Towards Physical Hybrid Systems

Katherine Cordwell and André Platzer

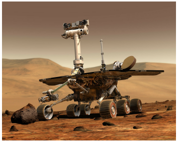
Carnegie Mellon University

August 29, 2019

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1252522. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of any other sponsoring institution. This research was also sponsored by the AFOSR under grant number FA9550-16-1-0288 and by the Alexander von Humboldt Foundation.

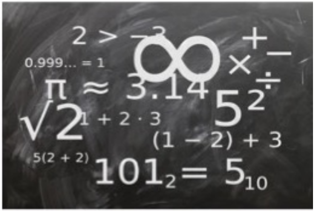
Safety-critical CPS

- How do we know that cyber-physical systems (CPS) are functioning correctly?



Safety-critical CPS

- How do we know that cyber-physical systems (CPS) are functioning correctly?
  - First step: model your CPS
  - *Hybrid systems* model CPS



Safety-critical CPS

- How do we know that cyber-physical systems (CPS) are functioning correctly?
  - First step: model your CPS
  - *Hybrid systems* model CPS

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

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

Follow the ODE subject to the domain constraint R

Loop

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

$$\alpha, \beta ::= \boxed{x := e} \mid ?P \mid x' = f(x) \& R \mid \alpha \cup \beta \mid \alpha ; \beta \mid \alpha^*$$

Assignment

Follow the ODE subject to the domain constraint R

Loop

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

$$\alpha, \beta ::= \boxed{x := e} \mid \boxed{?P} \mid x' = f(x) \& R \mid \alpha \cup \beta \mid \alpha ; \beta \mid \alpha^*$$

Assignment **Test**

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

$$\alpha, \beta ::= \boxed{x := e} \mid \boxed{?P} \mid \boxed{x' = f(x) \& R} \mid \alpha \cup \beta \mid \alpha ; \beta \mid \alpha^*$$

Assignment   **Test**   **Follow the ODE subject to the domain constraint R**   Loop

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

$$\alpha, \beta ::= \boxed{x := e} \mid \boxed{?P} \mid \boxed{x' = f(x) \& R} \mid \boxed{\alpha \cup \beta} \mid \alpha; \beta \mid \alpha^*$$

Assignment

Test

Follow the ODE subject to the domain constraint R

Run either α or β

Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?

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

Assignment

Test

Follow the ODE subject to the domain constraint R

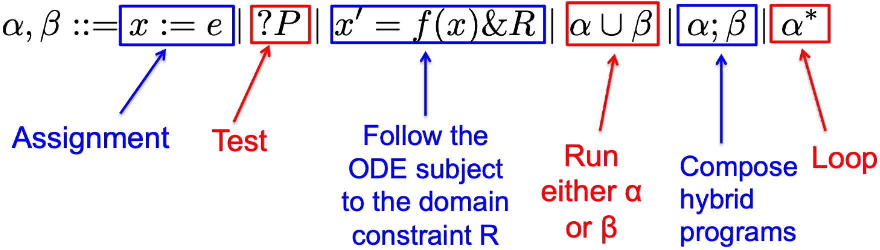
Run either α or β

Compose hybrid programs

Loop

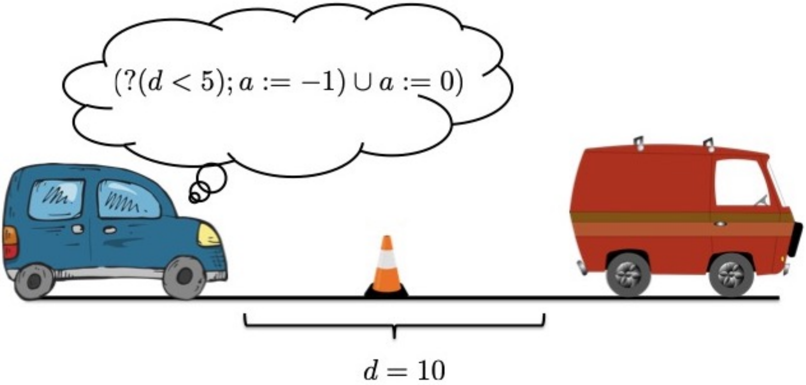
Then write hybrid systems in logic...

...with differential dynamic logic, perhaps?



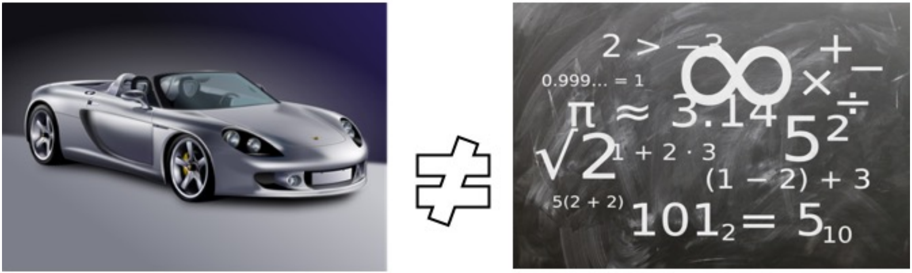
Problems?

- The model could be overly permissive



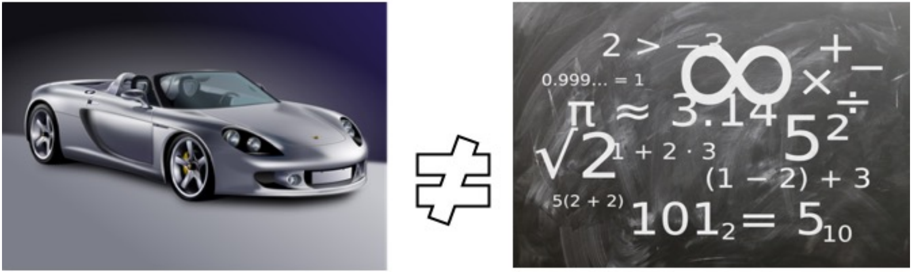
Problems?

- The model could be overly permissive
- Or the model could be overly strict



Problems?

- The model could be overly permissive
- Or the model could be overly strict
    - Logic is precise, physical systems are not
    - Note that we absolutely want to have precise safety guarantees



Math versus physics

- How can models be too strict?

Math versus physics

- How can models be too strict?
  - Models can classify systems as being unsafe on minutely small sets

Math versus physics

- How can models be too strict?
  - Models can classify systems as being unsafe on minutely small sets
- Is this realistic?

Math versus physics

- How can models be too strict?
  - Models can classify systems as being unsafe on minutely small sets
- Is this realistic?
  - No! Even math allows more imprecision than models

Math versus physics

- How can models be too strict?
  - Models can classify systems as being unsafe on minutely small sets
- Is this realistic?
  - No! Even math allows more imprecision than models
- Does it matter?

Math versus physics

- How can models be too strict?
  - Models can classify systems as being unsafe on minutely small sets
- Is this realistic?
  - No! Even math allows more imprecision than models
- Does it matter?
  - Yes! Physically unrealistic counterexamples can distract from real unsafeties of a system

Our Approach

- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*

Our Approach

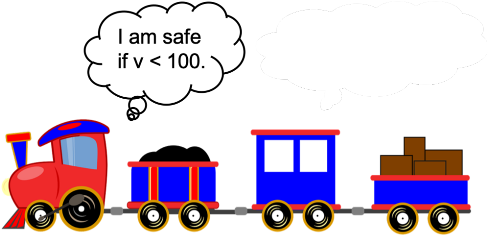
- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*
  - There are multiple ways to develop PHS

Our Approach

- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*
  - There are multiple ways to develop PHS
  - Our first foray into PHS stays very close to the usual notion of safety

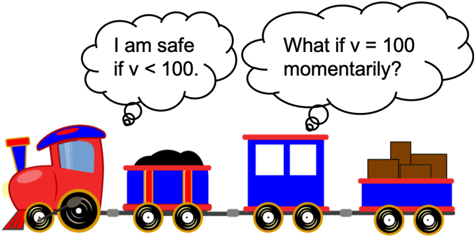
Our Approach

- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*
  - There are multiple ways to develop PHS
  - Our first foray into PHS stays very close to the usual notion of safety



Our Approach

- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*
  - There are multiple ways to develop PHS
  - Our first foray into PHS stays very close to the usual notion of safety



Our Approach

- We propose **physical hybrid systems** (PHS), which are systems that behave safely *almost everywhere*
  - There are multiple ways to develop PHS
  - Our first foray into PHS stays very close to the usual notion of safety
  - Our new logic (PdTL) is designed to ignore "very small", meaningless sets of safety violations along the execution trace of a system.

FAQ, anticipated

- Why not ask the user to edit the models?

FAQ, anticipated

- Why not ask the user to edit the models?
  - PdTL is capturing something that is even closer to the normal notion of safety
  - Also, we don't want to limit the models that a user can write

FAQ, anticipated

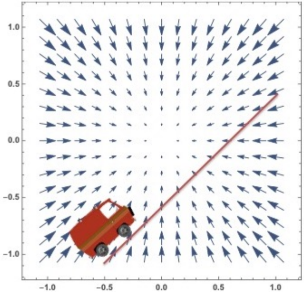
- Why not ask the user to edit the models?
  - PdTL is capturing something that is even closer to the normal notion of safety
  - Also, we don't want to limit the models that a user can write
- Why isn't this just solved by robustness?

Robustness

- Safe up to small perturbations
- Tool support, e.g. dReach

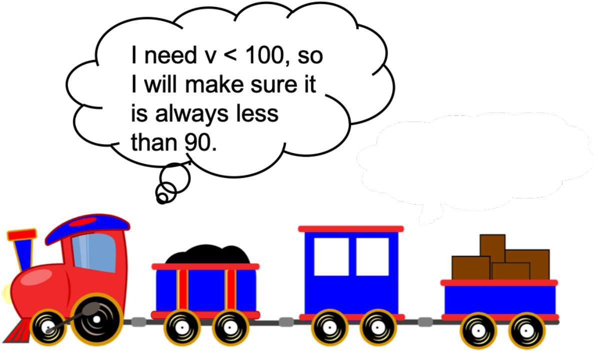
Robustness

- Safe up to small perturbations
- Tool support, e.g. dReach
- Models of CPS can and should be robust



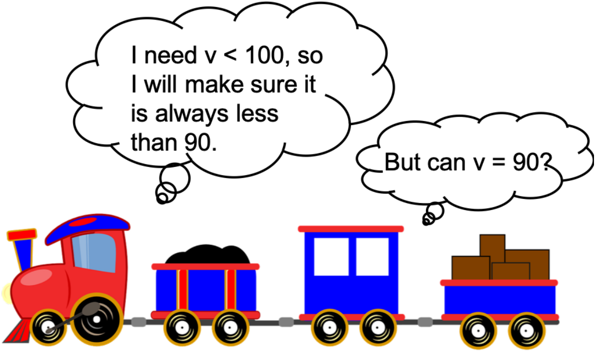
Robustness

- But robustness is only one piece of the puzzle. We're trying to do something different.



Robustness

- But robustness is only one piece of the puzzle. We're trying to do something different.



Robustness

- But robustness is only one piece of the puzzle. We're trying to do something different.
- Also, robustness often requires a reachability analysis and can be more limited in scope (no induction!)

Let's talk PdTL

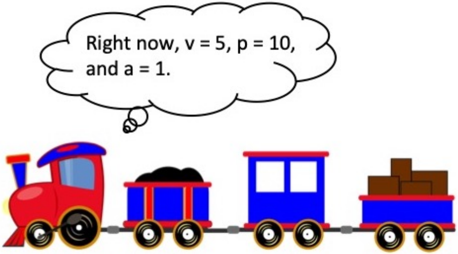
- *Physical differential temporal dynamic logic (PdTL)* extends dTL extends dL
- dTL rigorizes execution traces

Formulas in dTL (and PdTL!)

- State formulas
    - Evaluated in a state (at a snapshot in time)

Formulas in dTL (and PdTL!)

- State formulas
  - Evaluated in a state (at a snapshot in time)
  - States map variables to $\mathbb{R}$



Formulas in dTL (and PdTL!)

- State formulas
  - Evaluated in a state (at a snapshot in time)
  - States map variables to $\mathbb{R}$
- Trace formulas
  - Evaluated along execution traces (sequences of functions mapping intervals to states)

Traces in PdTL

$a := 1;\ ?(a = 1);\ \{x' = v, v' = a\}$

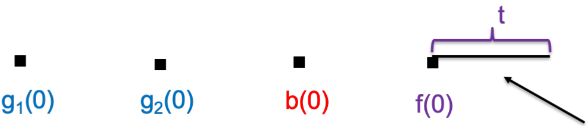
One trace: (g_1, g_2, b, f)

g_1: [0, 0] → states

g_2: [0, 0] → states

b: [0, 0] → states

f: [0, t] → states



$g_1(0)$: x = 0, v = 0, a = 0

$g_2(0)$: x = 0, v = 0, a = 1

b(0): x = 0, v = 0, a = 1

f(0): x = 0, v = 0, a = 1

x and v evolve continuously

PdTL

- Trace semantics
  - The same as in dTL, except we allow Carathéodory solutions to ODEs

PdTL

- Trace semantics
  - The same as in dTL, except we allow Carathéodory solutions to ODEs
- Formulas
  - The same state formulas as dTL
  - Instead of dTL's trace formulas, use $\Box_{\mathsf{tae}}$

PdTL

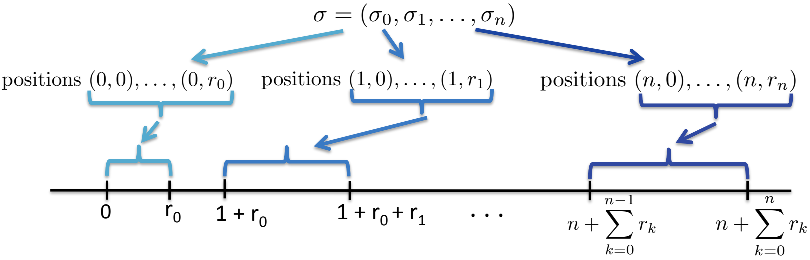
- Trace semantics
  - The same as in dTL, except we allow Carathéodory solutions to ODEs
- Formulas
  - The same state formulas as dTL
  - Instead of dTL's trace formulas, use $\Box_{\mathsf{tae}}$
  - Intuitively, $\sigma \models \Box_{\mathsf{tae}}\phi$ means ϕ holds except at only a "small" set of positions along the trace

PdTL

- Trace semantics
  - The same as in dTL, except we allow Carathéodory solutions to ODEs
- Formulas
  - The same state formulas as dTL
  - Instead of dTL's trace formulas, use $\Box_{\mathsf{tae}}$
  - Intuitively, $\sigma \models \Box_{\mathsf{tae}}\phi$ means ϕ holds except at only a "small" set of positions along the trace
  - Measure zero: mathematically rigorous notion of a very small set

PdTL

- How to get a measure on a trace? Map it to $\mathbb{R}$



PdTL

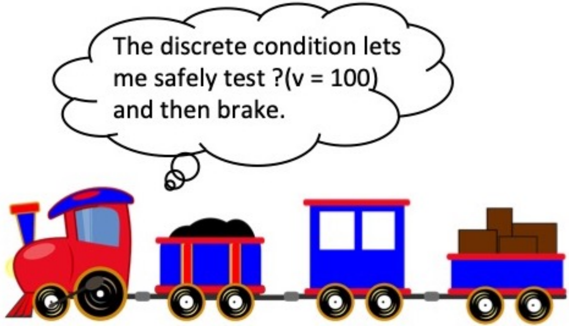
- For $\sigma \models \Box_{\mathsf{tae}}\phi$ to hold:
  - Need ϕ to be satisfied at almost all positions along the trace (continuous condition)

PdTL

- For $\sigma \models \Box_{\mathsf{tae}}\phi$ to hold:
  - Need ϕ to be satisfied at almost all positions along the trace (continuous condition)
  - On discrete portions of the trace, need ϕ to almost hold (discrete condition)

PdTL

- For $\sigma \models \Box_{\text{tae}}\phi$ to hold:
  - Need ϕ to be satisfied at almost all positions along the trace (continuous condition)
  - On discrete portions of the trace, need ϕ to almost hold (discrete condition)



Compelling logical properties

- Conservative extension of dL

Compelling logical properties

- Conservative extension of dL
- A proof calculus that is designed to:
  - Remove instances of $\square_{\text{tae}}$ when possible

$$[?P]\square_{\text{tae}}\phi \leftrightarrow \overline{\phi}$$

Compelling logical properties

- Conservative extension of dL
- A proof calculus that is designed to:
  - Remove instances of $\Box_{\mathsf{tae}}$ when possible
    $$[?P]\Box_{\mathsf{tae}}\phi \leftrightarrow \overline{\phi}$$
  - Reduce complicated formulas to structurally simpler formulas
    $$[\alpha \cup \beta]\Box_{\mathsf{tae}}\phi \leftrightarrow [\alpha]\Box_{\mathsf{tae}}\phi \land [\beta]\Box_{\mathsf{tae}}\phi$$

Compelling logical properties

- Conservative extension of dL
- A proof calculus that is designed to:
  - Remove instances of $\Box_{\mathsf{tae}}$ when possible
    $$[?P]\Box_{\mathsf{tae}}\phi \leftrightarrow \overline{\phi}$$
  - Reduce complicated formulas to structurally simpler formulas
    $$[\alpha \cup \beta]\Box_{\mathsf{tae}}\phi \leftrightarrow [\alpha]\Box_{\mathsf{tae}}\phi \wedge [\beta]\Box_{\mathsf{tae}}\phi$$
  - Do induction
    $$\frac{\overline{\phi} \vdash [\alpha]\Box_{\mathsf{tae}}\phi}{\overline{\phi} \vdash [\alpha^*]\Box_{\mathsf{tae}}\phi}$$

Compelling logical properties

- A major challenge: reasoning principles for ODEs

Compelling logical properties

- A major challenge: reasoning principles for ODEs
  - $[x' = f(x)]\Box_{\mathsf{tae}} P \leftrightarrow \overline{P} \wedge \forall t{\geq}0 Q$

Compelling logical properties

- A major challenge: reasoning principles for ODEs
  - $[x' = f(x)]\Box_{\mathsf{tae}} P \leftrightarrow \overline{P} \wedge \forall t {\geq} 0 Q$
  - P and Q are FOL formulas so that:
    "for almost all $t{\geq}0[x := y(t)]P$" $\iff$ $\forall t{\geq}\ 0\ Q$,
    where y(t) solves the ODE

Compelling logical properties

- A major challenge: reasoning principles for ODEs
  - $[x' = f(x)]\Box_{\text{tae}}P \leftrightarrow \overline{P} \wedge \forall t{\geq}0Q$
  - P and Q are FOL formulas so that: "for almost all $t{\geq}0[x := y(t)]P$" $\Longleftrightarrow$ $\forall t{\geq}\ 0\ Q$, where y(t) solves the ODE
- More complicated ODEs reasoning: a remaining challenge

Proof calculus

$[?]_{\mathsf{tae}}$ $\quad [?P]\Box_{\mathsf{tae}}\phi \leftrightarrow \overline{\phi}$

$[\cup]_{\mathsf{tae}}$ $\quad [\alpha \cup \beta]\Box_{\mathsf{tae}}\phi \leftrightarrow [\alpha]\Box_{\mathsf{tae}}\phi \wedge [\beta]\Box_{\mathsf{tae}}\phi$

$[:=]_{\mathsf{tae}}$ $\quad [x := e]\Box_{\mathsf{tae}}\phi \leftrightarrow \overline{\phi} \wedge [x := e]\overline{\phi}$

$[;]_{\mathsf{tae}}$ $\quad [\alpha;\beta]\Box_{\mathsf{tae}}\phi \leftrightarrow ([\alpha]\Box_{\mathsf{tae}}\phi \wedge [\alpha][\beta]\Box_{\mathsf{tae}}\phi)$

$\mathsf{I}_{\mathsf{tae}}$ $\quad [\alpha^*]\Box_{\mathsf{tae}}\phi \leftrightarrow \left(\overline{\phi} \wedge [\alpha^*](\overline{\phi} \to [\alpha]\Box_{\mathsf{tae}}\phi)\right)$

$[']_{\mathsf{tae}}$ $\quad [x' = f(x)]\Box_{\mathsf{tae}}P \leftrightarrow \overline{P} \wedge \forall t{\geq}0 Q$

$['\&]_{\mathsf{tae}}$ $\quad [x' = f(x)\&R]\Box_{\mathsf{tae}}P \leftrightarrow \overline{P} \wedge$
$\forall t{>}0\,((\forall 0{\leq}s{\leq}t\ [x := y(s)]R) \to Q)$

$$G_{\mathsf{tae}}\ \frac{\phi}{[\alpha]\Box_{\mathsf{tae}}\phi}$$

$$K_{\mathsf{tae}}\ \frac{\overline{\phi} \to \overline{\psi} \quad [\alpha]\Box_{\mathsf{tae}}(\phi \to \psi)}{[\alpha]\Box_{\mathsf{tae}}\phi \to [\alpha]\Box_{\mathsf{tae}}\psi}$$

$$\mathsf{TopCL}\ \frac{\phi \to \psi}{\overline{\phi} \to \overline{\psi}}$$

$$\mathsf{CGG}\ \ [\alpha]\Box_{\mathsf{tae}}\phi \to [\alpha]\overline{\phi}$$

Here, α and β are hybrid programs, ϕ and ψ are state formulas, P is a FOL formula, $y(t)$ solves $x' = f(x)$, and the formula Q in $[']_{\mathsf{tae}}$ and $['\&]_{\mathsf{tae}}$ is a FOL formula constructed for $P(y(t))$ so that "for almost all $t{\geq}0[x := y(t)]P$" is logically equivalent to "$\forall t{\geq}0\ Q$".

PdTL works on the train example

- Model:

$$a = 0 \land v = 0 \rightarrow [\Big(\big(((?(v<100); a := 1) \cup (?(v = 100); a := -1)\big); \{x' = v, v' = a \;\&\; 0 \leq v \leq 100\}\Big)^*]\Box_{\text{tae}} v<100$$

PdTL works on the train example

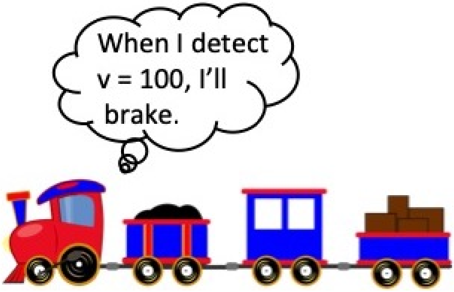
- Model:

$$a = 0 \land v = 0 \rightarrow [\big(((?(v<100); a := 1) \cup (?(v = 100); a := -1)); \\ \{x' = v, v' = a \; \& \; 0 \le v \le 100\}\big)^*]\Box_{\mathsf{tae}} v<100$$

- Key idea: Remove the loop with $\text{loop}_{\mathsf{tae}}$, split and simplify with $[;]_{\mathsf{tae}}$ and dL axioms, handle the ODE with $['\&]_{\mathsf{tae}}$, close with dL reasoning

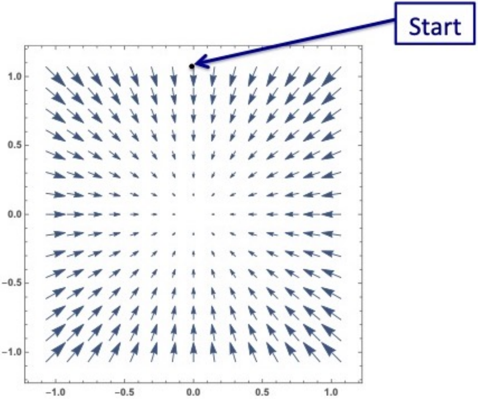
PdTL works on the train example

... and other event-triggered controllers



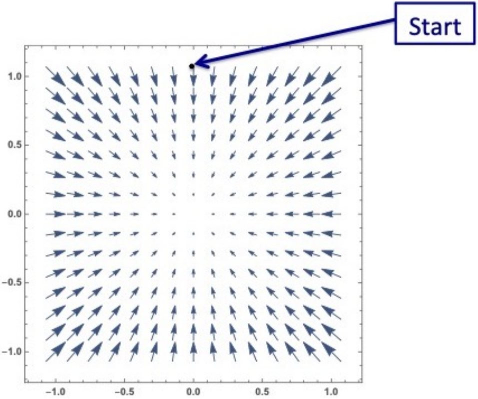
When else does it work?

- Start at $x = 0$ and $y = 1$, evolve along $x' = -x, y' = -y$, require $x^2 + y^2 < 1$



When else does it work?

- Start at $x = 0$ and $y = 1$, evolve along $x' = -x, y' = -y$, require $x^2 + y^2 < 1$
- Handover point glitch

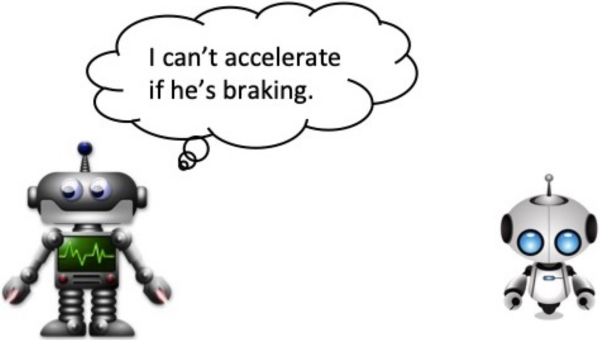


When else does it work?

- Handle glitches in continuous portions of program

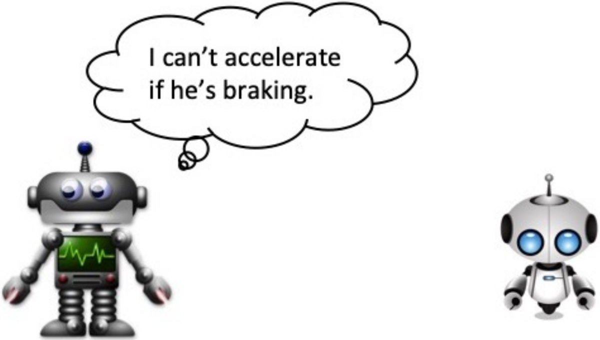
When else does it work?

- Handle glitches in continuous portions of program
- Two robots moving



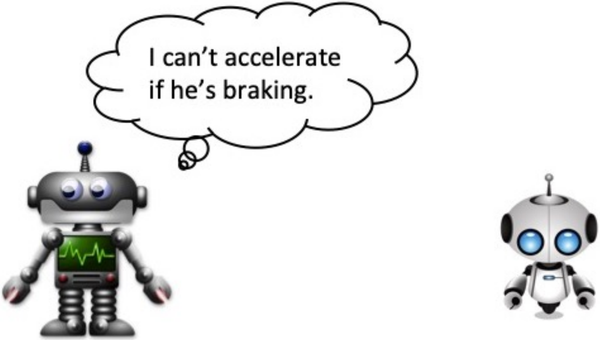
When else does it work?

- Model this with $\neg(a_1 \leq 0 \wedge a_2 \geq 0)$



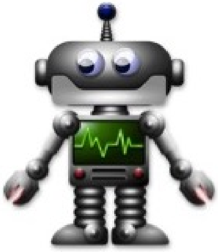
When else does it work?

- Model this with $\neg(a_1 \leq 0 \land a_2 \geq 0)$
- This is a small mistake. We should allow $a_1 = 0 \land a_2 = 0$



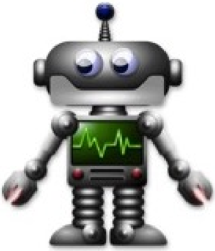
When else does it work?

- Postcondition $\neg(a_1 \leq 0 \land a_2 \geq 0)$
- Controller $a_1 := -1; a_2 := -1; \{a_1' = 1, a_2' = 1\}$



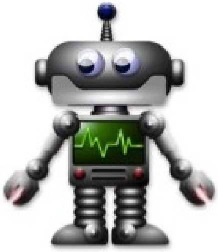
When else does it work?

- Postcondition $\neg(a_1 \leq 0 \land a_2 \geq 0)$
- Controller $a_1 := -1; a_2 := -1; \{a_1' = 1, a_2' = 1\}$
- This is tae safe (but not safe everywhere)



When else does it work?

- Postcondition $\neg(a_1 \leq 0 \wedge a_2 \geq 0)$
- Controller $a_1 := -1; a_2 := -1; \{a'_1 = 1, a'_2 = 1\}$
- This is tae safe (but not safe everywhere)
- $a_1 := -1; a_2 := -1; \{a'_1 = 1, a'_2 = 2\}$ is *not* tae safe



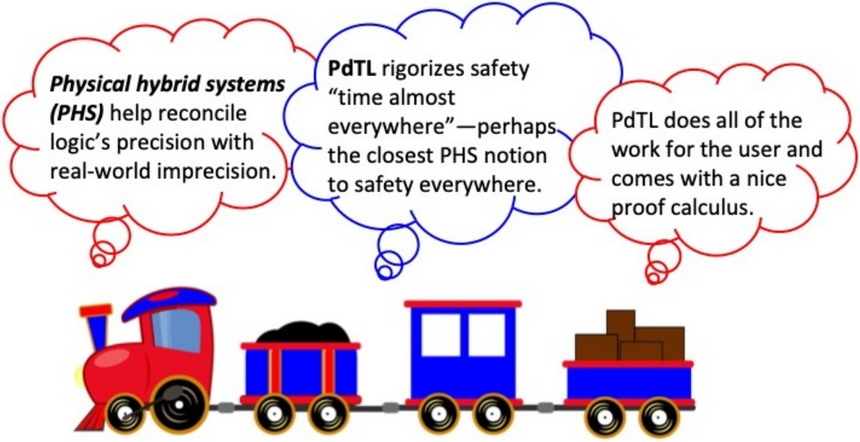
Conclusion

- PdTL formalizes the notion of safety “almost everywhere in time”

Conclusion

- PdTL formalizes the notion of safety "almost everywhere in time"
- Next up... more relaxed notions of PHS?

Questions?



Thank you!