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

James Gallicchio **Yong Kiam Tan** Stefan Mitsch André Platzer

Computer Science Department, Carnegie Mellon University

IJCAR, 10 Aug 2022

Outline

1. Hybrid System Verification
2. Implicit Definitions in Differential Dynamic Logic
3. Implementation in KeYmaera X
4. Conclusion

Outline

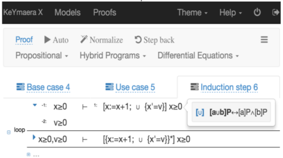
Motivation: Cyber-Physical Systems (CPSs)



Challenge: How can we formally ensure correctness for cyber-physical systems that feature interacting discrete and continuous dynamics?

Motivation: Cyber-Physical Systems (CPSs)

Model as hybrid system & specify correctness

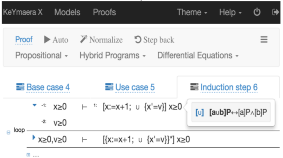


Hybrid system verification tool

Challenge: How can we formally ensure correctness for cyber-physical systems that feature interacting discrete and continuous dynamics?

Motivation: Cyber-Physical Systems (CPSs)

Model as hybrid system & specify correctness

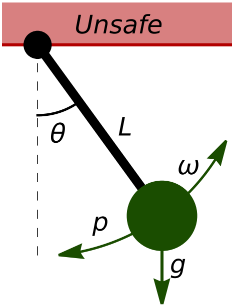


Hybrid system verification tool

Toy Example: safely pushed swing

Challenge: How can we formally ensure correctness for cyber-physical systems that feature interacting discrete and continuous dynamics?

Safely Pushed Swing

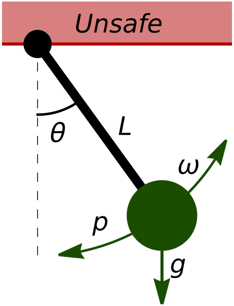


Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

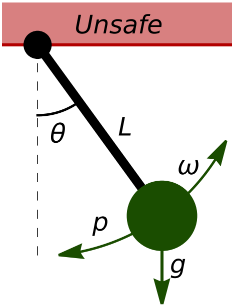
Safety:
Swing stays below horizontal

Challenges:

- Hybrid system model + specification

Need adequate modeling of interacting discrete & continuous dynamics

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

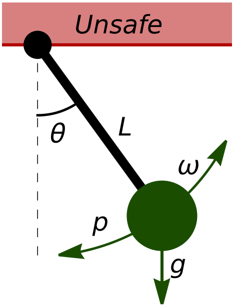
Safety:
Swing stays below horizontal

Challenges:

- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Challenges:

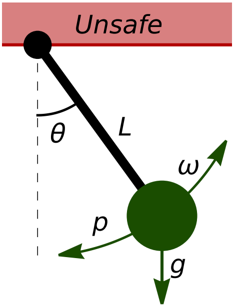
- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

- Proving safety & correctness

Need sound + (semi-)automated reasoning for hybrid dynamics

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Challenges:

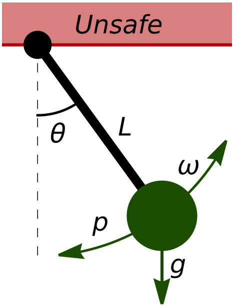
- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

- Proving safety & correctness

✓KeYmaera X theorem prover

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Challenges:

- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

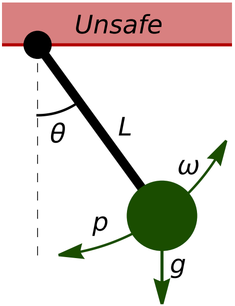
- Proving safety & correctness

✓KeYmaera X theorem prover

- User-defined functions

Need extensible support for new defs.

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Challenges:

- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

- Proving safety & correctness

✓KeYmaera X theorem prover

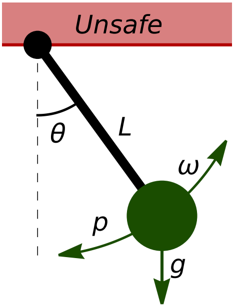
- User-defined functions

Need extensible support for new defs.

$$\sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \dots$$

Series defs. in foundational provers
×Lose hybrid system support & autom.

Safely Pushed Swing



Discrete controlled pushes p
Continuous ODEs:

$$\theta' = \omega, \omega' = -\frac{g}{L}\sin(\theta) - k\omega$$

Safety:
Swing stays below horizontal

Challenges:

- Hybrid system model + specification

✓Differential Dynamic Logic (dL)

- Proving safety & correctness

✓KeYmaera X theorem prover

- User-defined functions

✓**This Work:**
Definitions package for user-defined functions in dL and KeYmaera X

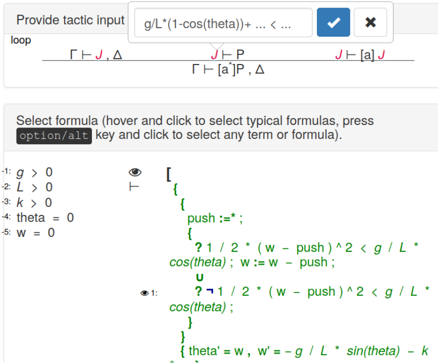
Domain-specific support for hybrid systems
e.g., $\sin(\theta)$ solves $s' = c, c' = -s$

KeYmaera X Package for Implicit Definitions

Modeling Interface:

```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
Problem
  g > 0 & L > 0 & k > 0 &
  theta = 0 & w = 0 /* Swing starts at rest */
  ->
  [{
    /* Discrete push allowed if it is safe to do so */
    {
      push :=*;
      if (1/2*(w-push)^2 < g/L *cos(theta))
        { w := w-push; }
    }
    /* Continuous dynamics */
    { theta' = w, w' = -g/L * sin(theta) - k*w }
  }*]
  /* Swing never crosses horizontal */
  (-pi()/2 < theta & theta < pi()/2)
End.
```

Proof Interface:

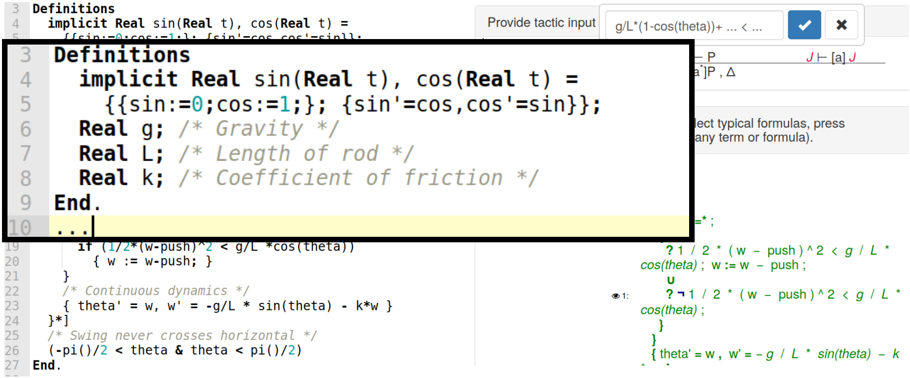


KeYmaera X Package for Implicit Definitions

Users define their desired functions using sugared syntax in KeYmaera X.

Proof Interface:

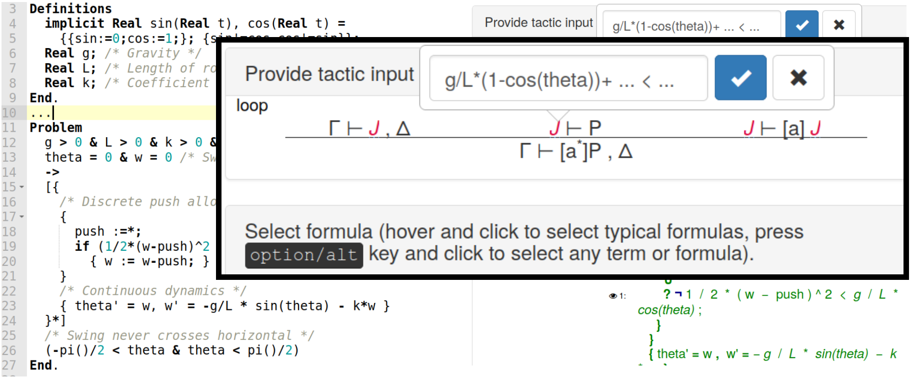
```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
```



KeYmaera X Package for Implicit Definitions

Users define their desired functions using sugared syntax in KeYmaera X.

Seamlessly use functions throughout existing specifications and proof methods.

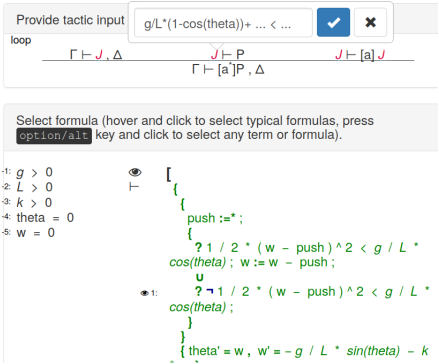


KeYmaera X Package for Implicit Definitions

Users define their desired functions using sugared syntax in KeYmaera X.

Seamlessly use functions throughout existing specifications and proof methods.

```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
Problem
  g > 0 & L > 0 & k > 0 &
  theta = 0 & w = 0 /* Swing starts at rest */
  ->
  [{
    /* Discrete push allowed if it is safe to do so */
    {
      push :=*;
      if (1/2*(w-push)^2 < g/L *cos(theta))
        { w := w-push; }
    }
    /* Continuous dynamics */
    { theta' = w, w' = -g/L * sin(theta) - k*w }
  }*]
  /* Swing never crosses horizontal */
  (-pi()/2 < theta & theta < pi()/2)
End.
```



Proof: ✔ All goals closed

Export proof | Browse proof | Redo proof

```
Provable(  ==>  g()>0&L()>0&k()>0&theta=0&w=0->... proved)
```

Outline

Background: Differential Dynamic Logic (dL)

Hybrid programs model hybrid systems; terms in red (polynomials, etc.)

$$\alpha, \beta \;::=\; x' = f(x) \,\&\, Q \mid x := e \mid ?Q \mid \alpha ; \beta \mid \alpha \cup \beta \mid \alpha^*$$

ODE | Assign | Test | Compose | Choice | Loop

Background: Differential Dynamic Logic (dL)

Hybrid programs model hybrid systems; terms in red (polynomials, etc.)

$$\alpha, \beta \ ::= \ x' = f(x) \,\&\, Q \mid x := e \mid ?Q \mid \alpha ; \beta \mid \alpha \cup \beta \mid \alpha^*$$

ODE | Assign | Test | Compose | Choice | Loop

Properties of hybrid program α are specified in dL's **formula** language.

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

Compare $\geq, >, =$ | And, Or, etc. | For all, Exists | ϕ true after all α runs | ϕ true after some α run

Background: Differential Dynamic Logic (dL)

Hybrid programs model hybrid systems; terms in red (polynomials, etc.)

$$\alpha, \beta \ ::= \ x' = f(x) \,\&\, Q \mid x := e \mid ?Q \mid \alpha; \beta \mid \alpha \cup \beta \mid \alpha^*$$

ODE | Assign | Test | Compose | Choice | Loop

Properties of hybrid program α are specified in dL's **formula** language.

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

Compare $\geq, >, =$ | And, Or, etc. | For all, Exists | ϕ true after all α runs | ϕ true after some α run

This Work: Expand term language with **implicitly defined functions**

$$f_{\ll\phi\gg}(t) = x \leftrightarrow \phi(x, t)$$

Function $f_{\ll\phi\gg}$ is interpreted using its **graph** characterized by ϕ.

Background: Differential Dynamic Logic (dL)

Hybrid programs model hybrid systems; terms in red (polynomials, etc.)

$$\alpha, \beta \ ::= \ x' = f(x) \,\&\, Q \mid x := e \mid ?Q \mid \alpha ; \beta \mid \alpha \cup \beta \mid \alpha^*$$

ODE | Assign | Test | Compose | Choice | Loop

Properties of hybrid program α are specified in dL's **formula** language.

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

Compare $\geq, >, =$ | And, Or, etc. | For all, Exists | ϕ true after all α runs | ϕ true after some α run

This Work: Expand term language w

$$f_{\ll \phi \gg}(t) =$$

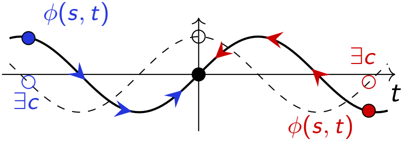
n.b. Not all dL formulas characterize graphs of (suitable) functions (see paper for restrictions).

Function $f_{\ll \phi \gg}$ is interpreted using its **graph** characterized by ϕ.

Differentially-Defined Functions

Example: Implicitly defined trigonometric sine function $\sin(t) = s$

$$\phi(s,t) \equiv \underbrace{\exists c \left\langle \begin{array}{l} \color{red}{s' = -c, c' = s, t' = -1 \cup} \\ \color{blue}{s' = c, c' = -s, t' = 1} \end{array} \right\rangle}_{ODE} \underbrace{\begin{pmatrix} s=0 \land \\ c=1 \land \\ t=0 \end{pmatrix}}_{Init.}$$

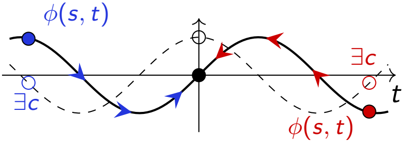


Intuition: Initial point is reachable by following ODE forward or backward.
$\Rightarrow \phi(s,t)$ characterizes graph of $\sin(t)$; similar characterization for $\cos(t)$.

Differentially-Defined Functions

Example: Implicitly defined trigonometric sine function $\sin(t) = s$

$$\phi(s,t) \equiv \exists c \underbrace{\left\langle \begin{array}{l} s' = -c, c' = s, t' = -1 \cup \\ s' = c, c' = -s, t' = 1 \end{array} \right\rangle}_{ODE} \underbrace{\begin{pmatrix} s = 0 \wedge \\ c = 1 \wedge \\ t = 0 \end{pmatrix}}_{Init.}$$



Intuition: Initial point is reachable by following ODE forward or backward.
$\Rightarrow \phi(s,t)$ characterizes graph of $\sin(t)$; similar characterization for $\cos(t)$.

General Case: Any projection of an ODE system solution is implicitly characterizable in dL (soundness proof in paper).

Thm. [JACM'20]: dL extended with Noetherian functions (incl. solutions of polynomial ODEs) has sound and complete ODE invariance reasoning.

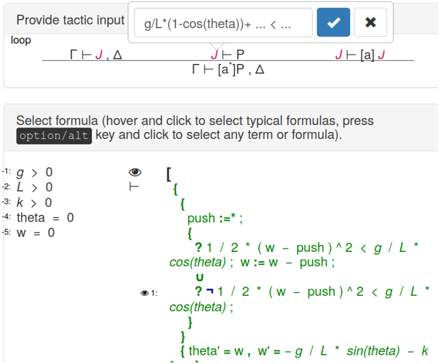
Outline

1. Hybrid System Verification
2. Implicit Definitions in Differential Dynamic Logic
3. Implementation in KeYmaera X
4. Conclusion

Modeling Interface:

```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
Problem
  g > 0 & L > 0 & k > 0 &
  theta = 0 & w = 0 /* Swing starts at rest */
  ->
  [{
    /* Discrete push allowed if it is safe to do so */
    {
      push :=*;
      if (1/2*(w-push)^2 < g/L *cos(theta))
        { w := w-push; }
    }
    /* Continuous dynamics */
    { theta' = w, w' = -g/L * sin(theta) - k*w }
  }*]
  /* Swing never crosses horizontal */
  (-pi()/2 < theta & theta < pi()/2)
End.
```

Proof Interface:



Implementation Details

Soundness-critical changes: Syntax & axiom schema for implicit defs.
Follows KeYmaera X's small trusted kernel design, $\approx$170 lines extension

Implementation Details

Soundness-critical changes: Syntax & axiom schema for implicit defs. Follows KeYmaera X's small trusted kernel design, ≈170 lines extension

Non-critical (core-adjacent): Syntactic sugar for parsing and UI pretty-printing of user-defined functions

Implementation Details

Soundness-critical changes: Syntax & axiom schema for implicit defs. Follows KeYmaera X's small trusted kernel design, ≈170 lines extension

Non-critical (core-adjacent): Syntactic sugar for parsing and UI pretty-printing of user-defined functions

Non-critical (user automation):

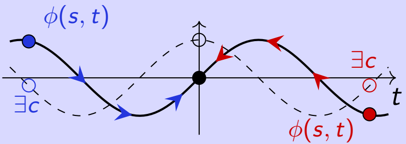
- Auto. derive base properties of functions from underlying ODEs:

Initial value: $\sin(0) = 0$

```
Provable(  ==>  sin<< ... >>(0)=0 proved)
```

Derivative: $\sin(e)' = \cos(e)(e)'$

```
Provable(  ==>
  (sin<< ... >>(x))'=cos<< ... >>(x)*(x)' proved)
```



- Prove additional arithmetic properties with ODE analysis (next slide)

Specialized Arithmetic Support

Adapt existing KeYmaera X sound abstraction & ODE analysis
$+$ arithmetic export to external real arithmetic solvers

$$x(\tanh(\lambda x) - \tanh(\lambda y)) + y(\tanh(\lambda x) + \tanh(\lambda y)) \leq 2\sqrt{x^2 + y^2}$$

Specialized Arithmetic Support

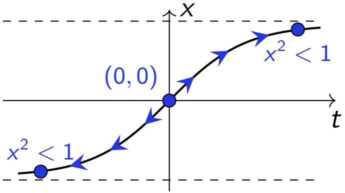
Adapt existing KeYmaera X sound abstraction & ODE analysis
+ arithmetic export to external real arithmetic solvers

ODE analysis

$$\tanh(\lambda x)^2 < 1 \land \tanh(\lambda y)^2 < 1 \rightarrow$$
$$x(\tanh(\lambda x) - \tanh(\lambda y)) + y(\tanh(\lambda x) + \tanh(\lambda y)) \leq 2\sqrt{x^2 + y^2}$$

Claim: $\tanh(t)^2 < 1$ for all t.

Intuition: Property is always preserved along ODE, forward and backward from initial point.



Specialized Arithmetic Support

Adapt existing KeYmaera X sound abstraction & ODE analysis
+ arithmetic export to external real arithmetic solvers

ODE analysis

$$\tanh(\lambda x)^2 < 1 \land \tanh(\lambda y)^2 < 1 \rightarrow$$
$$x(\tanh(\lambda x) - \tanh(\lambda y)) + y(\tanh(\lambda x) + \tanh(\lambda y)) \leq 2\sqrt{x^2 + y^2}$$
$$\Downarrow$$

Abstraction (replace tanh with fresh variables):

$${t_x}^2 < 1 \land {t_y}^2 < 1 \rightarrow x(t_x - t_y) + y(t_x + t_y) \leq 2\sqrt{x^2 + y^2}$$

Specialized Arithmetic Support

Adapt existing KeYmaera X sound abstraction & ODE analysis
+ arithmetic export to external real arithmetic solvers

ODE analysis

$$\tanh(\lambda x)^2 < 1 \land \tanh(\lambda y)^2 < 1 \rightarrow$$
$$x(\tanh(\lambda x) - \tanh(\lambda y)) + y(\tanh(\lambda x) + \tanh(\lambda y)) \leq 2\sqrt{x^2 + y^2}$$

$$\Downarrow$$

Abstraction (replace tanh with fresh variables):

$${t_x}^2 < 1 \land {t_y}^2 < 1 \rightarrow x(t_x - t_y) + y(t_x + t_y) \leq 2\sqrt{x^2 + y^2}$$

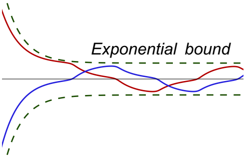
$$\Downarrow$$

Provable by solvers without native support for tanh

Proof: ✔ All goals closed

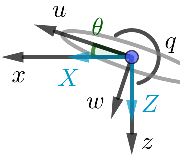
```
Provable(  ==>
  x*(tanh<< ... >>(lambda()*x)-tanh<< ... >>(lambda()*y)) +
  y*(tanh<< ... >>(lambda()*x)+tanh<< ... >>(lambda()*y)) <=
  2*(x^2+y^2)^(1/2) proved)
```

Examples (see paper [IJCAR'22])



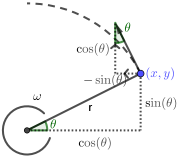
2 neuron interaction, asymptotic norm bound

```
Problem
  tau > 0 -> \forall eps( eps > 0 ->
    <{
      x' = -x/tau + tanh(lambda*x)
         - tanh(lambda*y),
      y' = -y/tau + tanh(lambda*x)
         + tanh(lambda*y)
    }>
    [{
      x' = -x/tau + tanh(lambda*x)
         - tanh(lambda*y),
      y' = -y/tau + tanh(lambda*x)
         + tanh(lambda*y)
    }]
    (x^2 + y^2)^(1/2) <= (2*tau + eps)
  )
End.
```

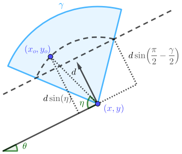


Invariants of longitudinal flight dynamics

```
Definitions
  ...
  Real inv1(Real z, Real u, Real w,
            Real theta, Real q) =
    M*z/Iyy + g*theta
    + (X/m-q*w)*cos(theta)
    + (Z/m+q*u)*sin(theta);
  ...
End.
...
Problem
  assmpts() & inv(x,z,u,w,theta,q)
  -> [motion;]inv(x,z,u,w,theta,q)
End.
```



Robot collision avoid., trajectory & vision limits



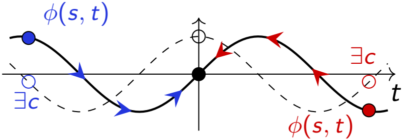
Takeaway: Package enables succinct models and powerful reasoning support for user-defined functions in KeYmaera X.

Outline

Summary

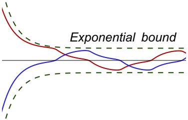
Theory: Implicit defs. in dL

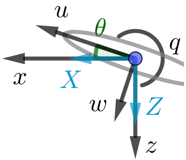
$$f_{\ll\phi\gg}(t) = x \leftrightarrow \phi(x, t)$$

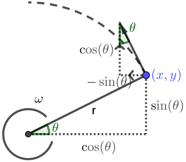


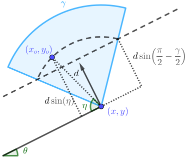
Practice: KeYmaera X package

```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
```





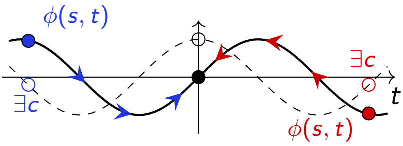




Summary

Theory: Implicit defs. in dL

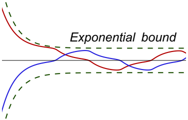
$$f_{\ll\phi\gg}(t) = x \leftrightarrow \phi(x,t)$$

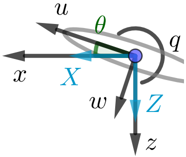


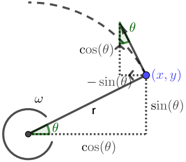
Future Work: Defining and reasoning about multivariate & non-smooth functions in dL

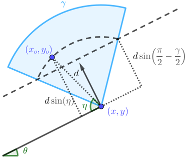
Practice: KeYmaera X package

```
Definitions
  implicit Real sin(Real t), cos(Real t) =
    {{sin:=0;cos:=1;}; {sin'=cos,cos'=sin}};
  Real g; /* Gravity */
  Real L; /* Length of rod */
  Real k; /* Coefficient of friction */
End.
...
```









Check it out: `http://keymaerax.org/keymaeraXfunc/`

References I

[1] Gallicchio, J., Tan, Y. K., Mitsch, S., and Platzer, A. (2022). Implicit definitions with differential equations for KeYmaera X (system description). In Blanchette, J., Kovacs, L., and Pattinson, D., editors, IJCAR, volume 13385 of LNCS, pages 723–733. Springer.

[2] Platzer, A. and Tan, Y. K. (2020). Differential equation invariance axiomatization. J. ACM, 67(1):6:1–6:66.