,t), t' = 1 \end{matrix} \right\rangle \begin{pmatrix} x = X \wedge \\ t = T \end{pmatrix} \tag{3}$$

Formula (3) says from point x_0, there exists a choice of the remaining coordinates $x_1, \ldots, x_n$ such that it is possible to follow the defining ODE either forward $x' = f(x,t), t' = 1$ or backward $x' = -f(x,t), t' = -1$ in time to reach the initial values $x = X$ at time $t = T$. In other words, the implicitly defined function $h_{\ll\phi(x_0,t)\gg}$ is the x_0-coordinate projected solution of the ODE starting from initial values X at initial time T. For example, the trigonometric functions used in Fig. 1 are differentially-definable as respective projections:

$$\phi_{\sin}(s,t) \equiv \langle c := *; \left\{ \begin{matrix} s' = -c, c' = s, t' = -1 \cup \\ s' = c, c' = -s, t' = 1 \end{matrix} \right\rangle \begin{pmatrix} s = 0 \wedge c = 1 \wedge \\ t = 0 \end{pmatrix} \tag{4}$$

$$\phi_{\cos}(c,t) \equiv \langle s := *; \left\{ \begin{matrix} s' = -c, c' = s, t' = -1 \cup \\ s' = c, c' = -s, t' = 1 \end{matrix} \right\rangle \begin{pmatrix} s = 0 \wedge c = 1 \wedge \\ t = 0 \end{pmatrix} \tag{5}$$

By Picard-Lindelöf [21, Thm. 2.2], the ODE $x' = f(x,t)$ has a unique solution $\Phi : (a,b) \to \mathbb{R}^{n+1}$ on an open interval (a,b) for some $-\infty \leq a < b \leq \infty$. Moreover, $\Phi(t)$ is C^∞ smooth in t because the ODE right-hand sides are dL terms with smooth interpretations [20]. Therefore, the side condition for Lemma 1 reduces to showing that Φ exists globally, i.e., it is defined on $t \in (-\infty, \infty)$.

Lemma 2 (Smooth interpretation). *If formula $\exists x_0\, \phi(x_0,t)$ is valid, $\phi(x_0,t)$ from (3) characterizes a smooth C^∞ function and axiom FI is sound for $\phi(x_0,t)$.*

Lemma 2 enables an implementation of axiom FI in KeYmaera X that combines a syntactic check (the interpretation has the shape of formula (3)) and a side condition check (requiring users to prove existence for their interpretations).

The addition of differentially-defined functions to dL strictly increases the deductive power of ODE invariants, a key tool in deductive ODE safety reasoning [21]. Intuitively, the added functions allow direct, syntactic descriptions of invariants, e.g., the exponential or trigonometric functions, that have effective invariance proofs using dL's complete ODE invariance reasoning principles [22].

Proposition 3 (Invariant expressivity). *There are valid polynomial dL differential equation safety properties which are provable using differentially-defined function invariants but are not provable using polynomial invariants.*

3 KeYmaera X Implementation

The implicit definition package adds interpretation annotations and axiom FI based on Lemma 2 in ≈170 lines of code extensions to KeYmaera X's soundness-critical core [7,16]. This section focuses on non-soundness-critical usability features provided by the package that build on those core changes.

3.1 Core-Adjacent Changes

KeYmaera X has a browser-based user interface with concrete, ASCII-based dL syntax [14]. The package extends KeYmaera X's parsers and pretty printers with support for interpretation annotations `h«...»(...)` and users can simultaneously define a family of functions as respective coordinate projections of the solution of an n-dimensional ODE (given initial conditions) with sugared syntax:

```
implicit Real h1(Real t), ..., hn(Real t) = {{initcond};{ODE}}
```

For example, the implicit definitions (4), (5) can be written with the following sugared syntax; KeYmaera X automatically inserts the associated interpretation annotations for the trigonometric function symbols, see the supplement [8] for a KeYmaera X snippet of formula ϕ_s from Fig. 1 using this sugared definition.

```
implicit Real sin(Real t), cos(Real t) =
    {{sin:=0; cos:=1;}; {sin'=cos, cos'=-sin}}
```

In fact, the functions $\sin, \cos, \exp$ are so ubiquitous in hybrid system models that the package builds their definitions in automatically without requiring users to write them explicitly. In addition, although arithmetic involving those functions is undecidable [11,24], KeYmaera X can export those functions whenever its external arithmetic tools have partial arithmetic support for those functions.

3.2 Intermediate and User-Level Proof Automation

The package automatically proves three important lemmas about user-defined functions that can be transparently re-used in all subsequent proofs:

1. It proves the side condition of axiom FI using KeYmaera X's automation for proving sufficient duration existence of solutions for ODEs [26] which automatically shows global existence of solutions for all affine ODEs and some univariate nonlinear ODEs. As an example of the latter, the hyperbolic tanh function is differentially-defined as the solution of ODE $x' = 1 - x^2$ with initial value $x = 0$ at $t = 0$ whose global existence is proved automatically.
2. It proves that the functions have initial values as specified by their interpretation, e.g., $\sin(0) = 0$, $\cos(0) = 1$, and $\tanh(0) = 0$.
3. It proves the *differential axiom* [20] for each function that is used to enable syntactic derivative calculations in dL, e.g., the differential axioms for $\sin, \cos$ are $(\sin(e))' = \cos(e)(e)'$ and $(\cos(e))' = -\sin(e)(e)'$, respectively. Briefly,

these axioms are automatically derived in a correct-by-construction manner using dL's syntactic version of the chain rule for differentials [20, Fig. 3], so the rate of change of $\sin(e)$ is the rate of change of $\sin(\cdot)$ with respect to its argument e, multiplied by the rate of change of its argument $(e)'$.

These lemmas enable the use of differentially-defined functions with all existing ODE automation in KeYmaera X [22,26]. In particular, since differentially-defined functions are univariate Noetherian functions, they admit complete ODE invariance reasoning principles in dL [22] as implemented in KeYmaera X.

The package also adds specialized support for arithmetical reasoning over differential definitions to supplement external arithmetic tools in proofs. First, it allows users to manually prove identities and bounds using KeYmaera X's ODE reasoning. For example, the bound $\tanh(\lambda x)^2 < 1$ used in the example α_n from Section 4 is proved by *differential unfolding* as follows (see supplement [8]):

$$\frac{\vdash \tanh(0)^2 < 1 \quad \tanh(\lambda v)^2{<}1 \vdash [\{v' = 1 \,\&\, v \leq x\} \cup \{v' = -1 \,\&\, v \geq x\}] \tanh(\lambda v)^2{<}1}{\vdash \tanh(\lambda x)^2 < 1}$$

This deduction step says that, to show the conclusion (below rule bar), it suffices to prove the premises (above rule bar), i.e., the bound is true at $v = 0$ (left premise) and it is preserved as v is evolved forward $v' = 1$ or backward $v' = -1$ along the real line until it reaches x (right premise). The left premise is proved using the initial value lemma for tanh while the right premise is proved by ODE invariance reasoning with the differential axiom for tanh [22].

Second, the package uses KeYmaera X's uniform substitution mechanism [20] to implement (untrusted) abstraction of functions with fresh variables when solving arithmetic subgoals, e.g., the following arithmetic bound for example α_n is proved by abstraction after adding the bounds $\tanh(\lambda x)^2 < 1, \tanh(\lambda y)^2 < 1$.

$$\textbf{Bound:} \quad x(\tanh(\lambda x) - \tanh(\lambda y)) + y(\tanh(\lambda x) + \tanh(\lambda y)) \leq 2\sqrt{x^2 + y^2}$$
$$\textbf{Abstracted:} \quad t_x^2 < 1 \land t_y^2 < 1 \to x(t_x - t_y) + y(t_x + t_y) \leq 2\sqrt{x^2 + y^2}$$

4 Examples

The definition package enables users to work with differentially-defined functions in KeYmaera X, including modeling and expressing their design intuitions in proofs. This section applies the package to verify various continuous and hybrid system examples from the literature featuring such functions.

Discretely driven pendulum. The specification ϕ_s from Fig. 1 contains a discrete loop whose safety property is proved by a loop invariant, i.e., a formula that is preserved by the discrete and continuous dynamics in each loop iteration [21]. The key invariant is $Inv \equiv \frac{g}{L}(1 - \cos\theta) + \frac{1}{2}\omega^2 < \frac{g}{L}$, which expresses that the total energy of the system (sum of potential and kinetic energy on the LHS) is less than the energy needed to cross the horizontal (RHS). The main steps are as follows (proofs for these steps are automated by KeYmaera X):

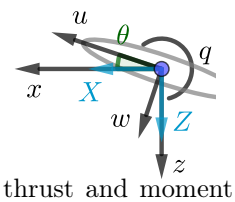
1. $Inv \rightarrow [\texttt{if}\left(\frac{1}{2}(\omega - p)^2 < \frac{g}{L}\cos(\theta)\right)\ \{\omega := \omega - p\}]Inv$, which shows that the discrete guard only allows push p if it preserves the energy invariant, and
2. $Inv \rightarrow [\{\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega\}]Inv$, which shows that Inv is an energy invariant of the pendulum's ODE.

Neuron interaction. The ODE α_n models the interaction between a pair of neurons [12]; its specification ϕ_n nests dL's diamond and box modalities to express that the system norm ($\sqrt{x^2+y^2}$) is asymptotically bounded by 2τ.

$$\alpha_n \equiv x' = -\frac{x}{\tau} + \tanh(\lambda x) - \tanh(\lambda y), y' = -\frac{y}{\tau} + \tanh(\lambda x) + \tanh(\lambda y)$$
$$\phi_n \equiv \tau > 0 \rightarrow \forall \varepsilon{>}0\ \langle \alpha_n \rangle [\alpha_n] \sqrt{x^2+y^2} \leq 2\tau + \varepsilon$$

The verification of ϕ_n uses differentially-defined functions in concert with KeYmaera X's symbolic ODE safety and liveness reasoning [26]. The proof uses a decaying exponential bound $\sqrt{x^2+y^2} \leq \exp(-\frac{t}{\tau})\sqrt{x_0^2+y_0^2} + 2\tau(1-\exp(-\frac{t}{\tau}))$, where the constants x_0, y_0 are symbolic initial values for x, y at initial time $t = 0$, respectively. Notably, the arithmetic subgoals from this example are all proved using abstraction and differential unfolding (Section 3) without relying on external arithmetic solver support for tanh.

Longitudinal flight dynamics. The differential equations α_a below describe the 6th order longitudinal motion of an airplane while climbing or descending [10,25]. The airplane adjusts its pitch angle θ with pitch rate q, which determines its axial velocity u and vertical velocity w, and, in turn, range x and altitude z (illustrated on the right). The physical parameters are: gravity g, mass m, aerodynamic thrust and moment M along the lateral axis, aerodynamic and thrust forces X, Z along x and z, respectively, and the moment of inertia I_{yy}, see [10, Section 6.2].



$$\alpha_a \equiv u' = \frac{X}{m} - g\sin(\theta) - qw, \qquad w' = \frac{Z}{m} + g\cos(\theta) + qu, \qquad q' = \frac{M}{I_{yy}},$$
$$x' = \cos(\theta)u + \sin(\theta)w, \qquad z' = -\sin(\theta)u + \cos(\theta)w, \qquad \theta' = q$$

The verification of specification $J \rightarrow [\alpha_a]J$ shows that the safety envelope $J \equiv J_1 \wedge J_2 \wedge J_3$ is invariant along the flow of α_a with algebraic invariants J_i:

$$J_1 \equiv \frac{Mz}{I_{yy}} + g\theta + \left(\frac{X}{m} - qw\right)\cos(\theta) + \left(\frac{Z}{m} + qu\right)\sin(\theta) = 0$$
$$J_2 \equiv \frac{Mz}{I_{yy}} - \left(\frac{Z}{m} + qu\right)\cos(\theta) + \left(\frac{X}{m} - qw\right)\sin(\theta) = 0 \quad J_3 \equiv -q^2 + \frac{2M\theta}{I_{yy}} = 0$$

Additional examples are available in the supplement [8], including: a bouncing ball on a sinusoidal surface [6,13] and a robot collision avoidance model [15].

5 Conclusion

This work presents a convenient mechanism for extending the dL term language with differentially-defined functions, thereby furthering the class of real-world

systems amenable to modeling and formalization in KeYmaera X. Minimal soundness-critical changes are made to the KeYmaera X kernel, which maintains its trustworthiness while allowing the use of newly defined functions in concert with all existing dL hybrid systems reasoning principles implemented in KeYmaera X. Future work could formally verify these kernel changes by extending the existing formalization of dL [3]. Further integration of external arithmetic tools [1,9,23] will also help to broaden the classes of arithmetic sub-problems that can be solved effectively in hybrid systems proofs.

Acknowledgments. We thank the anonymous reviewers for their helpful feedback on this paper. This material is based upon work supported by the National Science Foundation under Grant No. CNS-1739629. This research was sponsored by the AFOSR under grant number FA9550-16-1-0288.

References

1. Akbarpour, B., Paulson, L.C.: MetiTarski: An automatic theorem prover for real-valued special functions. J. Autom. Reasoning **44**(3), 175–205 (2010). https://doi.org/10.1007/s10817-009-9149-2
2. Bohrer, B., Fernández, M., Platzer, A.: dlι: Definite descriptions in differential dynamic logic. In: Fontaine, P. (ed.) CADE. LNCS, vol. 11716, pp. 94–110. Springer (2019). https://doi.org/10.1007/978-3-030-29436-6_6
3. Bohrer, B., Rahli, V., Vukotic, I., Völp, M., Platzer, A.: Formally verified differential dynamic logic. In: Bertot, Y., Vafeiadis, V. (eds.) CPP. pp. 208–221. ACM (2017). https://doi.org/10.1145/3018610.3018616
4. Boldo, S., Lelay, C., Melquiond, G.: Formalization of real analysis: a survey of proof assistants and libraries. Math. Struct. Comput. Sci. **26**(7), 1196–1233 (2016). https://doi.org/10.1017/S0960129514000437
5. Bonichon, R., Delahaye, D., Doligez, D.: Zenon : An extensible automated theorem prover producing checkable proofs. In: Dershowitz, N., Voronkov, A. (eds.) LPAR. LNCS, vol. 4790, pp. 151–165. Springer (2007). https://doi.org/10.1007/978-3-540-75560-9_13
6. Denman, W.: Automated verification of continuous and hybrid dynamical systems. Ph.D. thesis, University of Cambridge, UK (2015)
7. Fulton, N., Mitsch, S., Quesel, J., Völp, M., Platzer, A.: KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In: Felty, A.P., Middeldorp, A. (eds.) CADE. LNCS, vol. 9195, pp. 527–538. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_36
8. Gallicchio, J., Tan, Y.K., Mitsch, S., Platzer, A.: Implicit definitions with differential equations for KeYmaera X (system description). CoRR **abs/2203.01272** (2022), http://arxiv.org/abs/2203.01272
9. Gao, S., Kong, S., Clarke, E.M.: dReal: An SMT solver for nonlinear theories over the reals. In: Bonacina, M.P. (ed.) CADE. LNCS, vol. 7898, pp. 208–214. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38574-2_14
10. Ghorbal, K., Platzer, A.: Characterizing algebraic invariants by differential radical invariants. In: Ábrahám, E., Havelund, K. (eds.) TACAS. LNCS, vol. 8413, pp. 279–294. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_19

11. Gödel, K.: Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. Monatshefte Math. Phys. **38**(1), 173–198 (1931). https://doi.org/10.1007/BF01700692
12. Khalil, H.K.: Nonlinear systems. Macmillan Publishing Company, New York (1992)
13. Liu, J., Zhan, N., Zhao, H., Zou, L.: Abstraction of elementary hybrid systems by variable transformation. In: Bjørner, N., de Boer, F.S. (eds.) FM. LNCS, vol. 9109, pp. 360–377. Springer (2015). https://doi.org/10.1007/978-3-319-19249-9_23
14. Mitsch, S.: Implicit and explicit proof management in KeYmaera X. In: Proença, J., Paskevich, A. (eds.) F-IDE. EPTCS, vol. 338, pp. 53–67 (2021). https://doi.org/10.4204/EPTCS.338.8
15. Mitsch, S., Ghorbal, K., Vogelbacher, D., Platzer, A.: Formal verification of obstacle avoidance and navigation of ground robots. I. J. Robotics Res. **36**(12), 1312–1340 (2017). https://doi.org/10.1177/0278364917733549
16. Mitsch, S., Platzer, A.: A retrospective on developing hybrid systems provers in the KeYmaera family - A tale of three provers. In: Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Ulbrich, M. (eds.) Deductive Software Verification: Future Perspectives - Reflections on the Occasion of 20 Years of KeY, LNCS, vol. 12345, pp. 21–64. Springer (2020). https://doi.org/10.1007/978-3-030-64354-6_2
17. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reasoning **41**(2), 143–189 (2008). https://doi.org/10.1007/s10817-008-9103-8
18. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14509-4
19. Platzer, A.: The complete proof theory of hybrid systems. In: LICS. pp. 541–550. IEEE Computer Society (2012). https://doi.org/10.1109/LICS.2012.64
20. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reasoning **59**(2), 219–265 (2017). https://doi.org/10.1007/s10817-016-9385-1
21. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-63588-0
22. Platzer, A., Tan, Y.K.: Differential equation invariance axiomatization. J. ACM **67**(1) (2020). https://doi.org/10.1145/3380825
23. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation-based abstraction refinement. ACM Trans. Embed. Comput. Syst. **6**(1), 8 (2007). https://doi.org/10.1145/1210268.1210276
24. Richardson, D.: Some undecidable problems involving elementary functions of a real variable. J. Symb. Log. **33**(4), 514–520 (1968). https://doi.org/10.2307/2271358
25. Stengel, R.F.: Flight Dynamics. Princeton University Press (2004)
26. Tan, Y.K., Platzer, A.: An axiomatic approach to existence and liveness for differential equations. Formal Aspects Comput. **33**(4), 461–518 (2021). https://doi.org/10.1007/s00165-020-00525-0
27. Wiedijk, F.: Stateless HOL. In: Hirschowitz, T. (ed.) TYPES. EPTCS, vol. 53, pp. 47–61 (2009). https://doi.org/10.4204/EPTCS.53.4