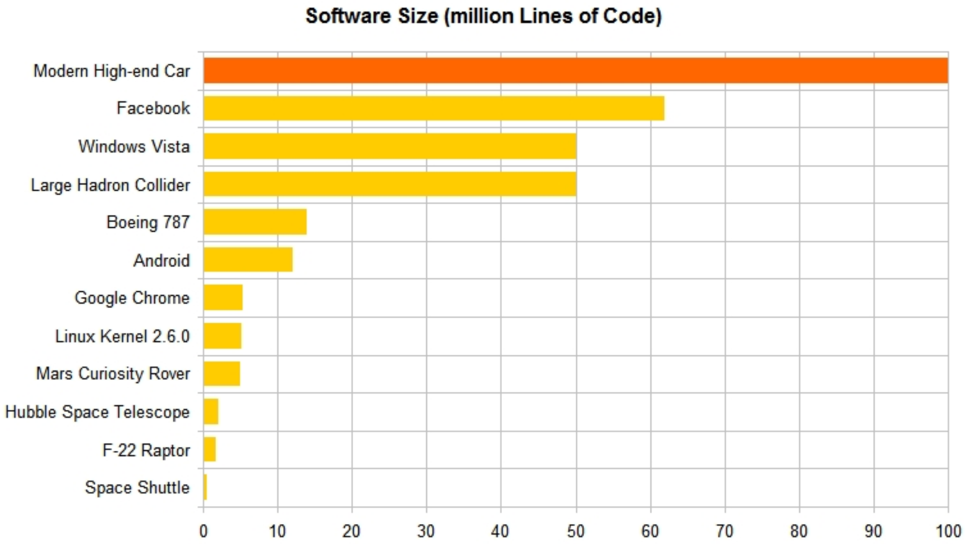
Safe AI for CPS

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

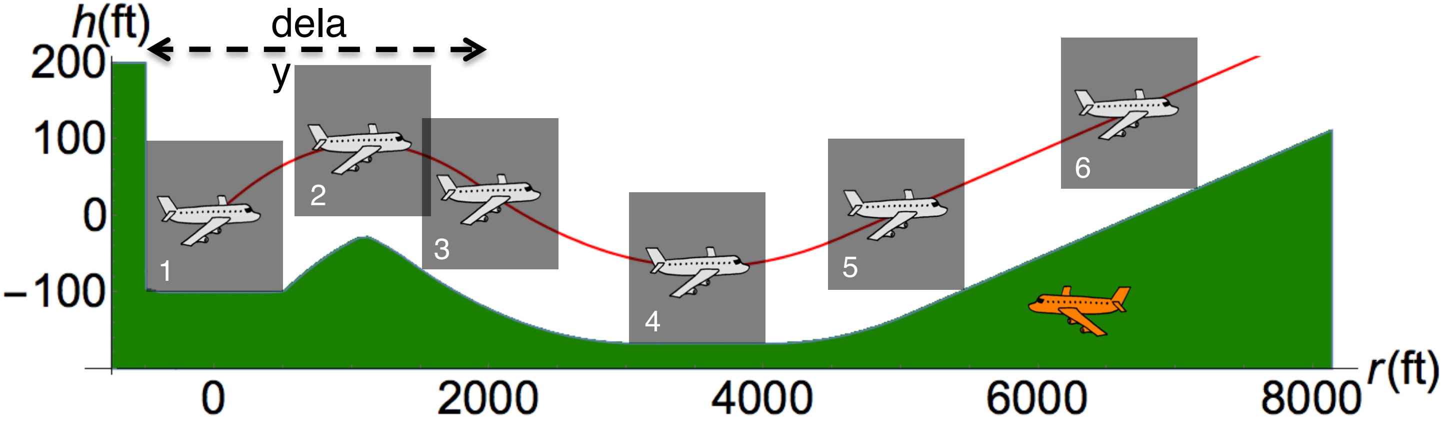
- Flexibility of learning
- Guarantees of CPS formal methods

Diametrically opposed: flexibility+adaptability versus predictability+simplicity

1. Cyber-Physical Systems with **Differential Dynamic Logic**
2. **Sandboxed reinforcement learning** is provably safe
3. **Model-update learning** addresses uncertainty with multiple models

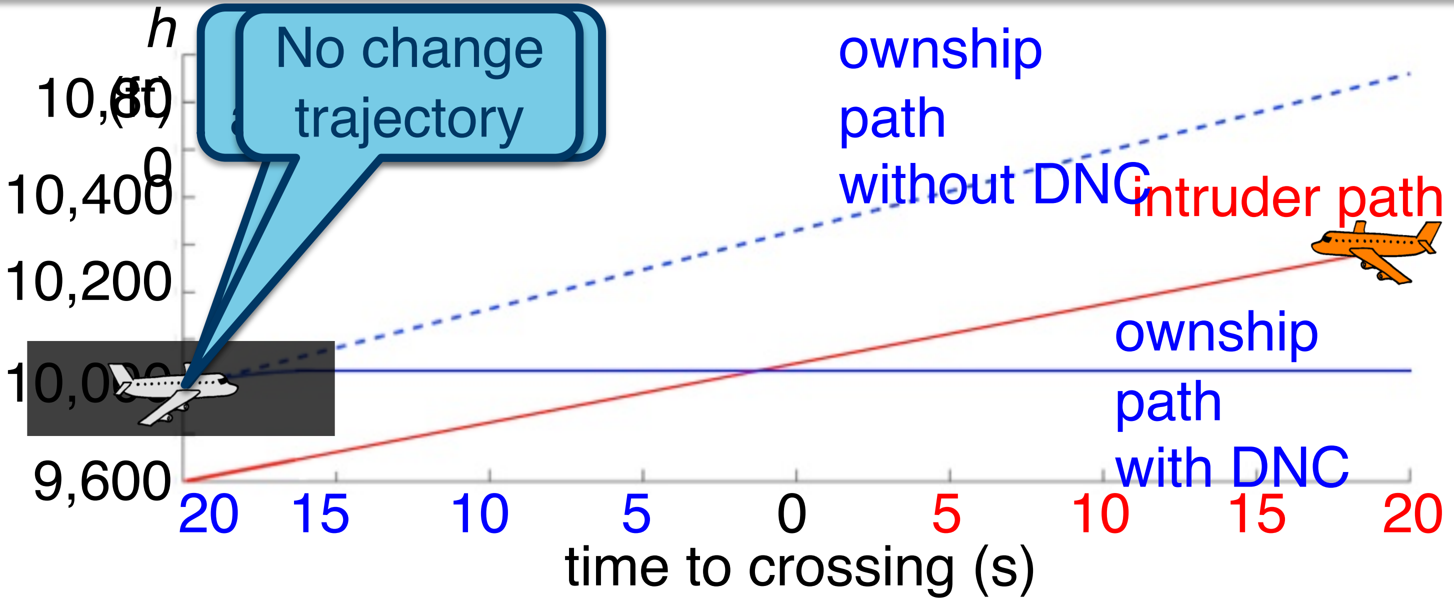
Airborne Collision Avoidance System ACAS X

- Developed by FAA to replace current TCAS in aircraft
- Approximately optimizes MDP on a grid
- Advisory from lookup tables with 5D interpolation regions
- Identified safe region per advisory and proved in KeYmaera X



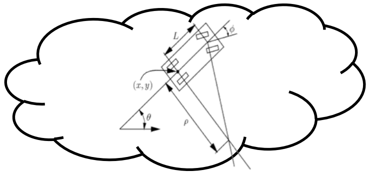
STTT'17

Comparison: ACAS X issues DNC



But CL1500 or no change would not lead to a collision

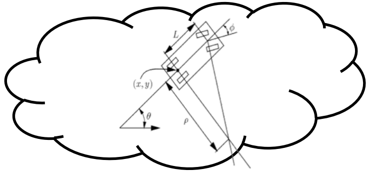
Model-Based Verification   Reinforcement Learning



φ

Model-Based Verification

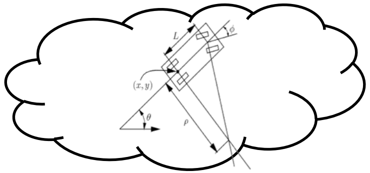
Reinforcement Learning



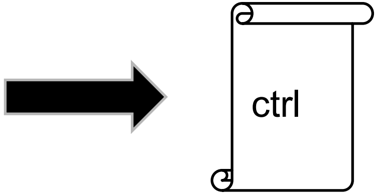
pos < stopSign

Model-Based Verification

Reinforcement Learning

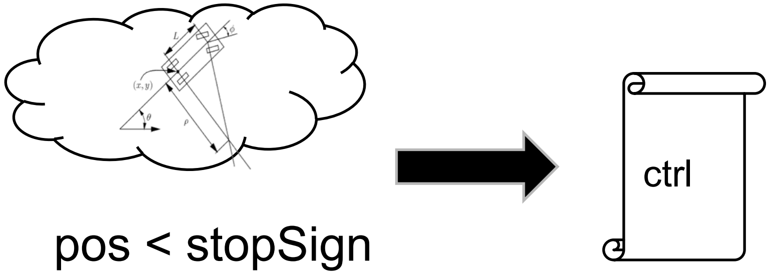


pos < stopSign



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

Benefits:

- Strong safety guarantees
- Automated analysis

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”

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

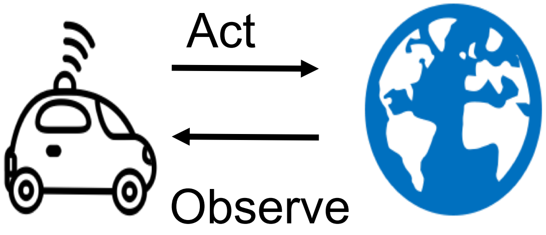
Benefits:

- 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



Model-Based Verification

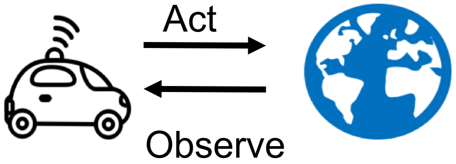
Benefits:

- 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

Model-Based Verification

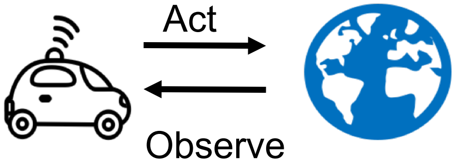
Benefits:

- 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

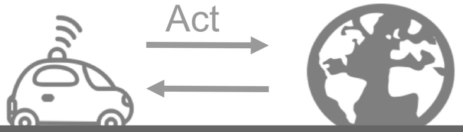
Bene

- S
- A

Draw

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

Reinforcement Learning



del

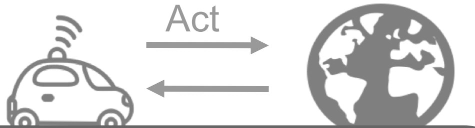
s

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

Goal: Provably correct reinforcement learning

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

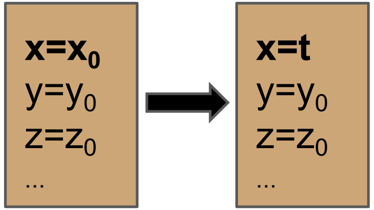
- No strong safety guarantees
- Proofs are obtained and checked by hand
- Formal proofs = decades-long proof development

Part I: Differential Dynamic Logic

Trustworthy Proofs for Hybrid Systems

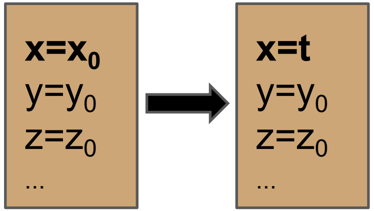
Hybrid Programs

$x := t$

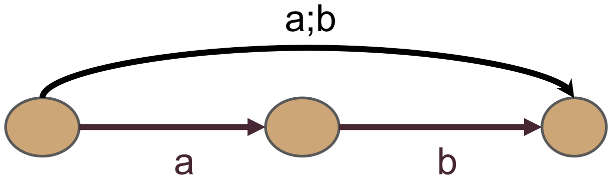


Hybrid Programs

$x := t$

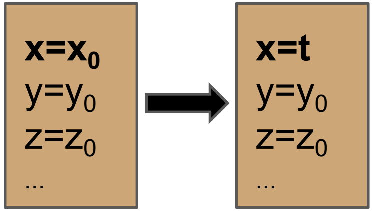


$a;b$

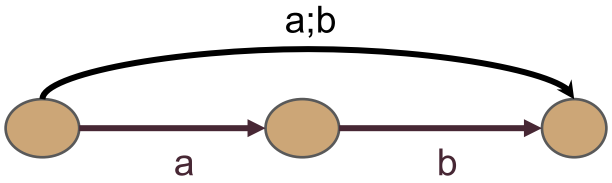


Hybrid Programs

x := t



a;b



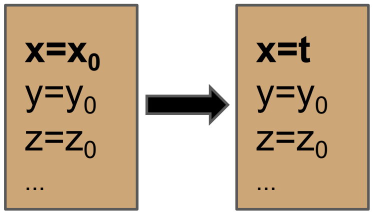
?P

If P is true: no change

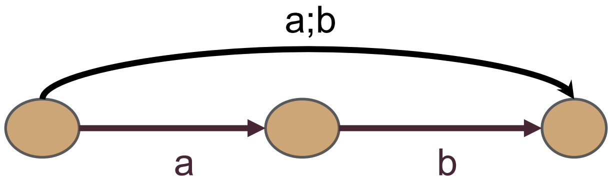
If P is false: terminate

Hybrid Programs

x := t



a;b

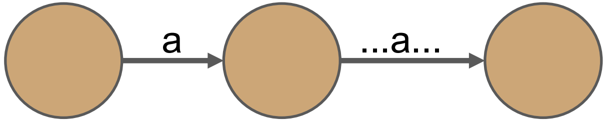


?P

If P is true: no change

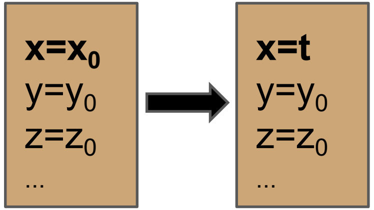
If P is false: terminate

a*

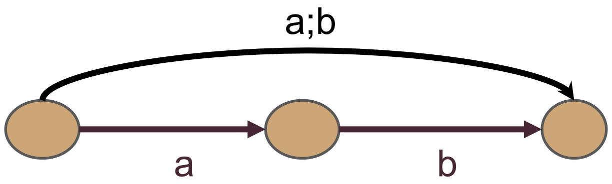


Hybrid Programs

x := t



a;b

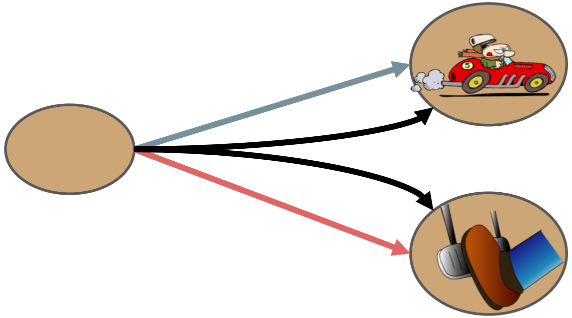


?P

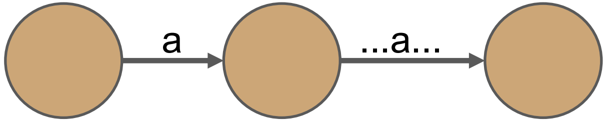
If P is true: no change

If P is false: terminate

a∪b

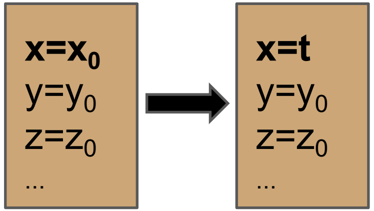


a*



Hybrid Programs

x := t

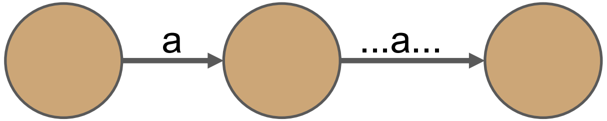


?P

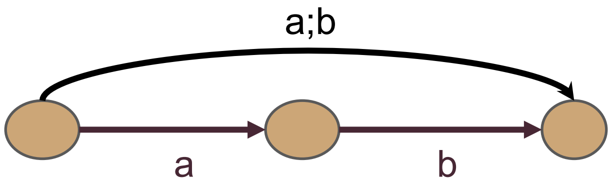
If P is true: no change

If P is false: terminate

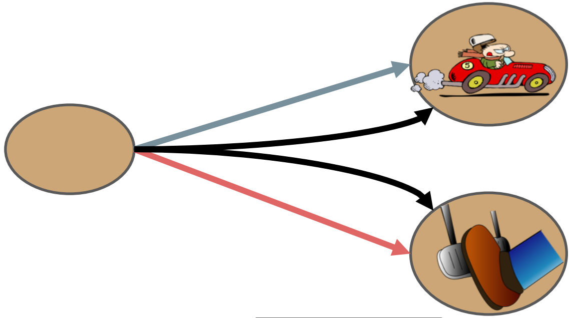
a*



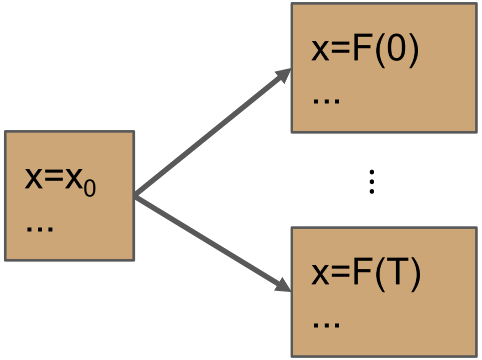
a;b



a∪b



x'=f(x)



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

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

Proofs give strong mathematical evidence of safety.

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

The Fundamental Question

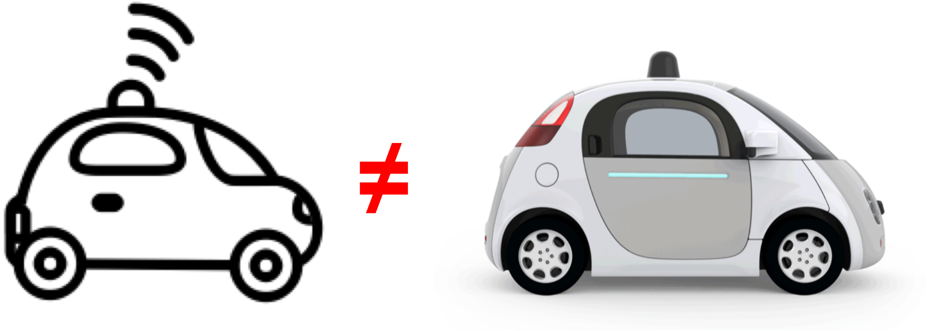
Why would our program not work if we have a *proof*?

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?

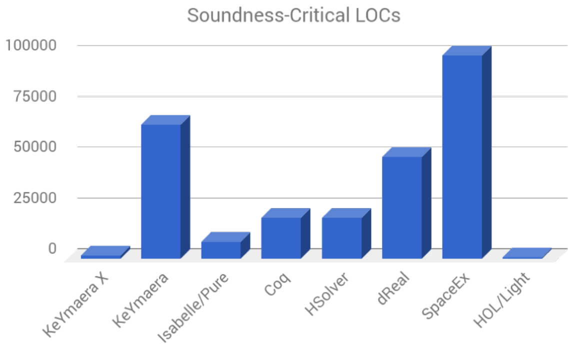


The Fundamental Question

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

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

2. Was the model accurate enough?



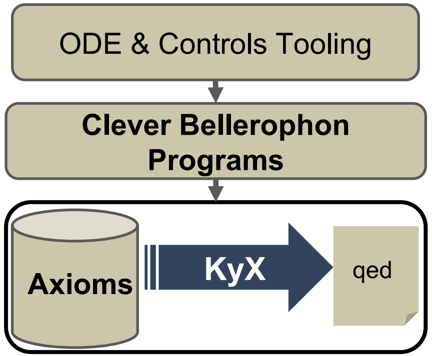
dI Tactic:

Side derivation:
(v ≥ v_0 − gt)′ ↔
...↔
...↔
...

H=r_p≥0 & r_a≥0 & g>0 & ...

DI Axiom:
[{x'=f&Q}]P↔([?Q]P←(Q→[{x'=f&Q}]P'))

Example:
[v′=r_pv²−g,t′=1]v ≥ v_0 − gt ↔
... ↔
[v′:=r_pv²−g][t′:=1]v′ ≥ −g*t′ ↔
r_pv²−g ≥ −g ↔
H→r_p≥0

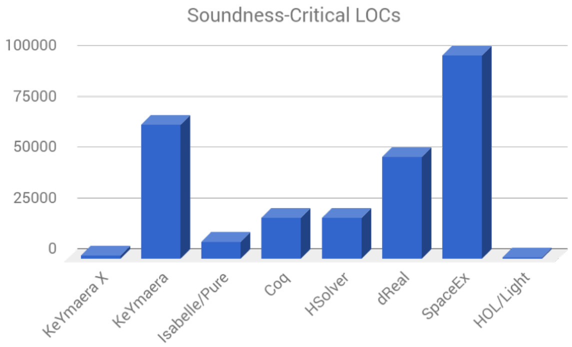


The Fundamental Question

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**



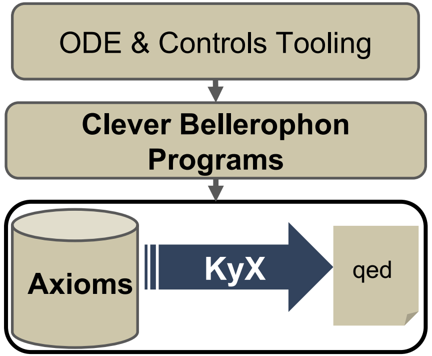
dI Tactic:

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

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

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

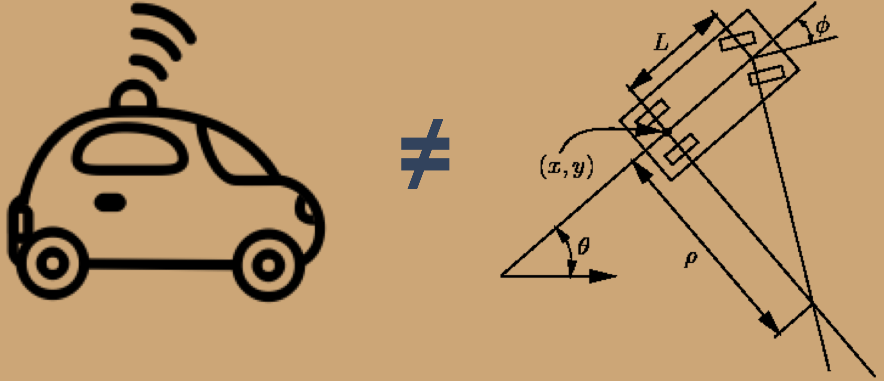
$H = r_p \geq 0 \;\&\; r_a \geq 0 \;\&\; g>0 \;\&\; \ldots$



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}
}*
```

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

Continuous motion

Model-Based Verification

Accurate, **analyzable** models often exist!

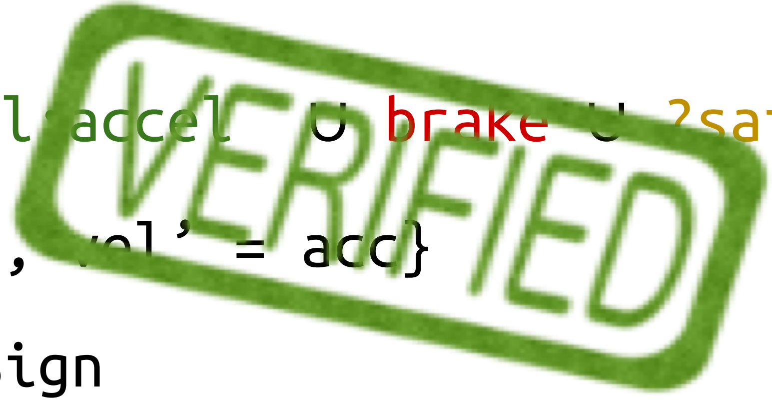
```
init → [{
   { ?safeAccel;accel  ∪ brake ∪ ?safeTurn; turn};
   {pos’ = vel, vel’ = acc}
}*]pos < stopSign
```

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 ∪ brake ∪ ?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:

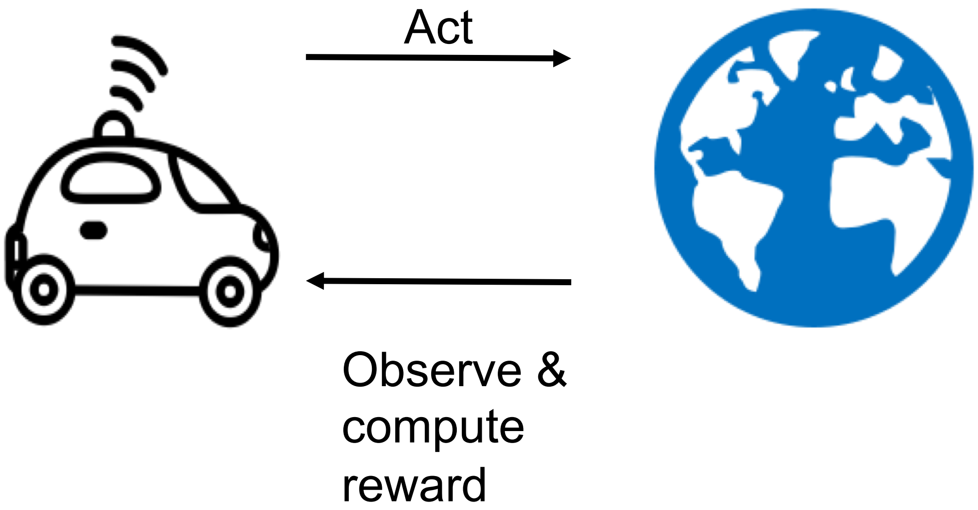
1. learns to resolve nondeterminism without sacrificing formal safety results

Safe RL Contribution

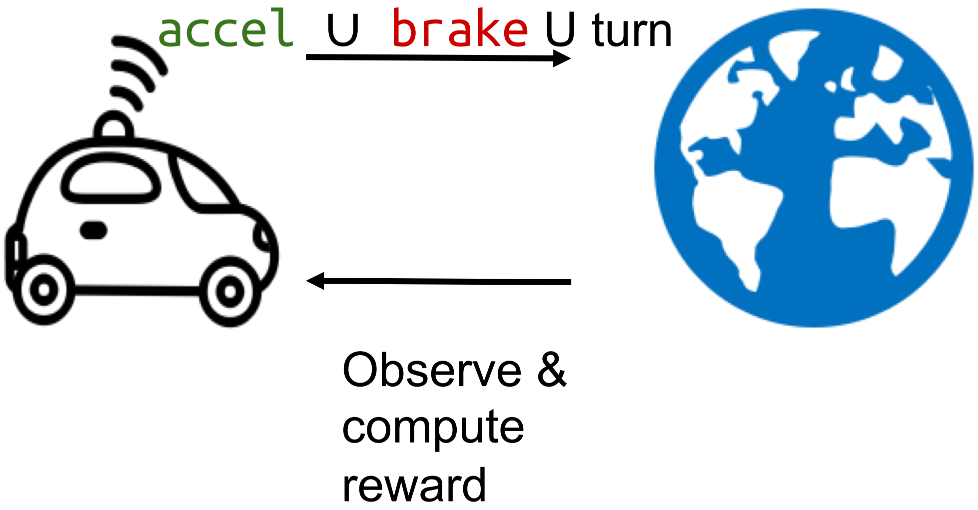
Justified Speculative Control is an approach toward provably safe reinforcement learning that:

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

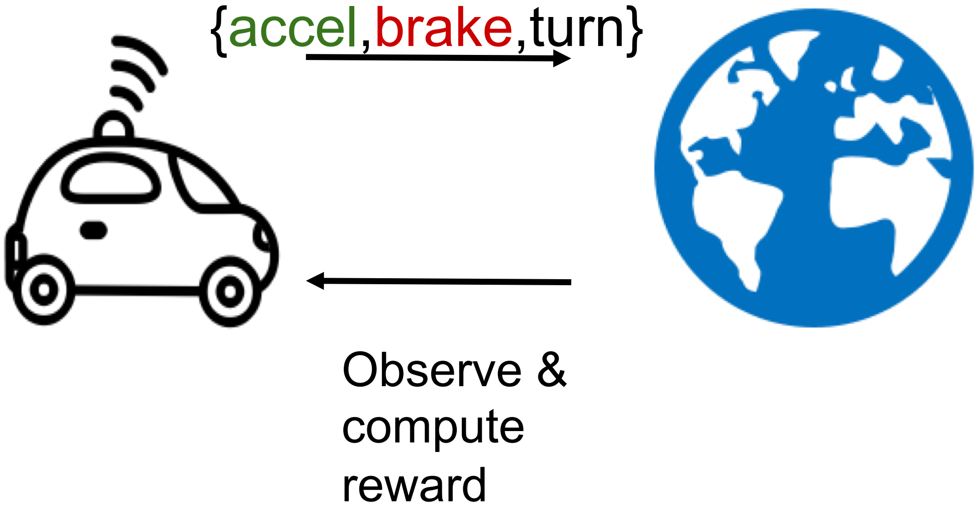
Learning to Resolve Non-determinism



Learning to Resolve Non-determinism



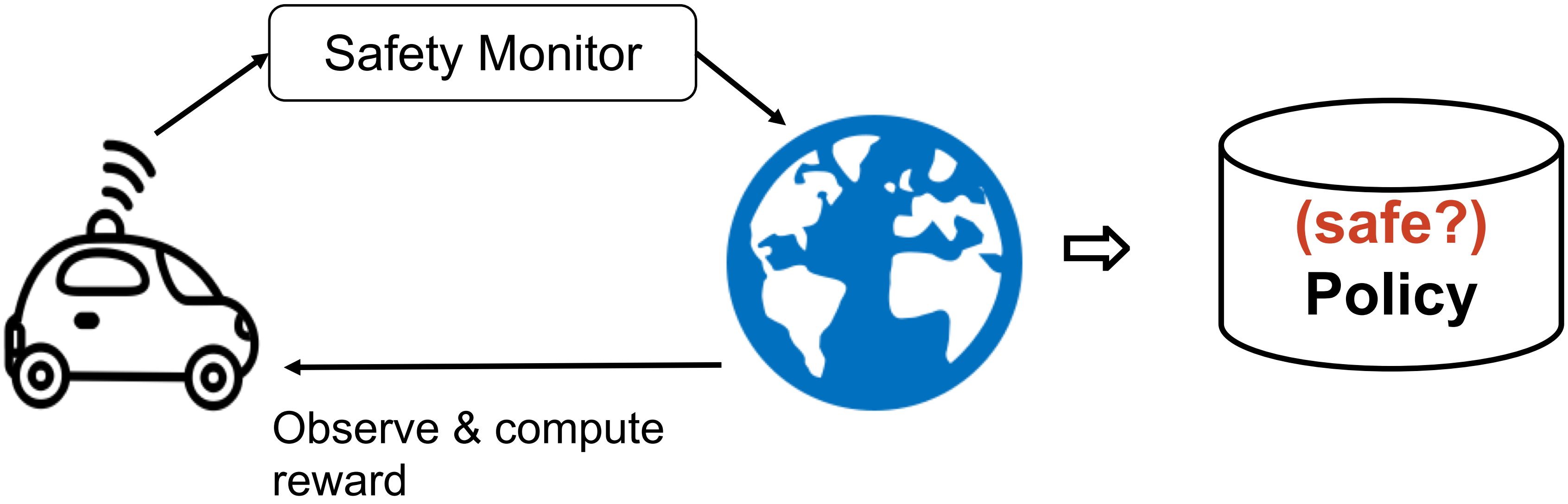
Learning to Resolve Non-determinism



Learning to Resolve Non-determinism

Learning to Resolve Non-determinism

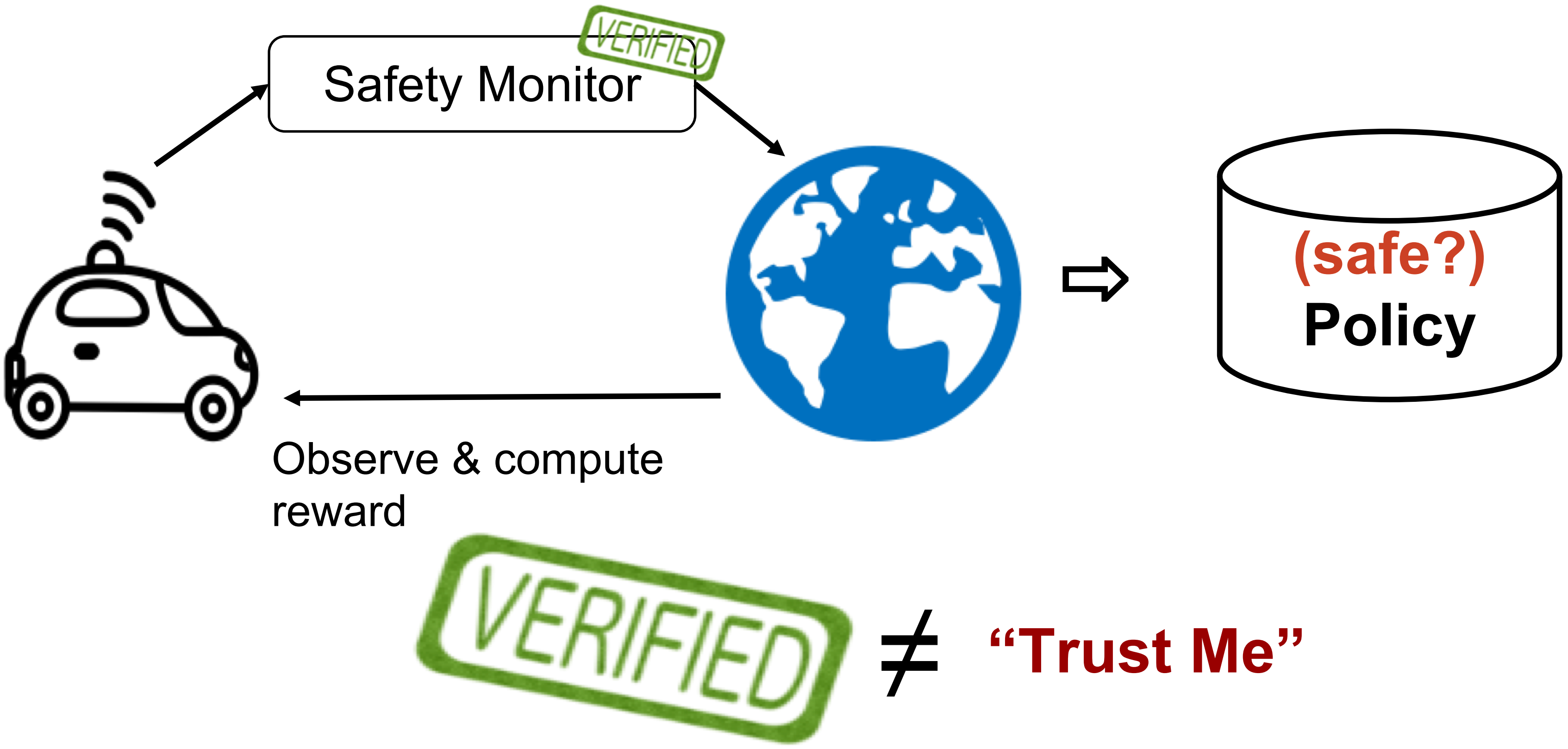
Learning to **Safely** Resolve Non-determinism



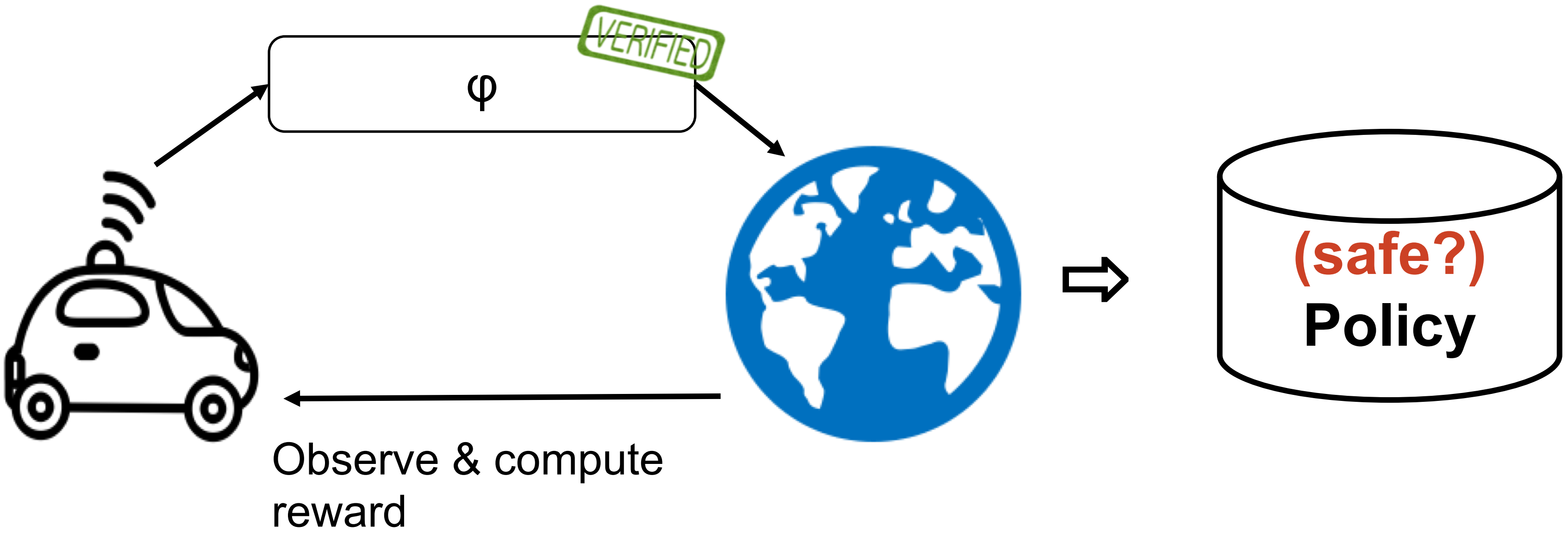
Useful to stay safe during learning

Crucial after deployment

Learning to **Safely** Resolve Non-determinism



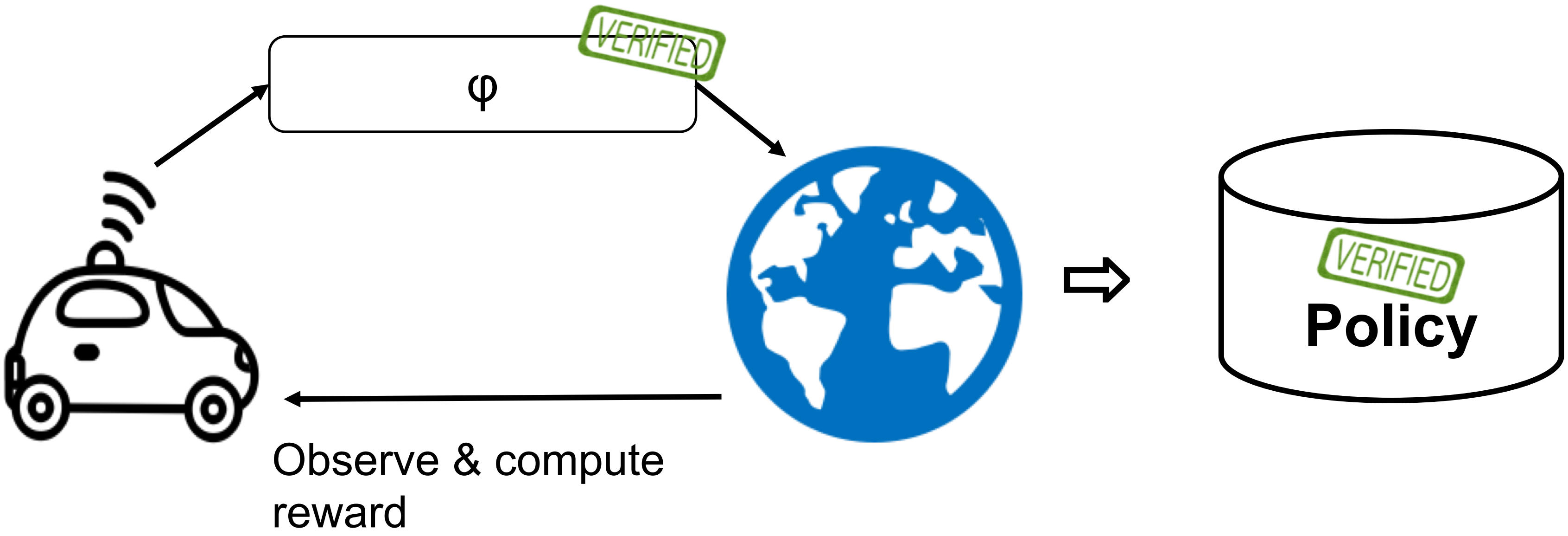
Learning to **Safely** Resolve Non-determinism



Use a theorem prover to extract:

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

Learning to **Safely** Resolve Non-determinism



Use a theorem prover to extract:

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

Learning to **Safely** Resolve Non-determinism

φ

VERIFIED

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

Observe & compute reward

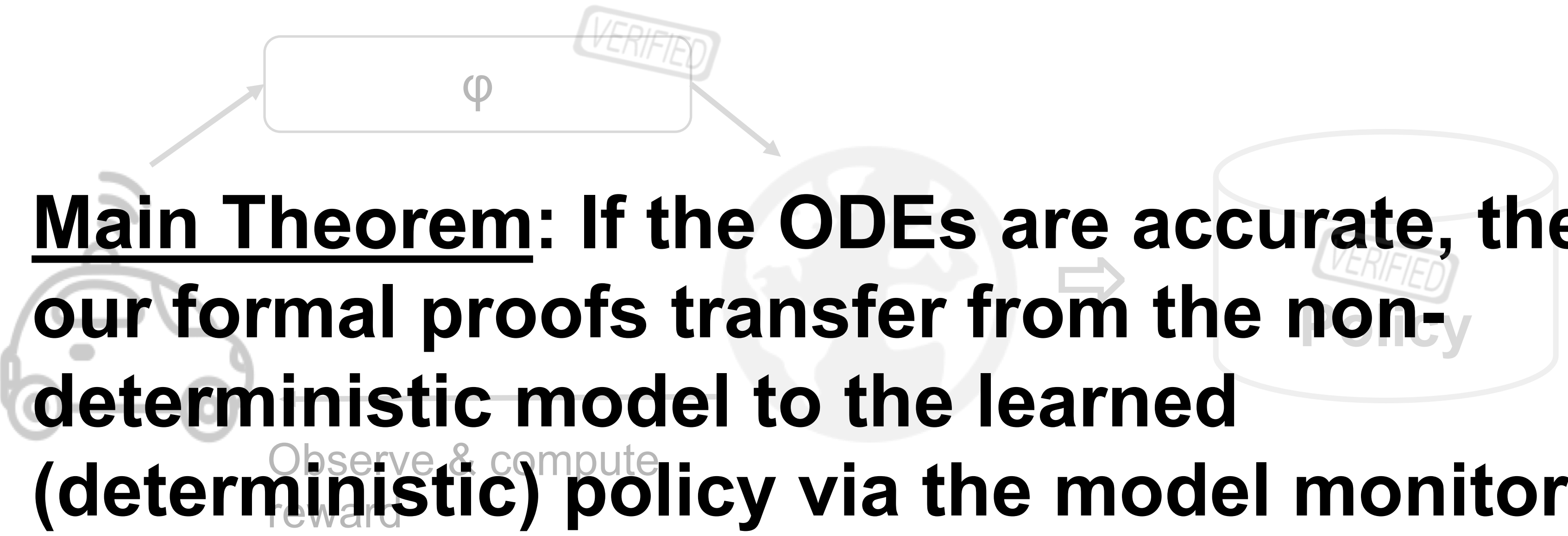
Policy

VERIFIED

Use a theorem prover to extract:

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

Learning to **Safely** Resolve Non-determinism

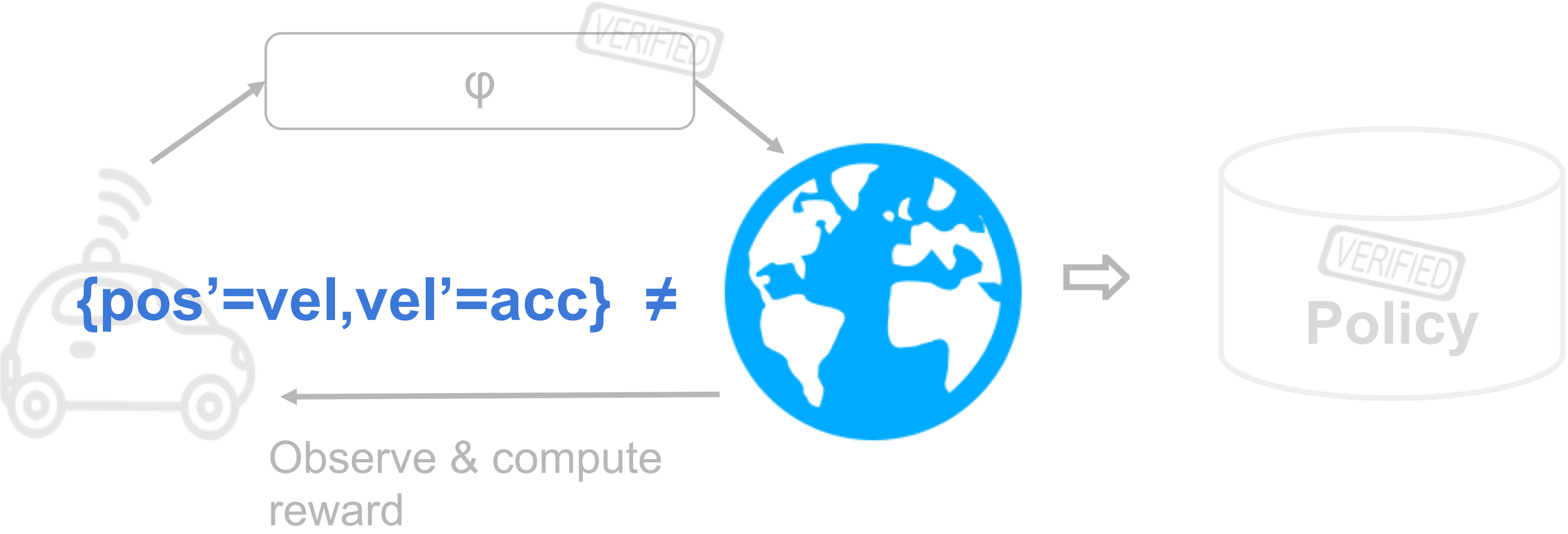


<u>Main Theorem</u>: If the ODEs are accurate, then our formal proofs transfer from the non-deterministic model to the learned (deterministic) policy via the model monitor.

Use a theorem prover to extract:

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

What about the physical model?



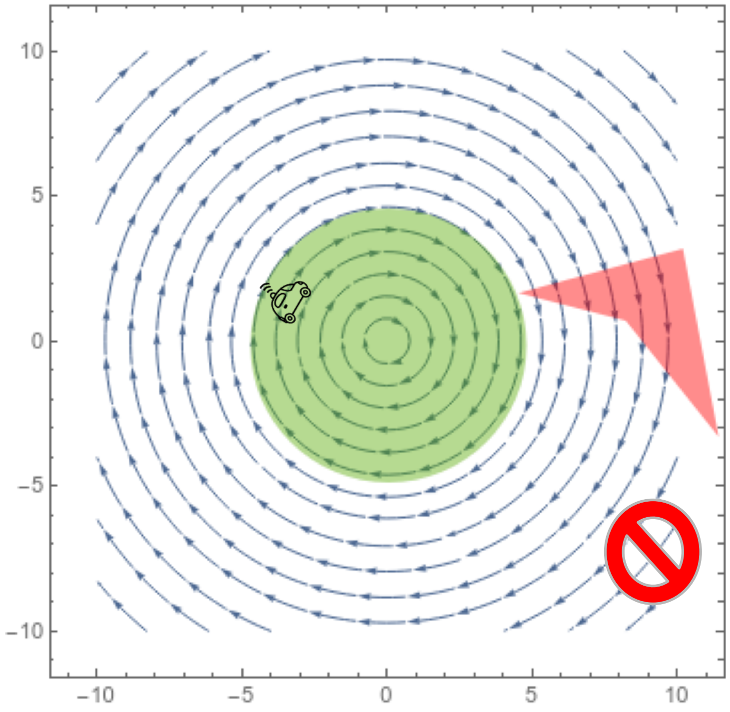
Use a theorem prover to extract:

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

What About the Physical Model?

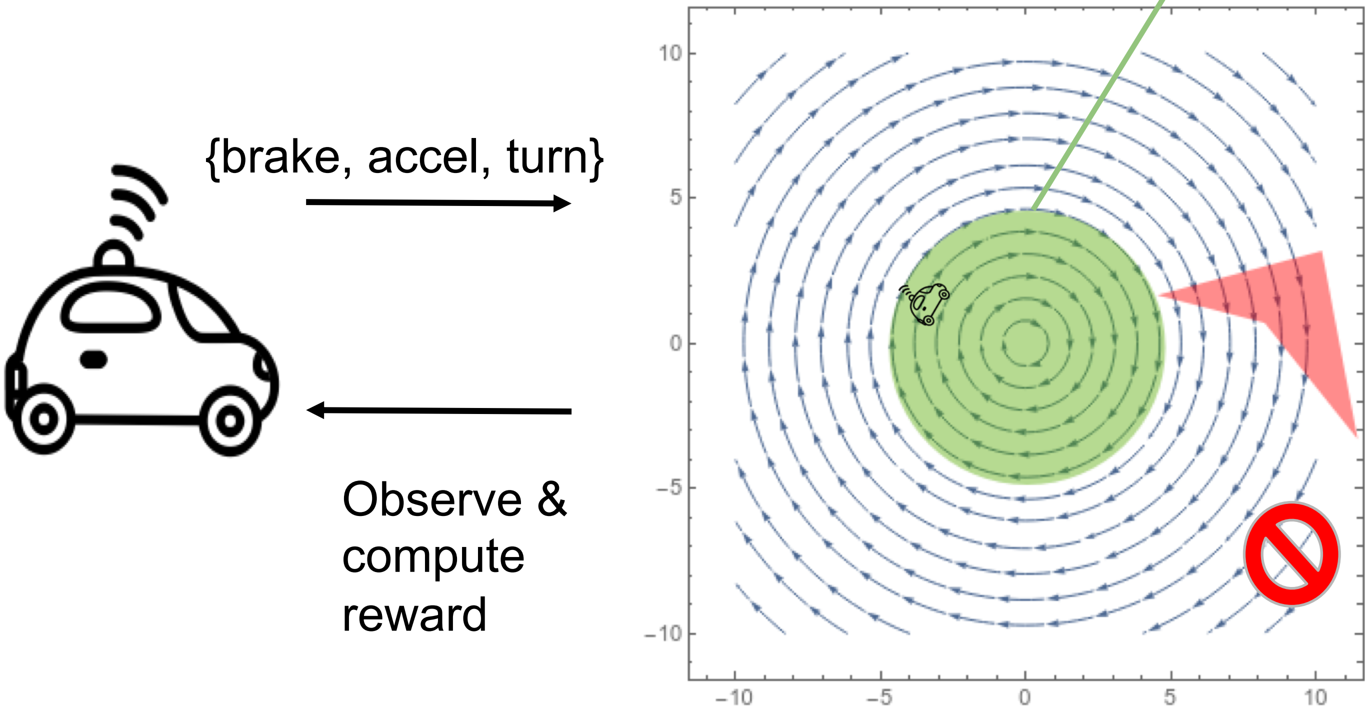
{brake, accel, turn}

Observe & compute reward



What About the Physical Model?

Model is accurate.

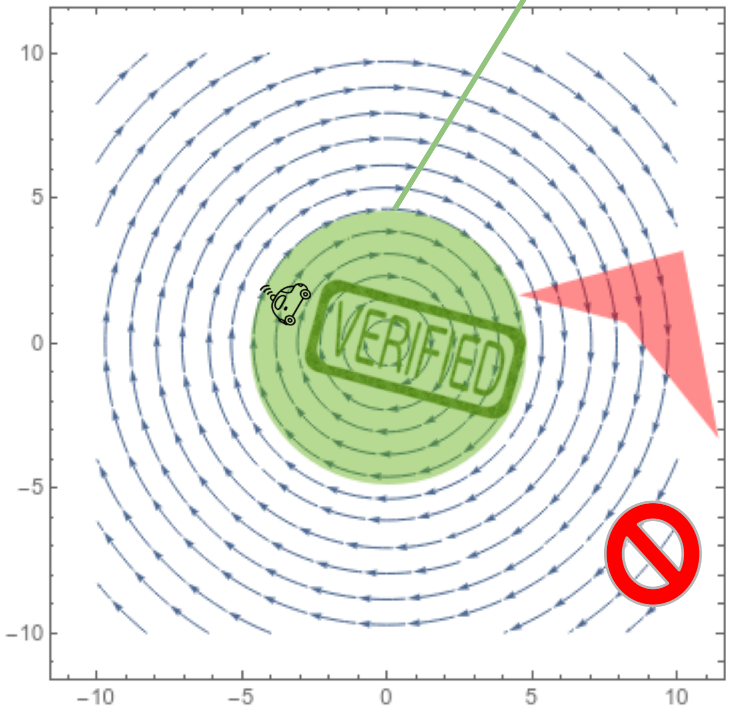


What About the Physical Model?

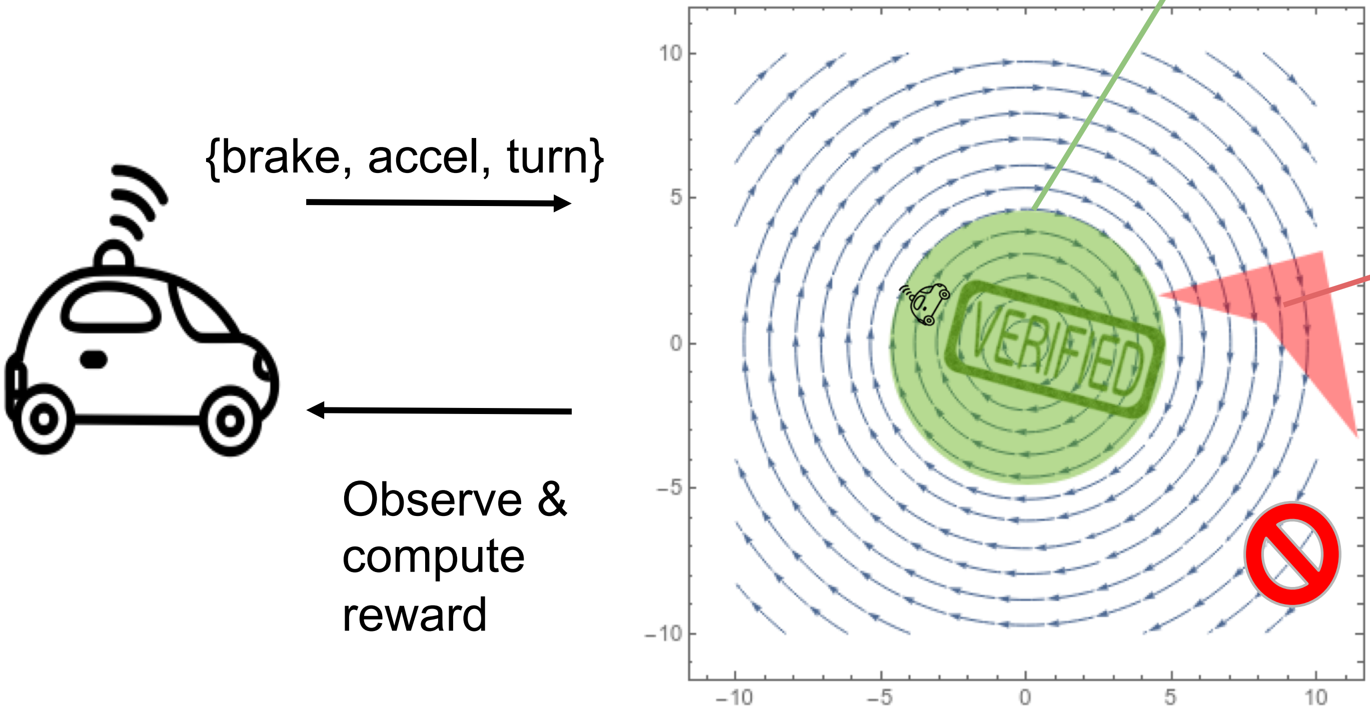
{brake, accel, turn}

Observe &
compute
reward

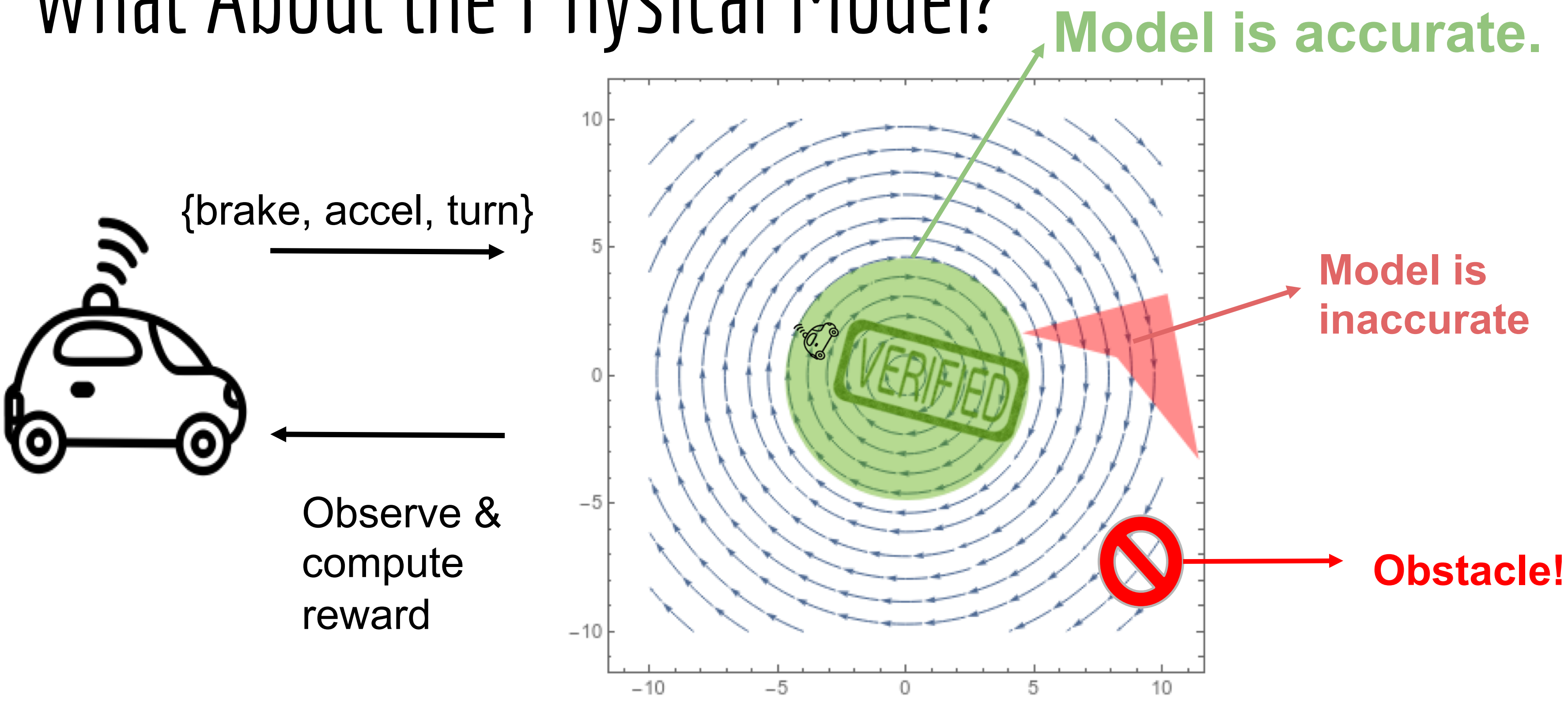
Model is accurate.



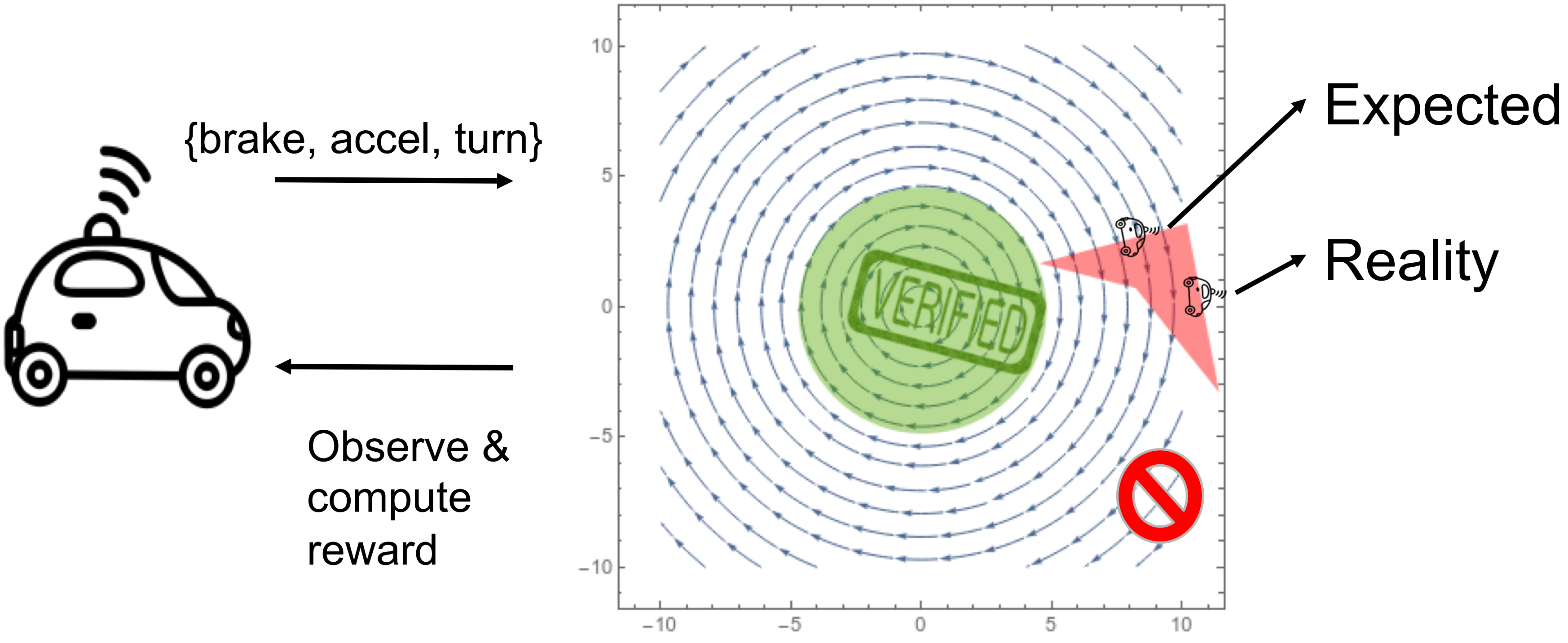
What About the Physical Model?



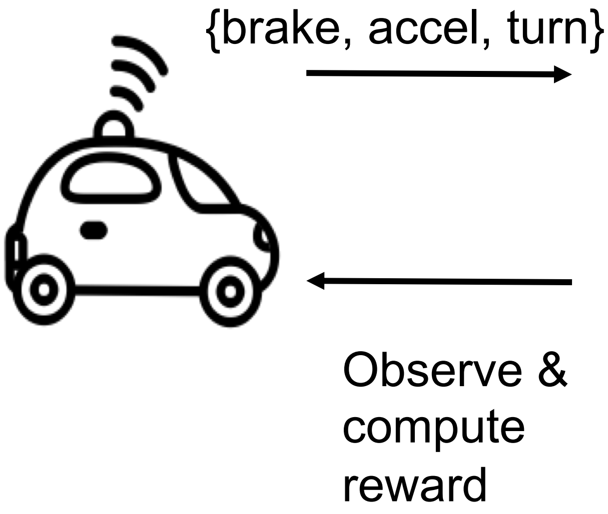
What About the Physical Model?

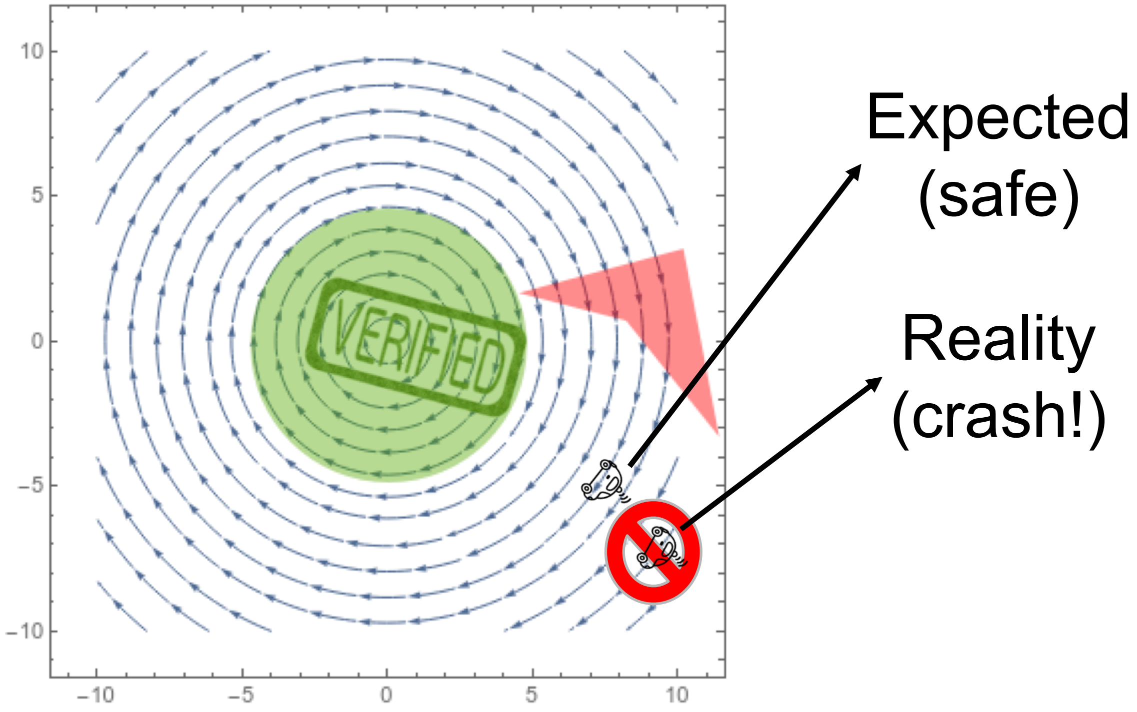


What About the Physical Model?

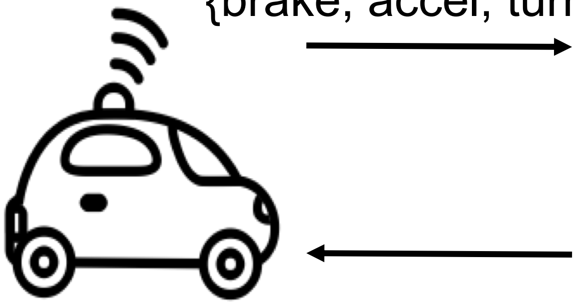


Speculation is Justified



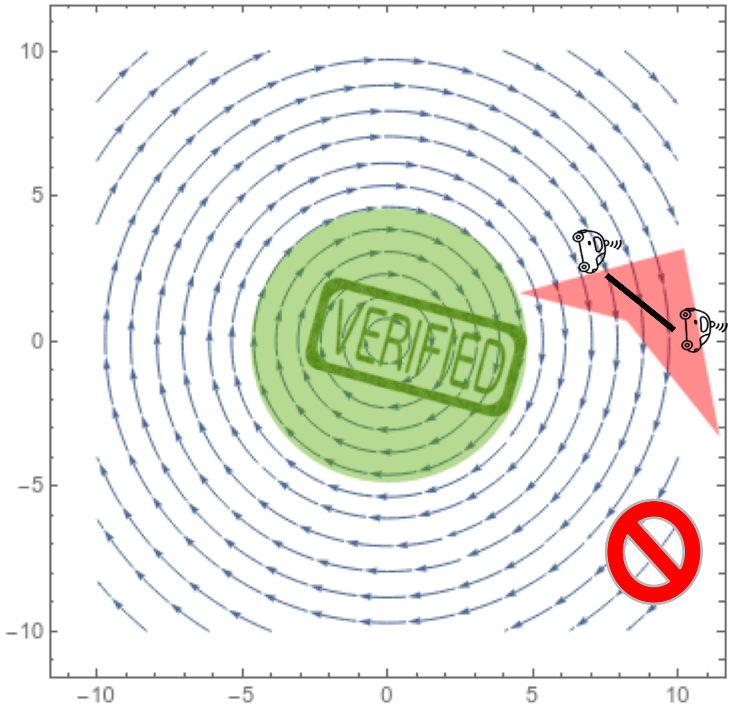


Leveraging Verification Results to Learn Better



{brake, accel, turn}

Observe &
compute
reward

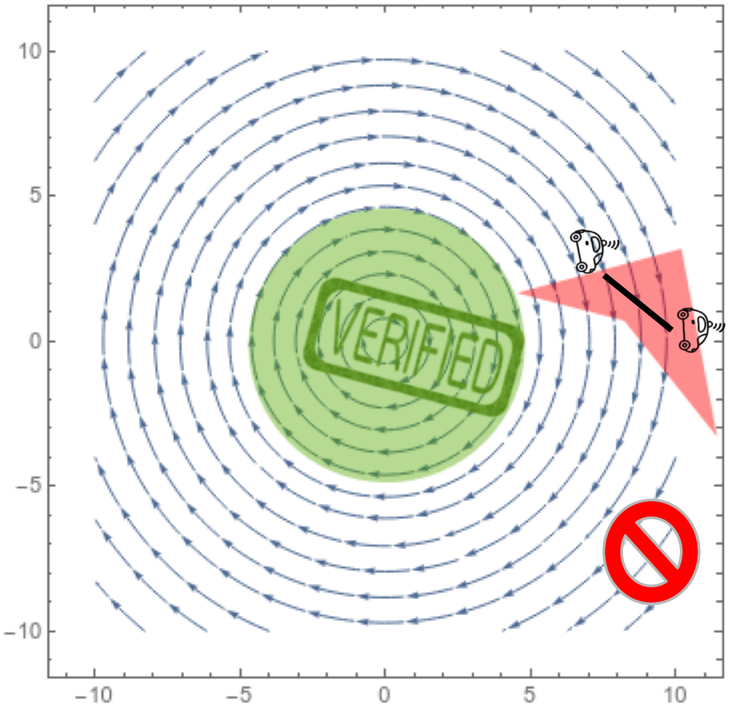


Use a real-valued version of the model monitor as a reward signal

Safe RL: How?

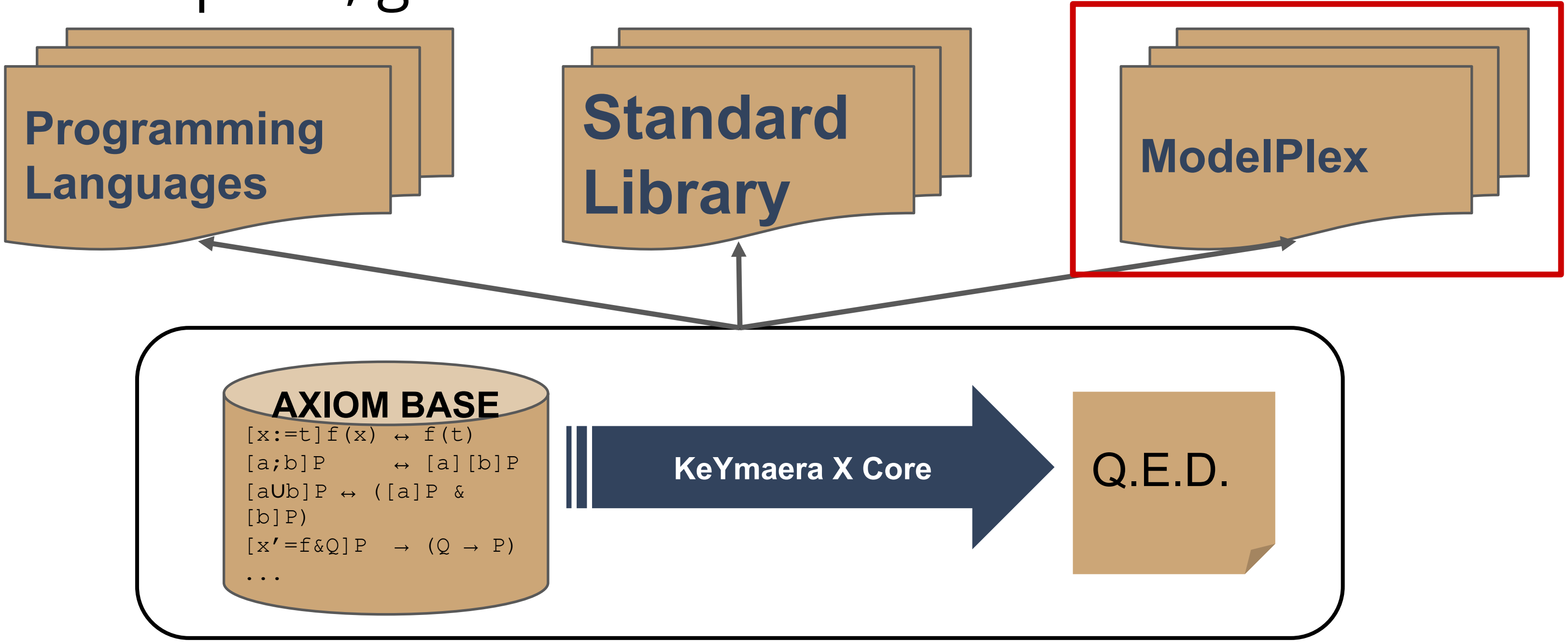
Details:

- ☐ 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

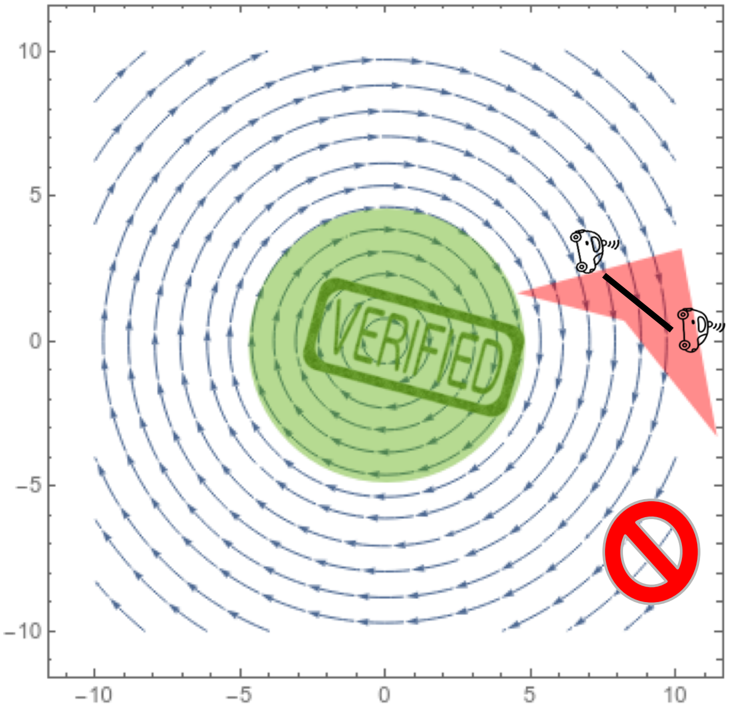
```
oldPos := read_sensor(GPS)
actuate(accel)
newPos := read_sensor(GPS)
if (QE(∃t. model_after(t) == newPos)):
        # No model deviation.
else:
        # Model deviation…?
```

Safe RL: How?

Details:

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

- ☐ Convert monitors into reward signals

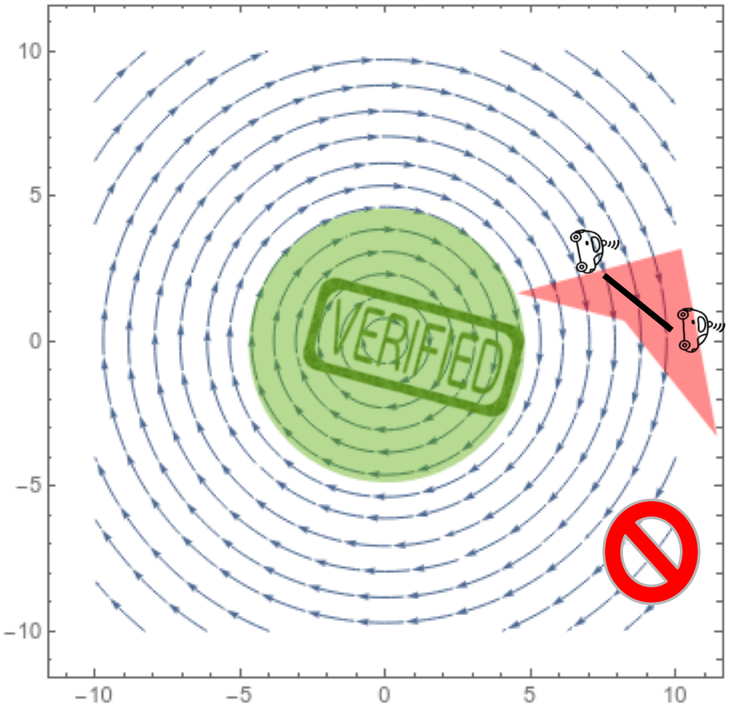


Safe RL: How?

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

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

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

An Example Monitor

init → [{

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

}*]safe

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

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

An Example Monitor

init → [{

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

}*]safe

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

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

An Example Monitor

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

init → [{

```
{?safeAccel;accel ∪ brake ∪ ?safeMaintain; maintainVel};

{pos' = vel, vel' = acc, t'=1}
```

}*]safe

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

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

Quantitative monitor as reward signal

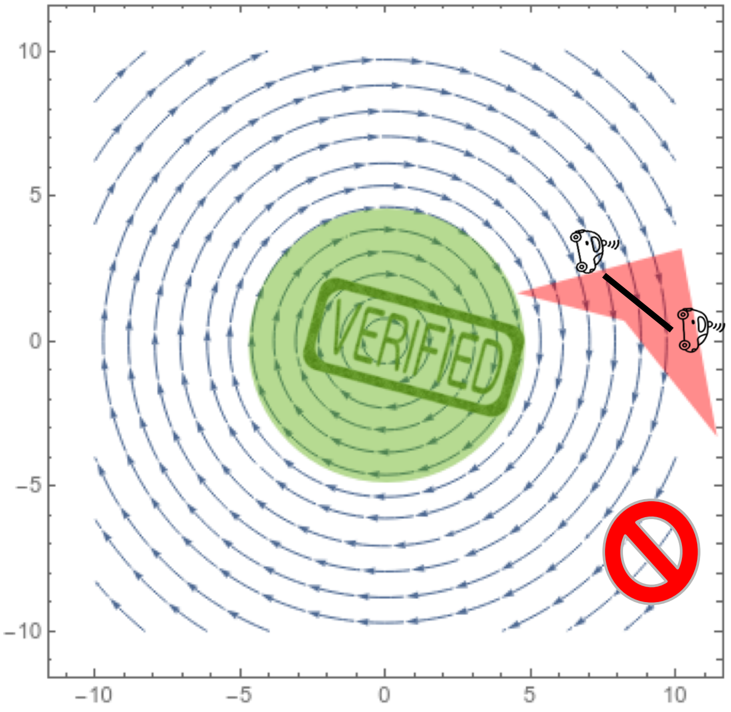
Safe RL: How?

Details:

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

Convert monitors into gradients:

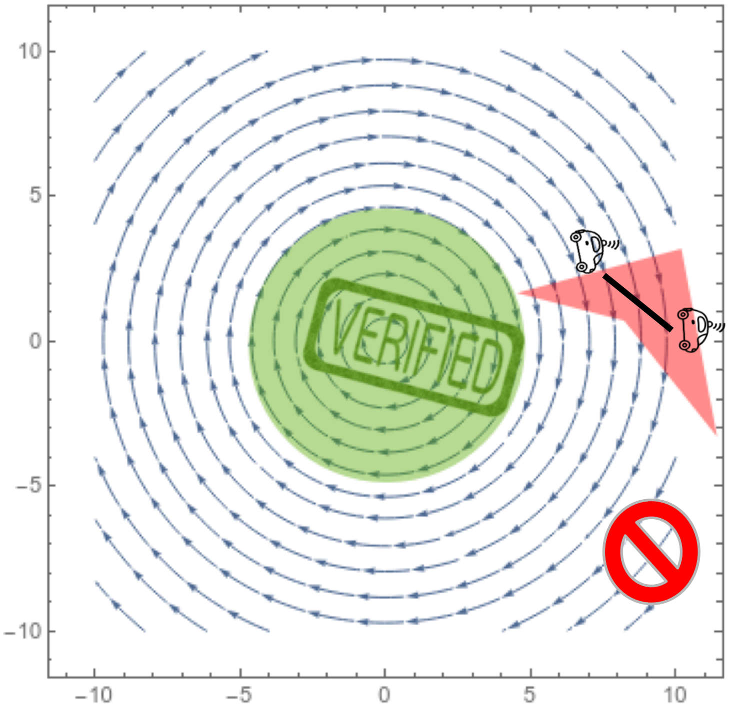
$(\mathbb{R}^n \rightarrow \mathbb{B}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R})$



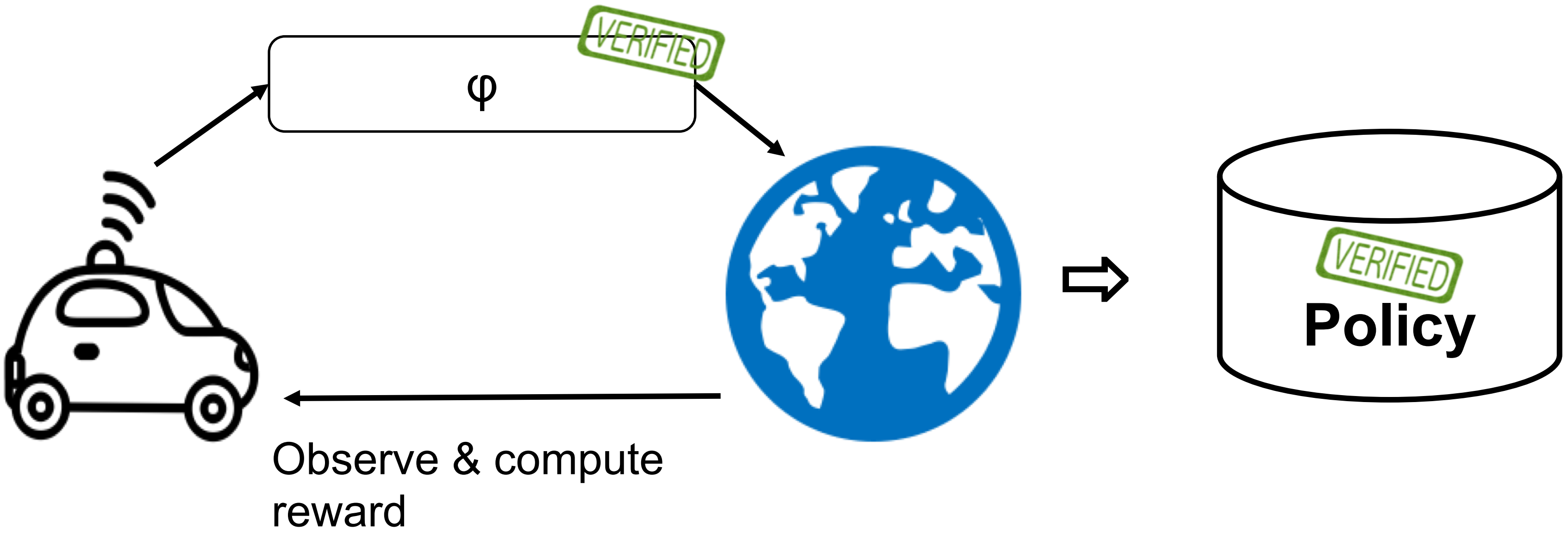
Safe RL: How?

Details:

- Runtime monitoring separates **modeled** from **unmodeled** state space.
- Convert **models** into gradients: ModelPlex $(\mathbb{R}^n \rightarrow \mathbb{B}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R})$



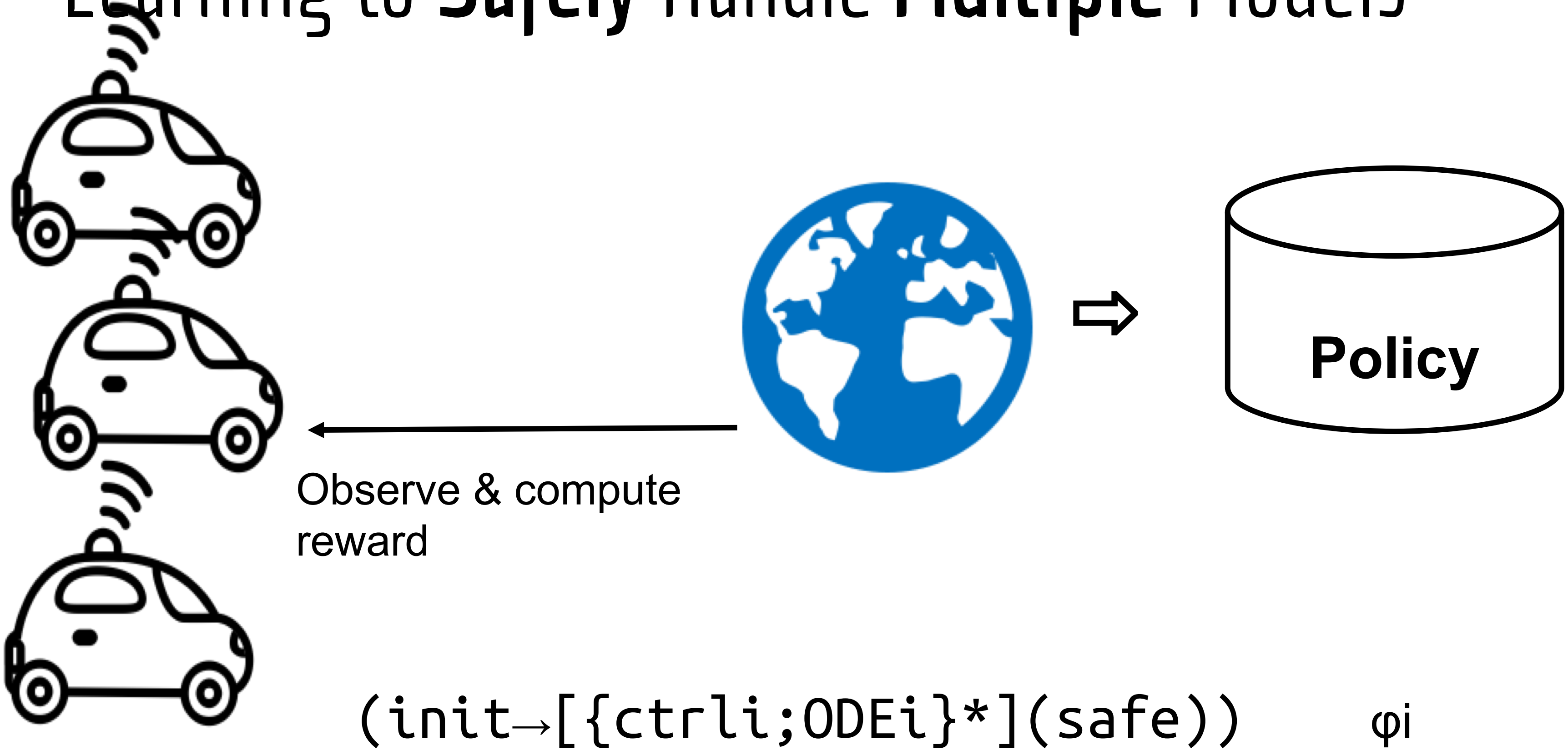
Learning to **Safely** Resolve Non-determinism



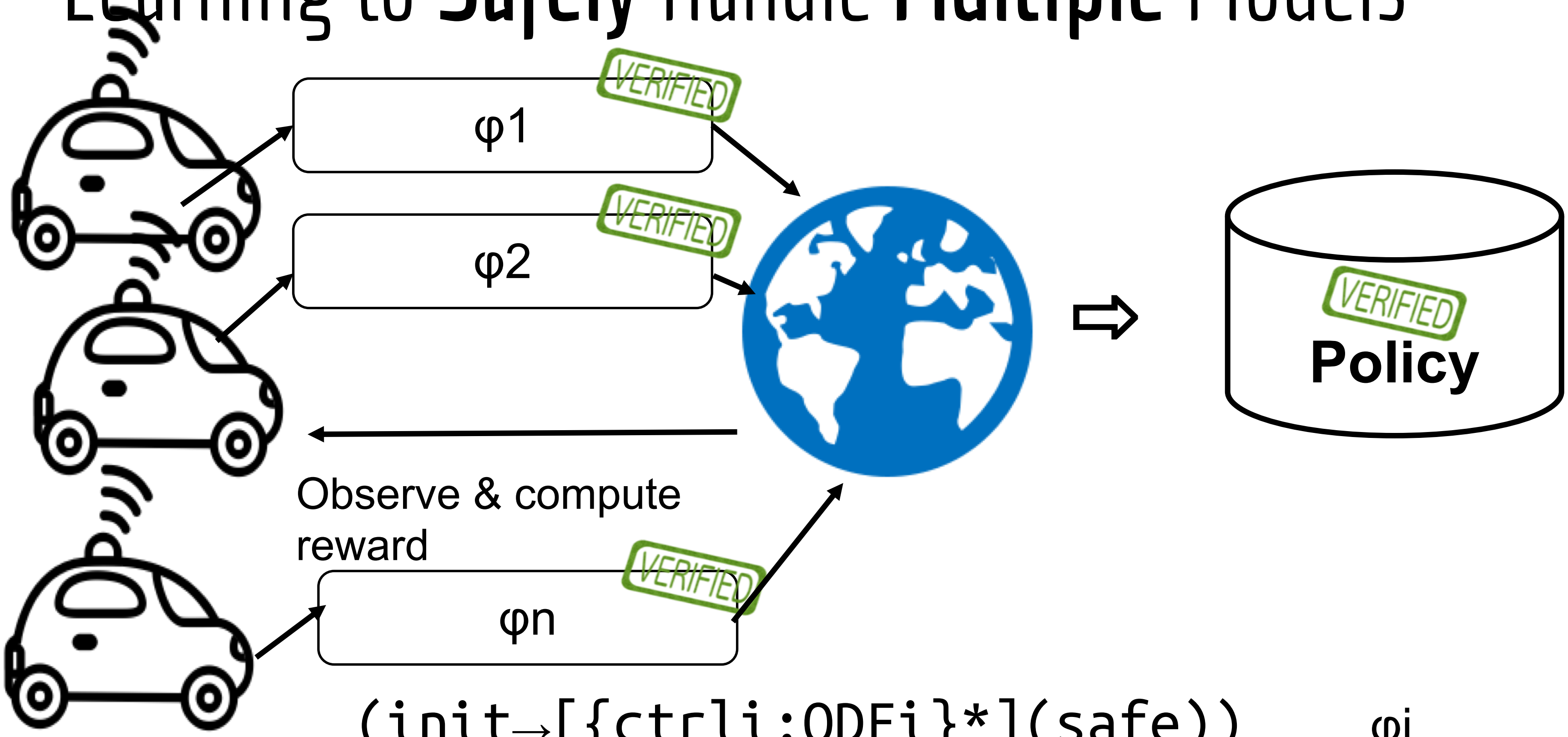
Use a theorem prover to extract:

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

Learning to **Safely** Handle **Multiple** Models

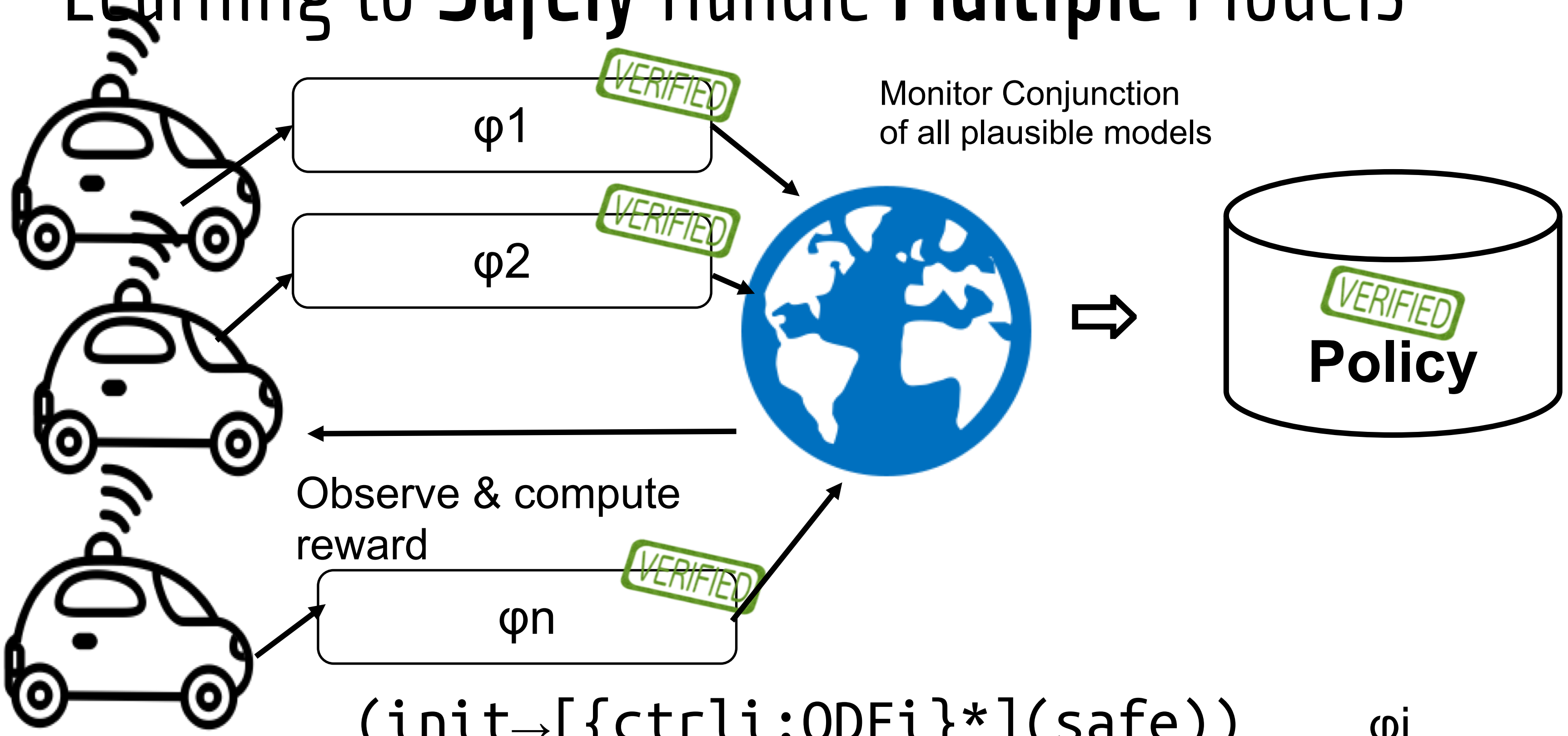


Learning to **Safely** Handle **Multiple** Models



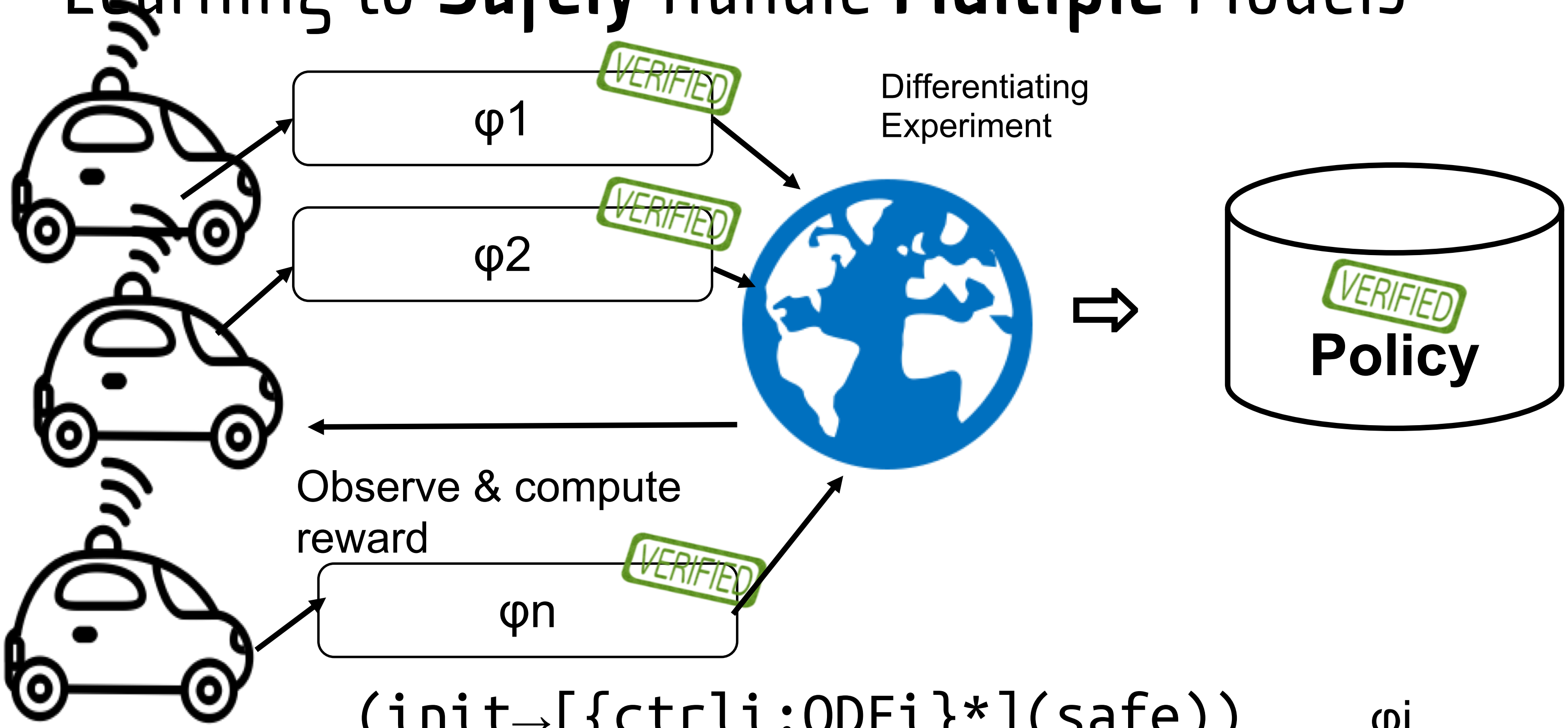
(init→[{ctrli;ODEi}*](safe)) φi

Learning to **Safely** Handle **Multiple** Models



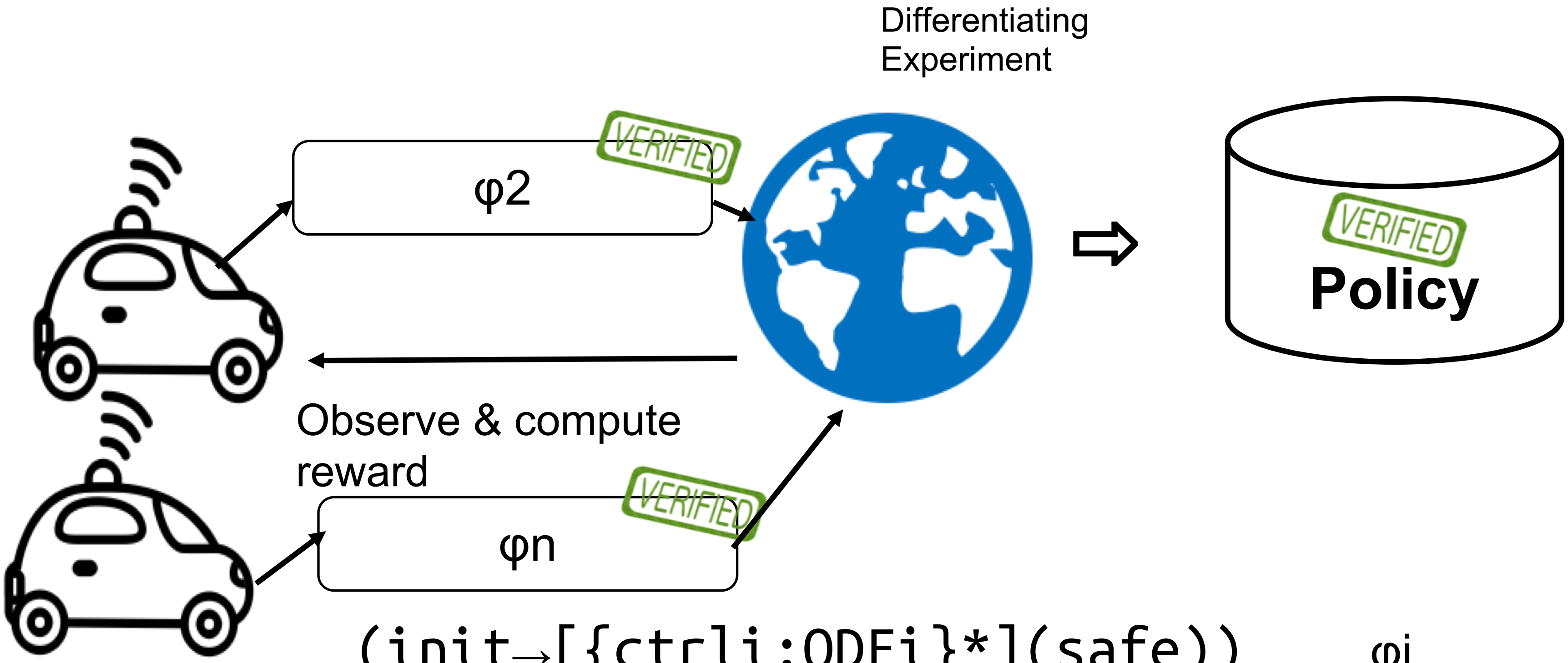
(init→[{ctrli;ODEi}*](safe))   φi

Learning to **Safely** Handle **Multiple** Models



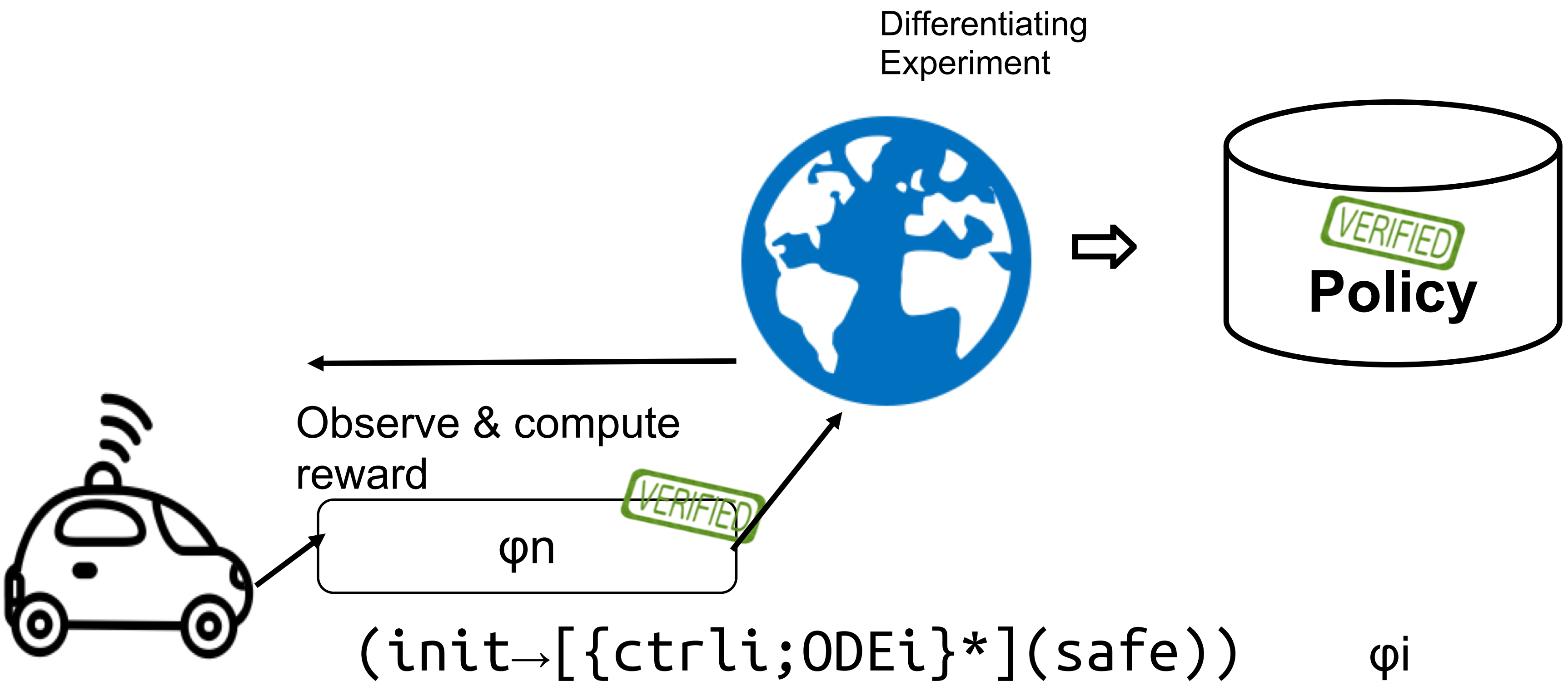
$(init \rightarrow [\{ctrli; ODEi\}^*](safe))$ φi

Learning to **Safely** Handle **Multiple** Models



(init→[{ctrli;ODEi}*](safe)) φi

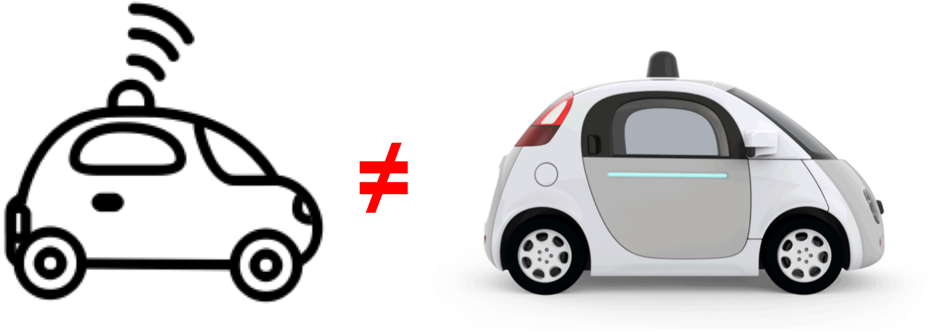
Learning to **Safely** Handle **Multiple** Models



Conclusion

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

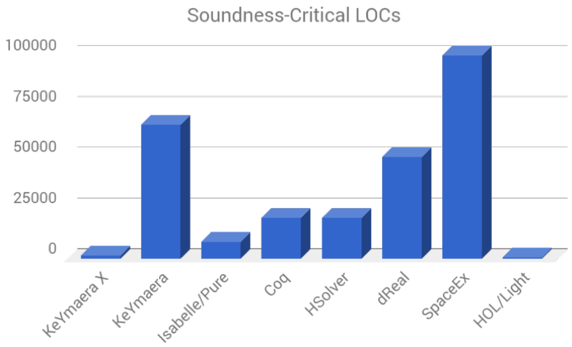
1. Was the proof correct?
2. Was the model accurate enough?



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?



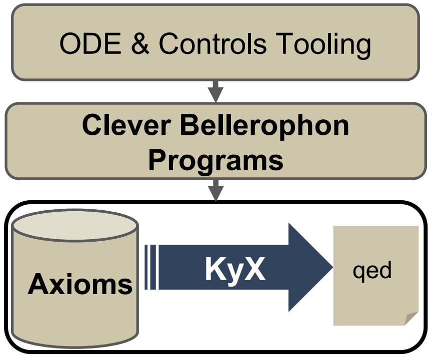
dI Tactic:

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

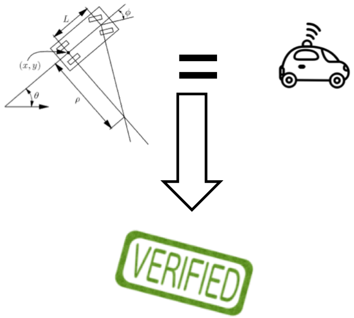
Example:
$[v' = r_p v^2 - g, t' = 1]\, v \geq v_0 - gt \quad \leftrightarrow$
$\ldots \quad \leftrightarrow$
$[v' := r_p v^2 - g][t' := 1]\, v' \geq -g*t' \quad \leftrightarrow$
$r_p v^2 - g \geq -g \quad \leftrightarrow$
$H \rightarrow r_p \geq 0$

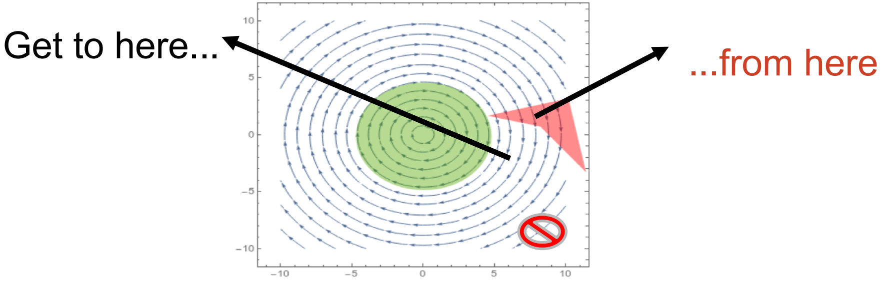


Conclusion

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**
3. With multiple possible models? **μ-learning**
4. When off-model? **Verification-preserving model update**

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

I Part: Elementary Cyber-Physical Systems

1. Differential Equations & Domains
2. Choice & Control
3. Safety & Contracts
4. Dynamical Systems & Dynamic Axioms
5. Truth & Proof
6. Control Loops & Invariants
7. Events & Responses
8. Reactions & Delays

II Part: Differential Equations Analysis

9. Differential Equations & Differential Invariants
10. Differential Equations & Proofs
11. Ghosts & Differential Ghosts
12. Differential Invariants & Proof Theory

III Part: Adversarial Cyber-Physical Systems

13-16. Hybrid Systems & Hybrid Games

IV Part: Comprehensive CPS Correctness