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
"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
Model-Based Verification

Reinforcement Learning

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

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”

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
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
**Model-Based Verification**

**Benefits:**
- Strong safety guarantees
- 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
Benefits:
- Strong safety guarantees
- Automated computational aids (ATP)

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

Model-Based Verification

Reinforcement Learning

Goal: Provably correct reinforcement learning
1. Learn Safety
2. Learn a Safe Policy
3. Justify claims of safety

- 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

\[ x := t \]

\[ x = x_0 \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ \ldots \]

\[ x = t \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ \ldots \]
Hybrid Programs

\[ x := t \]

\[ x = x_0 \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ ... \]

\[ x = t \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ ... \]
Hybrid Programs

If $P$ is true: no change

If $P$ is false: terminate

$x := t$

$x = x_0$
y$= y_0$
z$= z_0$
...

$a; b$

a

b
Hybrid Programs

\[ x := t \]

If P is true: no change

If P is false: terminate

\( ?P \)

\( a^* \)

\[ x = x_0 \]
\[ y = y_0 \]
\[ z = z_0 \]

...
Hybrid Programs

\[ x := t \]

\[ x = x_0 \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ ... \]
\[ x = t \]
\[ y = y_0 \]
\[ z = z_0 \]
\[ ... \]

If P is true: no change
If P is false: terminate

?P

\[ x = t \]
\[ y = y_0 \]
\[ z = z_0 \]

\[ a; b \]
\[ a \]
\[ b \]

\[ a \]
\[ a; b \]
\[ a; b \]

\[ a^* \]
\[ a \]
\[ ... a ... \]
Hybrid Programs

\[ x := t \quad \begin{array}{c} x=x_0 \\ y=y_0 \\ z=z_0 \\ \vdots \end{array} \quad \begin{array}{c} x=t \\ y=y_0 \\ z=z_0 \\ \vdots \end{array} \]

\[ ?P \]
If P is true: no change
If P is false: terminate

\[ a^* \]
\[ a \rightarrow \ldots a \rightarrow \]

\[ a; b \]

\[ a \rightarrow b \]

\[ a \cup b \]

\[ \begin{array}{c} x=F(0) \\ \vdots \end{array} \]

\[ x=x_0 \]
\[ x:=t \]
\[ x=x_0 \]
\[ y=y_0 \]
\[ z=z_0 \]
\[ \ldots \]
\[ x=F(0) \]
\[ x=F(T) \]
Approaching a Stopped Car

Is this property true?

\[
\left[ \{ \{ \text{accel} \cup \text{brake} \}; \ t:=0; \ \{ \text{pos}'=\text{vel},\text{vel}'=\text{accel},\ t'=1 \ & \ \text{vel}\ge0 \ & \ t\le T \} \right] \ast (\text{pos} \le \text{stoppedCarPos})
\]
Approaching a Stopped Car

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

\[
[\{\text{accel} \cup \text{brake}\}; t:=0; \{\text{pos}'=\text{vel}, \text{vel}'=\text{accel}, t'=1 & \text{vel} \geq 0 & t \leq T\}]*
\]

\((\text{pos} \leq \text{stoppedCarPos})\)
Approaching a Stopped Car

if we also assume the system is safe initially:

\[
\text{safeDistance}(\text{pos, vel, stoppedCarPos, } B) \rightarrow \\
[
\begin{array}{l}
\{ \{ \text{accel} \cup \text{brake} \}; \ t:=0; \ \{ \text{pos}'=\text{vel}, \text{vel}'=\text{accel}, t'=1 \ & \text{vel} \geq 0 \ & \text{t} \leq T \} \} \ast \\
\end{array}
\]
(pos \leq \text{stoppedCarPos})
safeDistance\( (pos, vel, stoppedCarPos, B) \) →

\[
\exists\{\{accel \cup brake\}; t:=0; \{pos' = pos, vel' = accel, t' = 1 \land vel \geq 0 \land t \leq T\} \}^* (pos \leq stoppedCarPos)\
\]
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 \textit{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?
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**
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!

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

**Accurate**, analyzable models often exist!

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

Accurate, analyzable models often exist!

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

**Accurate, analyzable** models often exist!

formal verification gives strong safety guarantees

\[\text{init} \rightarrow \left\{\begin{array}{l}
\{ \text{?safeAccel, } \text{accel} \cup \text{brake} \cup \text{?safeTurn; turn}\} ; \\
\{\text{pos}' = \text{vel}, \text{vel}' = \text{acc}\}
\end{array}\right\} \text{pos} < \text{stopSign}\]
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

=  
• 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?

\[
\{ \text{pos'} = \text{vel}, \text{vel'} = \text{acc} \}
\]

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

Only accurate sometimes
Model-Based Verification Isn’t Enough

Perfect, analyzable models don’t exist!

How to implement?

\[
\begin{align*}
\{ & \ ?\text{safeAccel};\text{accel} \cup \text{brake} \cup \ ?\text{safeTurn}; \text{turn}; \\
& \{ dx'=w*y, \ dy'=-w*x, \ldots \}
\end{align*}
\]

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

Observe & compute reward

Act
Learning to Resolve Non-determinism

accel $\cup$ brake $\cup$ turn

Observe & compute reward
Learning to Resolve Non-determinism

Observe & compute reward

{accel, brake, turn}
Learning to Resolve Non-determinism

{accel, brake, turn}

Observe & compute reward

Policy
Learning to Resolve Non-determinism

{accel, brake, turn} → (safe?) Policy

Observe & compute reward
Learning to **Safely** Resolve Non-determinism

- **Safety Monitor**
- Observe & compute reward

Useful to stay safe during learning

Crucial after deployment

Policy

(safe?)
Learning to **Safely** Resolve Non-determinism

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

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

Use a theorem prover to prove:

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

\[\varphi\]
Learning to **Safely** Resolve Non-determinism

Use a theorem prover to prove:

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

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

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

Use a theorem prover to prove:

\[(\text{init} \rightarrow [\{\{\text{accel} \cup \text{brake}\}; \text{ODEs}\}^*](\text{safe})) \neq \phi \]
What About the Physical Model?

{brake, accel, turn}

Observe & compute reward
What About the Physical Model?

Model is accurate.

{brake, accel, turn}

Observe & compute reward
What About the Physical Model?

Model is accurate.
What About the Physical Model?

Model is accurate.

Model is inaccurate.

{brake, accel, turn}

Observe & compute reward
What About the Physical Model?

{brake, accel, turn}

Observe & compute reward

Model is accurate.

Model is inaccurate

Obstacle!
What About the Physical Model?

{brake, accel, turn}

Expected

Reality

Observe & compute reward
Speculation is Justified

Observe & compute reward

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

Expected (safe)

Reality (crash!)
Leveraging Verification Results to Learn Better

{brake, accel, turn}

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

Observe & compute reward
Safe RL: How?

Details:

☐ Detect \textbf{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.

\[
[x:=t]f(x) \leftrightarrow f(t) \\
[a;b]P \leftrightarrow [a][b]P \\
[a\cup b]P \leftrightarrow ([a]P \land [b]P) \\
[x' = f\& Q]P \rightarrow (Q \rightarrow P) \\
\ldots
\]
Detecting unmodeled State Space

```plaintext
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 (\(\text{QE}(\exists t. \text{model}_\text{after}(t) == \text{newPos})\)):
  # No model deviation.
else:
  # Model deviation…?
Safe RL: How?

Details:

- Runtime monitoring separates **modeled** from **unmodeled** state space.
- Convert monitors into reward signals

![Diagram showing a state space with modeled and unmodeled regions, a verified area, and an area marked as forbidden.](image)
Safe RL: How?

Details:

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

- Convert monitors into reward signals:
  \((\mathbb{R}^n \rightarrow \mathbb{B}) \rightarrow (\mathbb{R}^n \rightarrow \mathbb{R})\)!
An Example

\[\text{init} \rightarrow \{\{\{\text{?safeAccel}; \text{accel} \cup \text{brake} \cup \text{?safeMaint}; \text{maintVel}\};\{\text{pos'} = \text{vel}, \text{vel'} = \text{acc}, t'=1\}\}^*\text{safe}\]
An Example Monitor

$\text{init} \rightarrow [\{

\{?\text{safeAccel};\text{accel} \cup \text{brake} \cup ?\text{safeMaintain}; \text{maintainVel}\};

\{\text{pos'} = \text{vel}, \text{vel'} = \text{acc}, \text{t'}=1\}
\}]*\text{safe}

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

(t_{\text{post}} \geq 0 \land a_{\text{post}} = 0 \land v_{\text{post}} = v \land p_{\text{post}} = vt_{\text{post}} + p) \lor \text{Etc.}$
An Example Monitor

\[
\text{init} \rightarrow \begin{cases} 
\{ \text{?safeAccel; accel} \cup \text{brake} \cup \text{?safeMaintain; maintainVel} \}; \\
\{ \text{pos'} = \text{vel}, \text{vel'} = \text{acc}, t' = 1 \} \end{cases}
\]

\[
\text{safe} \begin{cases} 
\text{pos'} = \text{vel}, \text{vel'} = \text{acc}, t' = 1 \end{cases}
\]

\[
(t_{\text{post}} \geq 0 \land a_{\text{post}} = \text{accel} \land v_{\text{post}} = \text{acc} \cdot t_{\text{post}} + v \land p_{\text{post}} = \text{acc} \cdot t_{\text{post}}^2/2 + v \cdot t_{\text{post}} + p) \lor
\]

\[
(t_{\text{post}} \geq 0 \land a_{\text{post}} = 0 \land v_{\text{post}} = v \land p_{\text{post}} = v \cdot t_{\text{post}} + p) \lor \text{Etc.}
\]
An Example: The Monitor

\[
\text{init} \rightarrow \left[ \begin{array}{c}
\{?\text{safeAccel};\text{accel} \cup \text{brake} \cup ?\text{safeMaintain}; \text{maintainVel}\}; \\
\{\text{pos'} = v, \text{vel'} = a, t' = 1\}
\end{array} \right] \]

\text{safe} \left( t_{\text{post}} \geq 0 \land a_{\text{post}} = \text{acc} \land v_{\text{post}} = \text{accel} t_{\text{post}} + v \land p_{\text{post}} = \text{acc} t_{\text{post}}^2/2 + v t_{\text{post}} + p \right) \lor \\
\left( t_{\text{post}} \geq 0 \land a_{\text{post}} = 0 \land v_{\text{post}} = v \land p_{\text{post}} = vt_{\text{post}} + p \right) \lor \text{Etc.}
An Example: The Monitor

init → [{

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

{pos' = vel, vel' = acc, t'=1}
}]*]safe

(t_{post} >= 0 ∧ a_{post} = acc ∧ v_{post} = accel t_{post} + v ∧ p_{post} = acc t_{post}^2/2 + v t_{post} + p) ∨

(t_{post} >= 0 ∧ a_{post} = 0 ∧ v_{post} = v ∧ p_{post} = vt_{post} + p) ∨ 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?

![Diagram](image.png)
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
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