Embedding Differential Dynamic Logic in PVS

J. Tanner Slagel,[1] Mariano Moscato,[2] Lauren White[1]
César Muñoz*,[1] Swee Balachandran†,[2] Aaron Dutle[1]

[1] NASA Langley Research Center
Hampton, VA, USA
j.tanner.slagel@nasa.gov

[2] National Institute of Aerospace
Hampton, VA, USA

* Currently at Amazon Web Services
† Currently at Xwing

July 2023 @ LSFA 2023: 18th International Workshop on Logical and Semantic Frameworks, with Applications

Overview

- **dL:** Differential Dynamic Logic for hybrid programs [1]
- **PVS:** Interactive theorem prover [2]

Result: Plaidypvs [3]

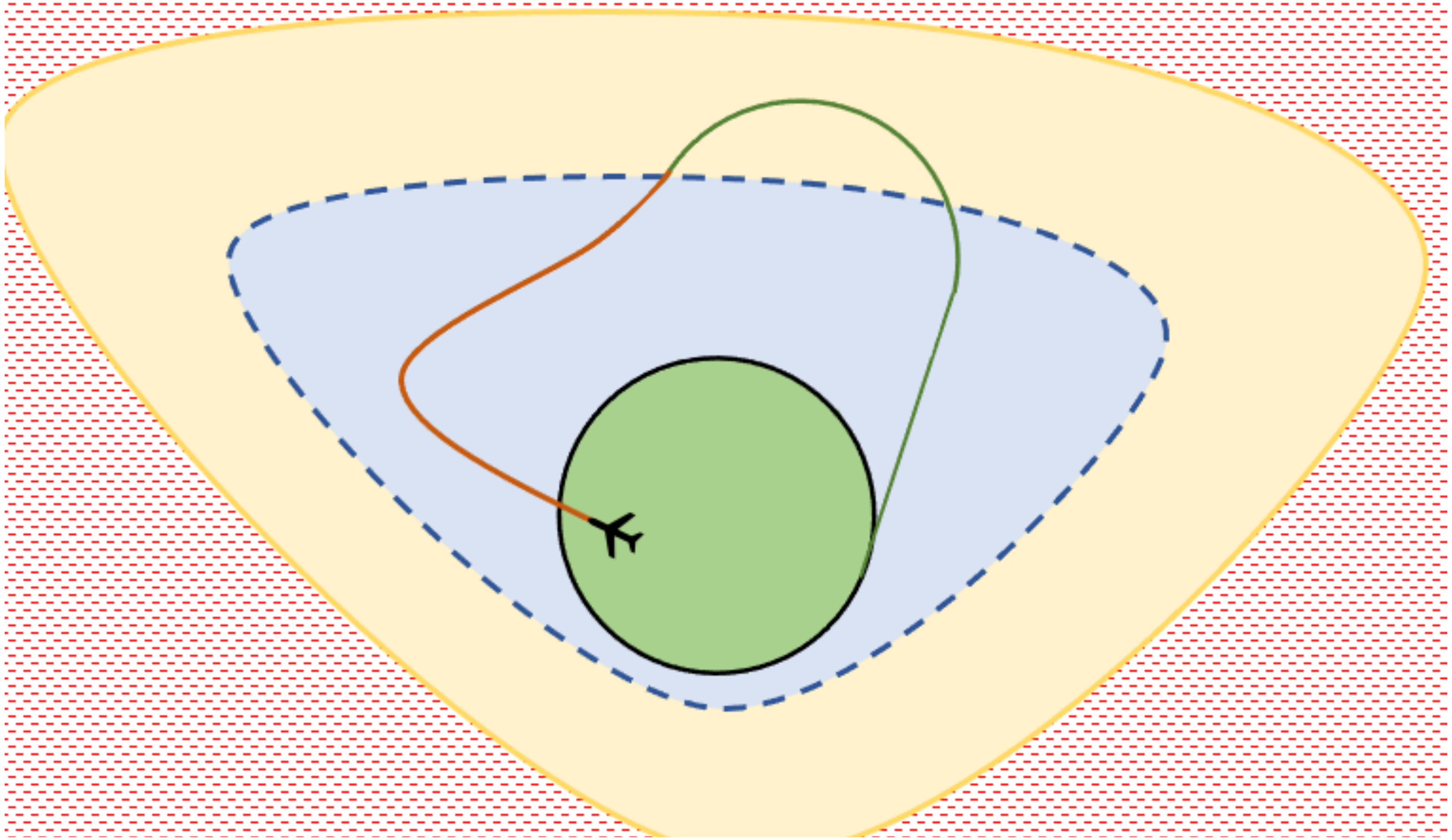
- Formally verified soundness of **dL**
- Fully operational in **PVS**
- Leveraging features of **PVS** to extend **dL**

[1] Differential Dynamic Logic website, André Platzer: https://symbolaris.com/logic/dL.html

[2] PVS website, SRI International: https://pvs.csl.sri.com

[3] Plaidypvs in NASA PVS Library: https://github.com/nasa/pvslib/tree/master/dL

Hybrid Systems



- **Hybrid system:** dynamical system that exhibits
  - Continuous behavior
  - Discrete behavior

Want

- **Formal specification** of hybrid systems
- **Formal reasoning** of hybrid systems

Hybrid Programs

Hybrid programs allow **formal specification** of hybrid systems:

- Discrete jump set:

$$(x_1 := \theta_1, \dots, x_n := \theta_n)$$

- Differential equations:

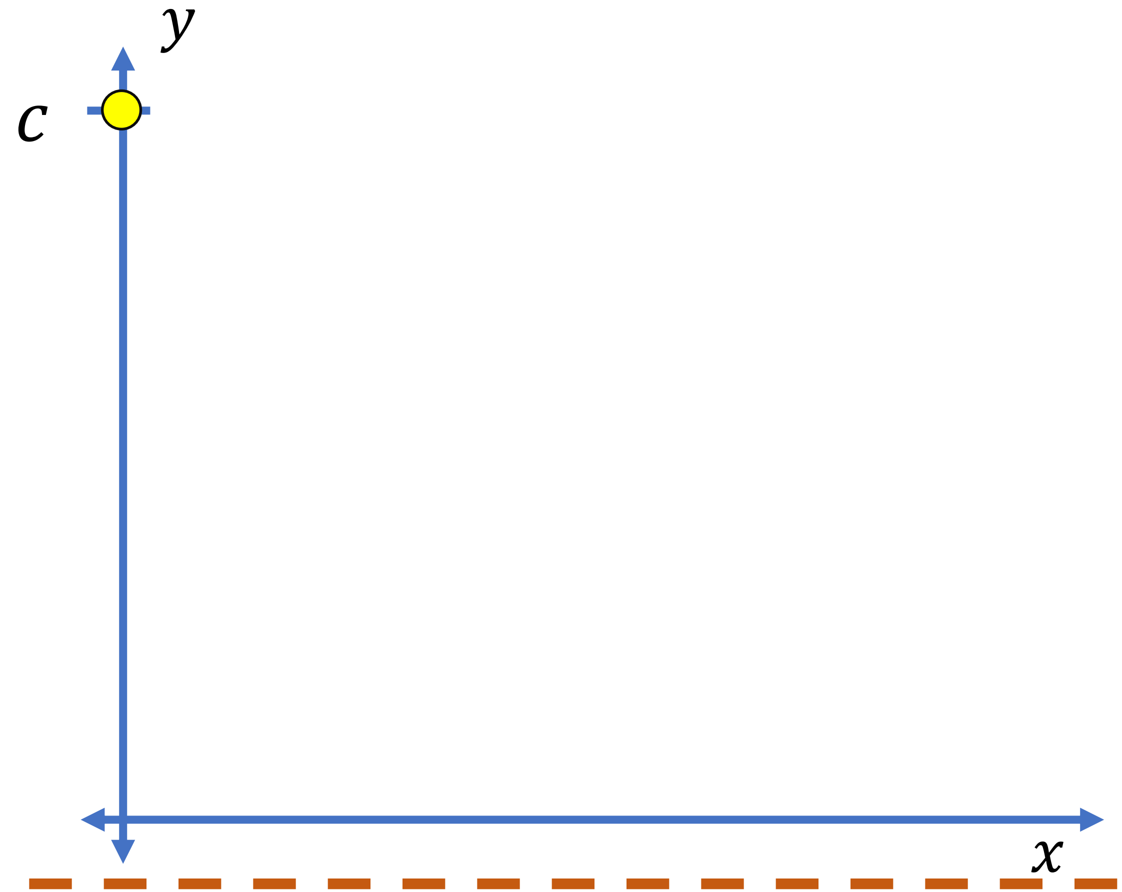
$$\{x_1' := \theta_1, \dots, x_n' := \theta_n \ \& \ \chi\}$$

  - $\{x_i\}_{i=1}^{n}$ variables
  - $\{\theta_i\}_{i=1}^{n}$ assignments (e.g. – functions of existing variable values)
  - χ first order formula that describes domain

Example:

$$(x := 0\,, y := c\,); \{x' = y, y' = -x \ \& \ y \geq 0\}$$

Hybrid Programs



Example:

$$(x := 0\,, y := c\,); \{x' = y, y' = -x \;\&\; y \geq 0\}$$

Hybrid Programs (continued)

For hybrid programs Hp_1, Hp_2, first-order formula χ:

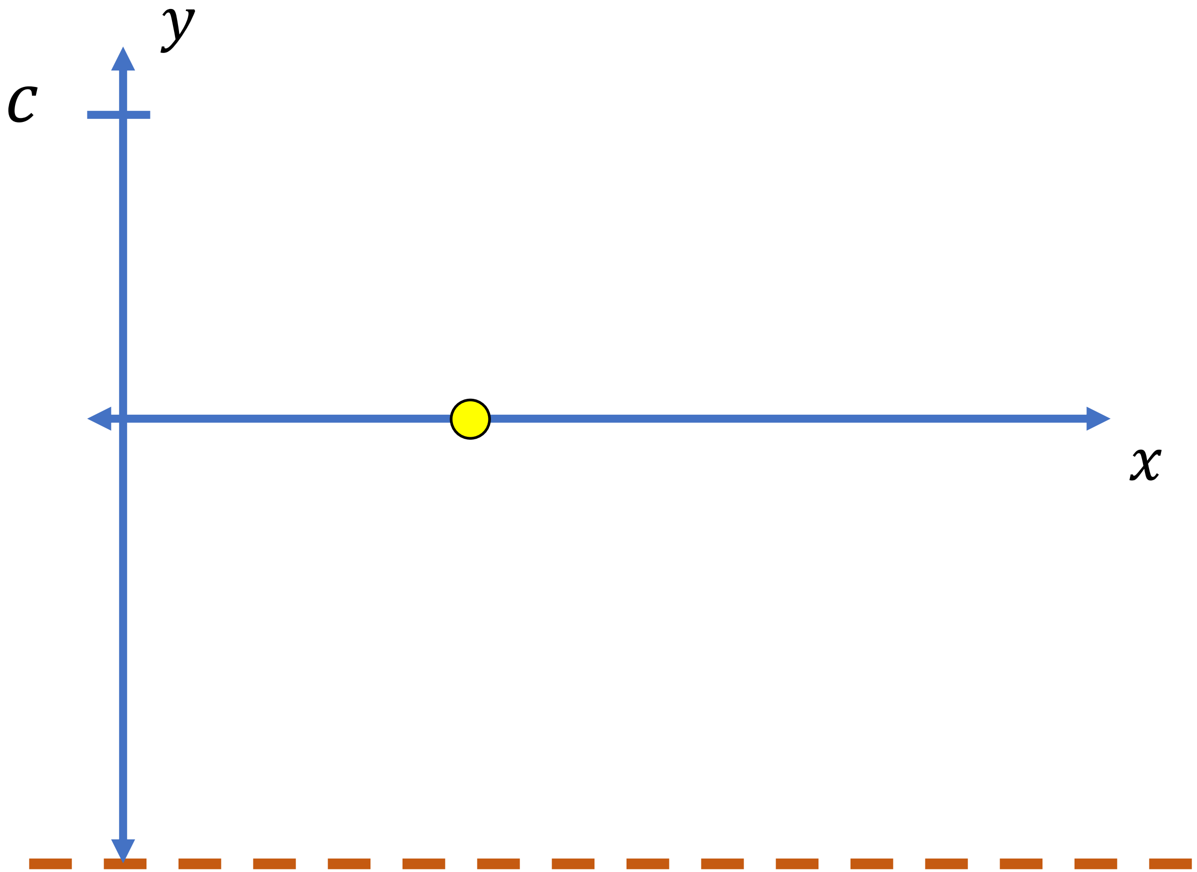
- Choice $(Hp_1 \cup Hp_2)$
- Sequence $(Hp_1; Hp_2)$
- Repeat $(Hp_1)^*$
- Test $(?\,\chi)$

Example:
$$\left\{\begin{array}{c}\big((?\,y > 0); \{x' = y, y' = -x \ \&\ y \geq 0\}\big) \\ \cup \\ \big((?\,y \leq 0); \{y' = -c\}\big)\end{array}\right\}^*$$

Hybrid Programs (continued)

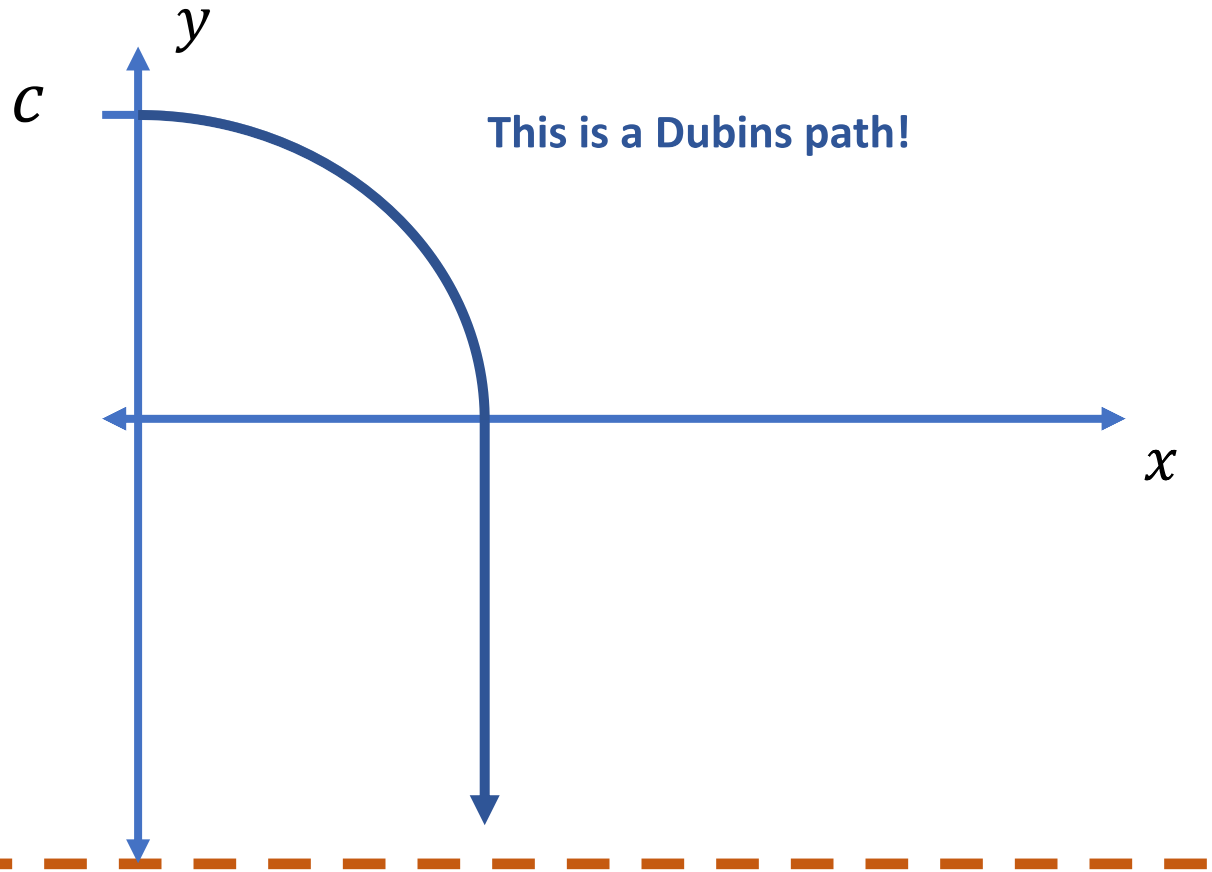
Example: $\left\{ \begin{array}{c} \big((?\,y > 0); \{x' = y, y' = -x \ \& \ y \geq 0\}\big) \\ \cup \\ \big((?\,y \leq 0); \{y' = -c\ \}\big) \end{array} \right\}^*$

Hybrid Programs (continued)



Example: $$\left\{\begin{array}{c} \left((?\,y > 0);\{x' = y, y' = -x \;\&\; y \geq 0\}\right) \\ \cup \\ \left((?\,y \leq 0);\{y' = -c\}\right) \end{array}\right\}^*$$

Hybrid Programs (continued)



This is a Dubins path!

Example:
$$\left\{ \begin{array}{c} ((?\,y > 0); \{x' = y, y' = -x \;\&\; y \geq 0\}) \\ \cup \\ ((?\,y \leq 0); \{y' = -c\}) \end{array} \right\}^*$$

dL: Differential Dynamic Logic

dL allows **formal reasoning** of hybrid programs:

- For hybrid program Hp and predicate P
  - All runs $[Hp]\mathrm{P}$
  - Some runs $\langle Hp\rangle\mathrm{P}$

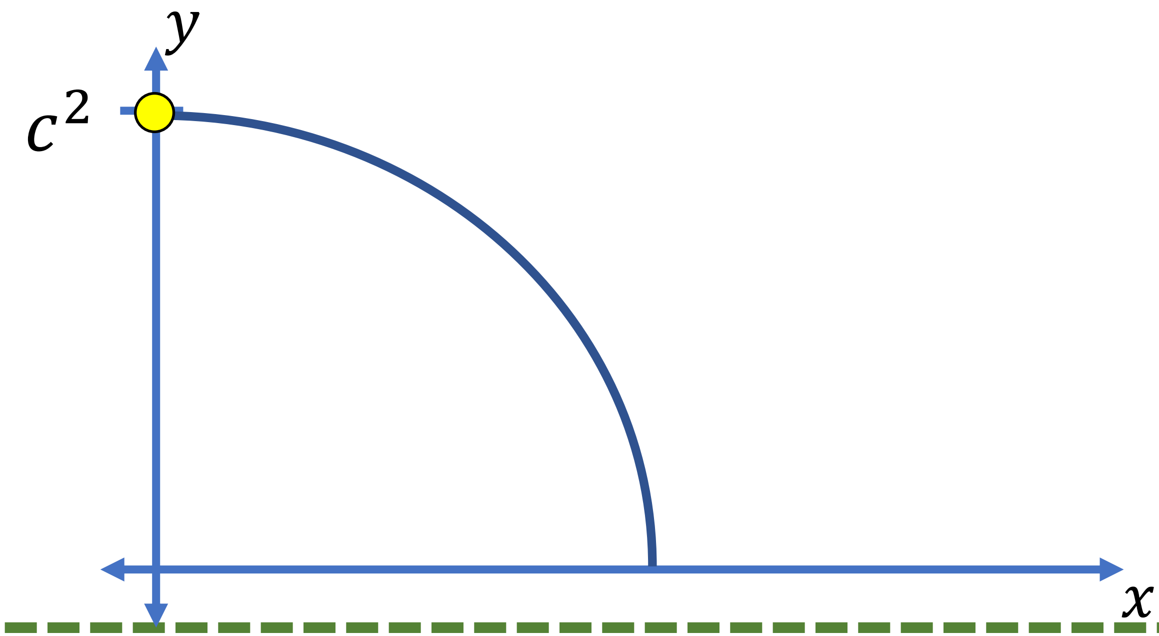
Example: Let

$$Hp \equiv \big((?\,y > 0); \{x' = y, y' = -x \ \& \ y \geq 0\}\big)$$
$$\mathrm{P} \ = (x^2 + y^2 = c^2),$$

then

$$\mathrm{y} = \mathrm{c}^2, \mathrm{x} = 0 \rightarrow [Hp]P \qquad \mathrm{y} = \mathrm{c}\,, \mathrm{x} = 0 \rightarrow \langle Hp\rangle(\mathrm{y} = 0)$$

dL: Differential Dynamic Logic



Example: Let

$$Hp \equiv \big((?\,y > 0); \{x' = y, y' = -x \ \& \ y \geq 0\}\big)$$
$$\mathrm{P} \ = (x^2 + y^2 = c^2),$$

then

$$\mathrm{y} = \mathrm{c}\,, \mathrm{x} = 0 \vdash [Hp]P \qquad \mathrm{y} = \mathrm{c}\,, \mathrm{x} = 0 \vdash \langle Hp \rangle (\mathrm{y} = 0)$$

dL: Differential Dynamic Logic – Rule Schema

Union axiom:

$$\frac{[Hp_1]P \wedge [Hp_2]P}{[Hp_1 \cup Hp_2]P}$$

Loop rule:

$$\frac{\Gamma \vdash J \qquad J \vdash [\alpha]J \qquad J \vdash P}{\Gamma \vdash [\alpha^*]P}$$

Differential invariant rule:

$$\frac{\Gamma, q(x) \vdash p(x) \qquad q(x) \vdash [x' := f(x)](p(x))'}{\Gamma \vdash [x' = f(x) \,\&\, q(x)]p(x)}$$

….and many more! [4]

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

dL: Proof

Differential invariant rule:

$$\frac{\Gamma, q(x) \vdash p(x) \qquad q(x) \vdash [x' := f(x)](p(x))'}{\Gamma \vdash [x' = f(x) \,\&\, q(x)]p(x)}$$

$$\mathrm{y} = \mathrm{c}\,, x = 0 \;\vdash [\{x' = y, y' = -x\}](\mathrm{x}^2 + \mathrm{y}^2 = \mathrm{c}^2)$$

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

dL: Proof

Differential invariant rule:

$$\frac{\Gamma, q(x) \vdash p(x) \qquad q(x) \vdash [x' := f(x)](p(x))'}{\boxed{\Gamma \vdash [x' = f(x) \,\&\, q(x)]p(x)}}$$

$$\boxed{\mathrm{y} = \mathrm{c}\,, x = 0 \;\vdash [\{x' = y, y' = -x\}](\mathrm{x}^2 + \mathrm{y}^2 = \mathrm{c}^2)}$$

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

dL: Proof

Differential invariant rule:

$$\frac{\boxed{\Gamma, q(x) \vdash p(x)} \quad \boxed{q(x) \vdash [x' := f(x)](p(x))'}}{\boxed{\Gamma \vdash [x' = f(x) \,\&\, q(x)]p(x)}}$$

Apply Di rule

$$\frac{\boxed{y = c, x = 0 \ \vdash \mathrm{x}^2 + \mathrm{y}^2 = \mathrm{c}^2} \quad \boxed{\vdash [x' := y, y' := -x](2\,\mathrm{x}\,\mathrm{x}' + 2\,\mathrm{y}\,\mathrm{y}' = 0)}}{\boxed{\mathrm{y} = \mathrm{c}\,, x = 0 \ \vdash [\{x' = y, y' = -x\}](\mathrm{x}^2 + \mathrm{y}^2 = \mathrm{c}^2)}}$$

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

dL: Proof

Differential invariant rule:

$$\frac{\boxed{\Gamma, q(x) \vdash p(x)} \quad \boxed{q(x) \vdash [x' := f(x)](p(x))'}}{\boxed{\Gamma \vdash [x' = f(x) \,\&\, q(x)]p(x)}}$$

$$\frac{\dfrac{\vdash 0^2 + c^2 = c^2 \qquad\qquad \vdash 2\,x\,y + 2\,y\,(-x) = 0}{\text{Apply substitution} \quad \boxed{y = c, x = 0 \vdash x^2 + y^2 = c^2} \quad \boxed{\vdash [x' := y, y' := -x](2\,x\,x' + 2\,y\,y' = 0)}}}{\text{Apply Di rule} \quad \boxed{y = c\,, x = 0 \vdash [\{x' = y, y' = -x\}](x^2 + y^2 = c^2)}}$$

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

dL: Proof

Differential invariant rule:

$$\frac{\boxed{\Gamma, q(x) \vdash p(x)} \quad \boxed{q(x) \vdash [x' := f(x)](p(x))'}}{\boxed{\Gamma \vdash [x' = f(x)\,\&\, q(x)]p(x)}}$$

$$\vdash 0 = 0 \quad ✓$$

Arithmetic! ✓

$$\vdash 0^2 + c^2 = c^2 \qquad \vdash 2\,x\,y + 2\,y\,(-x) = 0$$

Apply substitution

$$\boxed{y = c, x = 0 \vdash x^2 + y^2 = c^2} \qquad \boxed{\vdash [x' := y, y' := -x](2\,x\,x' + 2\,y\,y' = 0)}$$

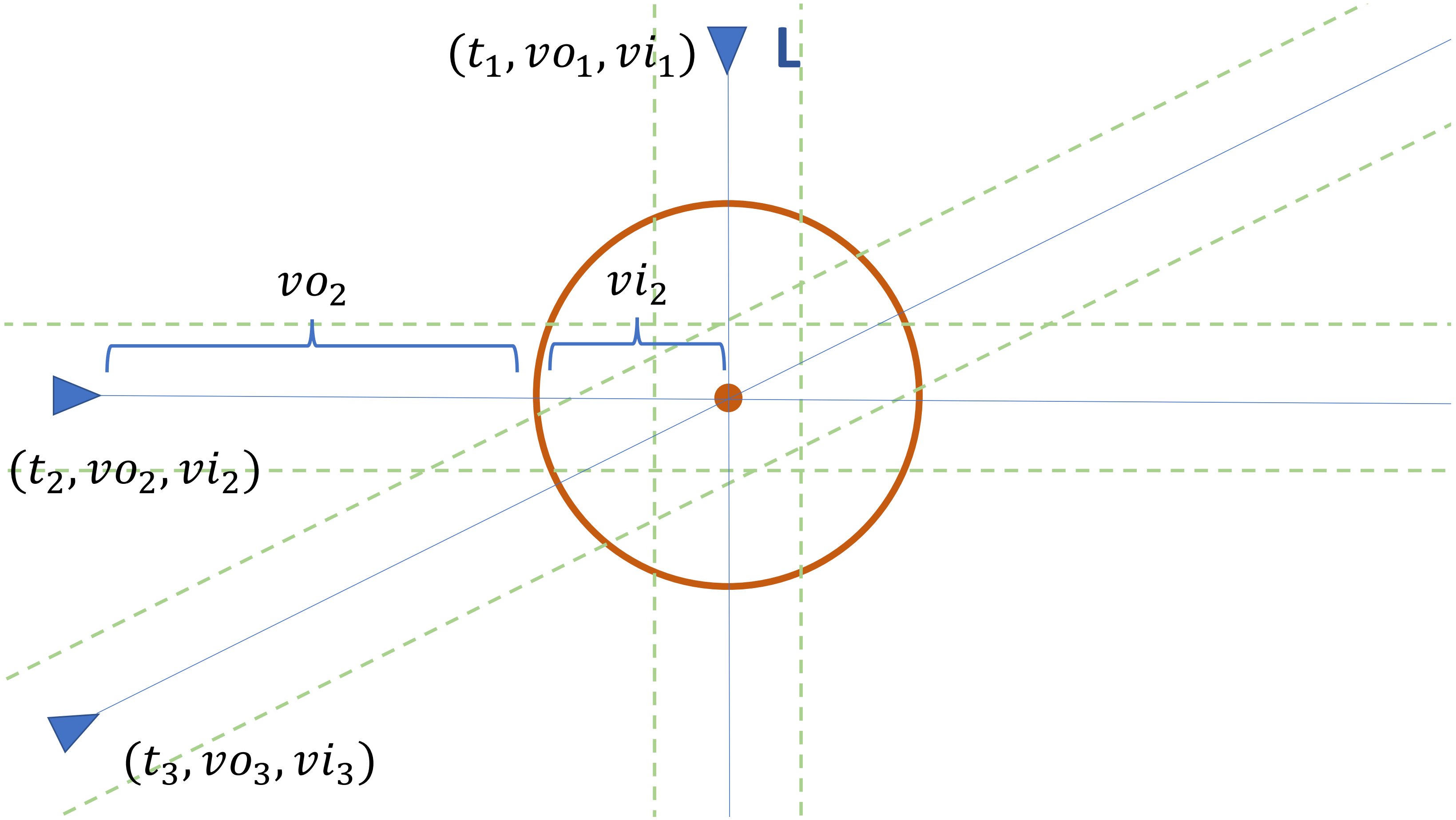
Apply Di rule

$$\boxed{y = c\,, x = 0 \vdash [\{x' = y, y' = -x\}](x^2 + y^2 = c^2)}$$

[4] André Platzer. 2018. Logical Foundations of Cyber-Physical Systems. Springer, Cham. https://doi.org/10.1007/978-3-319-63588-0

Outline

- Introduction
  - Hybrid programs
  - $\mathbf{dL}$
- Plaidypvs
  - PVS
  - Extensions of $\mathbf{dL}$ in PVS
  - Examples
- Concluding remarks



PVS

- “Prototype Verification System” developed by SRI International
- Interactive theorem prover
  - Higher order logic
  - Completely typed, dependent types
- Automation
  - Customizable tactics and strategies
- PVSio animation and rapid prototyping
- NASA PVS library [5]
  - 58 libraries
- Visual studio code extension [6]

```
deriv_test :

{-1}   b >= 0
{-2}   b > 0
{-3}   c /= 0
  |-------
[1]    deriv(LAMBDA (x: real): cos(x ^ 10 + b) + exp(x ^ 2) / c) =
        LAMBDA (x: real):
          -sin(x ^ 10 + b) * 10 * x ^ 9 + exp(x ^ 2) * 2 * x / c

>> (deriv)

Q.E.D.
```

[5] NASAlib, maintained by NASA Langley Formal Methods Group: https://github.com/nasa/pvslib
[6] VSCode-PVS, Paolo Masci: https://github.com/nasa/vscode-pvs

PVS – Prototype Verification System

Specification (.pvs)

```
% Define half
half(a:real,b:real | b>a):
  {r:real | abs(a-r) = abs(b-r)} =
    (a+b)/2

% Theorem about half
prove | show-prooflite
half_sq: THEOREM
 FORALL(a:real,b:real | b>a):
  EXISTS(n:posnat):
   a>n AND b>n
   IMPLIES half(a,b) < half(a^n,b^n)
```

Interactive theorem prover

```
half_sq.1.1 :

{-1}   b < b ^ 2
[-2]   a < a ^ 2
[-3]   a > 2
[-4]   b > 2
  |-------
[1]   half(a, b) < half(a ^ 2, b ^ 2)

>> (expand "half")

- Ctrl+SPACE shows the full list of commands.
- TAB autocompletes commands. Double click expands definitions.
```

PVS – Prototype Verification System

Proof (.prf)

```
half_sq : PROOF
(then (skeep)(inst 1 "2")(flatten)
  (spread (case "a<a^2")
   ((spread (case "b<b^2")
     ((then (expand "half")(mult-by 1 "2")(assert))
      (then (div-by 1 "b")(grind))))
    (then (div-by 1 "a")(grind)))))
QED half_sq
```

Interactive theorem prover

```
half_sq.1.1 :

{-1}   b < b ^ 2
[-2]   a < a ^ 2
[-3]   a > 2
[-4]   b > 2
  |-------
[1]   half(a, b) < half(a ^ 2, b ^ 2)

>> (expand "half")

- Ctrl+SPACE shows the full list of commands.
- TAB autocompletes commands. Double click expands definitions.
```

Hybrid Programs in PVS

Values of variables

```
Environment : TYPE = [nat->real]
%For example:
x: nat = 0
y: nat = 1
env: Environment = (LAMBDA(i:nat): 0)
  WITH [(x) := 10, (y) := -sqrt(5)]
```

Assignments

```
Assigns      : TYPE = MapExprInj
ODEs         : TYPE = MapExprInj
%For example
exp_ex: ODEs
= (: (x, val(x)), (y, val(y)+cnst(1)) :)
```

Functions on environments

```
%Predicates
BoolExpr    : TYPE = [Environment->bool]
%Quantified boolean expressions
QBoolExpr   : TYPE = [real->BoolExpr]
%Real-valued functions
RealExpr    : TYPE = [Environment->real]
%For example
val(i:nat): RealExpr
  = LAMBDA(env:Environment): env(i)
```

Hybrid Programs in PVS

Syntax of hybrid programs

```
HP : DATATYPE
BEGIN
  IMPORTING hp_def
  ASSIGN(assigns:Assigns) : assign?
  DIFF(odes:ODEs,be:BoolExpr) : diff?
  TEST(be:BoolExpr) : test?
  SEQ(stm1,stm2:HP) : seq?
  UNION(stm1,stm2:HP) : union?
  STAR(stm:HP) : star?
END HP
```

Semantics of hybrid programs

```
semantic_rel(hp:HP)(envi:Environment)
  (envo:Environment):
  INDUCTIVE bool = ...
```

Semantics of `ASSIGN(l:MapExprInj)`

```
(FORALL (i:below(length(l))) :
LET (k,re) = nth(l,i) IN
envo(k) = re(envi)) AND
FORALL (i:(not_in_map(l))) :envo(i) = envi(i)
```

Hybrid Programs in PVS

Recall:

$$x \geq 1 \land v \geq 0 \land a \geq 0 \ \vdash$$

$$\left[\left((a := a + 1); \{x' = v, v' = a\}\right)^*\right](x \geq 1)$$

In PVS:

```
hp_ex: LEMMA
LET alpha = STAR(SEQ(ASSIGN( (: a, val(a) + cnst(1) :)),
DIFF( (: (x, val(v)), (v,val(a)) :) ))) IN
   (: :) |- (: DLIMPLIES( (: val(x) >= cnst(1),
  val(v) >= cnst(0), val(a) >= cnst(0) :),
   ALLRUNS(alpha, val(x) >= cnst(1))):)
```

Plaidypvs

- **Formal verification** of soundness of $\mathbf{dL}^{[7]}$

- **Fully operational** embedding $\mathbf{dL}$

- **Extensions of $\mathbf{dL}$ in PVS**

[7] Previous Formal Verification of soundness of $\mathbf{dL}$ in Coq and Isabelle/Hol:
Brandon Bohrer, Vincent Rahli, Ivana Vukotic, Marcus Völp, and André Platzer. 2017. Formally verified differential dynamic logic.
In Proceedings of the 6th ACM SIGPLAN Conference on Certified Programs and Proofs.208–221. https://doi.org/10.1145/3018610.3018616

Formal Verification of Soundness of **dL**

Loop rule:

```
dl_loop : LEMMA
  FORALL (Gamma,Delta)(J,P)(A):
    (Gamma |- cons(J,Delta)) AND
    (J |- P) AND
    (J |- ALLRUNS(A,J)) IMPLIES
    (Gamma |- cons(ALLRUNS(STAR(A),P),Delta))
```

Differential invariant rule:

```
dl_dI: LEMMA
  FORALL  (P:(ddl_dom?))(Gamma,Delta)(nnP,Q)(ode:ODEs):
    derivable_up_nqbool?(max_var(ode),P)(nnP) AND
    ( cons(Q,Gamma) |- cons(nqb_to_be(nnP),Delta) ) AND
    ( Q |- SUB_DIFT(P,ode)(nnP) ) IMPLIES
    ( Gamma |-
     cons(ALLRUNS(DIFF(ode,DLAND(P,Q)),nqb_to_be(nnP)),Delta) )
```

81 proven rules/axioms of **dl** in PVS

Fully Operational Embedding of $\mathbf{dL}$

- Proof rules implemented as strategies in PVS
  - Fully operational $\mathbf{dL}$ within interactive prover console of PVS

```
discrete_loop_ex :

  |-------
{1}   (: val(x) >= cnst(1), val(v) >= cnst(0), val(a) >= cnst(0) :) |-
       (: ALLRUNS(STAR(SEQ(ASSIGN((: (a, val(a) + cnst(1)) :)),
                              DIFF((: (x, val(v)), (v, val(a)) :),
                                   DLBOOL(TRUE)))),
                 val(x) >= cnst(1)) :)

>> (dl-loop "val(x) >= cnst(1) AND val(v) >= cnst(0) AND val(a) >=cnst(0)")
```

Fully Operational Embedding of **dL**

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
discrete_loop_ex.1 :

  |-------
{1}   ((: val(x) >= cnst(1), val(v) >= cnst(0), val(a) >= cnst(0) :) |-
        (: DLAND(val(x) >= cnst(1),
                 DLAND(val(v) >= cnst(0), val(a) >= cnst(0))) :))

>> (dl-assert)
```

Fully Operational Embedding of **dL**

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
discrete_loop_ex.2 :

  |-------
{1}   ((: DLAND(val(x) >= cnst(1),
              DLAND(val(v) >= cnst(0), val(a) >= cnst(0))) :)
         |- (: val(x) >= cnst(1) :))

>> (dl-assert)
```

Fully Operational Embedding of dL

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
discrete_loop_ex.3 :

  |-------
{1}   ((: DLAND(val(x) >= cnst(1),
              DLAND(val(v) >= cnst(0), val(a) >= cnst(0))) :)
        |-
        (: ALLRUNS(SEQ(ASSIGN((: (a, val(a) + cnst(1)) :)),
                       DIFF((: (x, val(v)), (v, val(a)) :), DLBOOL(TRUE))),
                   DLAND(val(x) >= cnst(1),
                         DLAND(val(v) >= cnst(0), val(a) >= cnst(0)))) :))

>>
```

Fully Operational Embedding of **dL**

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
diff_inv_ex :

  |-------
{1}   ((: val(x) >= cnst(1), val(a) >= cnst(0) :) |-
        (: ALLRUNS(DIFF((: (x, val(v)), (v, val(a)) :), val(v) >= cnst(0)),
                   val(x) >= cnst(1)) :))

>> (dl-diffinv)
```

Fully Operational Embedding of **dL**

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
diff_inv_ex.1 :

  |-------
{1}   ((: val(v) >= cnst(0), (val(x) >= cnst(1)), val(a) >= cnst(0) :) |-
        (: (val(x) >= cnst(1)) :))

>> (dl-assert)
```

Fully Operational Embedding of **dL**

- Proof rules implemented as strategies in PVS
  - Fully operational **dL** within interactive prover console of PVS

```
diff_inv_ex.2 :

  |-------
{1}   ((: val(v) >= cnst(0) :) |- (: val(v) >= cnst(0) :))

>> (dl-assert)

Q.E.D.
```

Generalized Reasoning of Hybrid Programs

- **Fully typed** specification of hybrid programs
  - Reasoning at the type level (properties of groups of hybrid programs)
  - Reasoning for arbitrary hybrid programs (e.g., arbitrarily many variables)

```
behind: TYPE
  = {hp: (diff?) |
      (: behind?(odes(hp)) :) |- (: ALLRUNS(hp,behind?(odes(hp))) :)}

slow: TYPE
  = {hp: (diff?) |
    (: :) |- (: slower?(odes(hp)) :)}
```

- **A hybrid program of type `slow` is always of type `behind`**

```
slow_is_behind: JUDGEMENT
  slow SUBTYPE_OF behind
```

Summary

- **dL:** Differential Dynamic Logic for hybrid programs
- **PVS:** Interactive theorem prover

Result: Plaidypvs [3]

- Formal verification **dL**
- Fully operational in **PVS**
- Leveraging features of **PVS** to extend **dL**

[3] Plaidypvs in NASA PVS Library: https://github.com/nasa/pvslib/tree/master/dL