Safe Reinforcement Learning via Formal Methods

Nathan Fulton and André Platzer

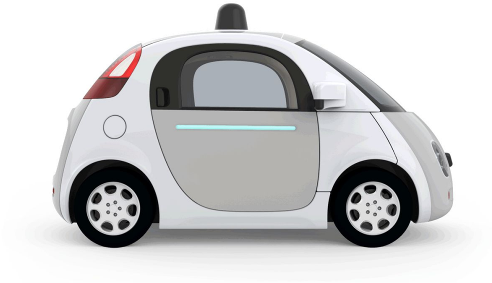
Carnegie Mellon University

Safe Reinforcement Learning via Formal Methods

Nathan Fulton and André Platzer

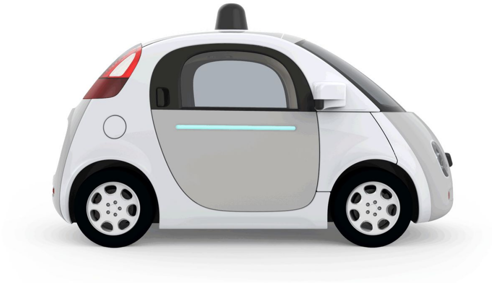
Carnegie Mellon University

Safety-Critical Systems



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

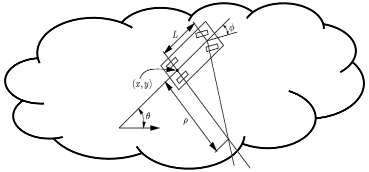
Autonomous Safety-Critical Systems



How can we provide people with **autonomous** cyber-physical systems they can bet their lives on?

Model-Based Verification

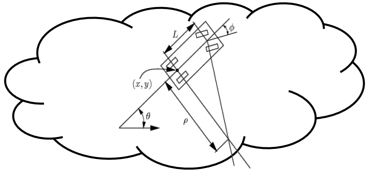
Reinforcement Learning



φ

Model-Based Verification

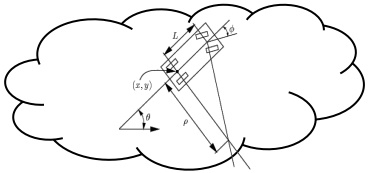
Reinforcement Learning



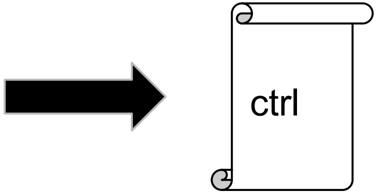
pos < stopSign

Model-Based Verification

Reinforcement Learning

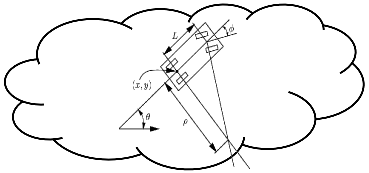


pos < stopSign

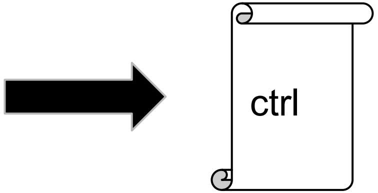


Model-Based Verification

Reinforcement Learning



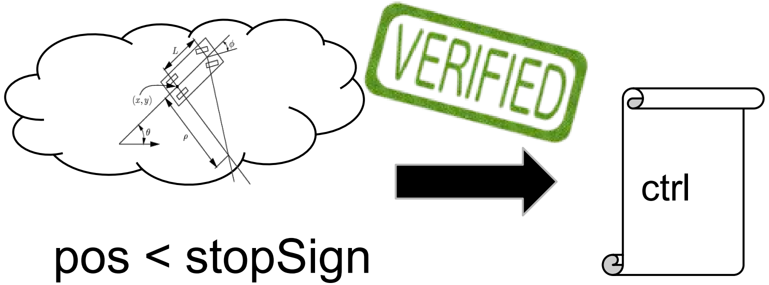
pos < stopSign



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

Model-Based Verification

Reinforcement Learning



pos < stopSign

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

Reinforcement Learning

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

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

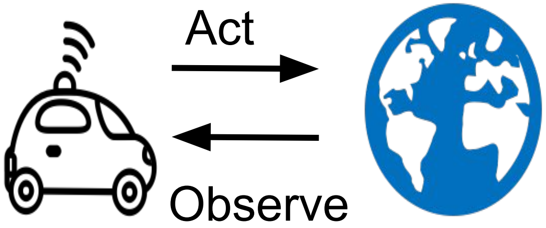
Benefits:

- 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

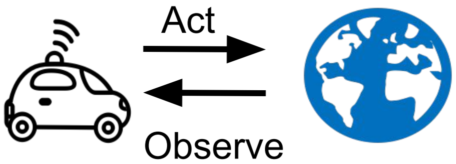
Benefits:

- 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

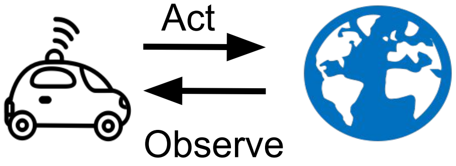
Benefits:

- 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:

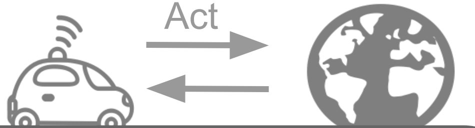
- No need for complete model
- Optimal (effective) policies

Drawbacks:

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

Model-Based Verification

Reinforcement Learning



Bene

Goal: Provably correct reinforcement learning

- S
- A

del
s

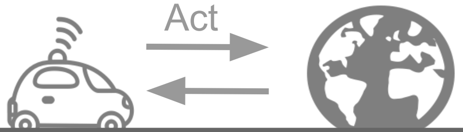
Draw

- 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



Goal: Provably correct reinforcement learning

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

Bene

- 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

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}
}*
```

discrete control

Continuous motion

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!

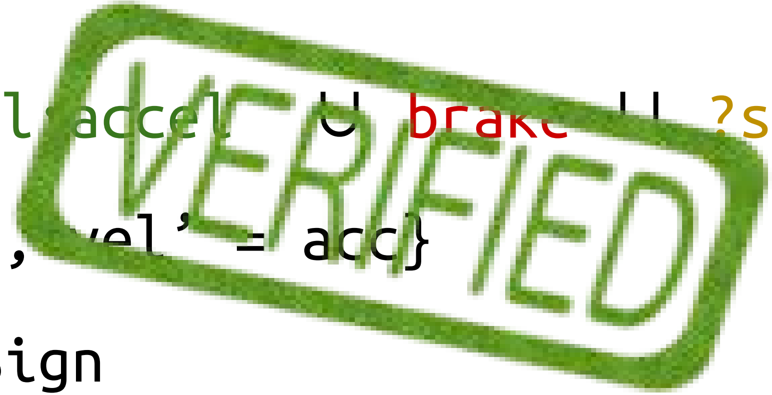
```
init → [{
    { ?safeAccel;accel  ∪ brake ∪ ?safeTurn; turn};
    {pos’ = vel, vel’ = acc}
}*]pos < stopSign
```

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 ∪ 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

Our Contribution

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

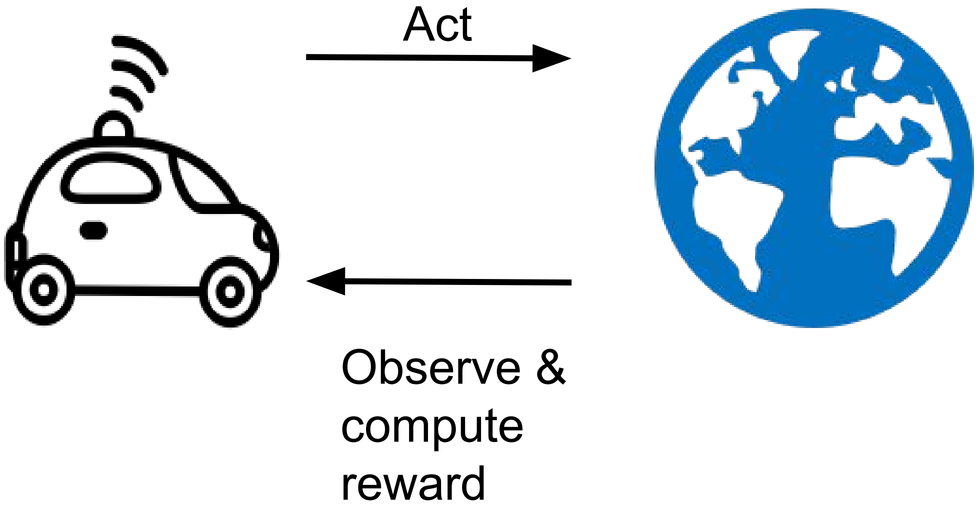
1. learns to resolve non-determinism without sacrificing formal safety results

Our Contribution

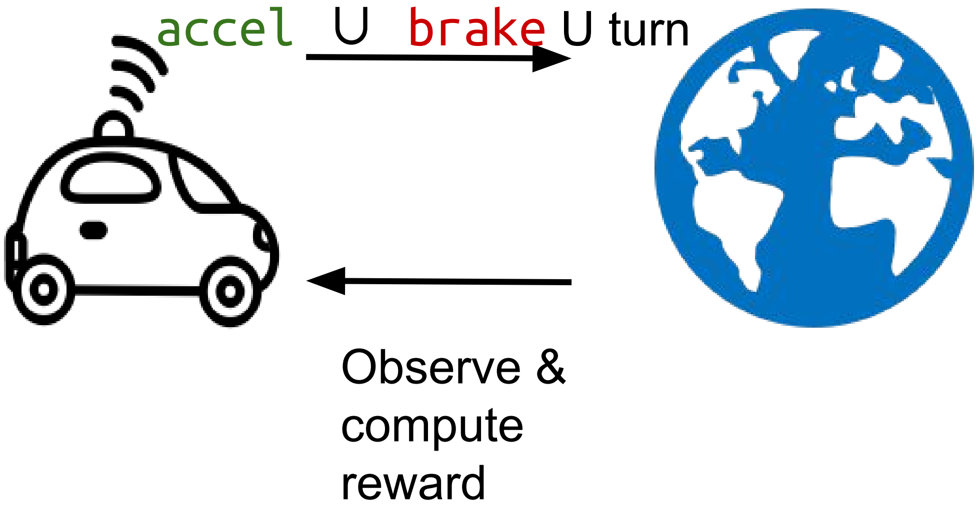
Justified Speculative Control is an approach toward provably safe reinforcement learning that:

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

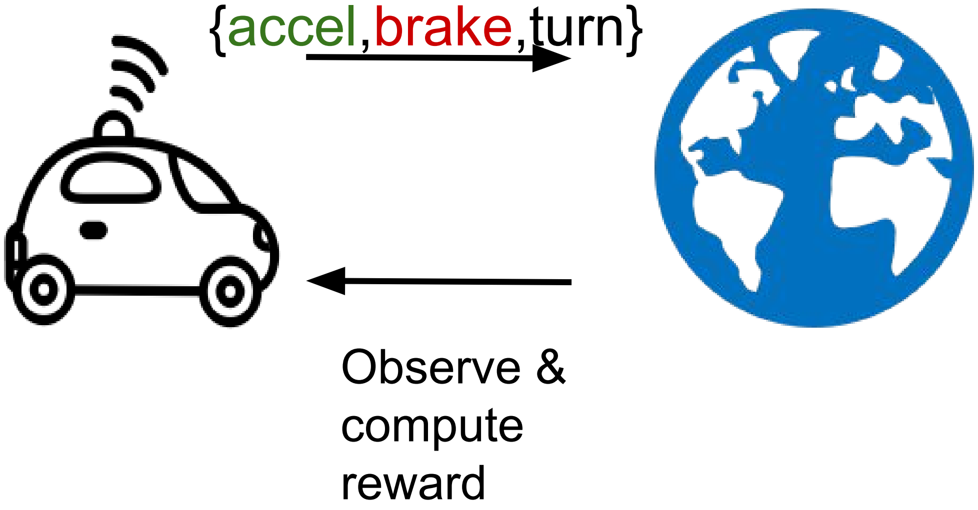
Learning to Resolve Non-determinism



Learning to Resolve Non-determinism



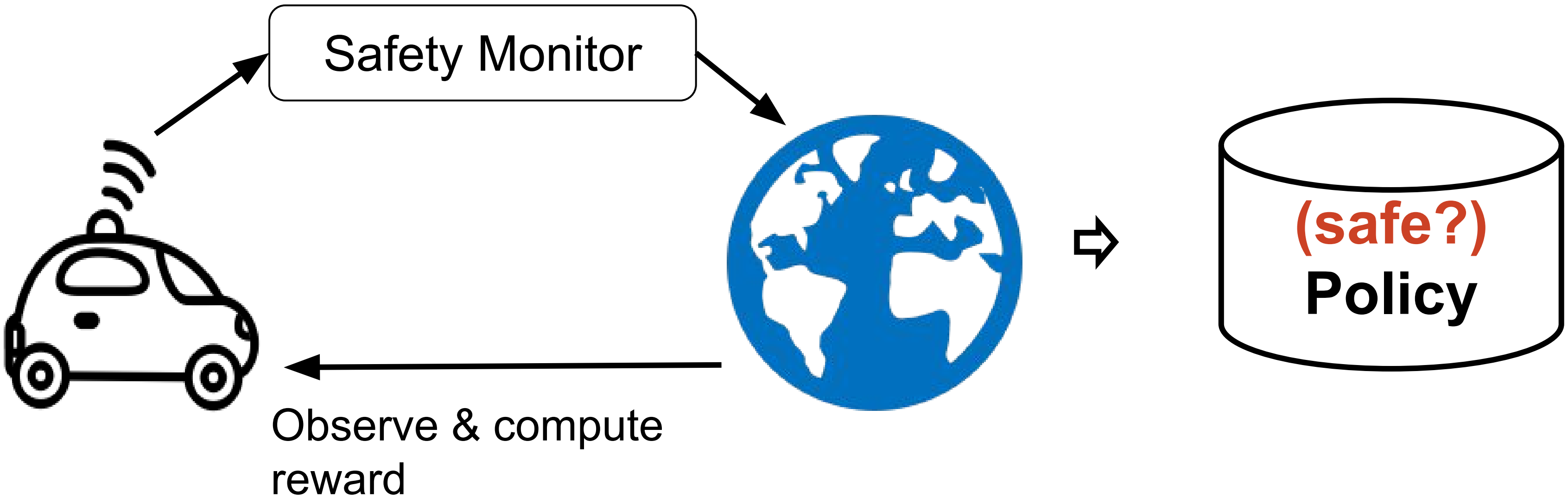
Learning to Resolve Non-determinism



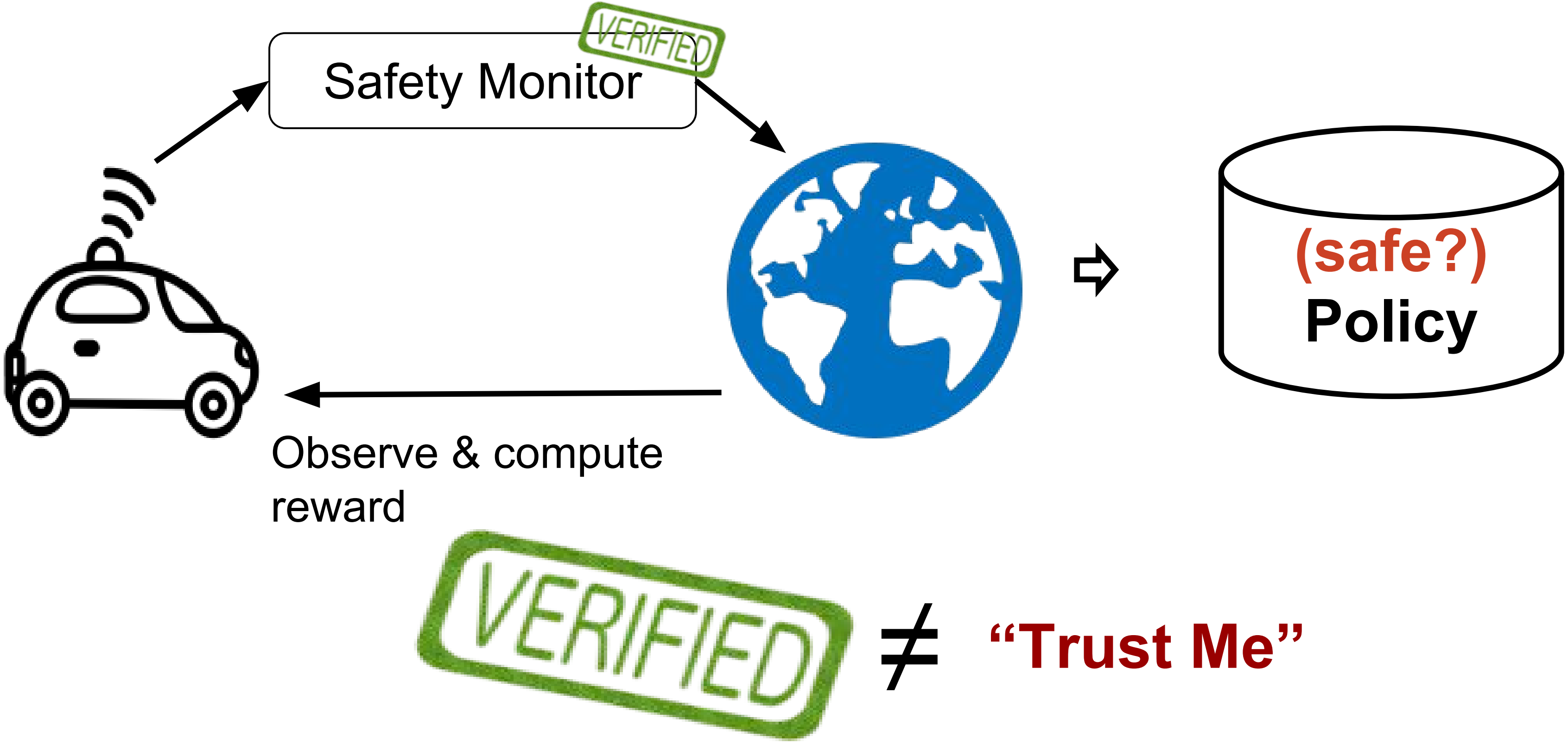
Learning to Resolve Non-determinism

Learning to Resolve Non-determinism

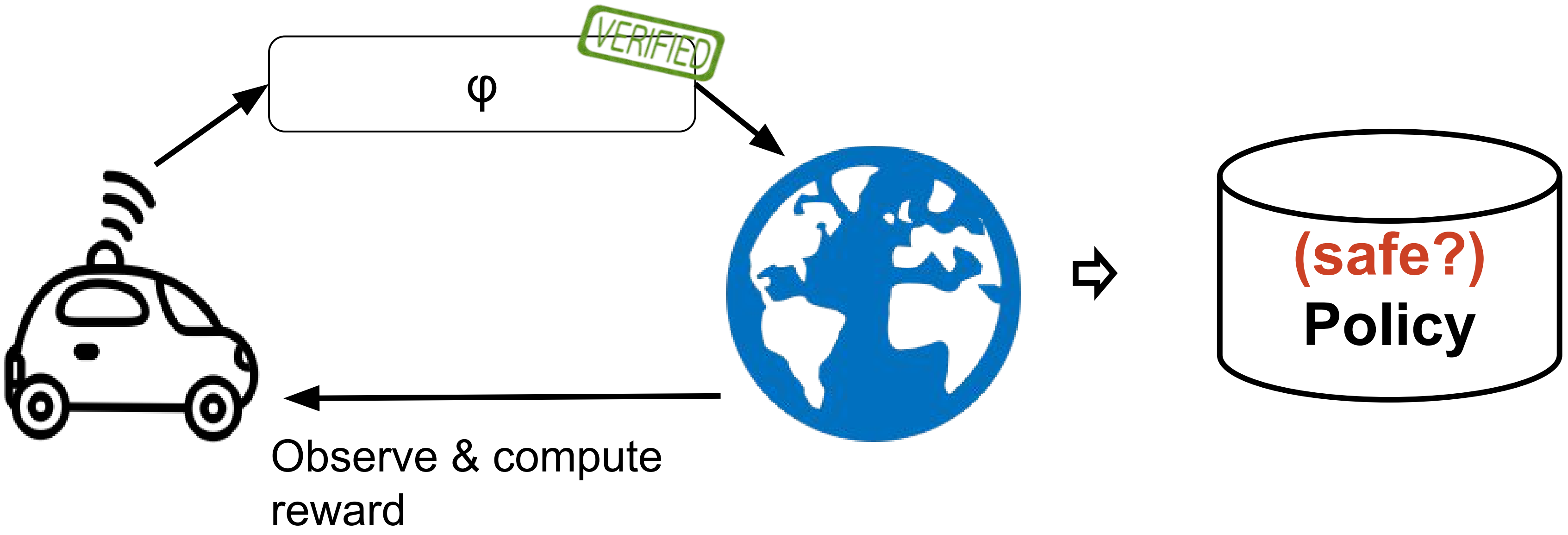
Learning to **Safely** Resolve Non-determinism



Learning to **Safely** Resolve Non-determinism



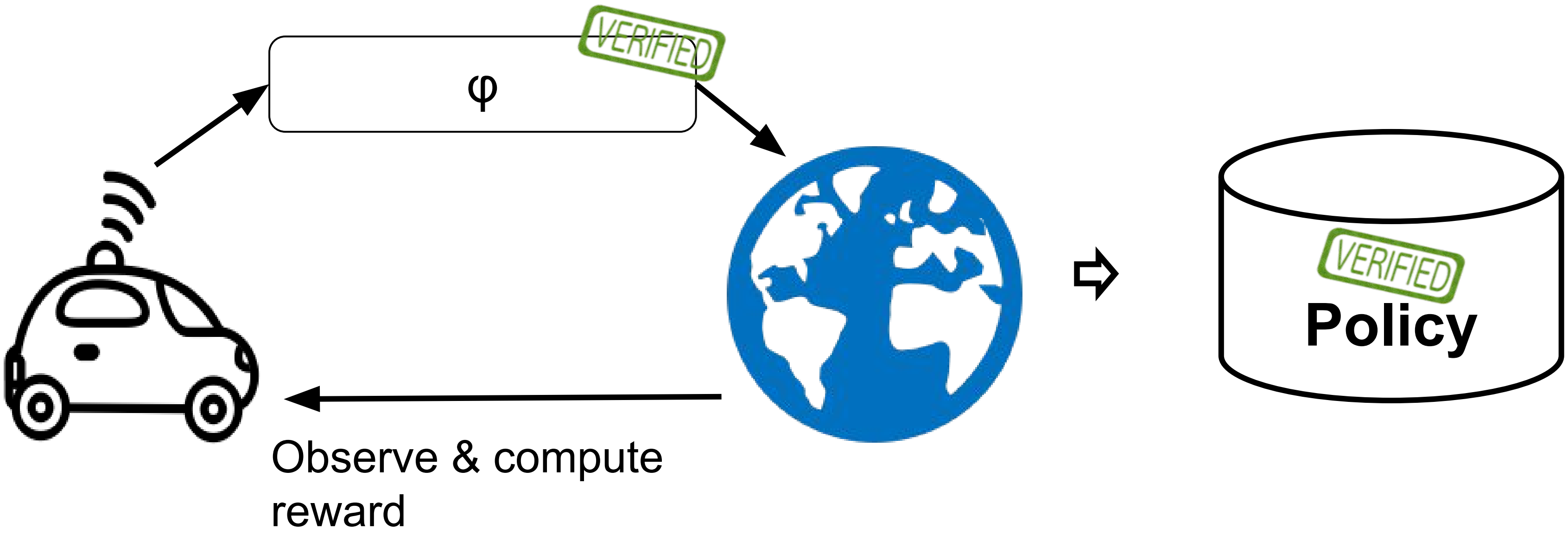
Learning to **Safely** Resolve Non-determinism



Use a theorem prover to prove:

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

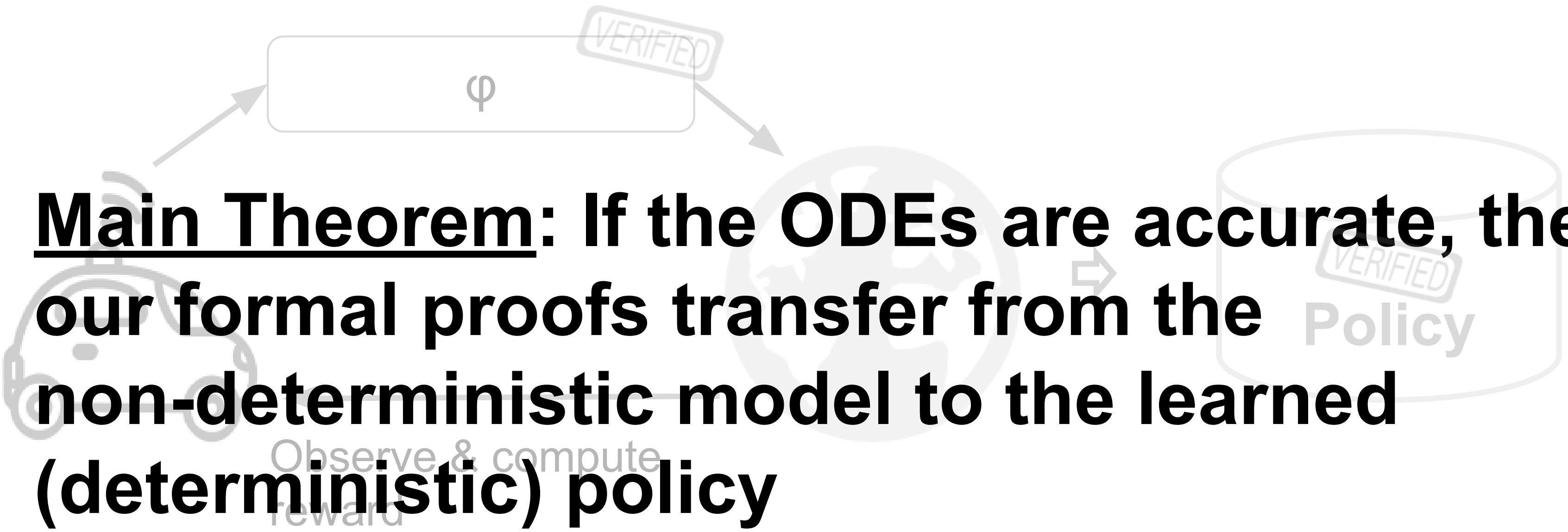
Learning to **Safely** Resolve Non-determinism



Use a theorem prover to prove:

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

Learning to **Safely** Resolve Non-determinism

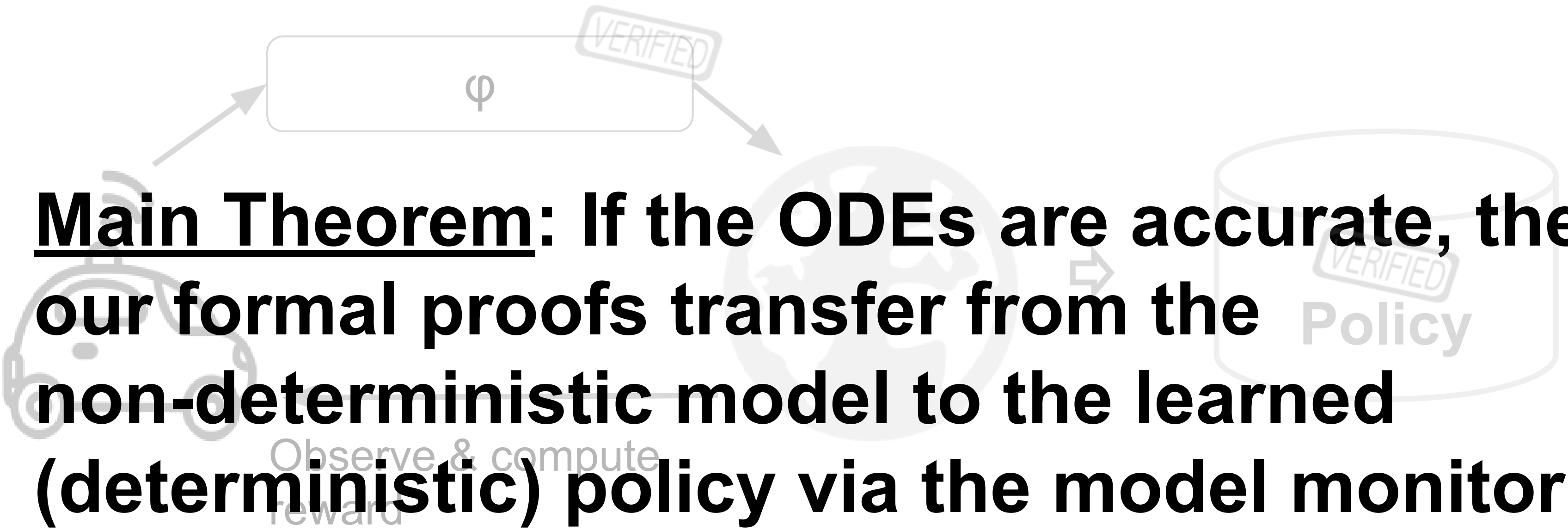


<u>Main Theorem</u>: If the ODEs are accurate, then our formal proofs transfer from the non-deterministic model to the learned (deterministic) policy

Use a theorem prover to prove:

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

Learning to **Safely** Resolve Non-determinism

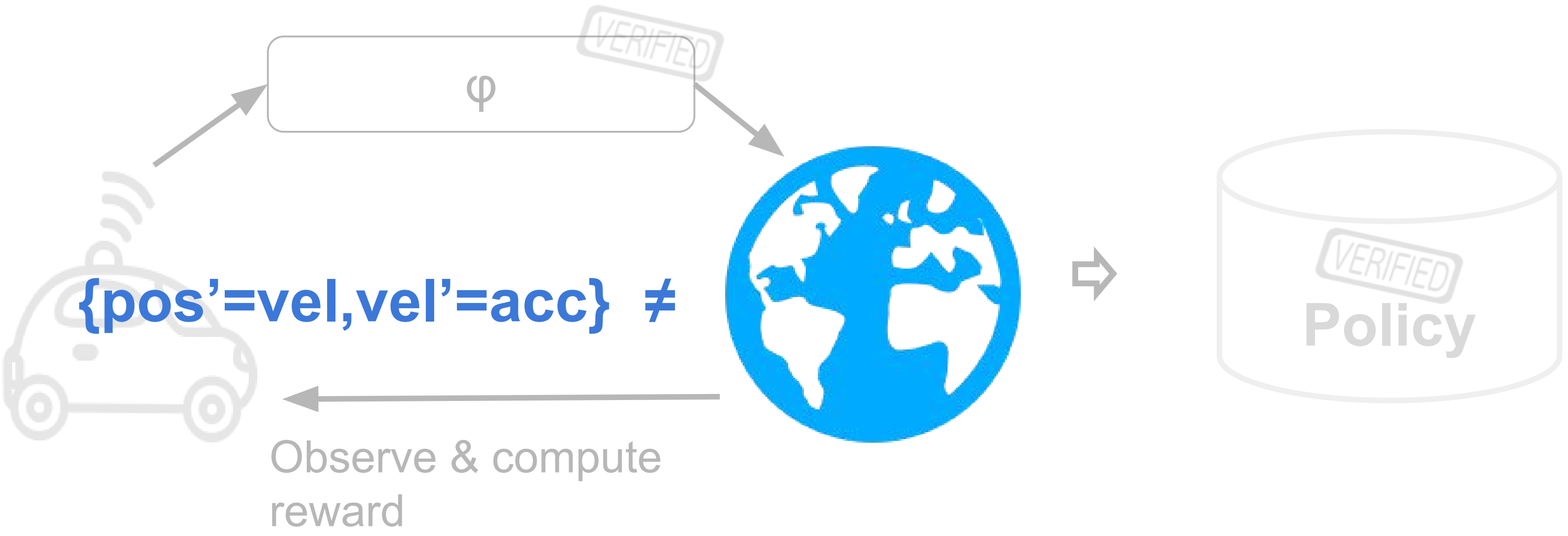


<u>Main Theorem</u>: If the ODEs are accurate, then our formal proofs transfer from the non-deterministic model to the learned (deterministic) policy via the model monitor.

Use a theorem prover to prove:

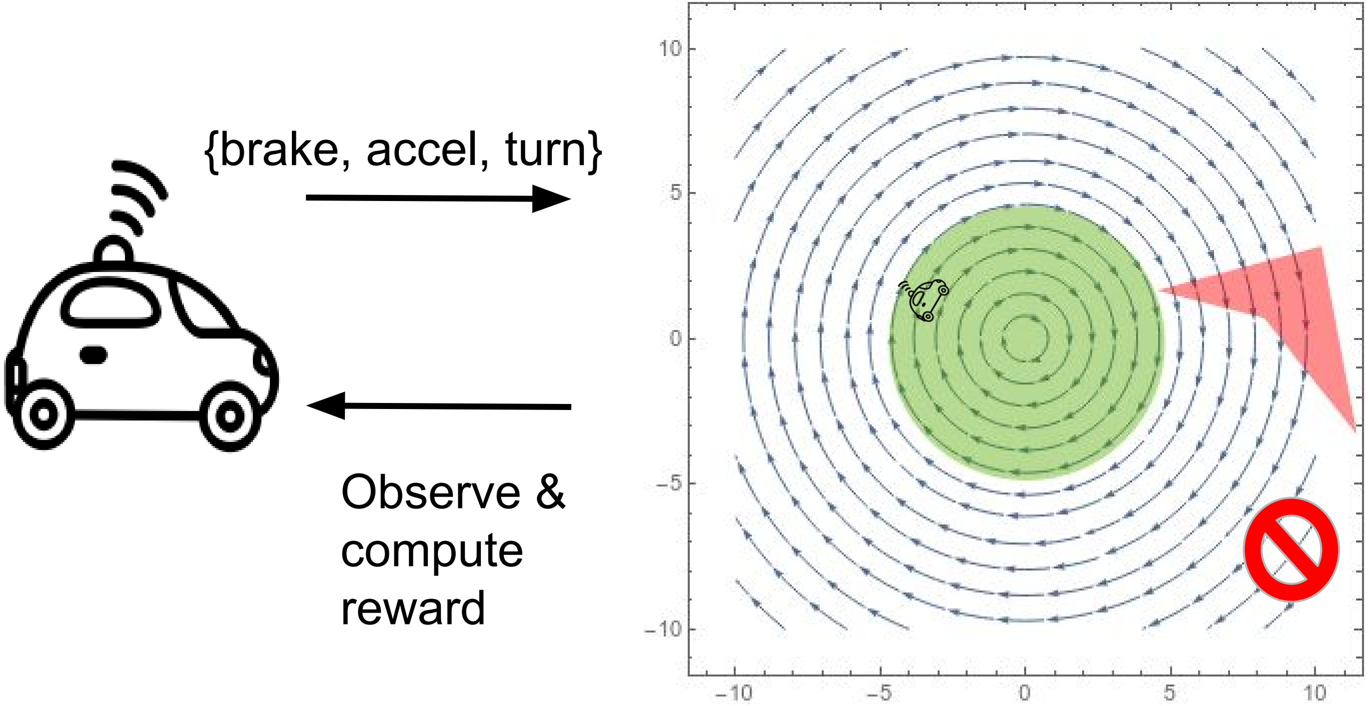
$$(init \rightarrow [\{\{accel \cup brake\};ODEs\}^*](safe)) \leftrightarrow \varphi$$

What about the physical model?

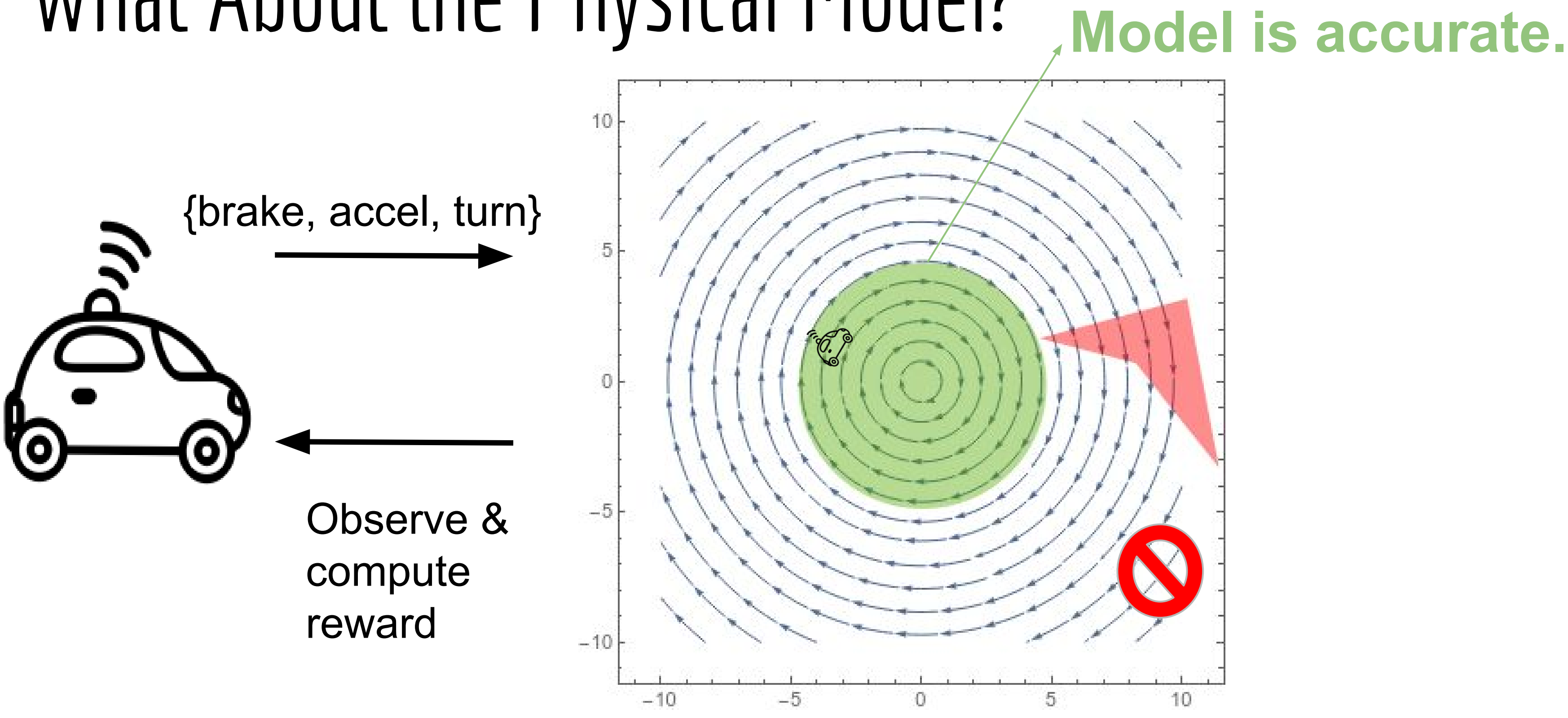


Use a theorem prover to prove: (init→ [{{accel∪brake};ODEs}*](safe)) ↔ φ

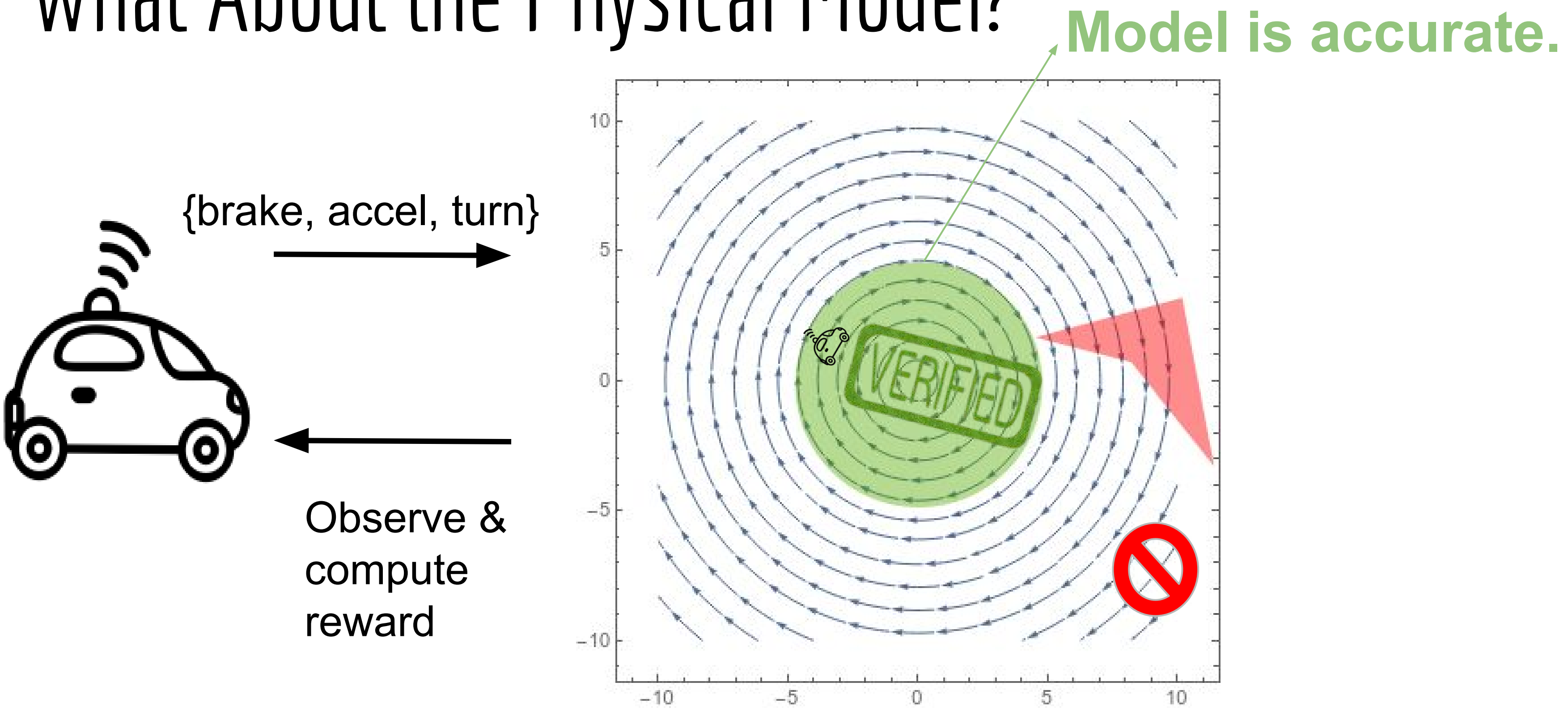
What About the Physical Model?



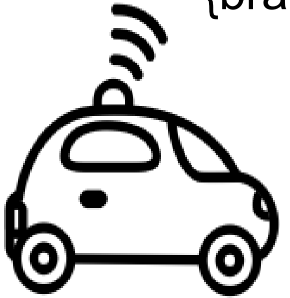
What About the Physical Model?



What About the Physical Model?

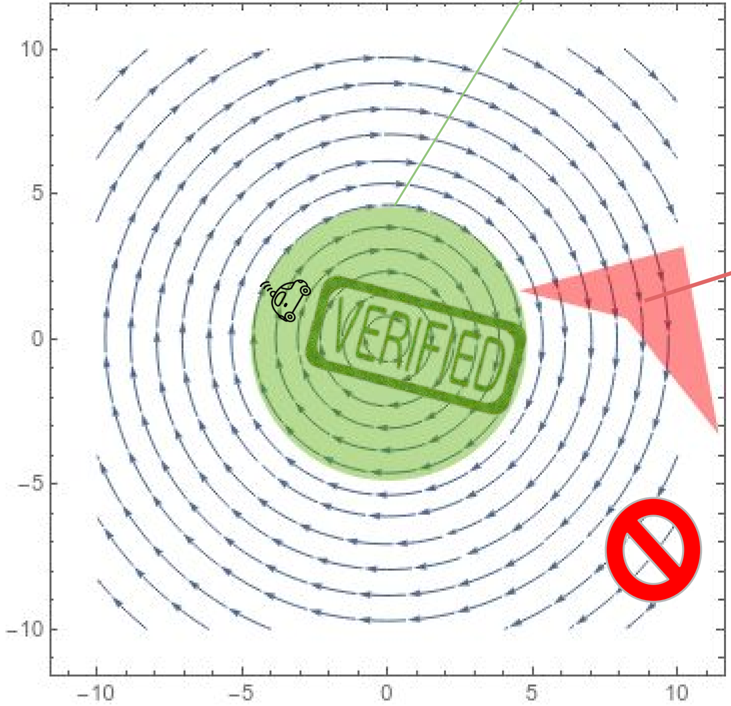


What About the Physical Model?



{brake, accel, turn}

Observe & compute reward

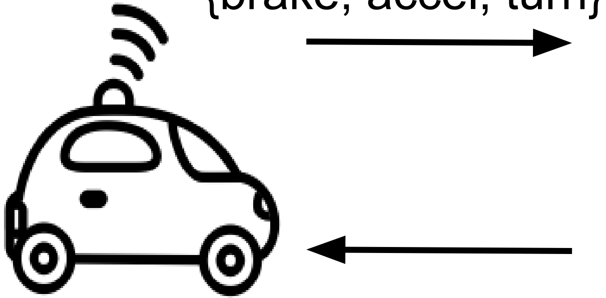


Model is accurate.

Model is inaccurate

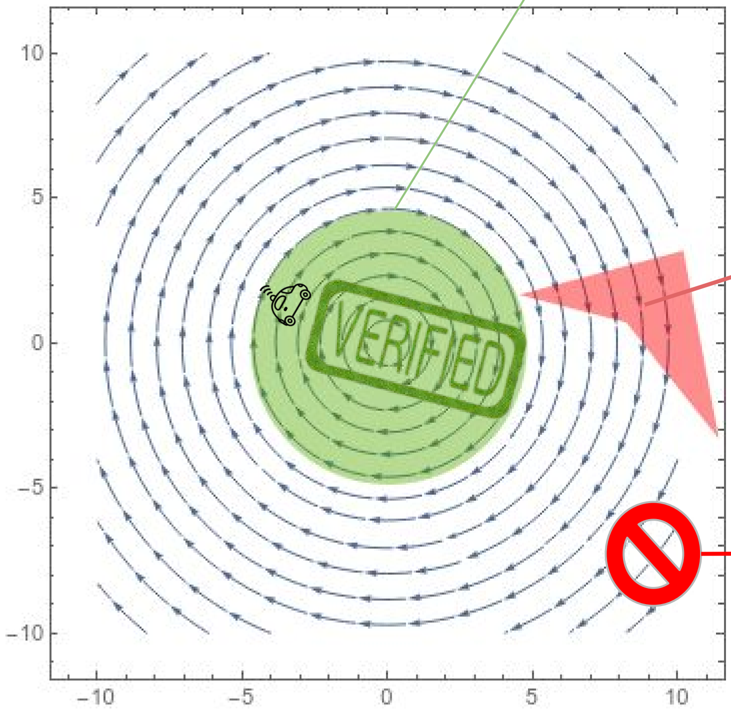
What About the Physical Model?

{brake, accel, turn}



Observe & compute reward

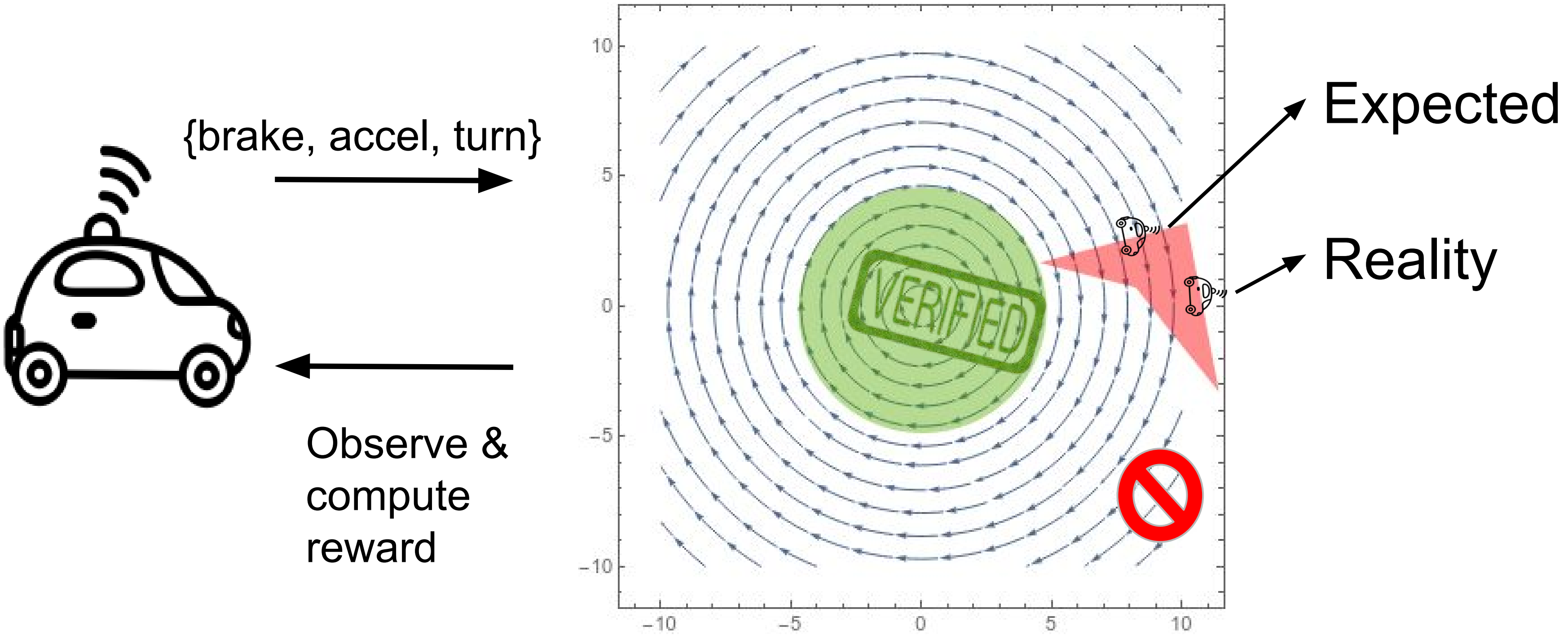
Model is accurate.



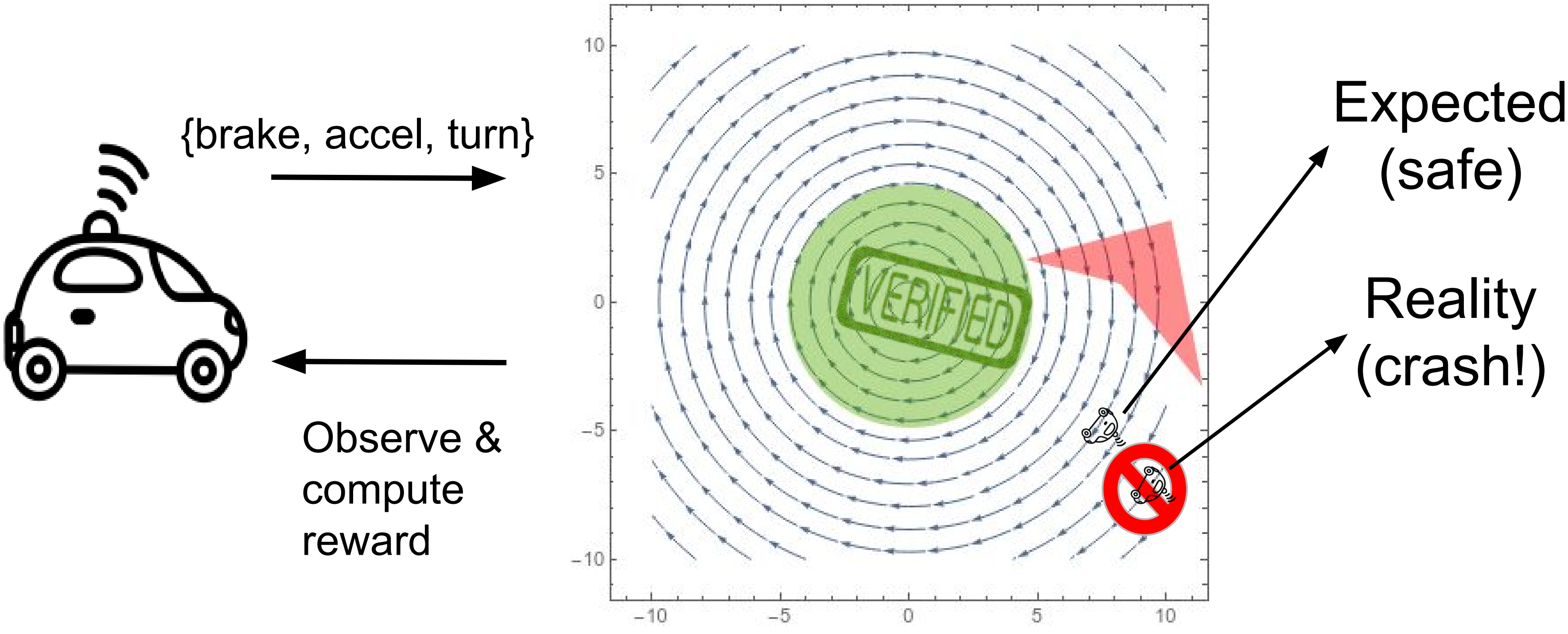
Model is inaccurate

Obstacle!

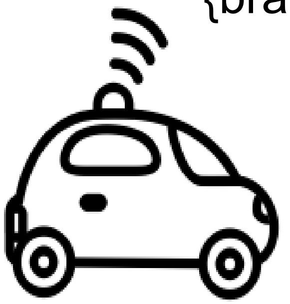
What About the Physical Model?



Speculation is Justified

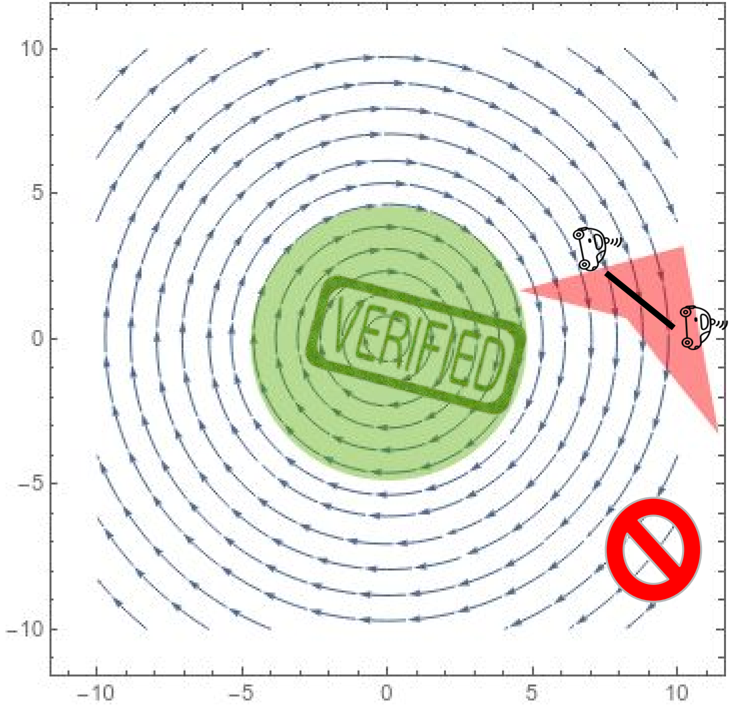


Leveraging Verification Results to Learn Better



{brake, accel, turn}

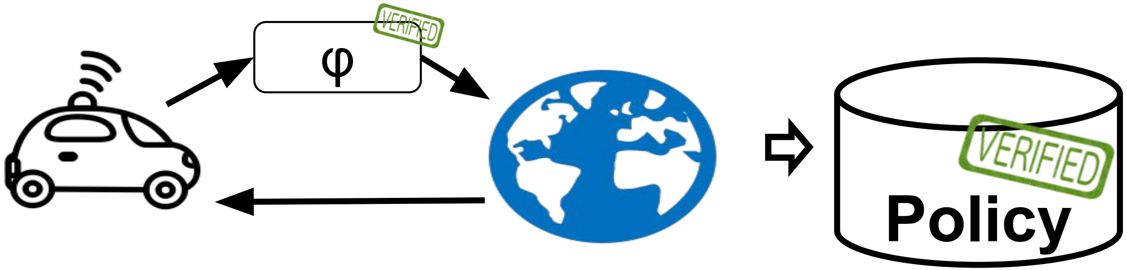
Observe & compute reward



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

Conclusion

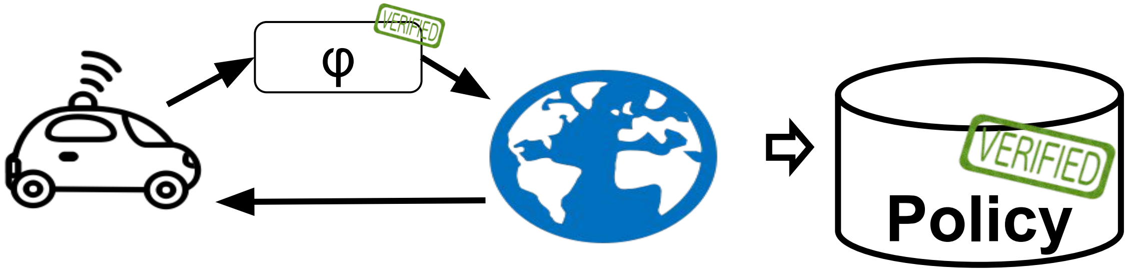
Justified Speculative Control provides the best of logic and learning:



Conclusion

Justified Speculative Control provides the best of logic and learning:

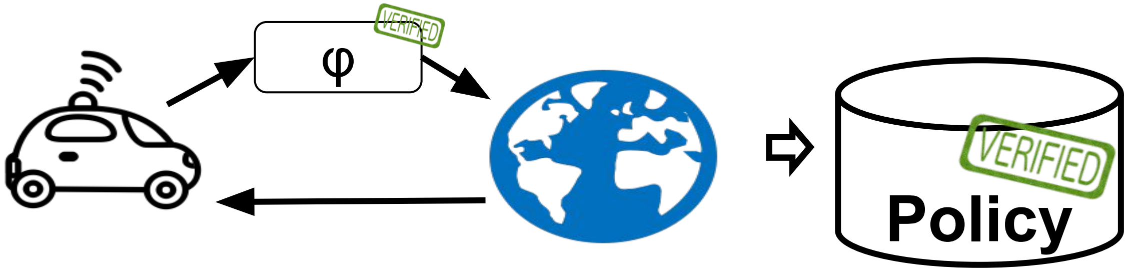
- Formally model the control system (**control + physics**)



Conclusion

Justified Speculative Control provides the best of logic and learning:

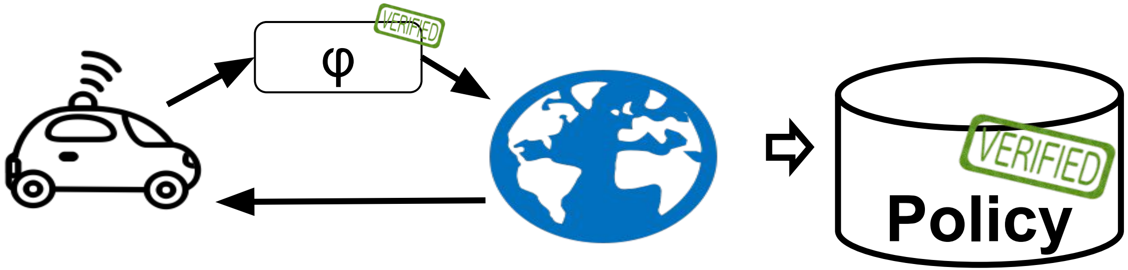
- Formally model the control system (**control + physics**)
- Learn how to resolve non-determinism in models.



Conclusion

Justified Speculative Control provides the best of logic and learning:

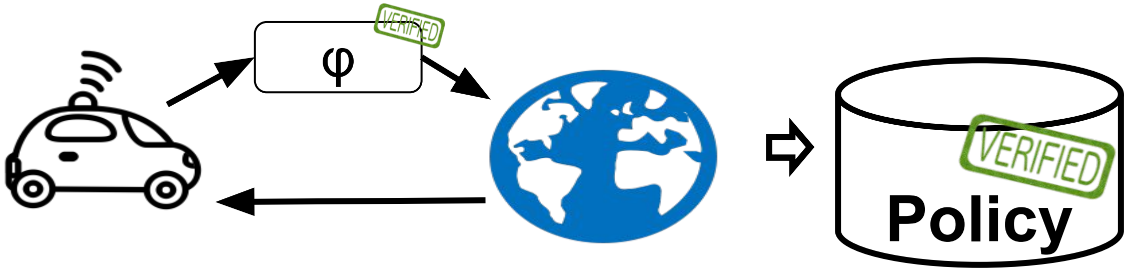
- Formally model the control system (**control + physics**)
- Learn how to resolve non-determinism in models.
- Leverage theorem proving to transfer **proofs** to learned policies.



Conclusion

Justified Speculative Control provides the best of logic and learning:

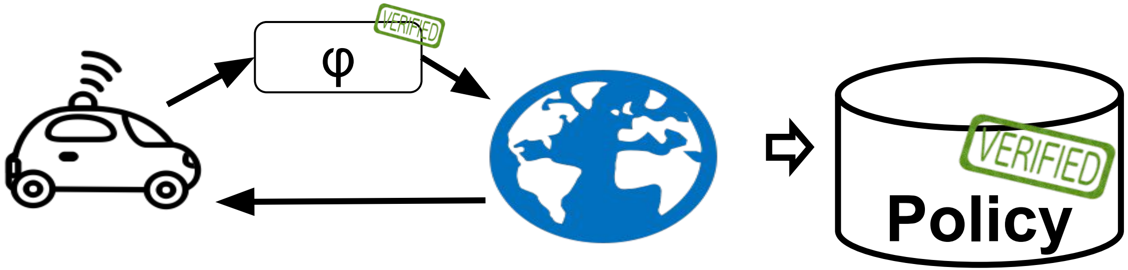
- Formally model the control system (**control + physics**)
- Learn how to resolve non-determinism in models.
- Leverage theorem proving to transfer **proofs** to learned policies.
- Unsafe **speculation is justified** when model deviates from reality



Conclusion

Justified Speculative Control provides the best of logic and learning:

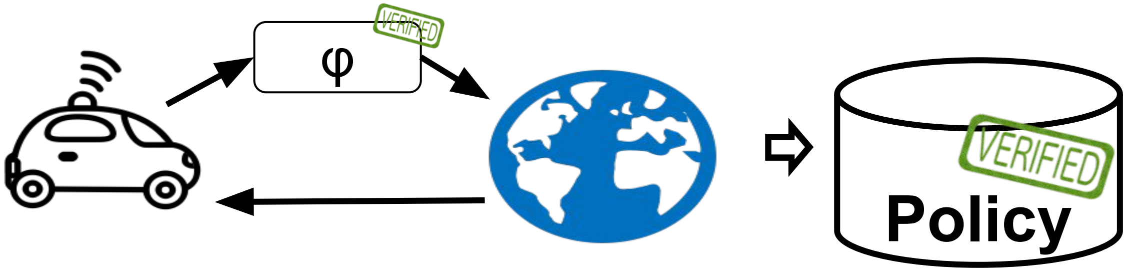
- Formally model the control system (**control + physics**)
- Learn how to resolve non-determinism in models
- Leverage theorem proving to transfer **proofs** to learned policies
- Unsafe **speculation is justified** when model deviates from reality, but **verification results can still be helpful**!



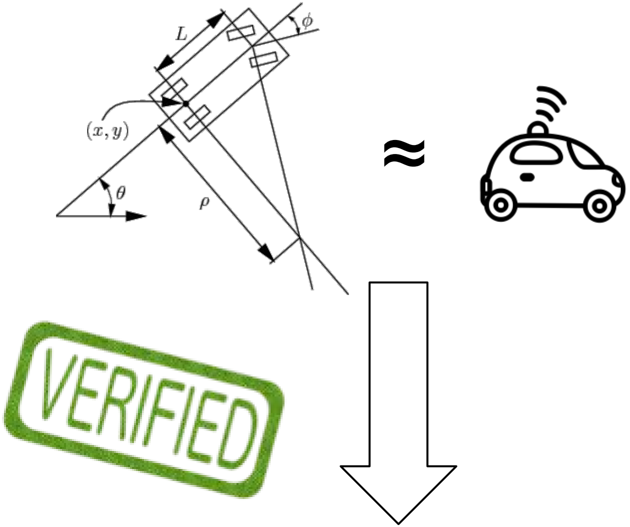
Conclusion

Justified Speculative Control provides the best of logic and learning:

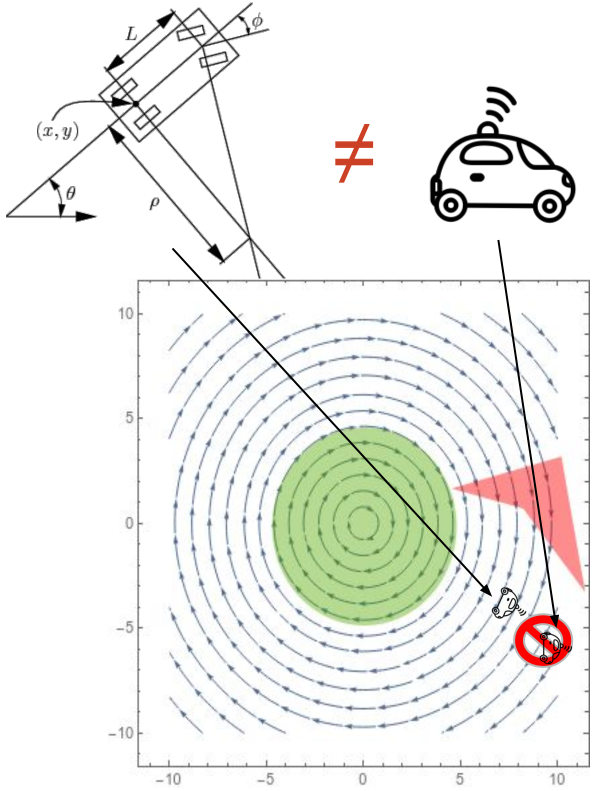
- Formally model the control system (**control + physics**)
- Learn how to resolve non-determinism in models
- Leverage theorem proving to transfer **proofs** to learned policies
- Unsafe **speculation is justified** when model deviates from reality, but **verification results can still be helpful**!



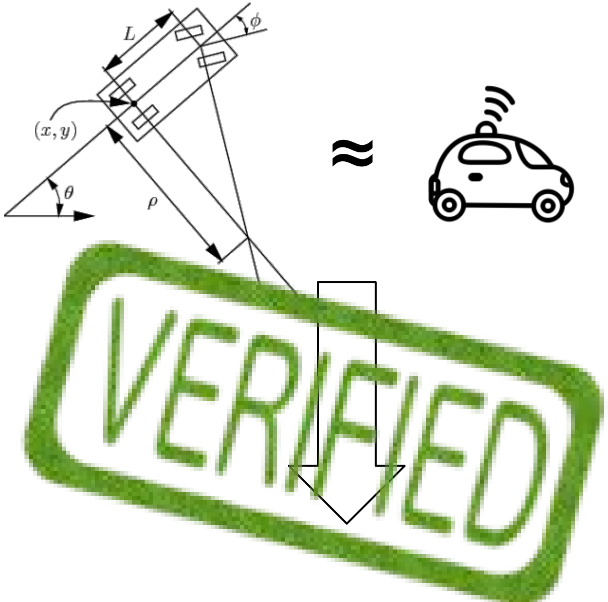
Justified Speculative Control

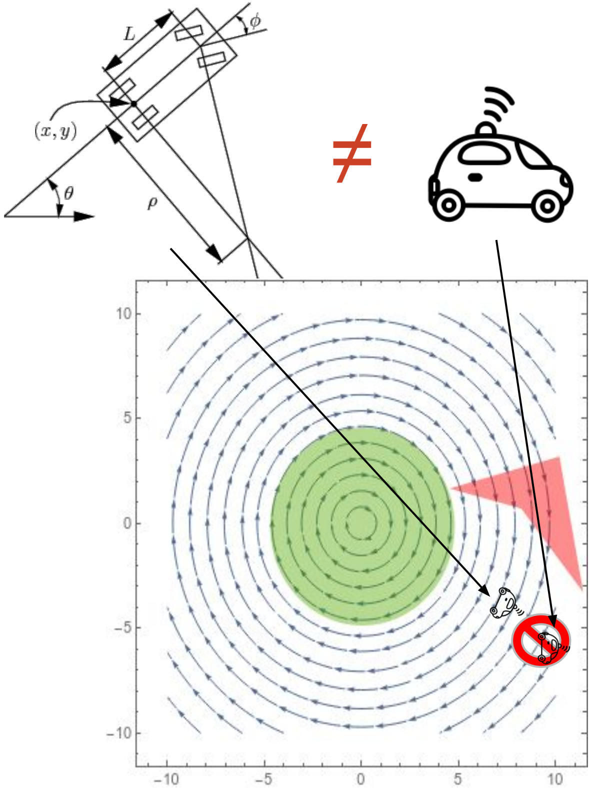


Learn over a constrained action space



Justified Speculative Control





Learn over a constrained action space

Safe Reinforcement Learning?

Policy deviates from model:

1. Policy is deterministic, verification result is set-valued.

Some Actions Aren't Always Safe

{accel,brake,turn} ≠ ?safeAccel; accel ∪ brake

Policy deviates from model:

1. Policy is deterministic, verification result is set-valued.

Some Actions Aren't Always Safe

{accel,brake,turn} ≠ ?safeAccel; accel ∪ brake

Policy deviates from model:

1. Policy is deterministic, verification result is set-valued.

Safe Reinforcement Learning?

Policy deviates from model:

1. Policy is deterministic, verification result is set-valued.

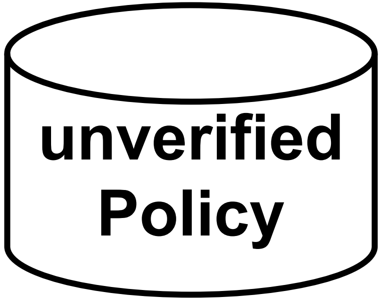
Physical Models are Approximations

Policy deviates from model:

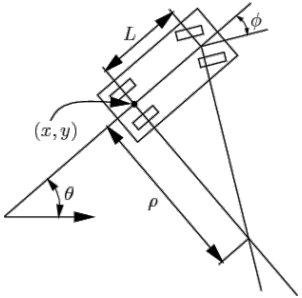
1. Policy is deterministic, verification result is set-valued.
2. Environment may not be accurately modeled.

Safety resolving non-determinism

$?\texttt{safeAccel};\ \texttt{accel} \cup \texttt{brake} \neq$



Sandboxing Reinforcement Learning

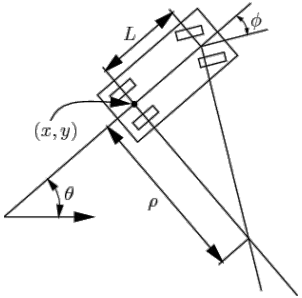


≈

"Accurate modulo determinism"

init → [{ {accel ∪ brake}; t:=0; continuousMotion }*](safe)

Sandboxing Reinforcement Learning



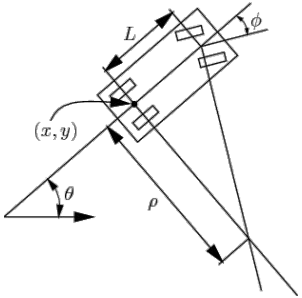
≈

"Accurate modulo determinism"

↓

Learn over a constrained action space

Sandboxing Reinforcement Learning



≈

"Accurate modulo determinism"

Learn over a constrained action space

Sandboxing Reinforcement Learning

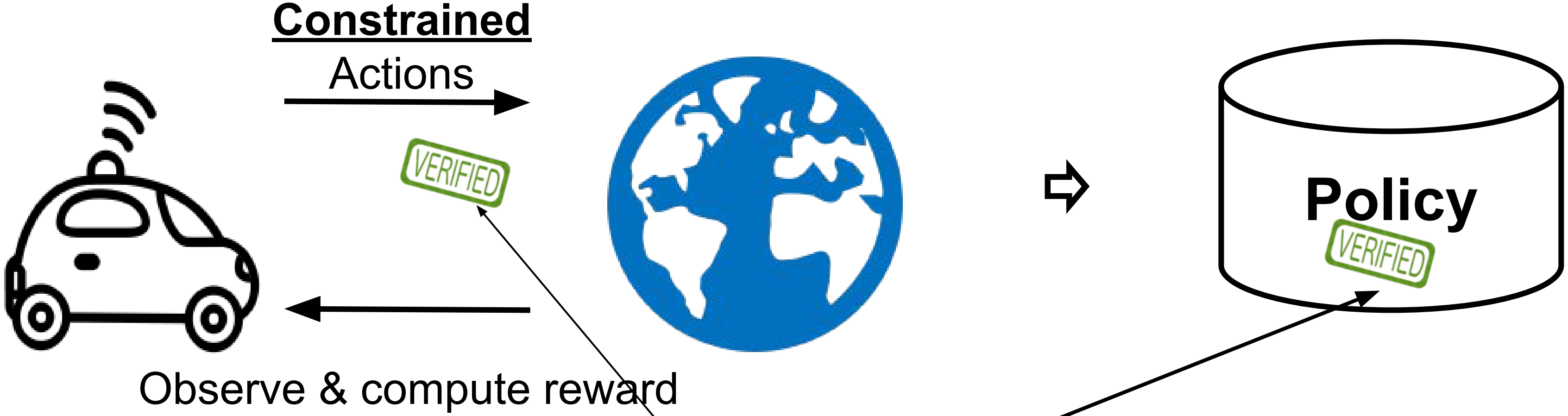
Theorem: **If the physical model is accurate** then verification results are preserved during learning and by learned policies.

Sandboxing Reinforcement Learning


```
init → [{ {accel ∪ brake}; t:=0; continuousMotion }*](safe)
```

Theorem: **If the physical model is accurate** then verification results are preserved during learning and by learned policies.

Sandboxing Reinforcement Learning




```
init → [{ {accel ∪ brake}; t:=0; continuousMotion }*](safe)
```

Theorem: If the physical model is accurate then **verification results are preserved during learning and by learned policies**.

~~Sandboxing~~ Safe Reinforcement Learning

Theorem 1 (JSCGeneric Explores Safely in Modeled Environments). *Assume a valid safety specification*

$$\models \; init \rightarrow [\{ctrl; plant\}^*]safe \qquad (3)$$

i.e., any repetition of $\{ctrl; plant\}$ *starting from a state in init will end in a state described by safe. Then* $u_i(s_i) \models safe$ *for all* u_i, s_i *satisfying the learning process for*

$$(init, (S, A, R, E), choose, update, done, CM, MM)$$

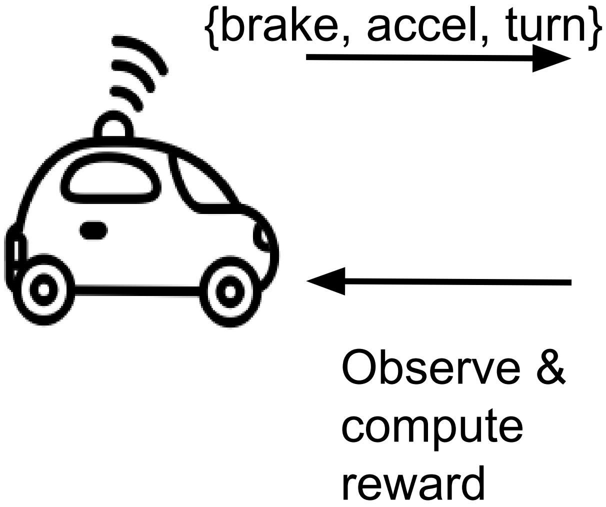
where CM and MM are the controller and model monitor for $init \rightarrow [\{ctrl; plant\}^*]safe$.

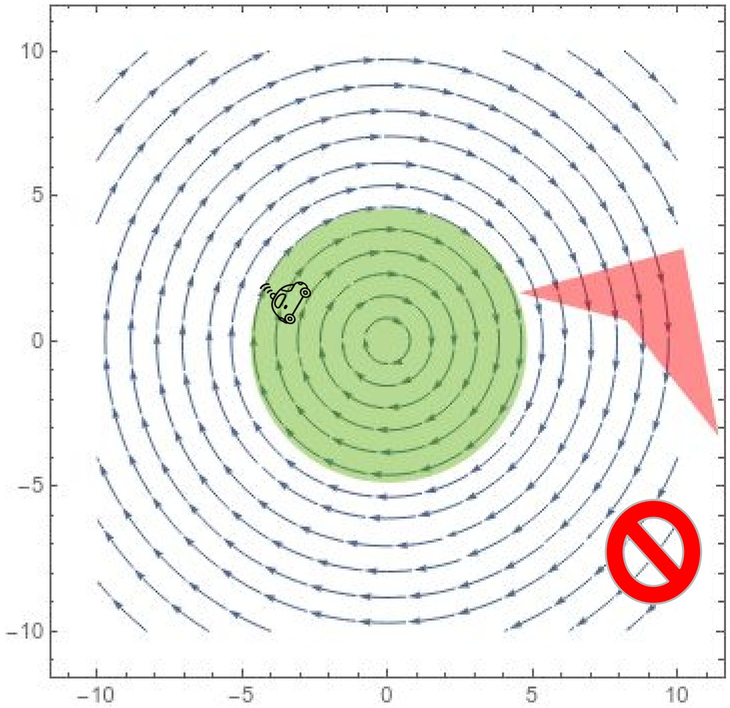
init → … (safe)

Theorem … ation

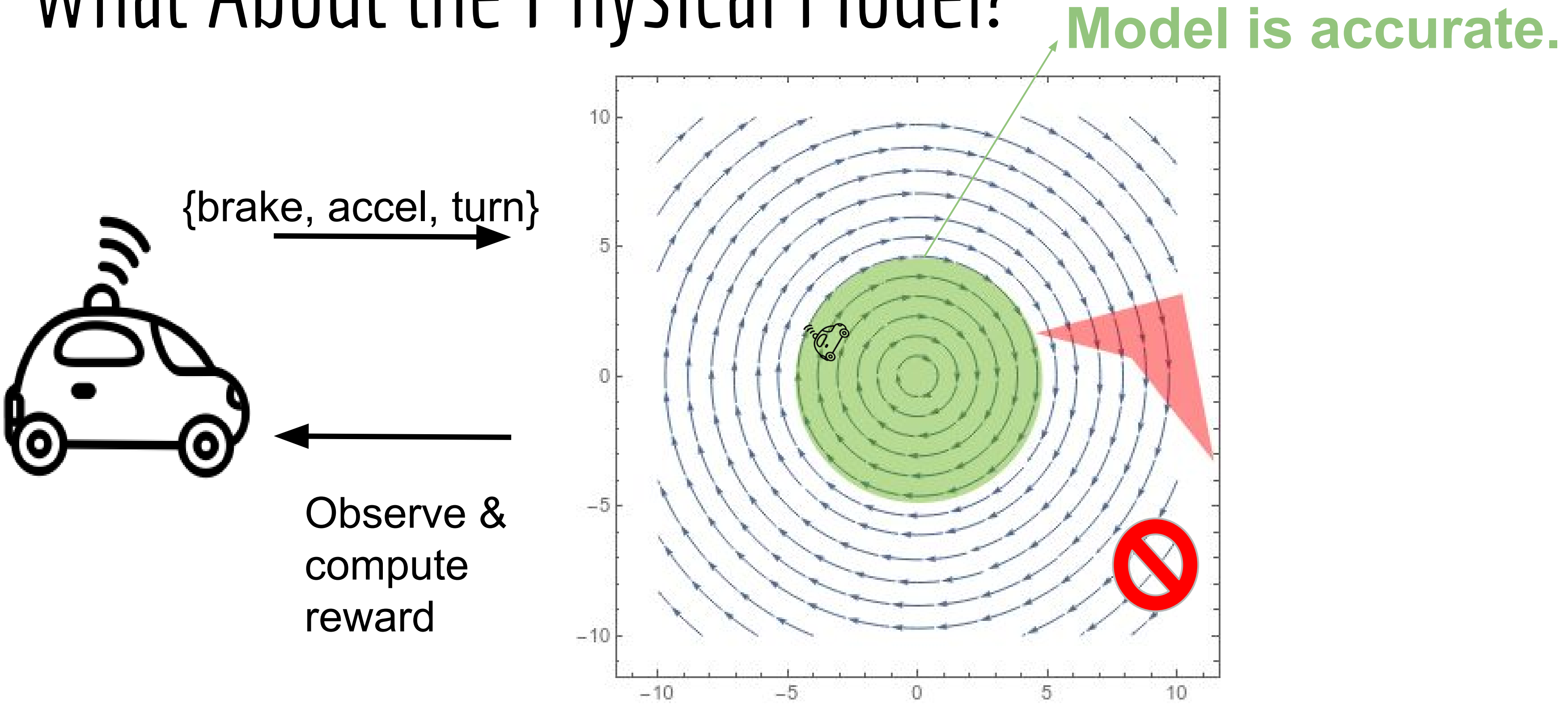
results are preserved by learned policies.

What About the Physical Model?



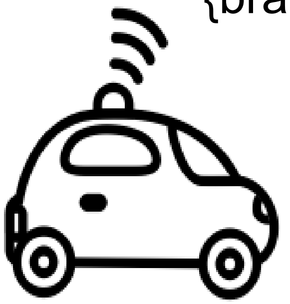


What About the Physical Model?



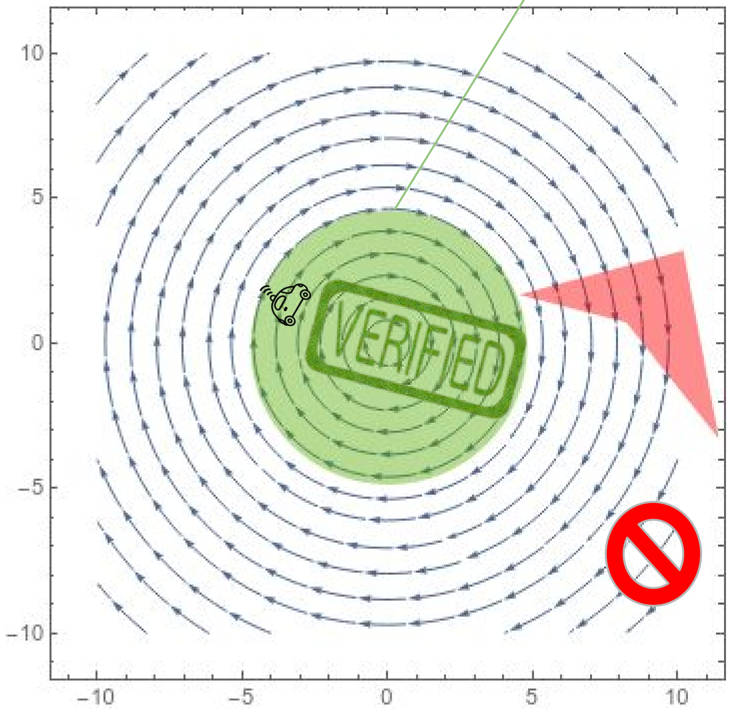
What About the Physical Model?

Model is accurate.



{brake, accel, turn}

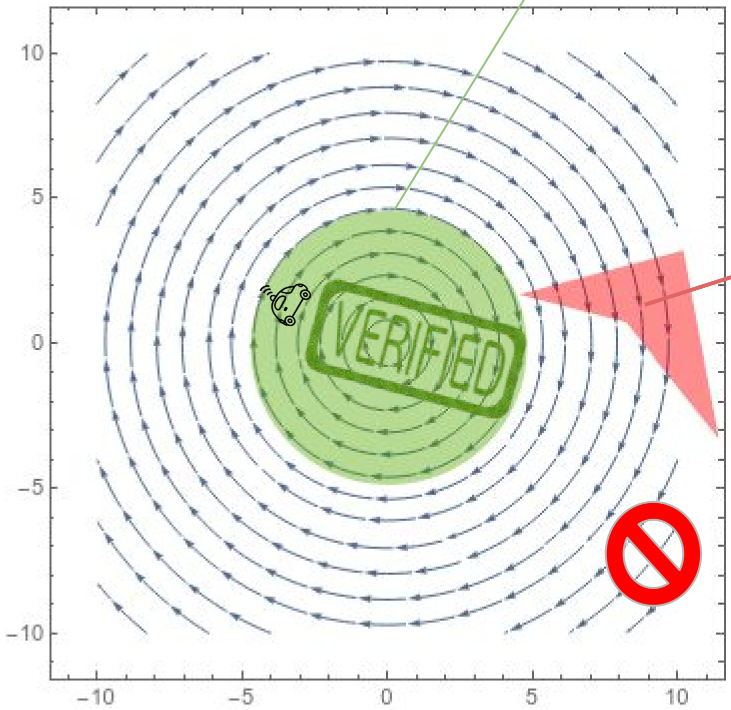
Observe & compute reward



What About the Physical Model?

{brake, accel, turn}

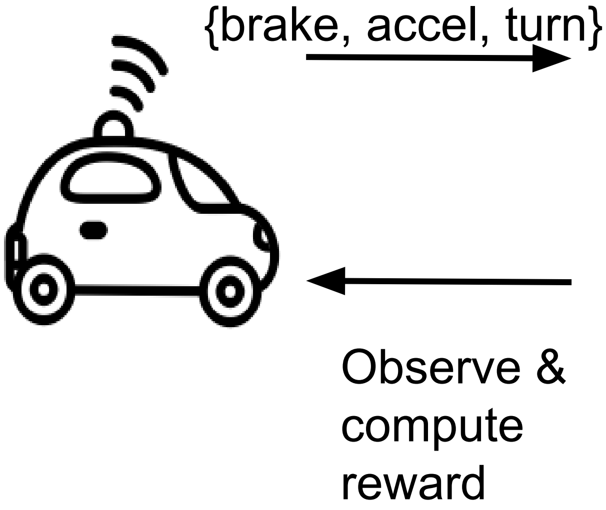
Observe & compute reward

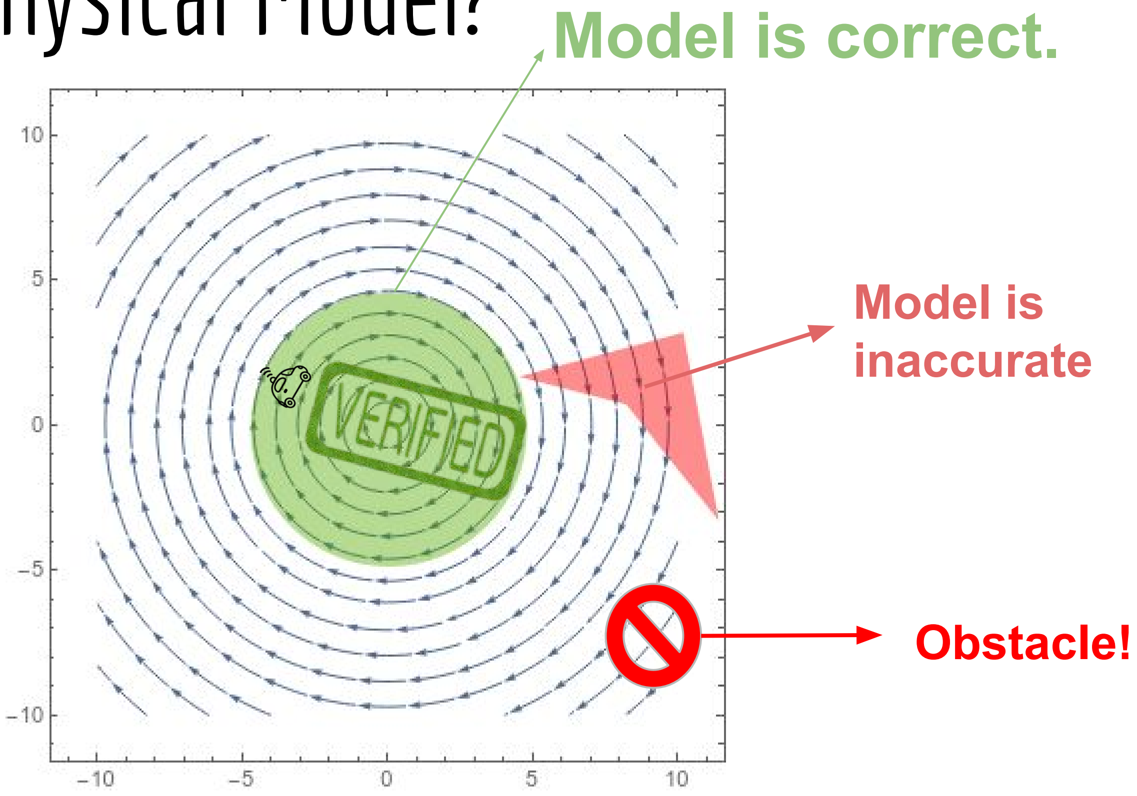


Model is correct.

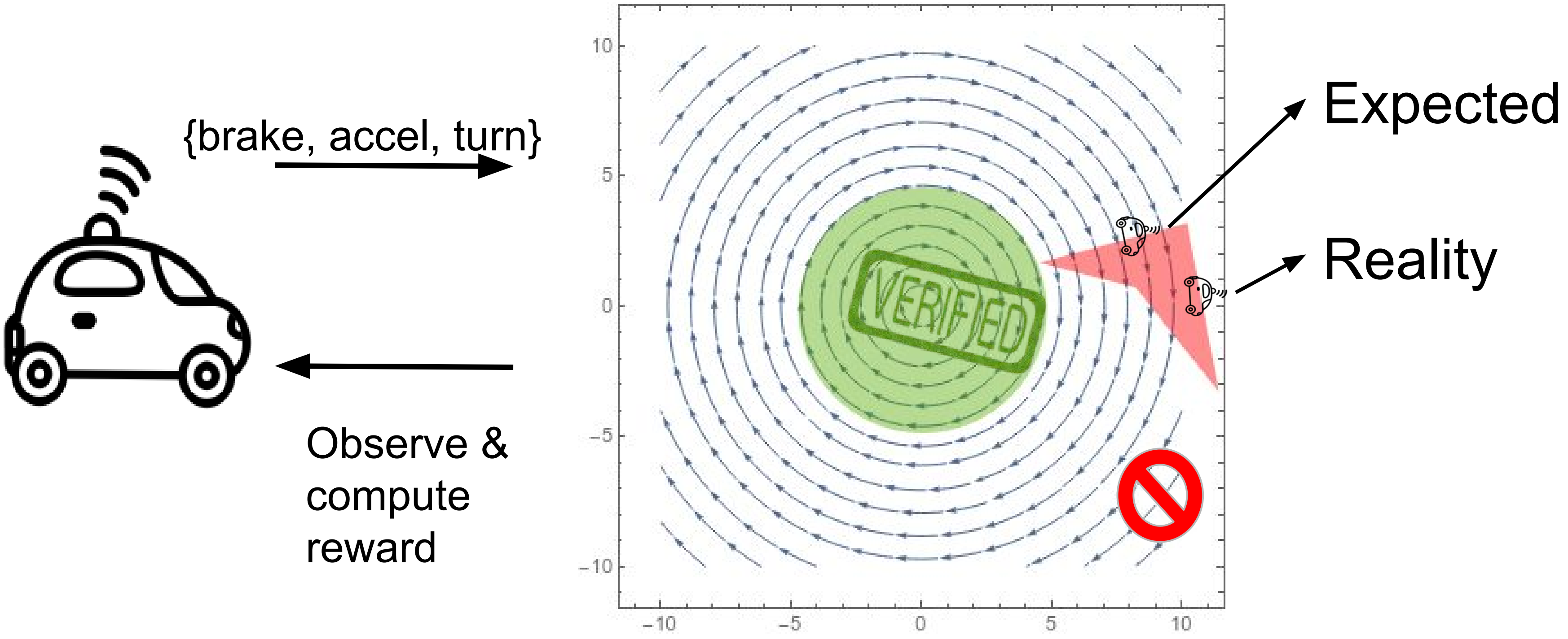
Model is inaccurate

What About the Physical Model?

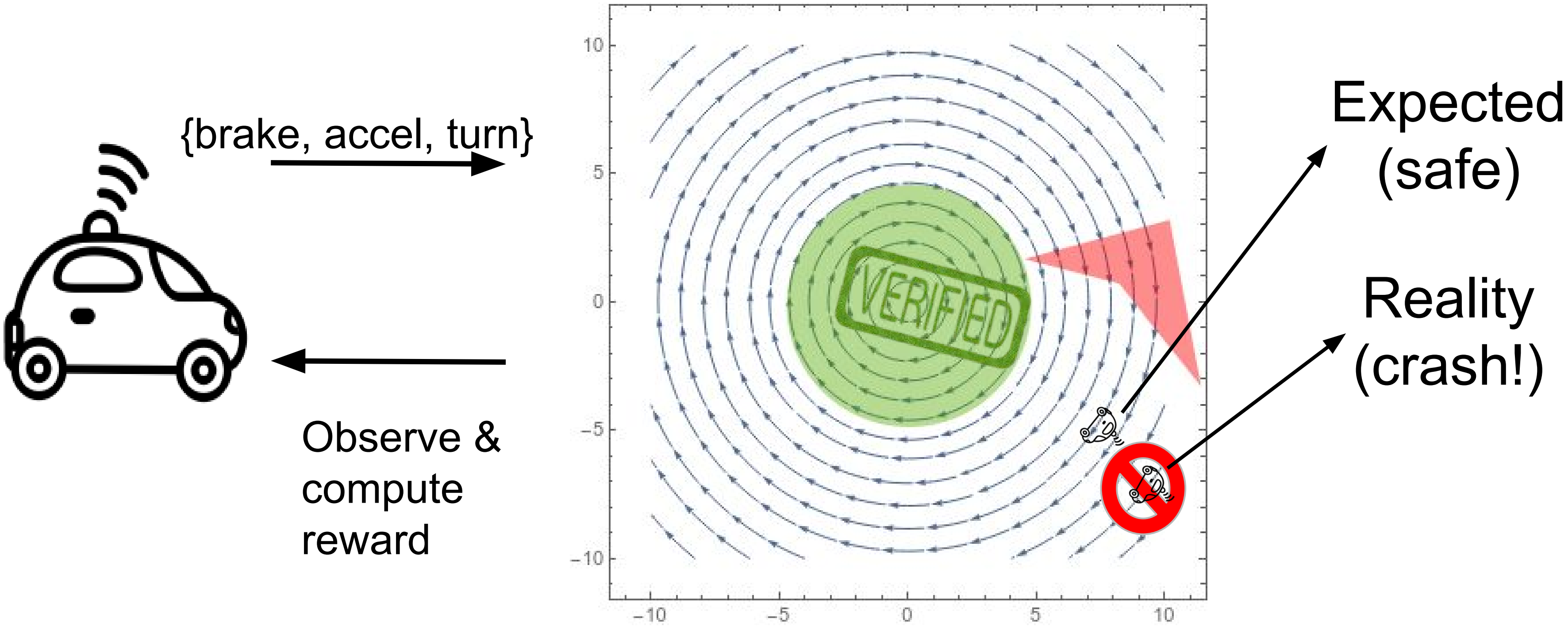




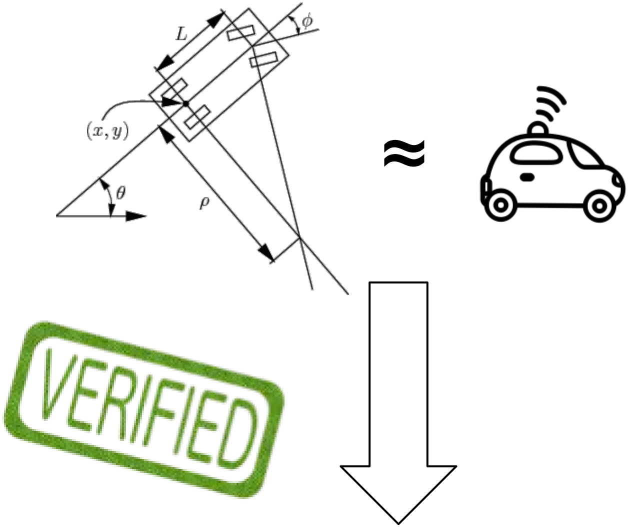
What About the Physical Model?

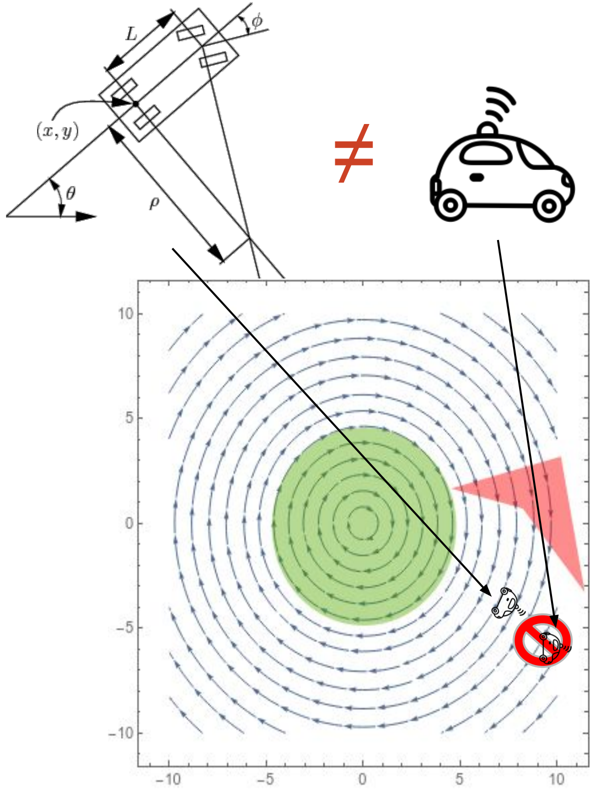


What About the Physical Model?



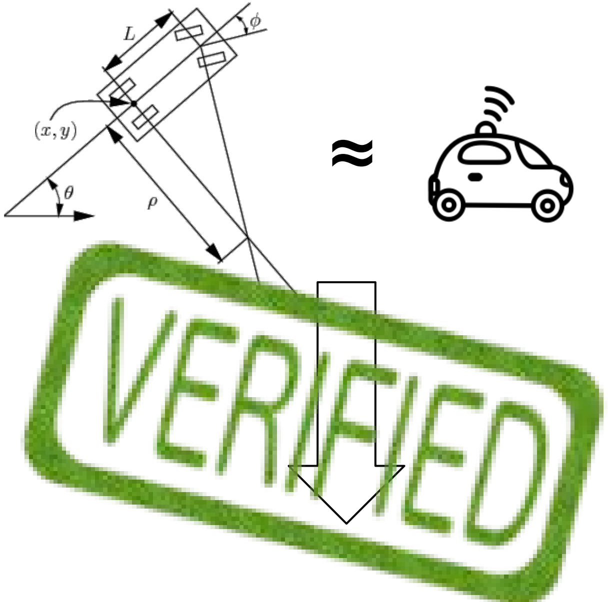
Justified Speculative Control



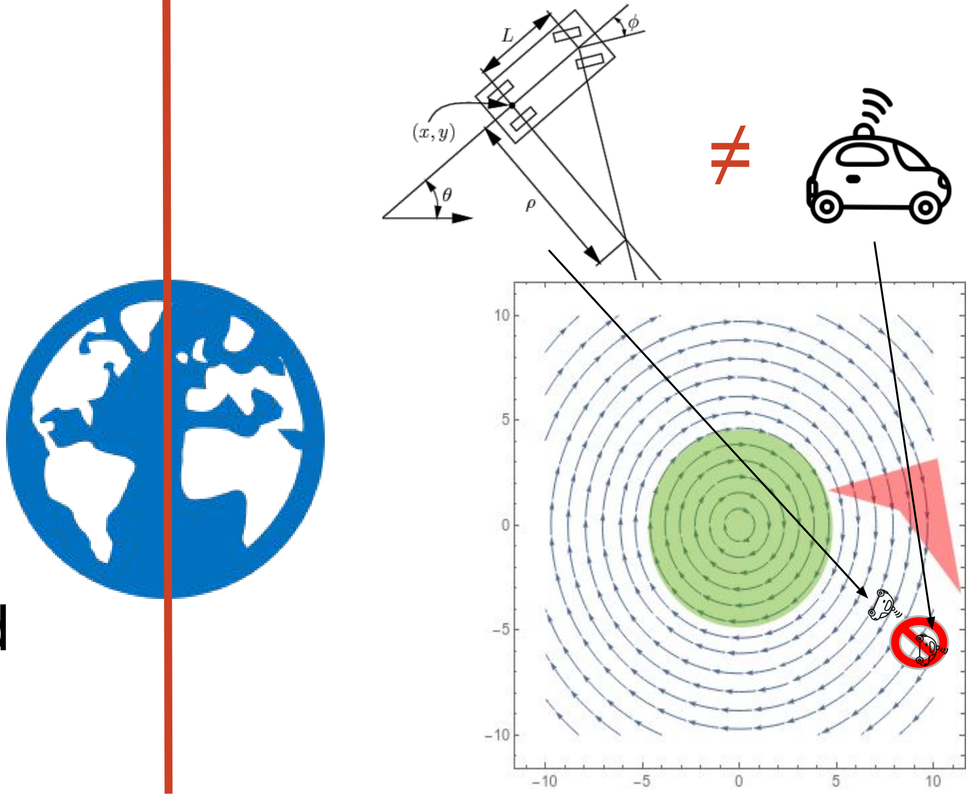


Learn over a constrained action space

Justified Speculative Control

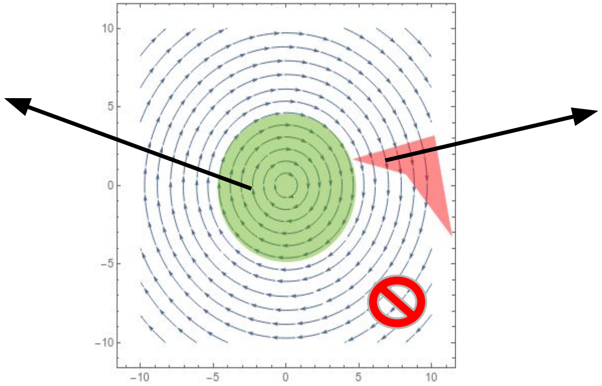


Learn over a constrained action space



Justified Speculative Control

Learn over a constrained action space



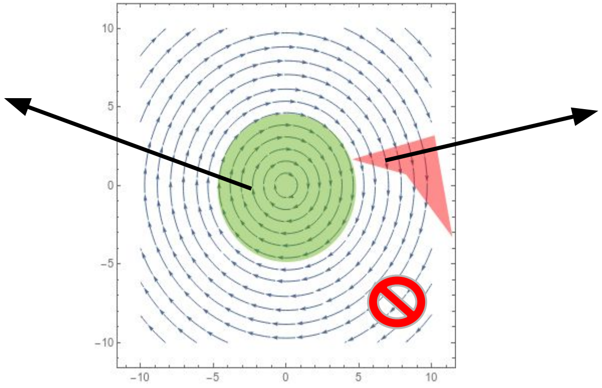
Learn

Some Questions:

1. How do we **know** when we're in unmodeled state space?
2. What do we **do** when we *are* in modeled state space?

Justified Speculative Control

Learn over a constrained action space



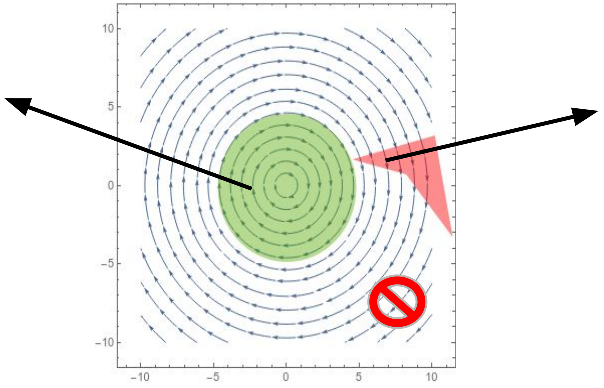
Learn

Some Questions:

1. How do we **know** when we're in unmodeled state space?
2. What do we **do** when we *are* in modeled state space?

Justified Speculative Control

Learn over a constrained action space



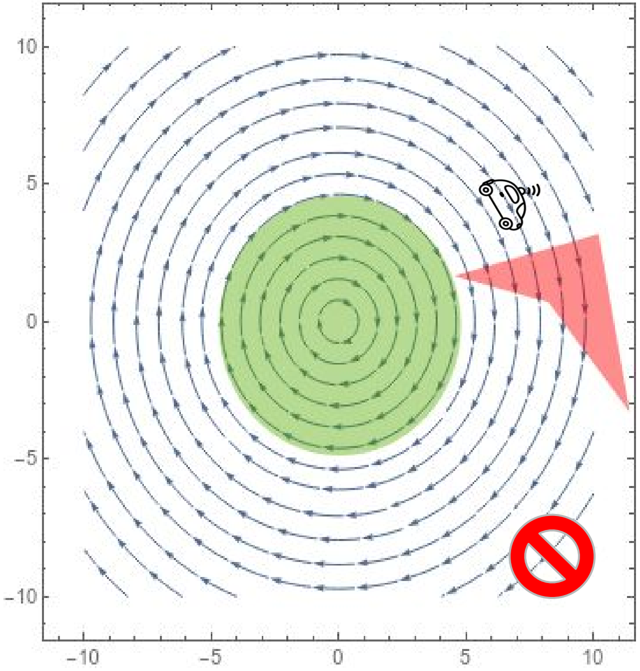
Learn

Theorem: Verification results are preserved outside of red region. But:

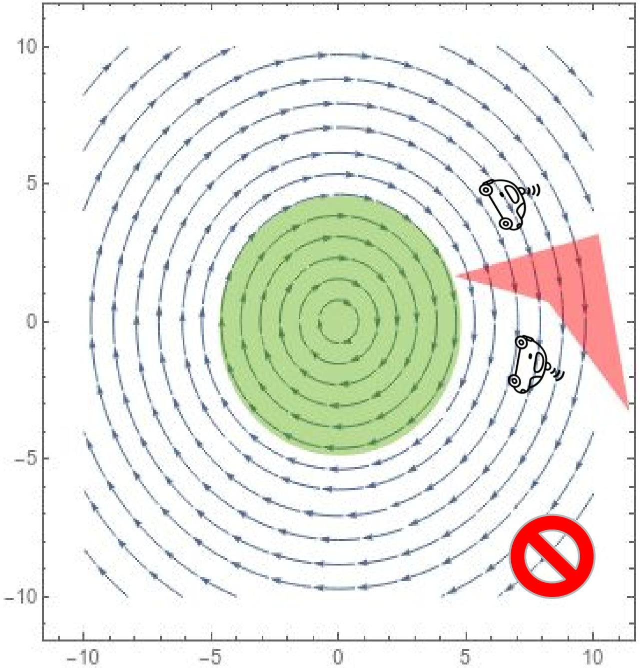
☒ How do we know when we're in unmodeled state space?

☐ **What do we do when we *are* in modeled state space?**

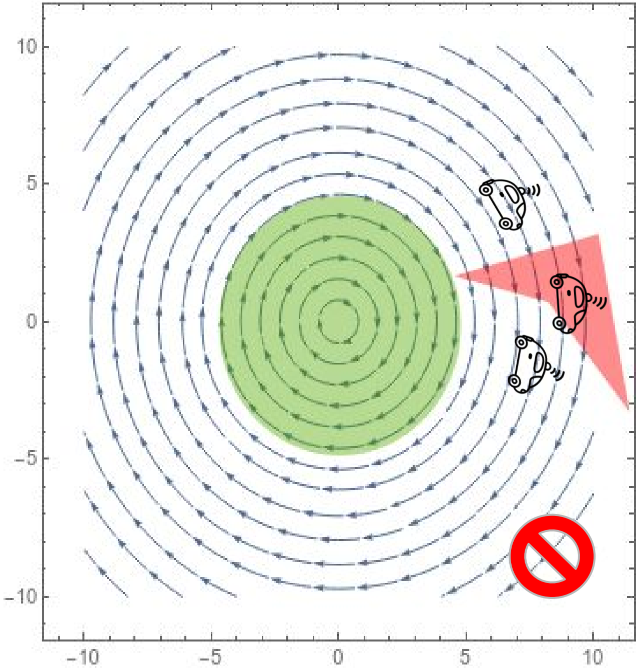
What do we do in unmodeled state-space?



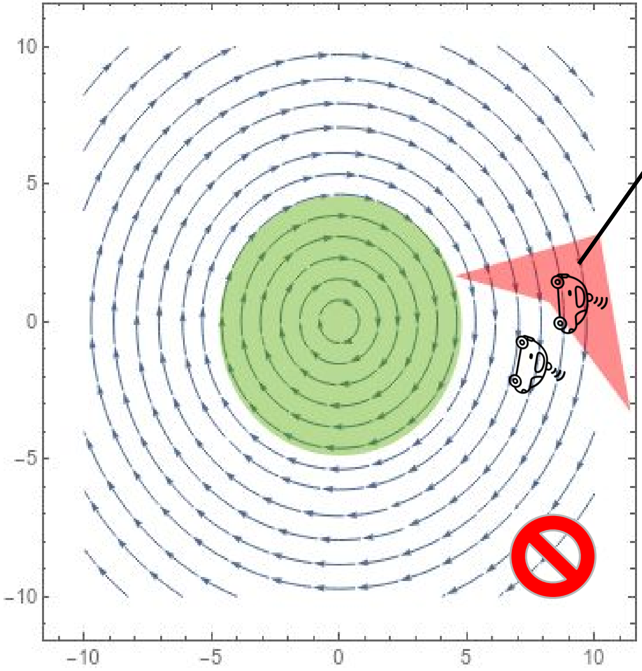
What do we do in unmodeled state-space?



What do we do in unmodeled state-space?

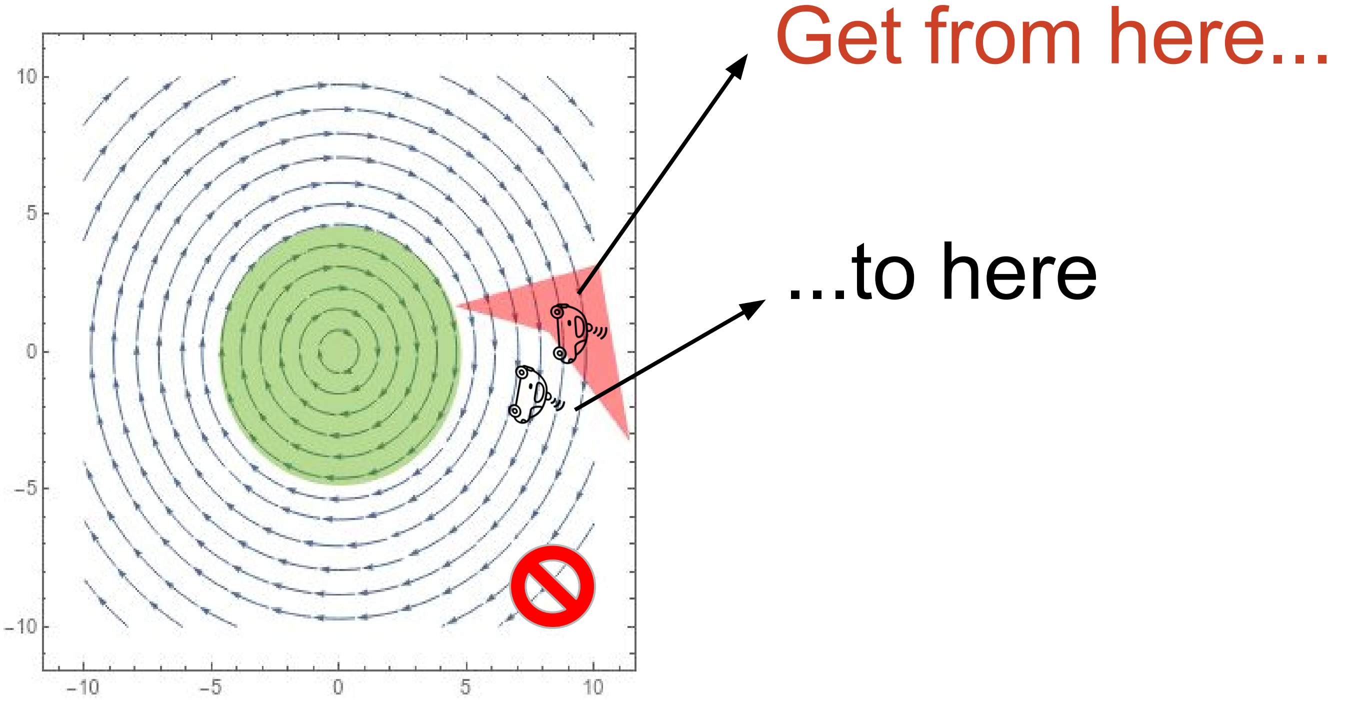


What do we do in unmodeled state-space?



Get from here...

What do we do in unmodeled state-space?



Leveraging Formal Methods during Learning

Own Car

Leader

Leveraging Formal Methods during Learning

Own Car

Leader

Perturbation	“Don’t hit the leader”	“Get back to modeled state space”
5%	3	2
25%	18	16
50%	41	24

Conclusion

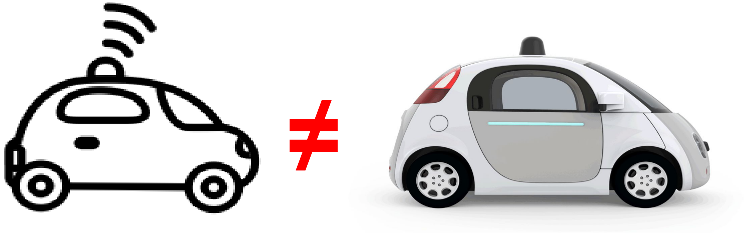
KeYmaera X + Justified Speculative Control:

1. Transfer **formal** verification results for **non-deterministic** control policies to policies obtained via a generic reinforcement learning algorithm.

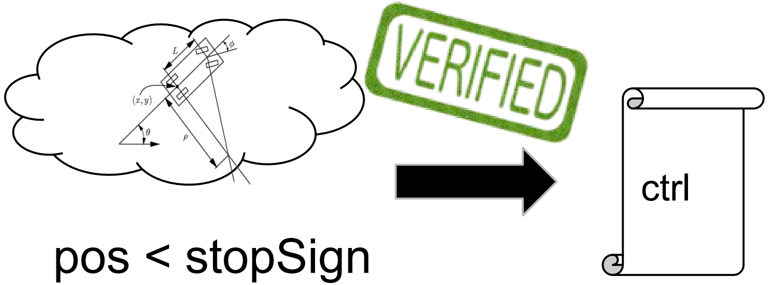
Conclusion

KeYmaera X + Justified Speculative Control:

1. Transfer **formal** verification results for **non-deterministic** control policies to policies obtained via a generic reinforcement learning algorithm.
2. Leverages insights obtained during verification to direct future learning.



Model-Based Verification



pos < stopSign

```
init → [{
   {?safeAccel; accel
    ∪ brake};
   t:=0; {pos'=vel,vel'=acc}
}*](pos < stopSign)
```

Reinforcement Learning