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
How can we provide people with autonomous cyber-physical systems they can bet their lives on?
Model-Based Verification  Reinforcement Learning
Model-Based Verification

Reinforcement Learning

\( \text{pos} < \text{stopSign} \)
Model-Based Verification

pos < stopSign

Reinforcement Learning
Approach: prove that control software achieves a specification with respect to a model of the physical system.
**Model-Based Verification**

pos < stopSign

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

- Strong safety guarantees
- Automated analysis
Benefits:

- Strong safety guarantees
- Automated analysis

Drawbacks:

- Control policies are typically non-deterministic: answers "what is safe", not "what is useful"
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.
**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
Benefits:
● Strong safety guarantees
● Automatic policies (ATP)

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

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

Goal: Provably correct reinforcement learning
Model-Based Verification

Benefits:

- Strong safety guarantees
- Atoms computational aids (ATP)

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

Goal: Provably correct reinforcement learning

1. Learn Safety
2. Learn a Safe Policy
3. Justify claims of safety
Model-Based Verification

Accurate, analyzable models often exist!

\{
  \{\textsf{safeAccel}\&\textsf{accel} \cup \textsf{brake} \cup \textsf{safeTurn}\&\textsf{turn}\};
  \{\textsf{pos'} = \textsf{vel}, \textsf{vel'} = \textsf{acc}\}
\}^*
Model-Based Verification

**Accurate**, analyzable models often exist!

\[
\{ \{\text{safeAccel};\text{accel} \cup \text{brake} \cup \text{safeTurn};\text{turn}\} ; \\
\{\text{pos'} = \text{vel}, \text{vel'} = \text{acc}\} \\
\}^* \quad \text{Continuous motion} \quad \text{discrete control}
\]
Model-Based Verification

**Accurate**, analyzable models often exist!

\[
\{ \\
\text{\{?safeAccel;accel} \cup \text{\textit{brake}} \cup \text{?safeTurn; turn}}; \\
\text{\{pos' = vel, vel' = acc}\}
\}^* \\
\text{Continuous motion} \quad \text{discrete, } \textit{non-deterministic} \quad \text{control}
\]
Model-Based Verification

**Accurate, analyzable** models often exist!

\[
\text{init} \rightarrow [\{
\{ ?\text{safeAccel};\text{accel} \cup \text{brake} \cup ?\text{safeTurn}; \text{turn}\};
\{\text{pos'} = \text{vel}, \text{vel'} = \text{acc}\}
\}^*] \text{pos} < \text{stopSign}
\]
Model-Based Verification

**Accurate, analyzable** models often exist!

formal verification gives strong safety guarantees

\[
\text{init} \rightarrow \begin{cases} 
\{ \; \text{?safeAccel; accel} \cup \text{brake} \; \cup \; \text{?safeTurn; turn} \}; \\
\{ \text{pos'} = \text{vel}, \; \text{vel'} = \text{acc} \} \\
\}^* \text{pos} < \text{stopSign}
\]
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

\[ \text{VERIFIED} \]

• 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!

\[
\textit{How to implement?} \\
\begin{align*}
\{ & \text{?safeAccel;accel} \cup \text{brake} \cup \text{?safeTurn; turn}\}; \\
\{ & \text{pos’ = vel, vel’ = acc}\}
\end{align*}
\]

\[
\text{Only accurate sometimes}
\]
Model-Based Verification Isn’t Enough

**Perfect**, analyzable models don’t exist!

How to implement?

\[
\{ \text{?safeAccel;accel} \cup \text{brake} \cup \text{?safeTurn; turn} \};
\]

\[
\{dx'=w*y, dy'=-w*x, \ldots\}
\]

\[*\]

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

Act

Observe & compute reward
Learning to Resolve Non-determinism

accel $\cup$ brake $\cup$ turn

Observe & compute reward
Learning to Resolve Non-determinism

{accel, brake, turn}

Observe & compute reward
Learning to Resolve Non-determinism

\{\text{accel, brake, turn}\}

Observe & compute reward

\rightarrow

Policy
Learning to Resolve Non-determinism

Observe & compute reward

\{\text{accel}, \text{brake}, \text{turn}\} \rightarrow \text{(safe?) Policy}
Learning to **Safely** Resolve Non-determinism

- Observe & compute reward
- Safety Monitor
- (safe?) Policy
Learning to **Safely** Resolve Non-determinism

- **Observe & compute reward**
- **Safety Monitor**
- **(safe?) Policy**

≠ “Trust Me”
Learning to **Safely** Resolve Non-determinism

Observe & compute reward

Use a theorem prover to prove:

\[
\text{(init} \rightarrow \{\{\text{accel} \cup \text{brake}\}; \text{ODEs}\}^{*})(\text{safe})\] \leftrightarrow \varphi
Learning to **Safely** Resolve Non-determinism

Use a theorem prover to prove:

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

Use a theorem prover to prove:

\[
\text{(init} \to \exists \{\{\text{accel} \cup \text{brake}\};\text{ODEs}\}^*(\text{safe})\) \leftrightarrow \varphi
\]
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.

Use a theorem prover to prove:

\[(\text{init} \rightarrow \{\text{accel} \cup \text{brake}\}; \text{ODEs})^* (\text{safe}) \leftrightarrow \varphi)\]
What about the physical model?

Use a theorem prover to prove: 

\[
\text{init} \rightarrow \neg \exists \{\text{pos}'=\text{vel}, \text{vel}'=\text{acc}\} \neq \emptyset
\]

\[
\{\text{accel} \cup \text{brake}\}; \text{ODEs}\}^*\text{(safe)} \leftrightarrow \varphi
\]
What About the Physical Model?

{brake, accel, turn}

Observe & compute reward
What About the Physical Model?

{brake, accel, turn}

Model is accurate.

Observe & compute reward
What About the Physical Model?

Model is accurate.

{brake, accel, turn}

Observe & compute reward
What About the Physical Model?

Observe & compute reward

{brake, accel, turn}

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?

Observe & compute reward

\{\text{brake, accel, turn}\}

Expected

Reality
Speculation is Justified

{brake, accel, turn}

Observe & compute reward

Expected (safe)

Reality (crash!)
Leveraging Verification Results to Learn Better

Observe & compute reward: {brake, accel, turn}

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

$\{\text{accel, brake, turn}\} \neq \text{safeAccel}; \text{ accel } \cup \text{ brake}$

Policy deviates from model:
1. Policy is deterministic, verification result is set-valued.
Some Actions Aren’t Always Safe
\{\text{accel, brake, turn}\} \neq \textcolor{red}{?safe\text{Accel}}; \text{ accel } \cup \text{ 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

\(?\text{safeAccel}; \text{accel} \cup \text{brake} \neq\) unverified Policy
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.
Theorem: If the physical model is accurate then verification results are preserved during learning and by learned policies.
Sandboxing Reinforcement Learning

Theorem: If the physical model is accurate then verification results are preserved during learning and by learned policies.
Theorem 1 (JSCGeneric Explores Safely in Modeled Environments). Assume a valid safety specification

\[ \models \text{init} \rightarrow \left[ \{\text{ctrl}; \text{plant}\}^* \right] \text{safe} \] (3)

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

\[ (\text{init}, (S, A, R, E), \text{choose}, \text{update}, \text{done}, \text{CM}, \text{MM}) \]

where CM and MM are the controller and model monitor for \( \text{init} \rightarrow \left[ \{\text{ctrl}; \text{plant}\}^* \right] \text{safe} \).
What About the Physical Model?

Observe & compute reward

{brake, accel, turn}
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 accurate.
What About the Physical Model?

Observe & compute reward

Model is correct.

Model is inaccurate

{brake, accel, turn}
What About the Physical Model?

{brake, accel, turn}

Model is correct.

Obstacle!

Model is inaccurate
What About the Physical Model?

{brake, accel, turn}

Observe & compute reward

Expected

Reality
What About the Physical Model?

{brake, accel, turn}

Observe & compute reward

Expected (safe)

Reality (crash!)
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

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

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

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?

Get from here...

...to here
Leveraging Formal Methods during Learning

Own Car

Leader
Leveraging Formal Methods during Learning

<table>
<thead>
<tr>
<th>Perturbation</th>
<th>“Don’t hit the leader”</th>
<th>“Get back to modeled state space”</th>
</tr>
</thead>
<tbody>
<tr>
<td>5%</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>25%</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>50%</td>
<td>41</td>
<td>24</td>
</tr>
</tbody>
</table>
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.
$$\text{pos} < \text{stopSign}$$

$$\text{init} \rightarrow \{ \{ \text{?safeAccel}; \text{accel} \cup \text{brake}; \}
\text{\quad t:=0; \{pos'=vel,vel'=acc\} \}
\}^* \text{\quad (pos < stopSign)}$$