Formal Verification of RL-based Approaches

Amin Ahantab  
amin.ahantab@tum.de

René Filip  
rene.filip@tum.de

SS 2018
Testing does not suffice

BUSINESS NEWS  FEBRUARY 12, 2014 / 3:55 PM / 4 YEARS AGO

Toyota to recall 1.9 million Prius cars for software defect in hybrid system

Reuters Staff

TOKYO (Reuters) - Toyota Motor Corp (7203.T) on Wednesday issued a recall covering all 1.9 million of the third-generation Prius cars sold worldwide, due to a programming glitch in their hybrid system.

Testing does not suffice

“Program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence.”

Edsger W. Dijkstra
Topics

1. Why Formal Verification?
2. Verifiable Properties
3. Formal Verification Methods
4. Summary and Challenges

1. Introduction to Formal Verification
2. Verification of an MDP
3. Verification of a Policy
4. Runtime Verification
1. Why Formal Verification?
Models become too complex!

[2] Human-level control through deep reinforcement learning
Adversarial Examples

1. Why Formal Verification?

Safe and Fast Exploration

1. Why Formal Verification?

- Exploration too dangerous and inefficient
- Inject domain knowledge via specifications
- Adoption to new environment
Optimization vs Safety

Problem → Clever RL Algorithm → Optimized Policy

+ Specification → Theorem Solver → Safe Policy

Optimized and Safe Policy?

1. Why Formal Verification?
2. Verifiable Properties
Model Verification

Does my MDP reflect the reality?

Will at least 95% of my runs end up in a good state?
Policy Verification

Environment → RL → Optimized Policy

Is my policy stable?
Can I ensure certain actions won’t happen in a specific state?
Can it deal with perturbations?
Code Verification

RL Implementation
1. if (p < 0.1) {
2.     // do something
3. } else {
4.     // do something else
5. }

Did I implement the code correctly?

MDP  RL  Optimized Policy

Challenges in the Verification of Reinforcement Learning Algorithms
Algorithm Verification

Will my algorithm converge to a (safe) policy?

MDP \rightarrow RL \rightarrow Optimized Policy

2. Verifiable Properties

Runtime Verification

Guarantee only safe actions!
Can I accelerate the training?

Environment $\xrightarrow{\text{RL}}$ Optimized Policy

Specifications $\xrightarrow{\text{Monitor}}$ Model

2. Verifiable Properties

3.1 Introduction to Formal Verification
## Comparison of Approaches

<table>
<thead>
<tr>
<th>Testing/Simulation</th>
<th>Program analysis</th>
<th>Deductive methods</th>
<th>Model checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset of executions</td>
<td>Superset of possible executions</td>
<td>Superset of possible executions</td>
<td>All possible executions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Eg. (all humans are mortal) ∧</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Sokrates is a human) →</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Sokrates is mortal)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>All possible executions</td>
<td></td>
</tr>
</tbody>
</table>

| + cost-efficient       | + can find absence of             | + complete                                       | + works automatically                               |
|                        | obvious bugs (E.g. division      |                                                  | + complete                                          |
|                        | by 0)                             |                                                  | + can be used for                                   |
|                        | + efficient                       |                                                  | concurrent, reactive &                              |
|                        |                                   |                                                  | distributed systems                                 |
| - incomplete           | - incomplete                      | - can yield false positive                       | - computationally costly                             |
|                        | - can yield false positive        | - difficult for concurrent                      |                                                     |
|                        |                                   | systems                                          |                                                     |
|                        |                                   | - Limited by human                               |                                                     |
|                        |                                   | prover                                           |                                                     |

Probabilistic Model Checking of a MDP

Example:

verify following specification: “The Probability that the car will stop at a red light is 0.999999“
What is Model checking?

Model checking are methods that:

• verify whether a given system satisfies a given specification
• work automatically
• prove correctness of the system w.r.t. to the specification
• or exhibit a counterexample
Model checking problem in Propositional logic

• Problem:
  • Given a valuation $\beta$ and a formula $F$ of propositional logic
  • check whether $\beta$ is a model of $F$

• Solution:
  • Replace the atomic propositions by their truth values in $\beta$, then use a truth table to evaluate to 1 or 0.

• Examples:
  • Let $\beta_1(A) = 1$ and $\beta_1(B) = 0$. Then $\beta_1 \not\models A \land B$ and $\beta_1 \models B$
  • Let $\beta_2(A) = 1$ and $\beta_2(B) = 1$. Then $\beta_2 \models A \land B$ and $\beta_2 \not\models \neg B$
Temporal logic

- Truth values of atomic proposition may change with time
- Example: Truth values of A in the course of Anna’s life

Possible statements:
Anna will **eventually** be an architect (at some point in the future)
Anna is an architect until she retires.

Extension of propositional log with temporal connectives (eventually, until)
### Approaches for temporal logics

- **Linear-time temporal logics (LTL)**
  - Formulae with temporal operators
  - Evaluated w.r.t (infinite) sequences of valuations
  - Question of interest: Do all sequences of valuations satisfy a given LTL formula?

- **Computation-tree logic (CTL, CTL*)**
  - Considers (infinite) trees of valuations
  - Question of interest: Does this tree satisfy a given CTL formula?
  - Interpretation: non-determinism, multiple possible developments
Model checking in CTL: Example

- CTL formula: $\text{AGEF brake}$
- Interpretation: “From any state you can always reach the brake state”

Computational tree:
Model Checking in LTL Example

• LTL formula : $G(\neg (\text{brake} \land \text{accelerate}))$

• Interpretation: “It never brakes and accelerates at the same time”

LTL path:
CTL vs LTL

3.2 Verification of an MDP
Probabilistic Model checking for MDPs

- MDPs are probabilistic and non-deterministic systems
- Formally a tuple \( (S, s_{\text{init}}, \text{Steps}, L) \) where:
  - \( S \) is a finite set of states ("state space")
  - \( s_{\text{init}} \in S \) is the initial state
  - \( \text{Steps} : S \rightarrow 2^{\text{ActxDist} (S)} \) is the transition probability function, where \( \text{Act} \) is a set of actions and \( \text{Dist} (S) \) is the set of discrete probability distributions over the set \( S \)
  - \( L : S \rightarrow 2^{\text{AP}} \) is a labelling with atomic propositions
Probabilistic Model checking for MPDs in PCTL

• PCTL: Extension of CTL with a probabilistic operator

• Example
  • “if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95”

• Complexity is linear in $|\phi|$ and polynomial in $|S|$

• $S :=$ State space of MDP

• $\phi :=$ PCTL formula
Probabilistic Model checking for MPDs in pLTL

• Same concept as PCTL: probabilities of set of path formulae
• Complexity is doubly exponential in $|\psi|$ and polynomial in $|S|$
• $S := \text{State space of MDP}$
• $\psi := \text{LTL formula}$
Probabilistic Model Checking Tool: PRISM

• PRISM: Probabilistic symbolic model checker
  • Developed at Birmingham/Oxford University, since 1999
  • Free and open source

• Model checking of:
  • PCTL, pLTL ...

• Symbolic data structures
  • State explosion problem: Compact storage by exploiting regularity (using Binary decision diagrams)

• PRISM website: http://www.prismmodelchecker.org/
3.3 Verification of a Policy
Verification of a Policy

<table>
<thead>
<tr>
<th>$Q(S, A)$</th>
<th>Action 1</th>
<th>Action 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1</td>
<td>0.75</td>
<td>1.23</td>
</tr>
<tr>
<td>State 2</td>
<td>0.01</td>
<td>10</td>
</tr>
<tr>
<td>State 3</td>
<td>1.4</td>
<td>0.23</td>
</tr>
<tr>
<td>State 4</td>
<td>10</td>
<td>9.9</td>
</tr>
<tr>
<td>State 5</td>
<td>3</td>
<td>3.333</td>
</tr>
</tbody>
</table>

trivial to verify

typically large

non-linear
Verification of a Policy

\[ [x - \varepsilon_x, x + \varepsilon_x]^d \]

\[ [y - \varepsilon_y, y + \varepsilon_y]^d \]

“accelerate car”
Verification of a Neural Network

Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks

using activation function $\text{ReLU} := \max(0, x)$
Simple Neural Network

is it possible that
Input: $x_1 \in [0, 1]$ and $x_4 \in [0.5, 1]$?

find $x_1, x_2, x_3$ and $x_4$ such that

$x_1 \in [0, 1]$
$x_2 = \max(0, x_1)$
$x_3 = \max(0, -x_1)$
$x_4 = \max(0, x_2 + x_3)$
$x_4 \in [0.5, 1]$

difficult...
1. Split and Rewrite Equations

\[
\begin{align*}
    x_2^w &= x_1 \\
    x_3^w &= -x_1 \\
    x_4 &= x_2^a + x_3^a \\
    x_2^w - x_1 &= x_5 \\
    x_3^w + x_1 &= x_6 \\
    x_4 - x_2^a - x_3^a &= x_7
\end{align*}
\]
2. Find assignment for new set of equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ReLU Constraints $x_2, x_3$

\[ x_2^w - x_1 = x_5 \]
\[ x_3^w + x_1 = x_6 \]
\[ x_4 - x_2^a - x_3^a = x_7 \]
## 2. Find assignment for new set of equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ReLU Constraints $x_2, x_3$

\[
\begin{align*}
    x_2^w - x_1 &= x_5 \\
    x_3^w + x_1 &= x_6 \\
    x_4 - x_2^a - x_3^a &= x_7
\end{align*}
\]

Update

\[
\begin{align*}
    x_4 &:= x_4 + 0.5 \\
    x_7 &:= x_7 + 0.5
\end{align*}
\]
2. Find assignment for new set of equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ReLU Constraints $x_2, x_3$

$$x_2^w - x_1 = x_5$$
$$x_3^w + x_1 = x_6$$
$$x_4 - x_2^a - x_3^a = x_7$$

Pivot

$$x_7 \leftrightarrow x_2^a$$

3.3 Verification of a Policy
2. Find assignment for new set of equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$\begin{align*}
  x_2^w - x_1 &= x_5 \\
  x_3^w + x_1 &= x_6 \\
  x_4 - x_7 - x_3^a &= x_2^a
\end{align*}$

ReLU Constraints $x_2, x_3$

Update

$\begin{align*}
  x_7 &:= x_7 - 0.5 \\
  x_2^a &:= x_2^a + 0.5
\end{align*}$
2. Find assignment for new set of equations

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0.5</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ReLU Constraints $x_2, x_3$

\[
x_2^w - x_1 = x_5
\]
\[
x_3^w + x_1 = x_6
\]
\[
x_4 - x_7 - x_3^a = x_2^a
\]

Update
\[
x_2^w := x_2^w + 0.5
\]
\[
x_5 := x_5 + 0.5
\]
## Final Result

<table>
<thead>
<tr>
<th>Variable</th>
<th>Assignment</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.5</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$x_2^w$</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_2^a$</td>
<td>0.5</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^w$</td>
<td>-0.5</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$x_3^a$</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$x_7$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ReLU Constraints $x_2, x_3$

\[
x_2^w - x_1 = x_5 \\
x_3^w + x_1 = x_6 \\
x_4 - x_7 - x_3^a = x_2^a
\]

This assignment solves the equation and ReLU constraints

"$x_1 \in [0,1]$ and $x_4 \in [0.5, 1]$ is possible"
Problems

• Implementation difficult (numerical errors)

• How to define neighborhood of an image?
  • Noise?
  • Missing data?

• What about other activation functions?
  • Needs to be linear
  • Approximation
3.4 Runtime Verification
"Safe Reinforcement Learning via Formal Methods"

Abstract

Formal verification provides a high degree of confidence in the safety of critical systems, but its application to machine learning is challenging. The paper presents a novel approach to ensure the safety of reinforcement learning agents through formal verification. We leverage formal methods to construct a safe controller for a given environment, ensuring that the RL agent does not violate safety constraints.

In this paper, we propose a method for Safe Reinforcement Learning via Formal Methods. The method integrates formal verification techniques with reinforcement learning to guarantee safety properties. The paper is structured as follows:

1. Introduction: Background and motivation for the need of formal methods in reinforcement learning.
2. Safety Specifications: Formulating safety properties that the RL agent must adhere to.
3. Safe Controller Design: Construction of a safe controller that meets the specified safety properties.
4. Runtime Verification: Monitoring the RL agent's behavior to ensure compliance with safety constraints.
5. Case Studies: Application of the method to real-world scenarios.

The paper contributes to the field of safe machine learning by providing a framework that ensures the safety of learning agents through formal methods.

3.4 Runtime Verification

[8] Safe Reinforcement Learning via Formal Methods
Hybrid Program $\alpha$

Either choose $a := A$ or $a := 0$

$((a := A \cup a := 0); \{p' = v, v' = a\})^*$

Controller
Either accelerate with $A$
or break

Model
differential equations
describe behavior
Full specification $d\mathcal{L}$ formula

precondition

$$v \geq 0 \land A > 0 \rightarrow \left[ (a := A \cup a := 0); \{p' = v, v' = a\}^* \right] v \geq 0$$

postcondition

If our car moves forward with non-negative velocity after choosing actions, it still has a non-negative velocity.
Controller and Model Monitor

Formula Transformations

\[ d\mathcal{L} \rightarrow \text{ModelPlex} \]

True or False

Controller Monitor Code

True or False

Model Monitor Code

Is my action safe?

Does our Model comply with the reality?

3.4 Runtime Verification

[8] Safe Reinforcement Learning via Formal Methods
Controller Monitor Example

\[ v \geq 0 \land A > 0 \rightarrow \left( (a := A \cup a := 0); \{p' = v, v' = a\}^{*} \right) v \geq 0 \]

```
return t_post >= 0 AND
    a_post == A AND
    v_post == A*t_post + v_prev AND
    p_post == A*t_post^2 / 2 + v_prev*t_post + p_pre) OR

    t_post >= 0 AND
    a_post == 0 AND
    v_post == v_pre AND
    p_post == v_pos*t_post + p_pre
```
Algorithm

1. \texttt{JSCGeneric}(init, (S,A,R,E), choose, update, done, CM, MM) {
2. \hspace{1em} prev := curr := init;
3. \hspace{1em} while (!done(curr)) {
4. \hspace{2em} \textbf{if} (MM(prev, curr)) {
5. \hspace{3em} u := choose\{a \in A \mid CM(a, curr)\};
6. \hspace{2em} } else {
7. \hspace{3em} u := choose(A); // do something more clever (expert, ...)
8. \hspace{2em} }
9. \hspace{1em} prev := curr;
10. \hspace{1em} curr := E(u, prev);
11. \hspace{1em} update(prev, u, curr);
12. \hspace{1em} }
13. }
4. Summary and Challenges
Summary

• There are many good reasons to formally verify a system!

• Model Checking

• MDP Verification via PRISM

• Policy Verification of a Neural Network

• Runtime Verification via Hybrid Programs
Open Questions/Problems

• Models might not reflect reality, or are too simple (wrong assumptions)

• Specification might be incomplete / wrong

• Environment might change (Eg. Stickers on a traffic sign)

• Complexity of Model Checking
Questions?


Literature

• [1] Reuters, “Toyota to recall 1.9 million Prius cars for software defect in hybrid system”, 


• [5] Probabilistic Model Checking by David Parker: 
  http://www.prismmodelchecker.org/lectures/pmc/ (25.07.2018)
