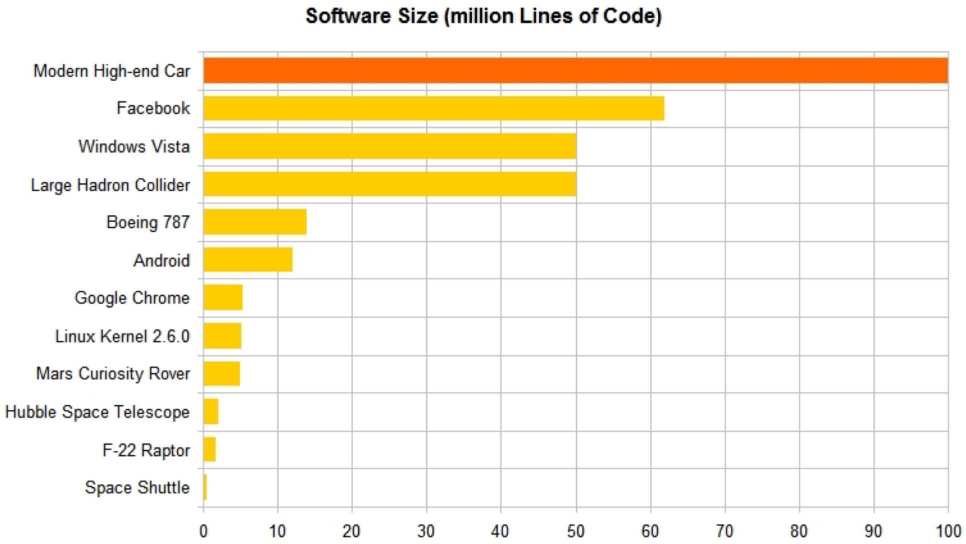
Safe Reinforcement Learning via Formal Methods

André Platzer
Carnegie Mellon University
Joint work with Nathan Fulton

Safety-Critical Systems

"How can we provide people with cyber-physical systems they can bet their lives on?" - Jeannette Wing

Safety-Critical Systems



"How can we provide people with cyber-physical systems they can bet their lives on?" - Jeannette Wing

This Talk

Ensure the safety of Autonomous Cyber-Physical Systems.

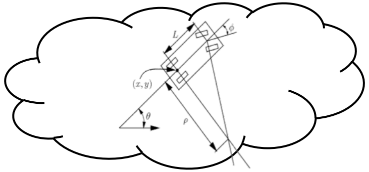
Best of both worlds: learning together with CPS safety
- Flexibility of learning
- Guarantees of CPS formal methods

Diametrically opposed: flexibility+adaptability versus predictability+simplicity

1. Cyber-Physical Systems with **Differential Dynamic Logic**
2. **Sandboxed reinforcement learning** is provably safe

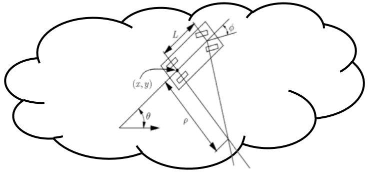
Model-Based Verification

Reinforcement Learning



φ

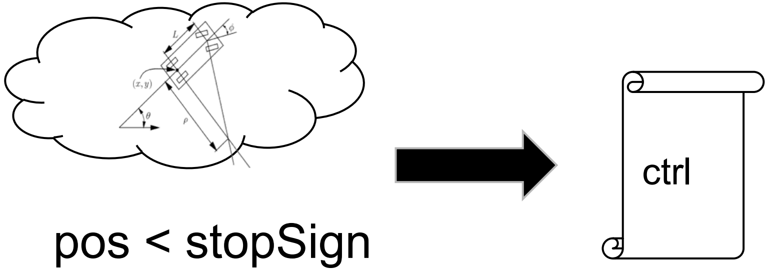
Model-Based Verification   Reinforcement Learning



pos < stopSign

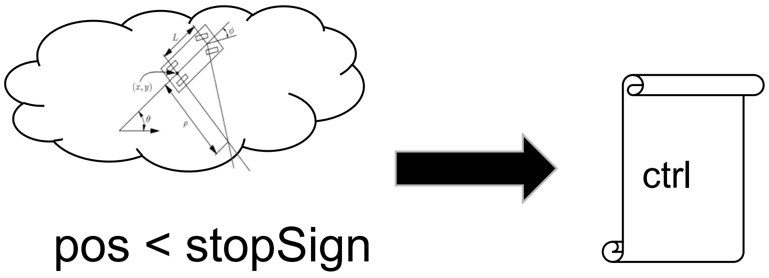
Model-Based Verification

Reinforcement Learning



pos < stopSign

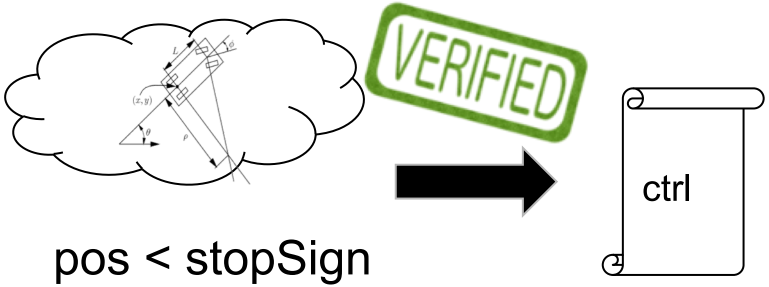
Model-Based Verification Reinforcement Learning



Approach: prove that control software achieves a specification with respect to a model of the physical system.

Model-Based Verification

Reinforcement Learning



Approach: prove that control software achieves a specification with respect to a model of the physical system.

Model-Based Verification   Reinforcement Learning

φ

Benefits:

- Strong safety guarantees
- Automated analysis

Model-Based Verification

Benefits:

- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers “what is safe”, not “what is useful”

Reinforcement Learning

Model-Based Verification

Reinforcement Learning

φ

Benefits:

- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers “what is safe”, not “what is useful”
- Assumes accurate model

Model-Based Verification

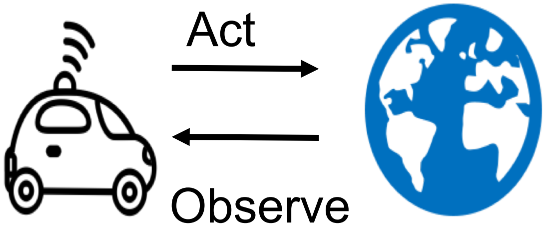
Benefits:

- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers “what is safe”, not “what is useful”
- Assumes accurate model.

Reinforcement Learning



<table><tr><td>

Model-Based Verification

Benefits:

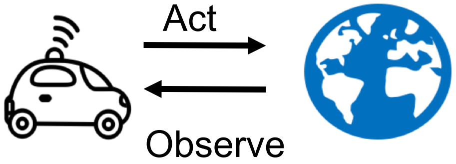
- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers "what is safe", not "what is useful"
- Assumes accurate model.

</td><td>

Reinforcement Learning



Benefits:

- No need for complete model
- Optimal (effective) policies

</td></tr></table>

Model-Based Verification

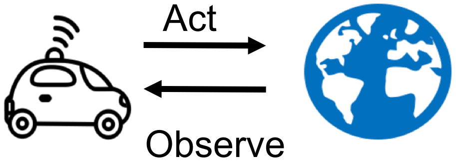
Benefits:

- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers “what is safe”, not “what is useful”
- Assumes accurate model.

Reinforcement Learning



Benefits:

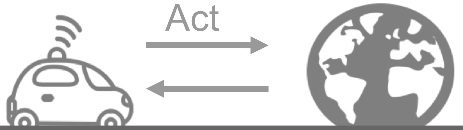
- No need for complete model
- Optimal (effective) policies

Drawbacks:

- No strong safety guarantees
- Proofs are obtained and checked by hand
- Formal proofs = decades-long proof development

Model-Based Verification

Reinforcement Learning



Bene

Goal: Provably correct reinforcement learning

- S
- A

del

s

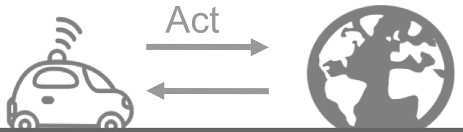
Draw

- Control policies are typically non-deterministic: answers "what is safe", not "what is useful"
- Assumes accurate model

- No strong safety guarantees
- Proofs are obtained and checked by hand
- Formal proofs = decades-long proof development

Model-Based Verification

Reinforcement Learning



Bene

- S
- A

del
s

Goal: Provably correct reinforcement learning

1. Learn Safety
2. Learn a Safe Policy
3. Justify claims of safety

Draw

- Control policies are typically non-deterministic: answers “what is safe”, not “what is useful”
- Assumes accurate model

- No strong safety guarantees
- Proofs are obtained and checked by hand
- Formal proofs = decades-long proof development

Part I: Differential Dynamic Logic

Trustworthy Proofs for Hybrid Systems

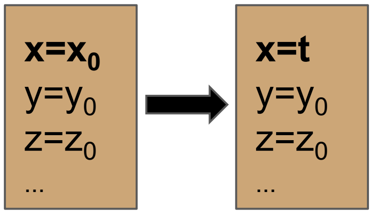
Hybrid Programs

$x := t$

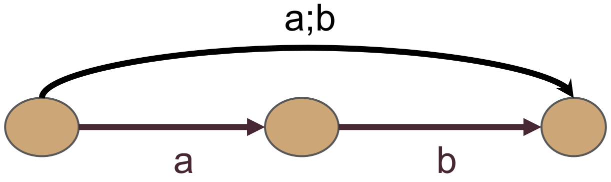
$\mathbf{x=x_0}$ $y=y_0$ $z=z_0$ …	$\Rightarrow$	$\mathbf{x=t}$ $y=y_0$ $z=z_0$ …

Hybrid Programs

$x := t$

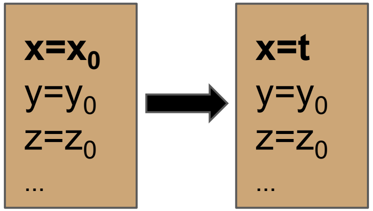


$a;b$

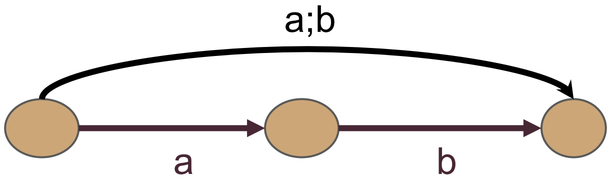


Hybrid Programs

$x := t$



$a;b$



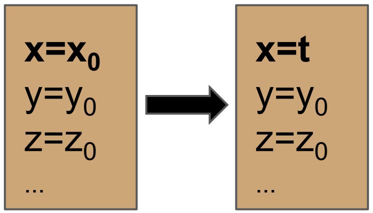
$?P$

If P is true: no change

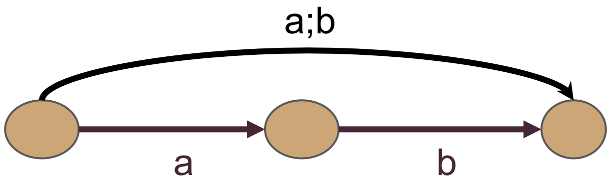
If P is false: terminate

Hybrid Programs

$x := t$



a;b

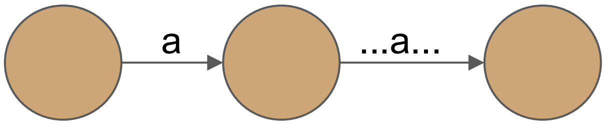


?P

If P is true: no change

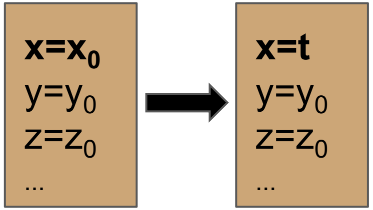
If P is false: terminate

a*

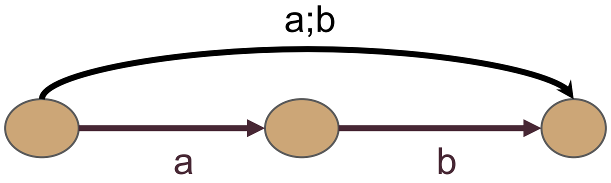


Hybrid Programs

$x := t$



$a;b$

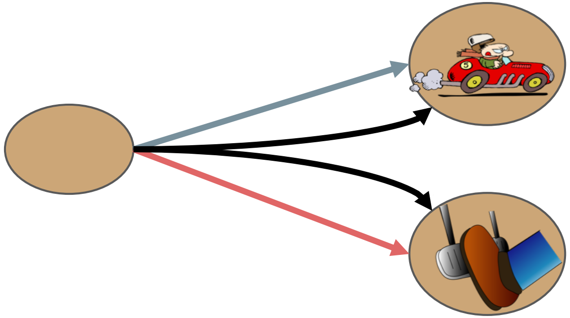


$?P$

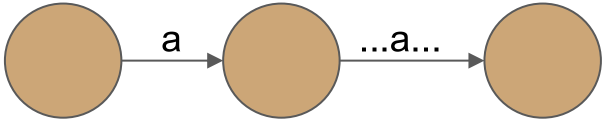
If P is true: no change

If P is false: terminate

$a \cup b$

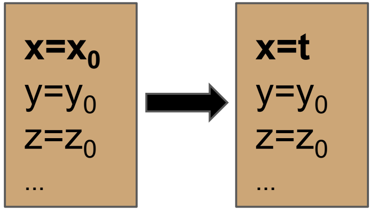


a^*

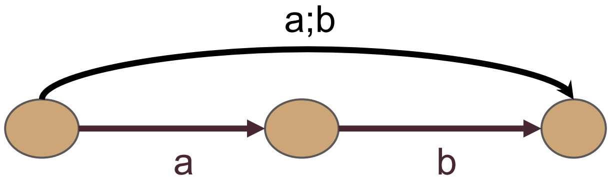


Hybrid Programs

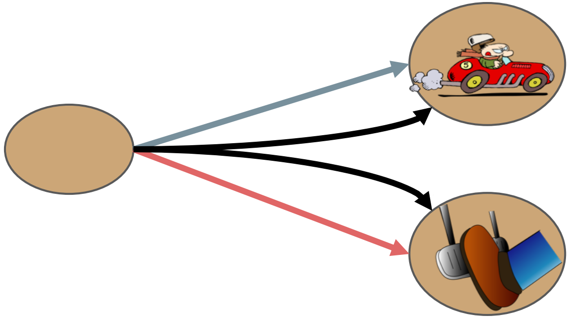
$x := t$



$a;b$



$a \cup b$

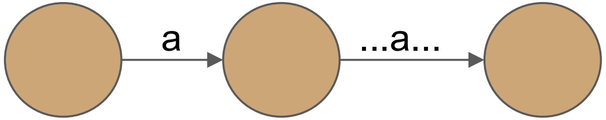


?P

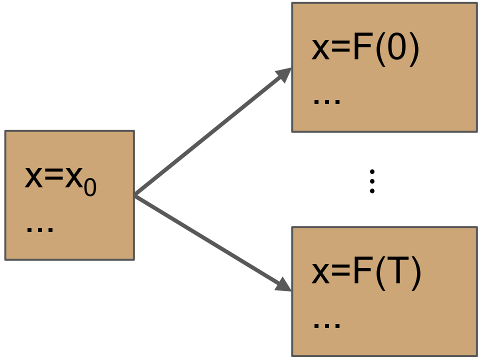
If P is true: no change

If P is false: terminate

a^*



$x' = f$



Approaching a Stopped Car

Own Car

Stopped Car

Is this property true?

```
[
 { {accel ∪ brake}; t:=0; {pos’=vel,vel’=accel,t’=1 & vel≥0 & t≤T} }*
](pos <= stoppedCarPos)
```

Approaching a Stopped Car

Own Car

Stopped Car

Assuming we only accelerate when it’s safe to do so, is this property true?

```
[
 { {accel ∪ brake}; t:=0; {pos’=vel,vel’=accel,t’=1 & vel≥0 & t≤T} }*
](pos <= stoppedCarPos)
```

Approaching a Stopped Car

Own Car

`safeDistance(pos,vel,stoppedCarPos,B)`

Stopped Car

if we also assume the system is safe initially:

```
safeDistance(pos,vel,stoppedCarPos,B) →

[

 { {accel ∪ brake}; t:=0; {pos’=vel,vel’=accel,t’=1 & vel≥0 & t≤T} }*

](pos <= stoppedCarPos)
```

Approaching a Stopped Car


```
safeDistance(pos,vel,stoppedCarPos,B) →

[

 { {accel ∪ brake}; t:=0; {pos'=vel,vel'=accel,t'=1 & vel≥0 & t≤T} }*

](pos <= stoppedCarPos)
```

The Fundamental Question

Why would our program not work if we have a *proof*?

The Fundamental Question

Why would our program not work if we have a *proof*?

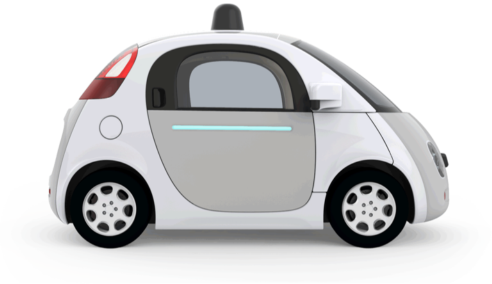
1. Was the proof correct?

The Fundamental Question

Why would our program not work if we have a *proof*?

1. Was the proof correct?
2. Was the model accurate enough?

≠

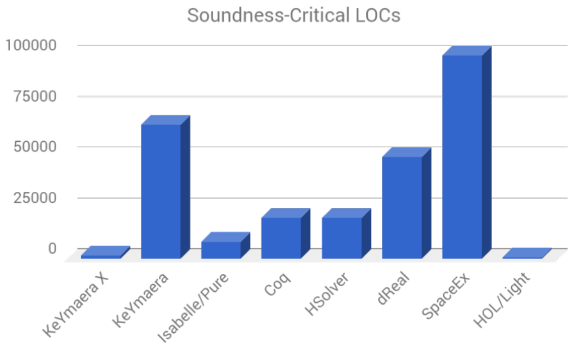


The Fundamental Question

Why would our program not work if we have a *proof*?

1. Was the proof correct? **KeYmaera X** 

2. Was the model accurate enough?



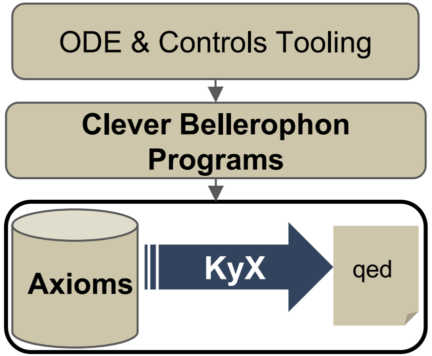
dI Tactic:

Side derivation:
$(v \geq v_0 - gt)' \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots$

$H = r_p \geq 0 \;\&\; r_a \geq 0 \;\&\; g > 0 \;\&\; \ldots$

DI Axiom:
$[\{x'=f \& Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f \& Q\}]P'))$

Example:
$[v'=r_p v^2-g, t'=1]v \geq v_0 - gt \quad \leftrightarrow$
$\ldots \quad \leftrightarrow$
$[v':=r_p v^2-g][t':=1]v' \geq -g*t' \quad \leftrightarrow$
$r_p v^2-g \geq -g \quad \leftrightarrow$
$H \rightarrow r_p \geq 0$

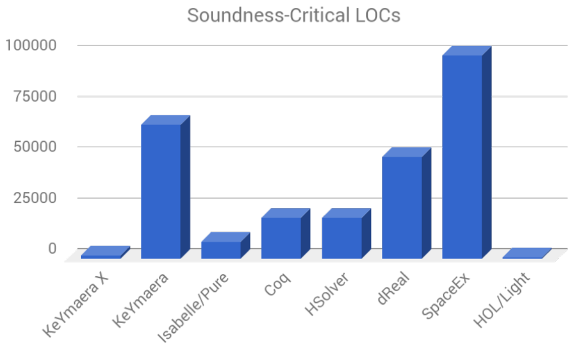


The Fundamental Question

Why would our program not work if we have a *proof*?

1. Was the proof correct? **KeYmaera X** 

2. Was the model accurate enough? **Safe RL**



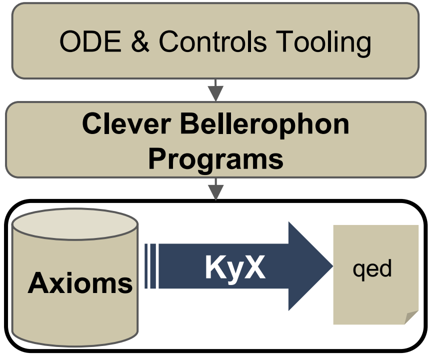
dI Tactic:

Side derivation:
$(v \geq v_0 - gt)' \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots$

$H = r_p \geq 0$ & $r_a \geq 0$ & $g > 0$ & ...

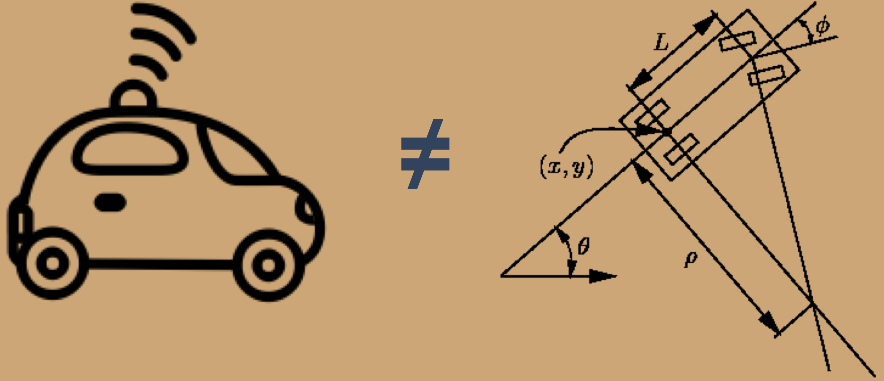
DI Axiom:
$[\{x'=f \& Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f \& Q\}]P'))$

Example:
$[v'=r_p v^2-g, t'=1]v \geq v_0 - gt \leftrightarrow$
$\ldots \leftrightarrow$
$[v':=r_p v^2-g][t':=1]v' \geq -g*t' \leftrightarrow$
$r_p v^2-g \geq -g \leftrightarrow$
$H \rightarrow r_p \geq 0$



Part II: Justified Speculative Control

Safe reinforcement learning in partially modeled environments



AAAI 2018

Model-Based Verification

Accurate, analyzable models often exist!

```
{
   {?safeAccel;accel   ∪ brake ∪ ?safeTurn; turn};
   {pos’ = vel, vel’ = acc}
}*
```

Model-Based Verification

Accurate, analyzable models often exist!

```
{
    {?safeAccel;accel    ∪ brake ∪ ?safeTurn; turn};
    {pos' = vel, vel' = acc}
}*
```

Continuous motion

discrete control

Model-Based Verification

Accurate, analyzable models often exist!

```
{
    {?safeAccel;accel   ∪ brake ∪ ?safeTurn; turn};
    {pos’ = vel, vel’ = acc}
}*
```

Continuous motion

discrete, ***non-deterministic*** control

Model-Based Verification

Accurate, **analyzable** models often exist!

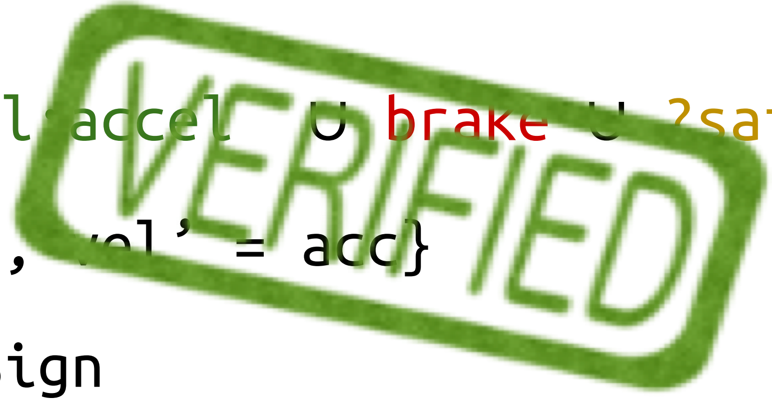
```
init → [{
   { ?safeAccel;accel  ∪ brake ∪ ?safeTurn; turn};
   {pos’ = vel, vel’ = acc}
}*]pos < stopSign
```

Model-Based Verification

Accurate, **analyzable** models often exist!

formal verification gives strong safety guarantees

```
init → [{
    { ?safeAccel;accel ∪ brake ∪ ?safeTurn; turn};
    {pos’ = vel, vel’ = acc}
}*]pos < stopSign
```



Model-Based Verification

Accurate, **analyzable** models often exist!

formal verification gives strong safety guarantees

=

- **Computer-checked proofs of safety specification.**

Model-Based Verification

Accurate, **analyzable** models often exist!

formal verification gives strong safety guarantees

=

- **Computer-checked proofs of safety specification**
- **Formal proofs mapping model to runtime monitors**

Model-Based Verification Isn't Enough

Perfect, analyzable models don't exist!

Model-Based Verification Isn't Enough

Perfect, analyzable models don't exist!

How to implement?

```
{
    { ?safeAccel;accel  U  brake  U  ?safeTurn; turn};
    {pos' = vel, vel' = acc}
}*
```

Only accurate sometimes

Model-Based Verification Isn't Enough

Perfect, analyzable models don't exist!

How to implement?

```
{
  { ?safeAccel;accel ∪ brake ∪ ?safeTurn; turn};
  {dx'=w*y, dy'=-w*x, ...}
}*
```

Only accurate sometimes

Safe RL Contribution

Justified Speculative Control is an approach toward provably safe reinforcement learning that:

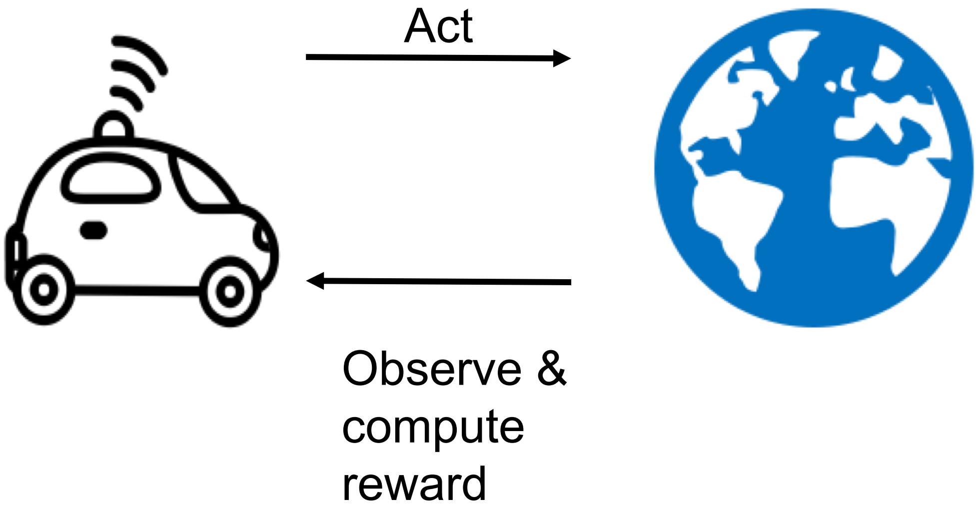
1. learns to resolve non-determinism without sacrificing formal safety results

Safe RL Contribution

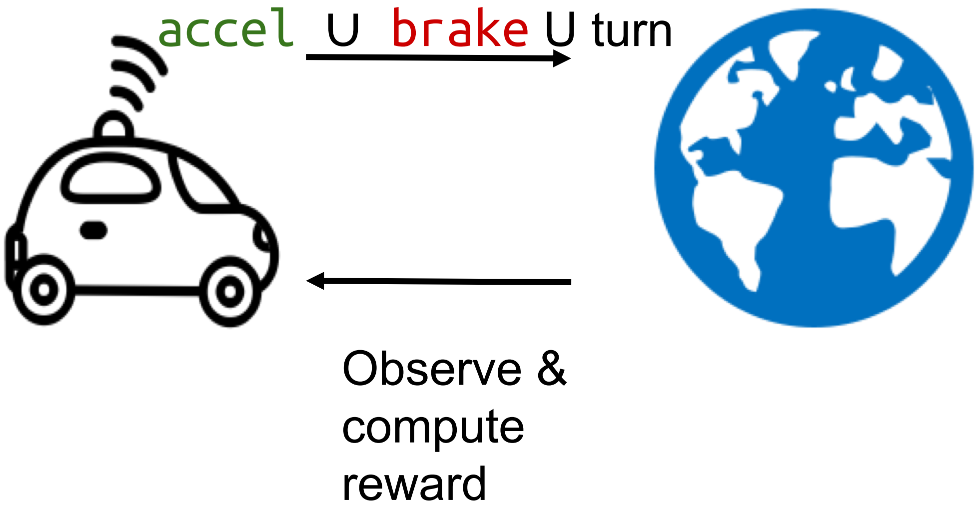
Justified Speculative Control is an approach toward provably safe reinforcement learning that:

1. learns to resolve non-determinism without sacrificing formal safety results
2. allows and directs speculation whenever model mismatches occur

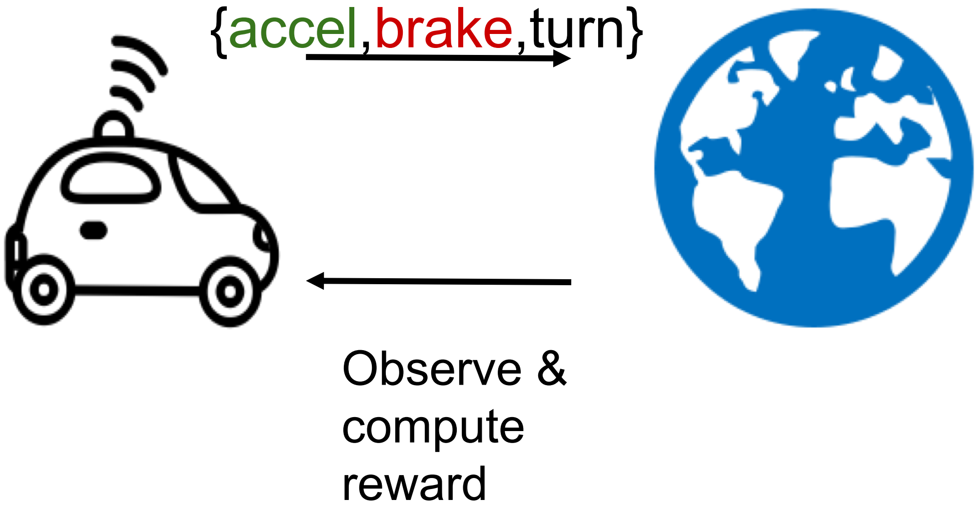
Learning to Resolve Non-determinism



Learning to Resolve Non-determinism



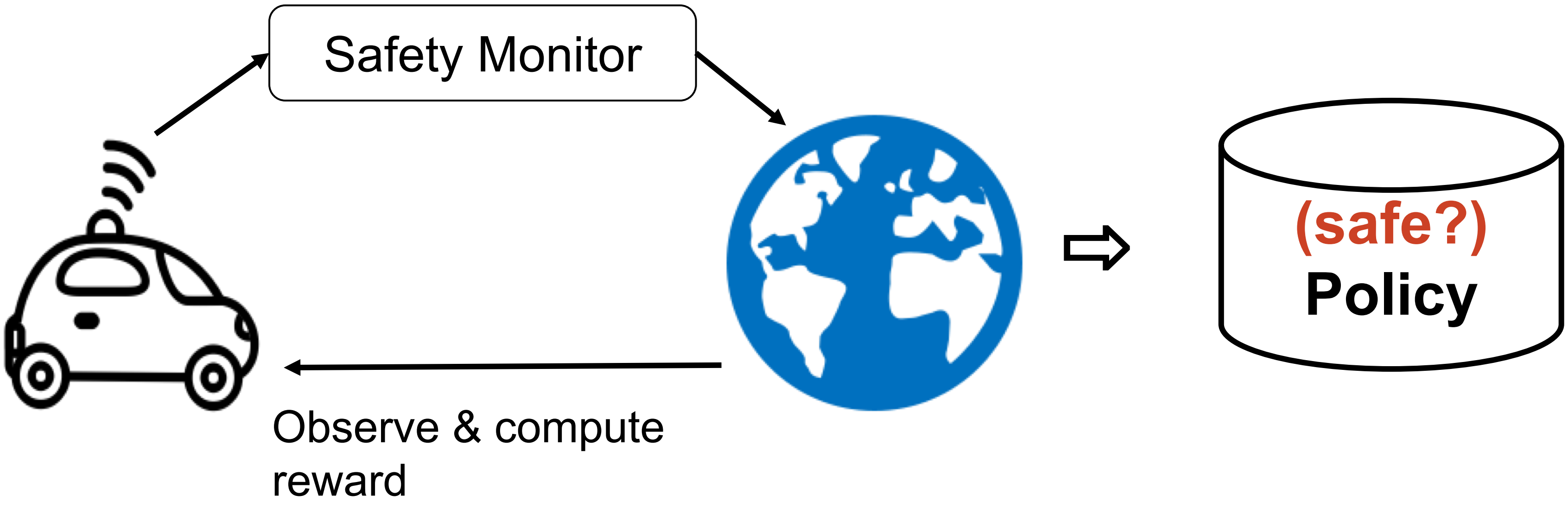
Learning to Resolve Non-determinism



Learning to Resolve Non-determinism

Learning to Resolve Non-determinism

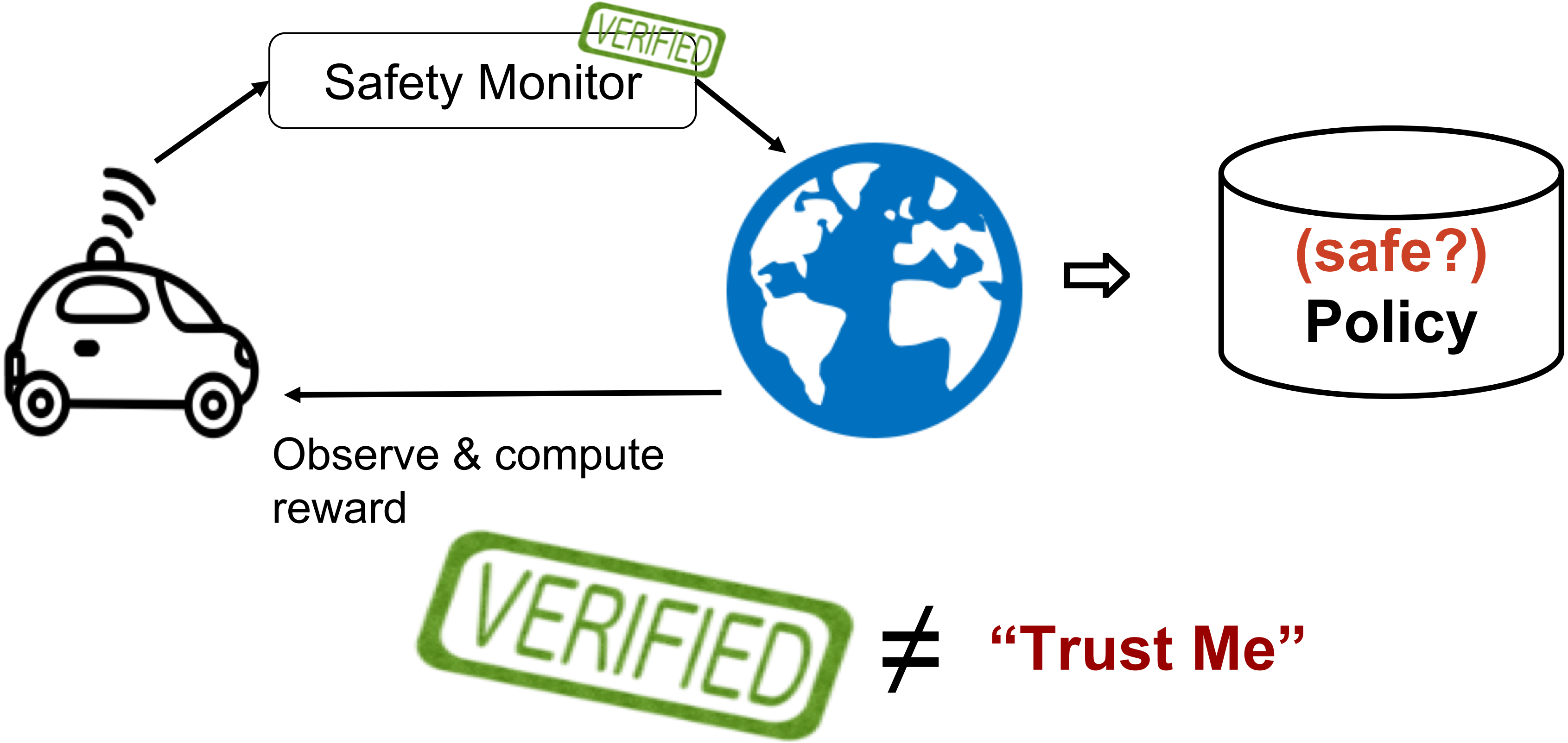
Learning to **Safely** Resolve Non-determinism



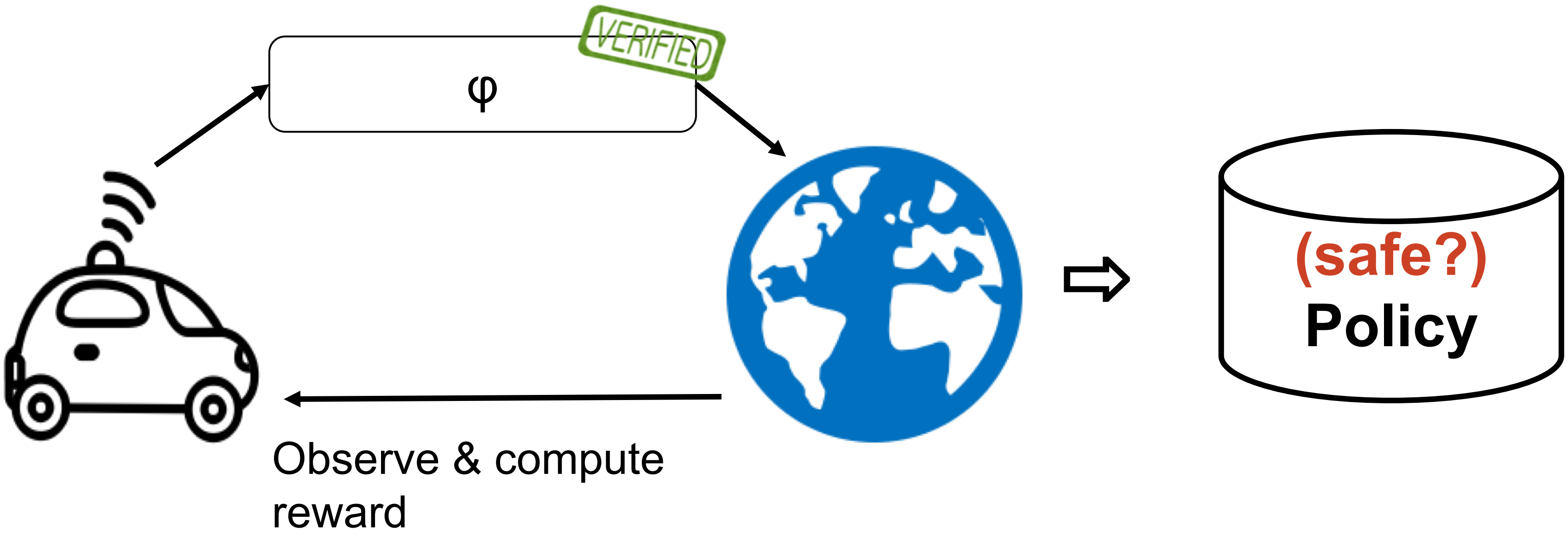
Useful to stay safe during learning

Crucial after deployment

Learning to **Safely** Resolve Non-determinism



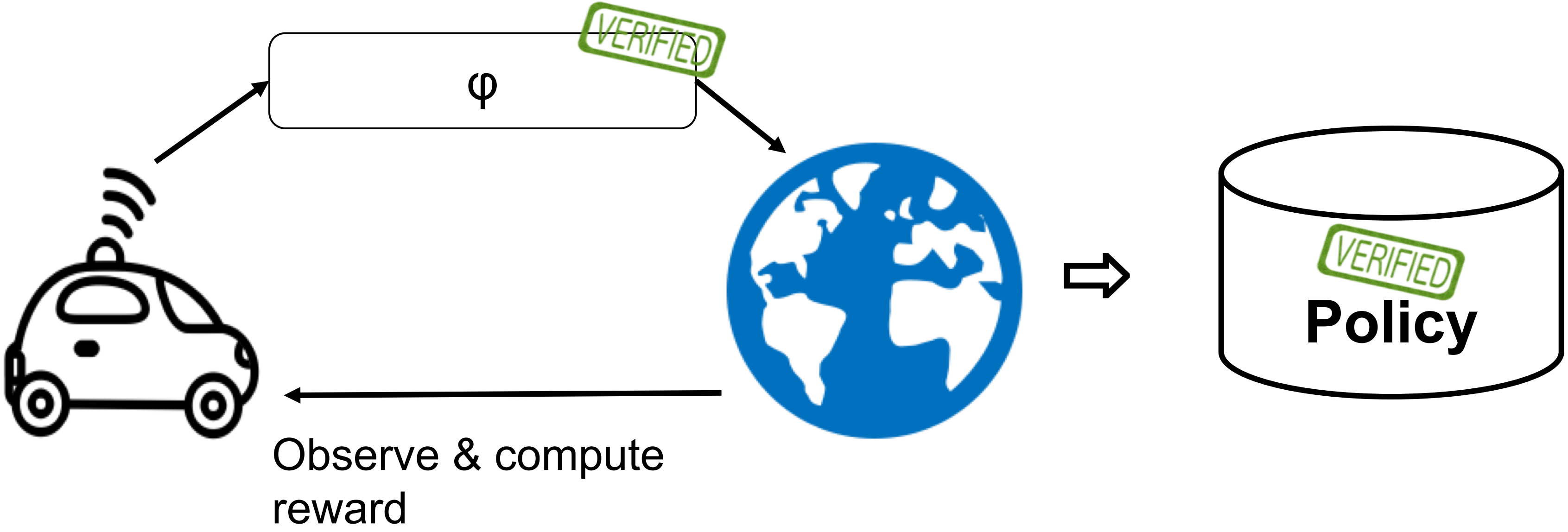
Learning to **Safely** Resolve Non-determinism



Use a theorem prover to prove:

$$(init \rightarrow [\{\{accel \cup brake\};ODEs\}^*](safe)) \qquad \varphi$$

Learning to **Safely** Resolve Non-determinism



Use a theorem prover to prove:

(init→[{{accel∪brake};ODEs}*](safe))   φ

Learning to **Safely** Resolve Non-determinism

<u>Main Theorem</u>: If the ODEs are accurate, then our formal proofs transfer from the non-deterministic model to the learned (deterministic) policy

φ

Observe & compute reward

Policy

Use a theorem prover to prove:

(init→[{{accel∪brake};ODEs}*](safe)) φ

Learning to **Safely** Resolve Non-determinism

Main Theorem: If the ODEs are accurate, then our formal proofs transfer from the non-deterministic model to the learned (deterministic) policy via the model monitor.

φ

VERIFIED

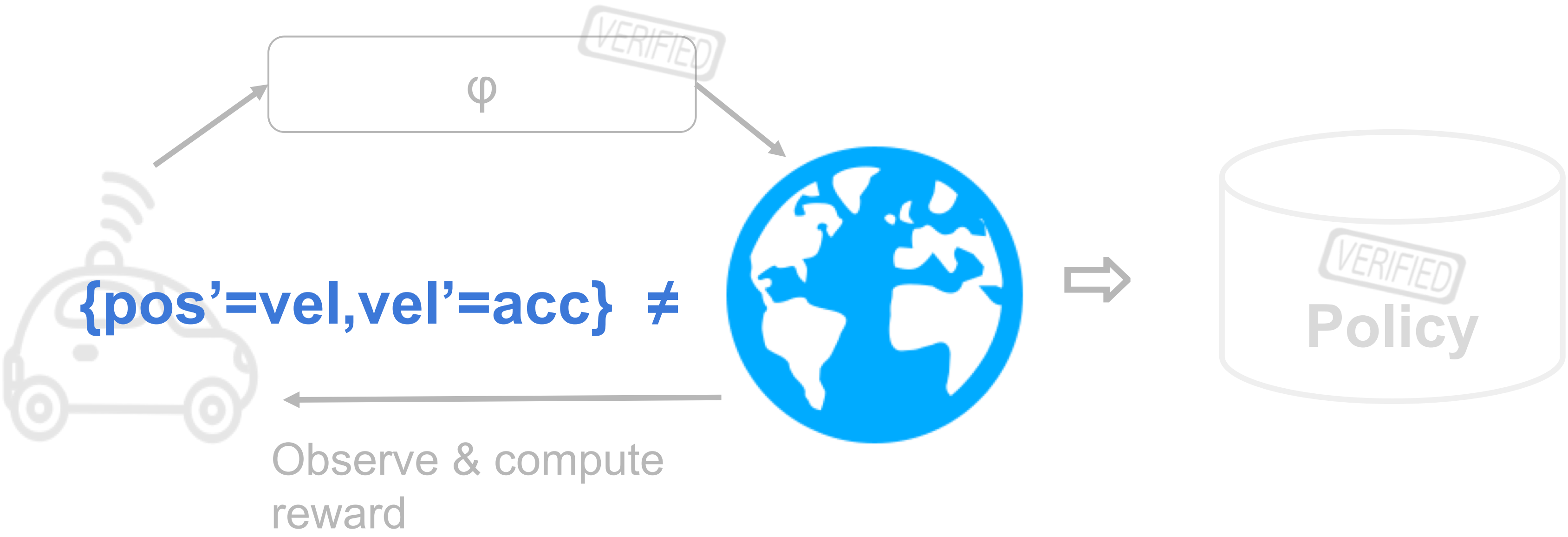
Policy

Observe & compute reward

Use a theorem prover to prove:

$(init \rightarrow [\{\{accel \cup brake\};ODEs\}^*](safe))$ φ

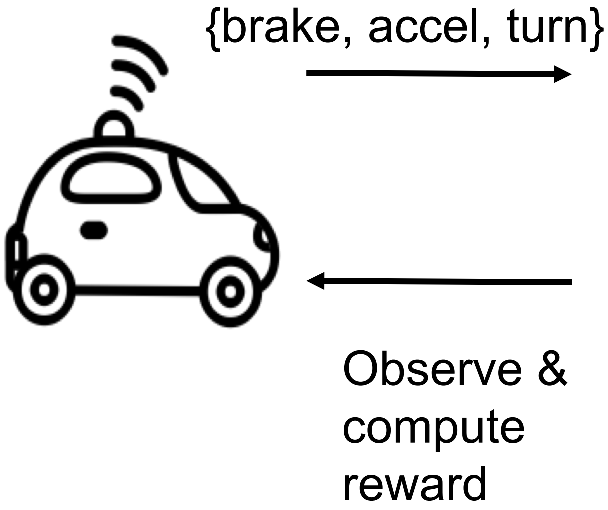
What about the physical model?

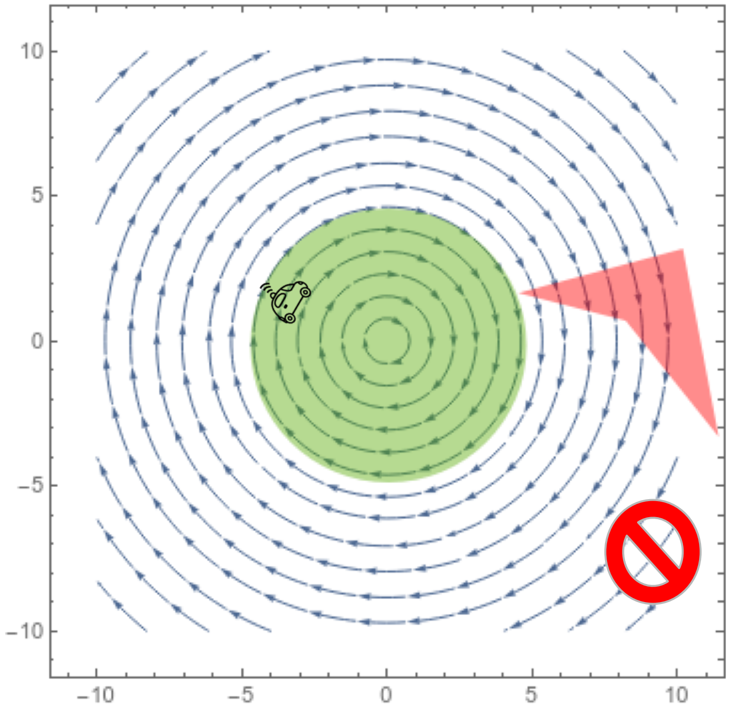


Use a theorem prover to prove:

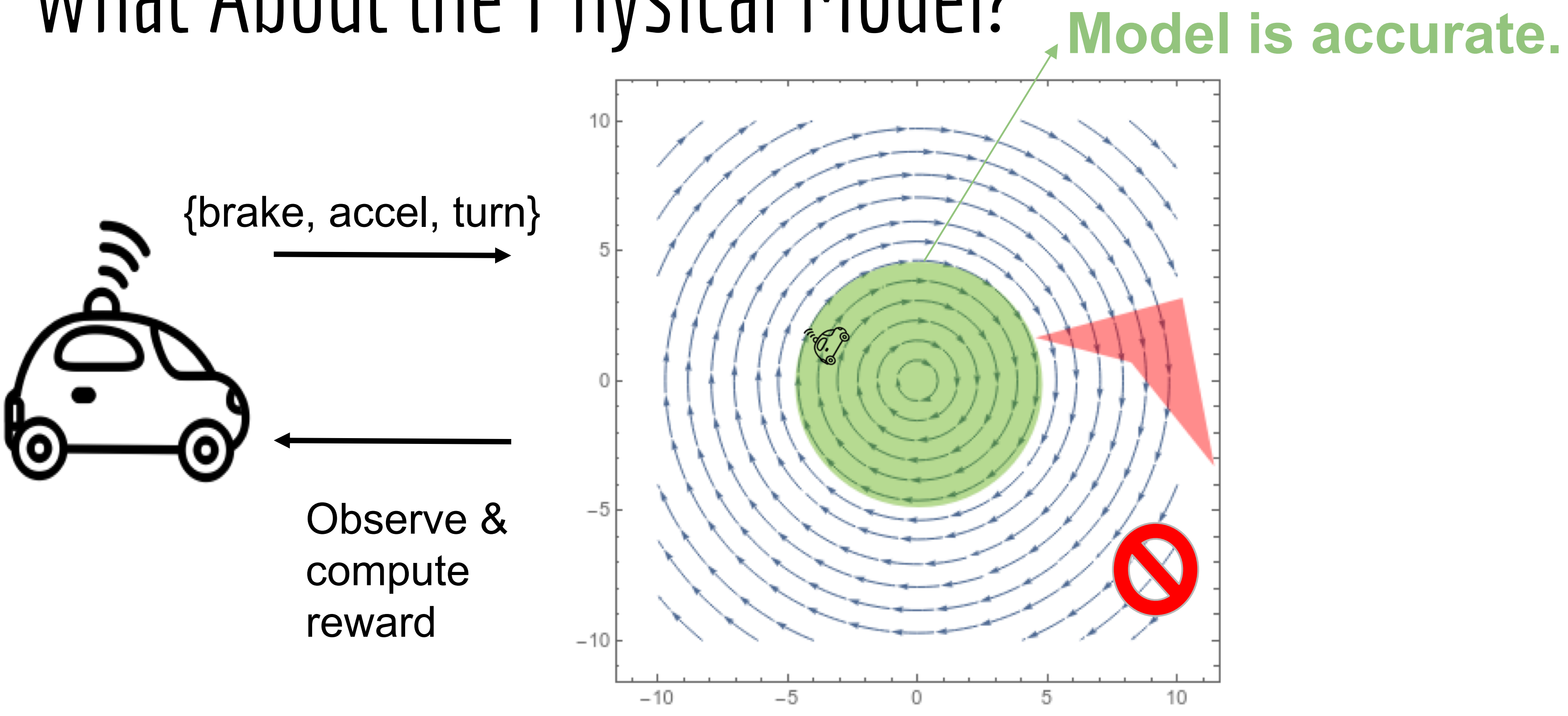
$(init \rightarrow [\{\{accel \cup brake\};ODEs\}^*](safe))$ φ

What About the Physical Model?

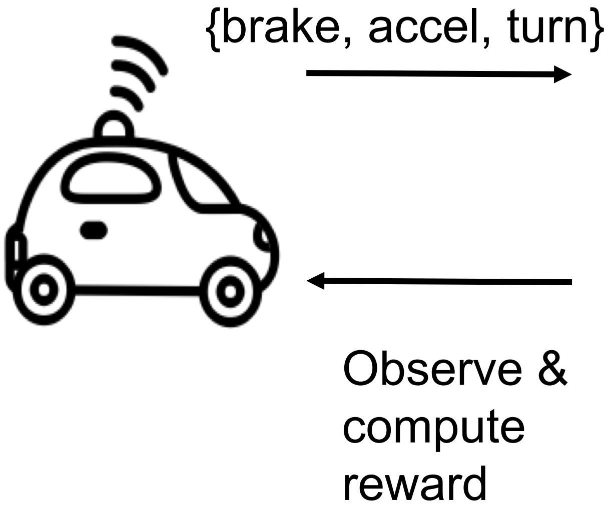


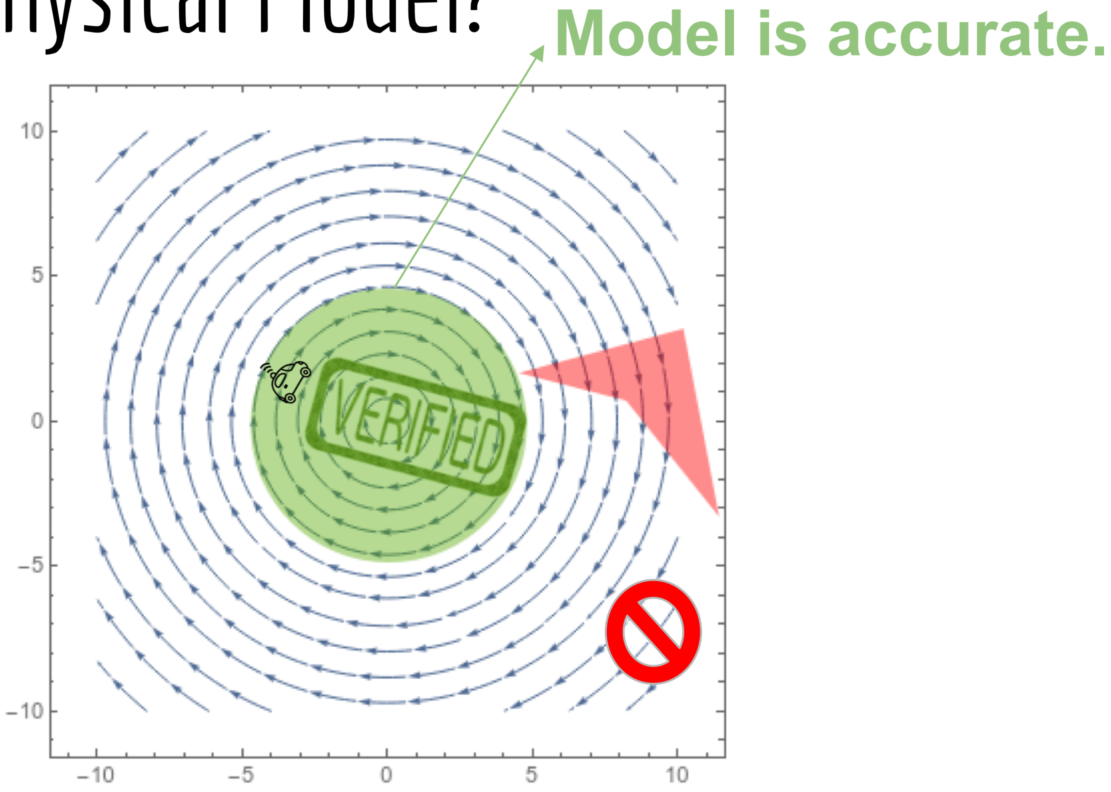


What About the Physical Model?

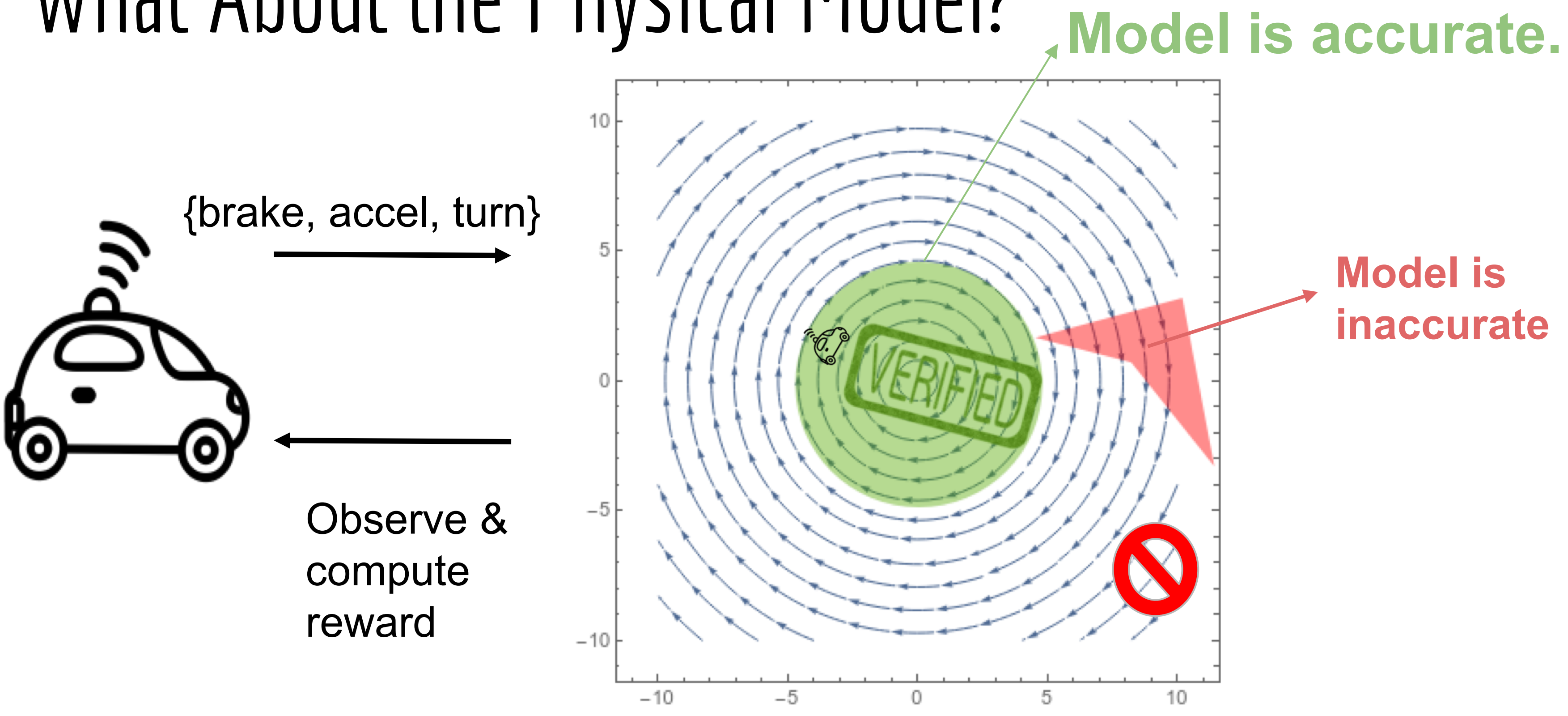


What About the Physical Model?





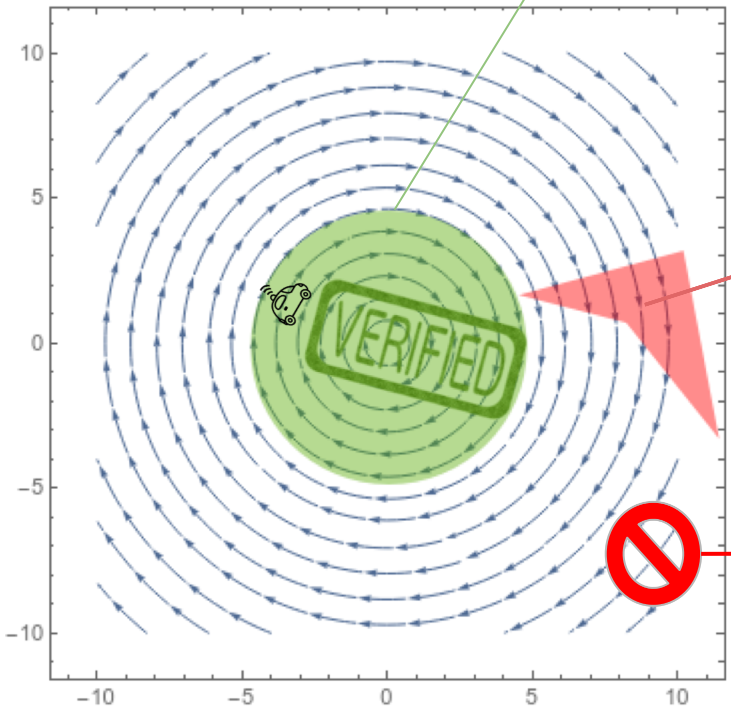
What About the Physical Model?



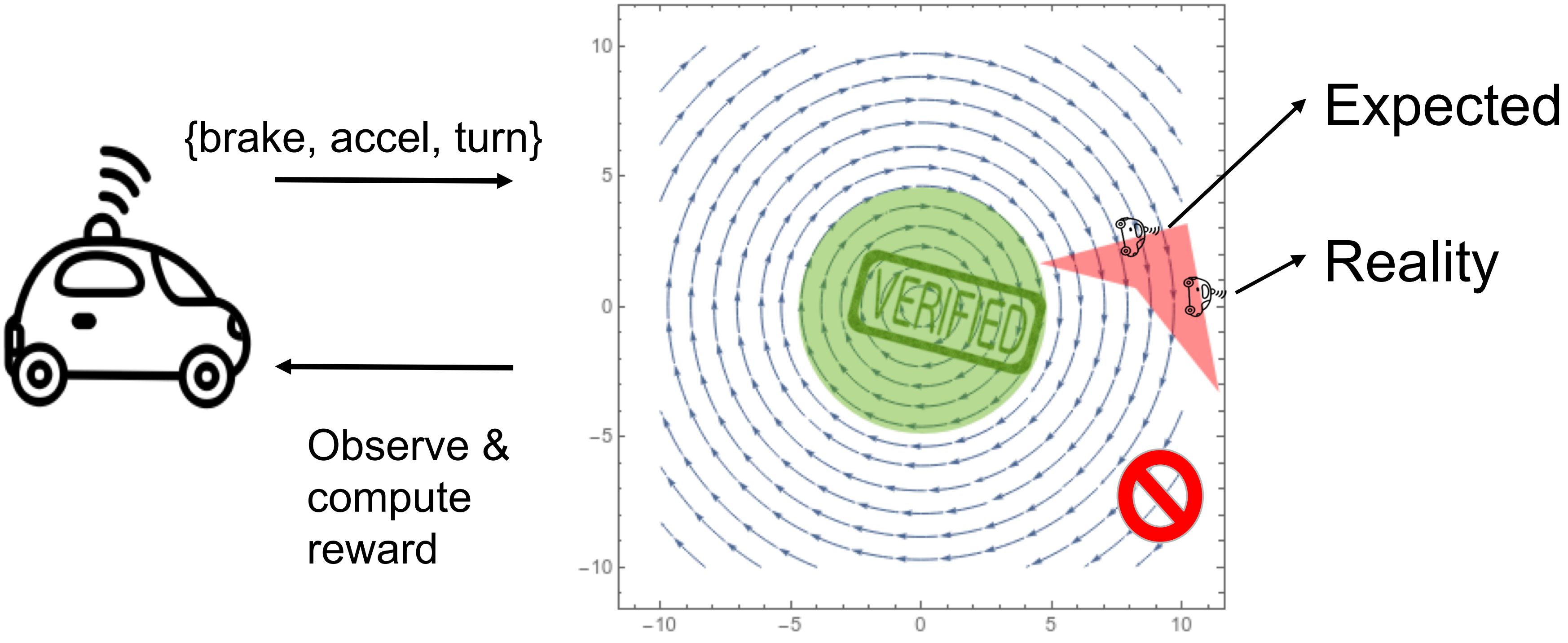
What About the Physical Model?

{brake, accel, turn}

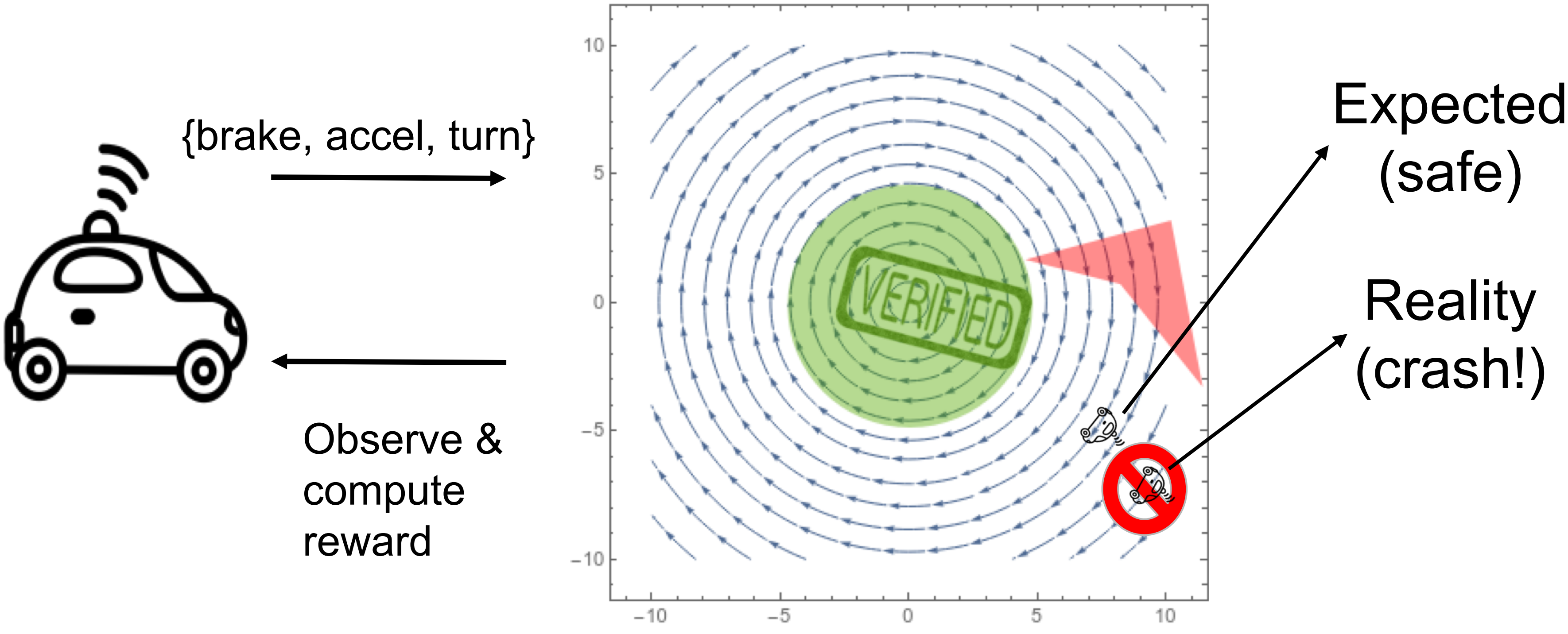
Observe & compute reward



What About the Physical Model?



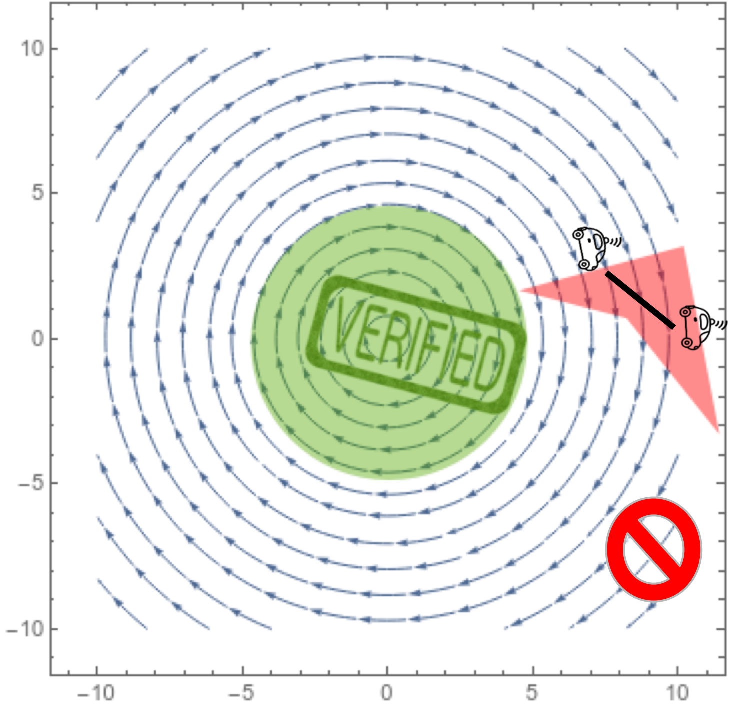
Speculation is Justified



Leveraging Verification Results to Learn Better

{brake, accel, turn}

Observe & compute reward

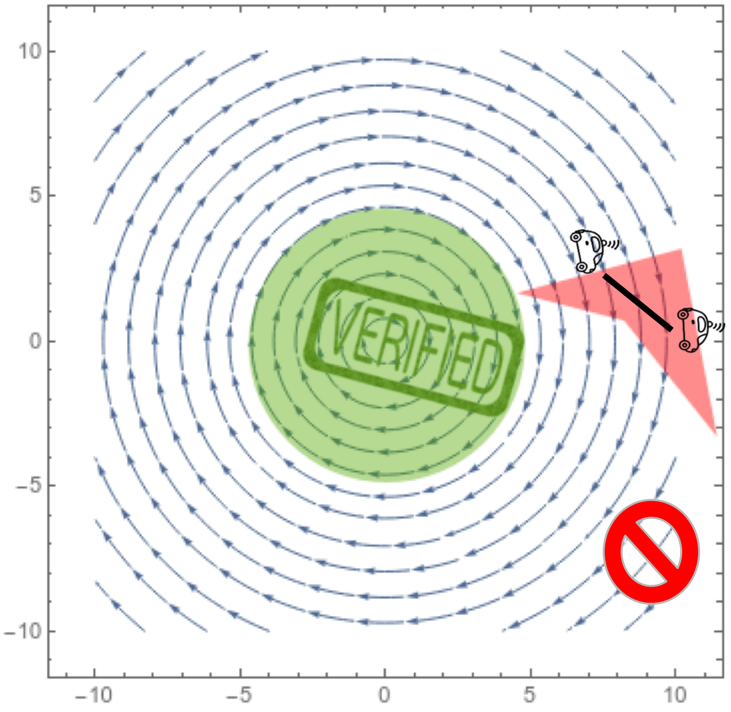


Use a real-valued version of the model monitor as a reward signal

Safe RL: How?

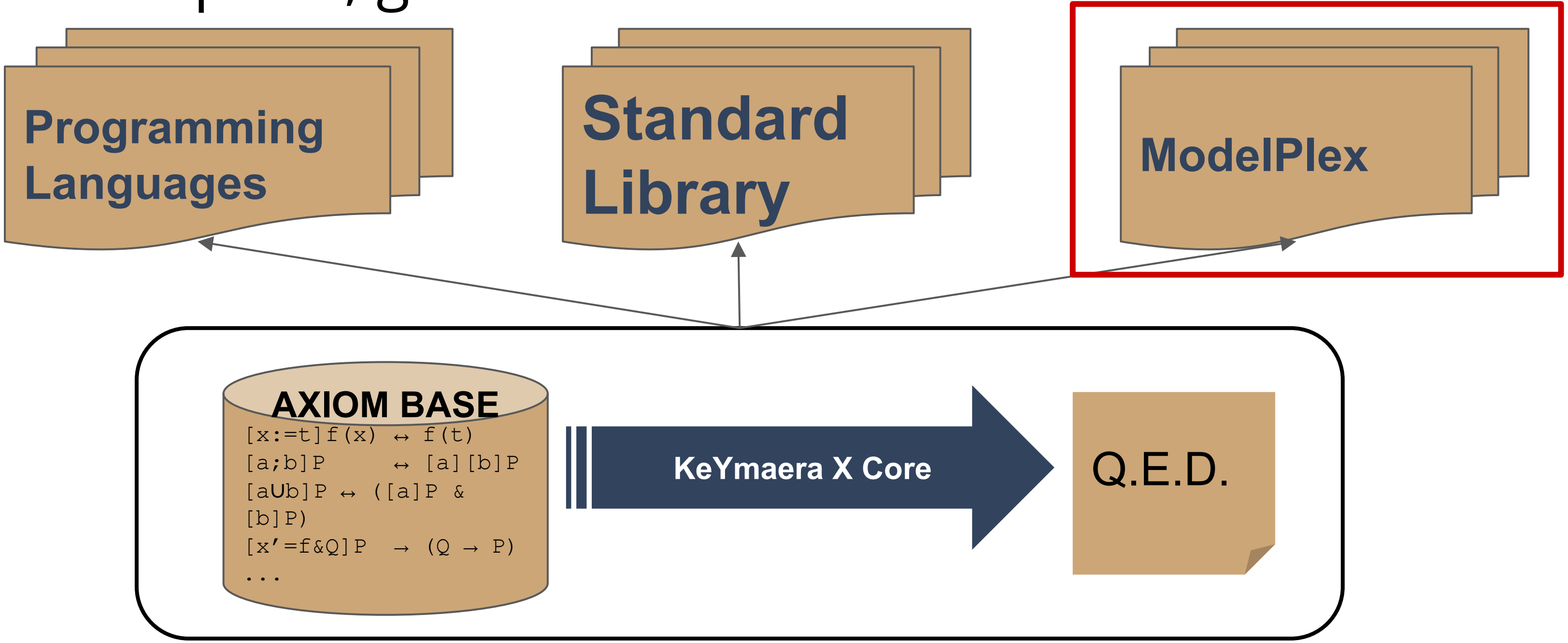
Details:

- ☐ Detect **modeled** vs **unmodeled** state space correctly at runtime.
- ☐ Convert monitors into reward signals



Detecting unmodeled State Space

The ModelPlex algorithm, implemented using Bellerophon, generates **verified runtime monitors**.



Detecting unmodeled State Space

```
oldPos := read_sensor(GPS)
actuate(accel)
newPos := read_sensor(GPS)
if (∃t. model_after(t) == newPos):
        # No model deviation.
else:
        # Model deviation…?
```

Detecting unmodeled State Space

```
oldPos := read_sensor(GPS)
actuate(accel)
newPos := read_sensor(GPS)
if (∃t. model_after(t) == newPos):
      # No model deviation.
else:
      # Model deviation…?
```

Detecting unmodeled State Space

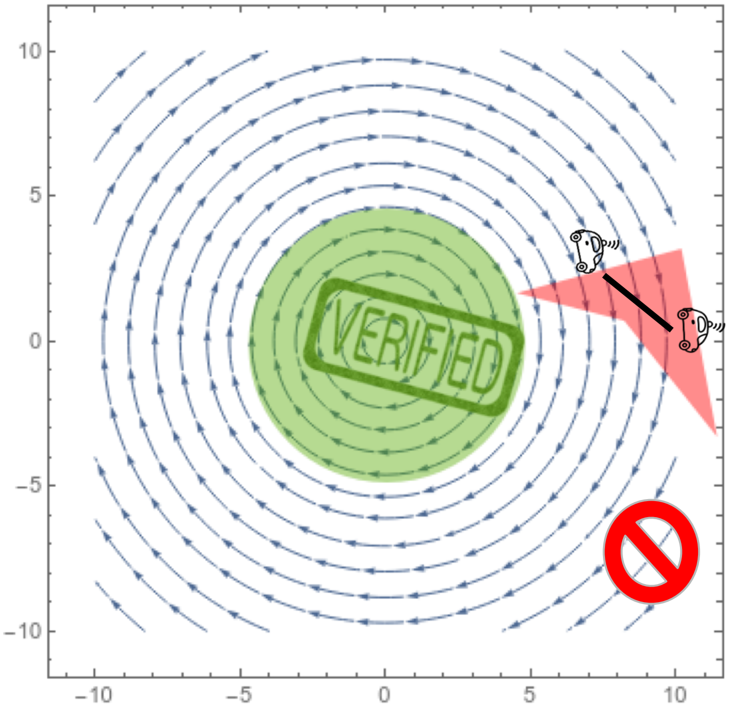
```
oldPos := read_sensor(GPS)
actuate(accel)
newPos := read_sensor(GPS)
if (QE(∃t. model_after(t) == newPos)):
      # No model deviation.
else:
      # Model deviation…?
```

Safe RL: How?

Details:

Runtime monitoring separates **modeled** from **unmodeled** state space.

- ☐ Convert monitors into reward signals

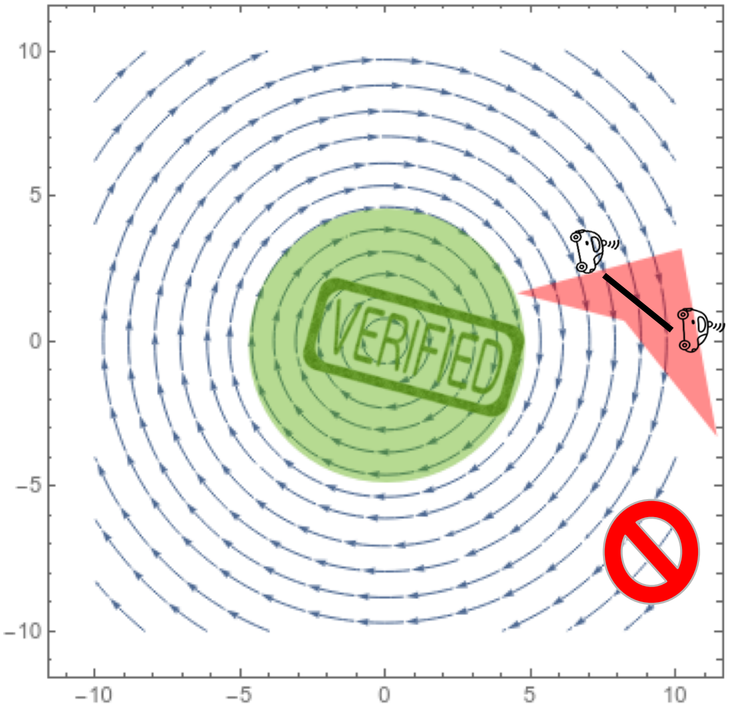


Safe RL: How?

Details:

Runtime monitoring separates **modeled** from **unmodeled** state space.

☐ Convert monitors into reward signals: $(\mathbb{R}^n \to \mathbb{B}) \to (\mathbb{R}^n \to \mathbb{R})$!?



An Example

init → [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaint; maintVel};
{pos’ = vel, vel’ = acc, t’=1}
```

}*]safe

An Example Monitor

init → [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaintain; maintainVel};
{pos' = vel, vel' = acc, t'=1}
```

}*]safe

$(t_{post} >= 0 \wedge a_{post} = acc \wedge v_{post} = acc\ t_{post} + v \wedge p_{post} = acc\ t_{post}^2/2 + v\ t_{post} + p) \vee$

$(t_{post} >= 0 \wedge a_{post} = 0 \wedge v_{post} = v \wedge p_{post} = vt_{post} + p) \vee$ Etc.

An Example Monitor

init $\rightarrow$ [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaintain; maintainVel};
{pos' = vel, vel' = acc, t'=1}
```

}*]safe

$(t_{post} >= 0 \land a_{post} = accel \land \mathbf{v_{post} = acc\ t_{post} + v} \land p_{post} = acc\ t_{post}^2/2 + v\ t_{post} + p) \lor$

$(t_{post} >= 0 \land a_{post} = 0 \land \boldsymbol{v_{post} = v} \land p_{post} = vt_{post} + p) \lor$ Etc.

An Example: The Monitor

init → [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaintain; maintainVel};
{pos' = vel, vel' = acc, t'=1}
```

}*]safe

$(t_{post} >= 0 \wedge a_{post} = acc \wedge v_{post} = accel\ t_{post} + v \wedge \mathbf{p_{post} = acc\ t_{post}^2/2 + v\ t_{post} + p}) \vee$

$(t_{post} >= 0 \wedge a_{post} = 0 \wedge v_{post} = v \wedge \boldsymbol{p_{post} = vt_{post} + p}) \vee$ Etc.

An Example: The Monitor

- Q.E. for RCF
- ODE solutions backed by proofs

init → [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaintain; maintainVel};
{pos’ = vel, vel’ = acc, t’=1}
```

}*]safe

$(t_{post} >= 0 \wedge a_{post} = acc \wedge v_{post} = accel\ t_{post} + v \wedge \mathbf{p_{post} = acc\ t_{post}{}^2/2 + v\ t_{post} + p}) \vee$

$(t_{post} >= 0 \wedge a_{post} = 0 \wedge v_{post} = v \wedge \boldsymbol{p_{post} = vt_{post} + p}) \vee$ Etc.

Quantitative monitor as reward signal

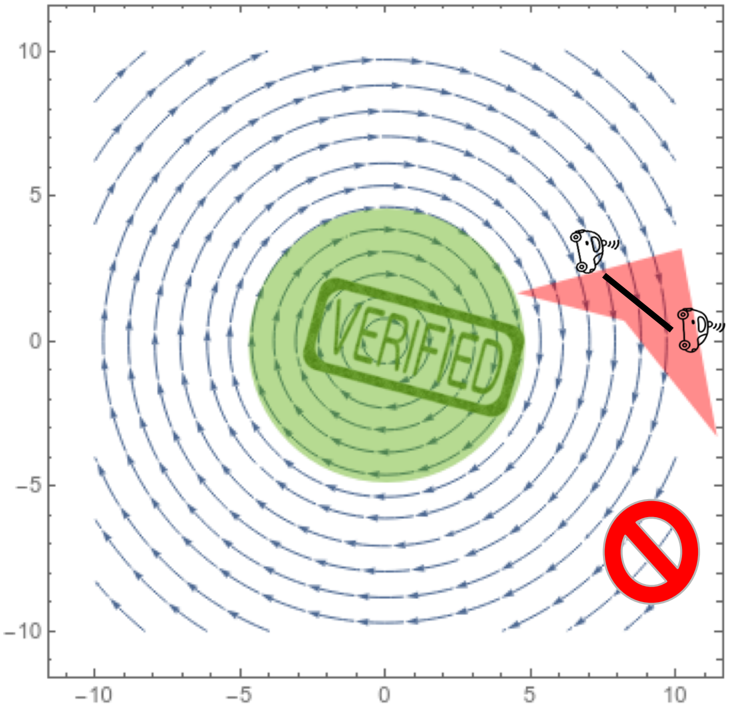
Safe RL: How?

Details:

Runtime monitoring separates **modeled** from **unmodeled** state space.

Convert monitors into gradients:

$(\mathbb{R}^n \to \mathbb{B}) \to (\mathbb{R}^n \to \mathbb{R})$



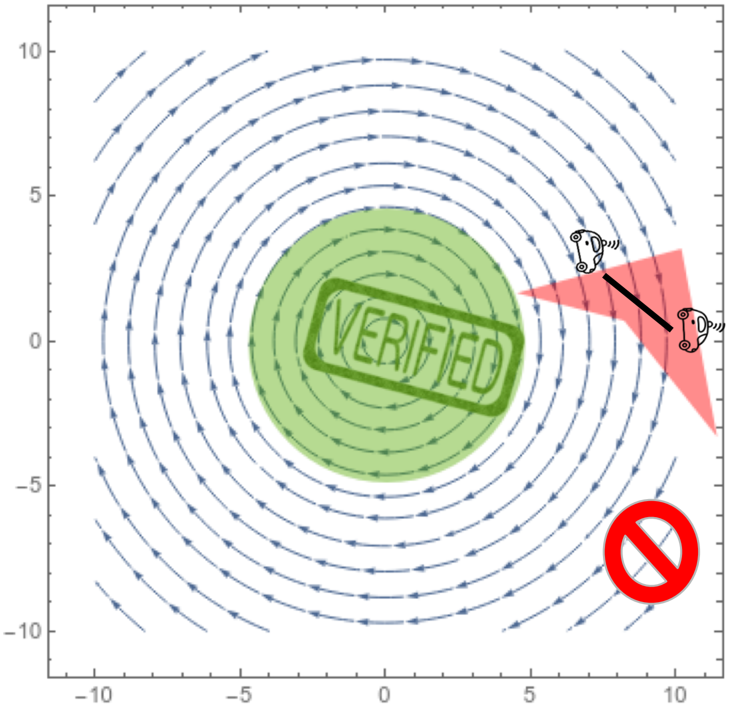
Safe RL: How?

Details:

Runtime monitoring separates **modeled** from **unmodeled** state space.

Convert **models** into gradients: ModelPlex

$(\mathbb{R}^n \rightarrow \mathbb{B}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R})$

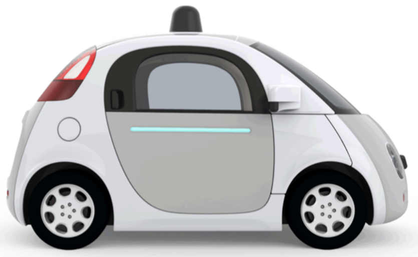


Conclusion

KeYmaera X + Justified Speculative Control provide strong safety guarantees for learning-enabled CPS.

1. Was the proof correct?
2. Was the model accurate enough?

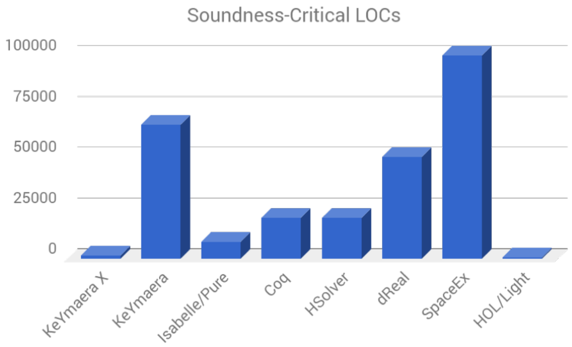
≠



Conclusion

KeYmaera X + Justified Speculative Control provide strong safety guarantees for learning-enabled CPS.

1. Was the proof correct? **KeYmaera X**
2. Was the model accurate enough?



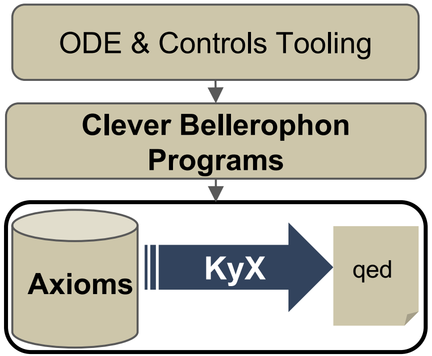
dI Tactic:

Side derivation:
$(v \geq v_0 - gt)' \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots \leftrightarrow$
$\ldots$

$H = r_p \geq 0$ & $r_a \geq 0$ & $g > 0$ & ...

DI Axiom:
$[\{x'=f \& Q\}]P \leftrightarrow ([?Q]P \leftarrow (Q \rightarrow [\{x'=f \& Q\}]P'))$

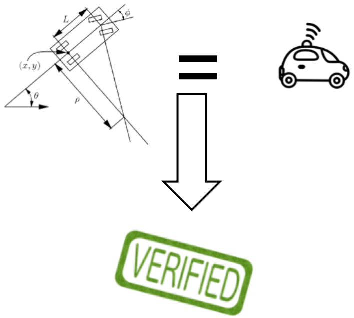
Example:
$[v'=r_p v^2 - g, t'=1] v \geq v_0 - gt \leftrightarrow$
$\ldots \leftrightarrow$
$[v' := r_p v^2 - g][t' := 1] v' \geq -g * t' \leftrightarrow$
$r_p v^2 - g \geq -g \leftrightarrow$
$H \rightarrow r_p \geq 0$

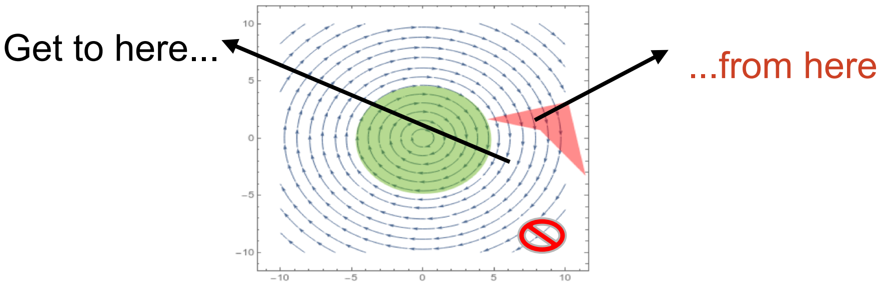


Conclusion

KeYmaera X + Justified Speculative Control provide strong safety guarantees for learning-enabled CPS.

1. Was the proof correct? **KeYmaera X**
2. Was the model accurate enough? **Justified Speculation**





Conclusion

KeYmaera X + Justified Speculative Control provide strong safety guarantees for learning-enabled CPS.

1. Was the proof correct? **KeYmaera X**
2. Was the model accurate enough? **Justified Speculation**

Web	keymaeraX.org
Online Demo	web.keymaeraX.org
Open Source (GPL)	github.com/LS-Lab/KeYmaeraX-release

Acknowledgments

Students and postdocs of the Logical Systems Lab at Carnegie Mellon
Brandon Bohrer, Nathan Fulton, Sarah Loos, João Martins, Yong Kiam Tan
Khalil Ghorbal, Jean-Baptiste Jeannin, Stefan Mitsch

A. Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer 2018