Safety and Reliability in (Adaptive) Cyber-Phyisical Systems

Armando Tacchella

University of Genoa

CPS Summer School 2018

http://www.cerbero-h2020.eu/summer-school-2018/

Alghero, 17-21 September, 2018

Cyber-Physical Systems (CPSs)

Cyber-Physical Systems (CPSs)

Pretty much everyone in the audience has an idea about CPSs...

Copyright: Chesky@fotolia.com

Cyber-Physical Systems (CPSs)

Pretty much everyone in the audience has an idea about CPSs...

Copyright: Chesky@fotolia.com

...maybe it is **your own** idea, but that's ok!

CPS = Robot (for this tutorial only!)

- Research on cooperative human-robot interaction

CPS = Robot (for this tutorial only!)

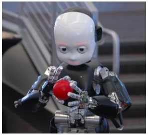
- Research on cooperative human-robot interaction
- Robots must be made **adaptable** and **safe**

CPS = Robot (for this tutorial only!)

- Research on cooperative human-robot interaction
- Robots must be made **adaptable** and **safe**
- Focus is on
  - checking requirements of **control software**
  - **learning to interact** with the environment
  - using **formal models** and techniques

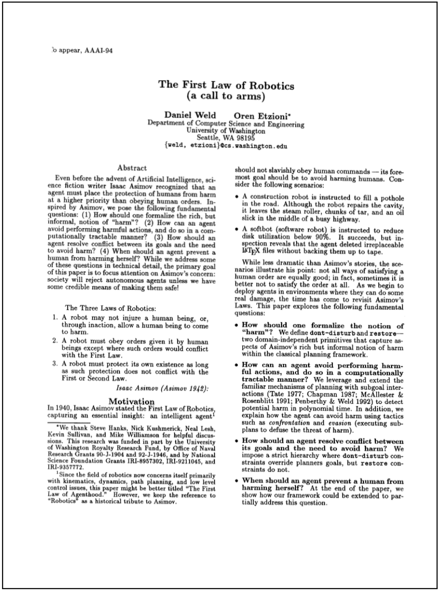
CPS = Robot (for this tutorial only!)

- Research on cooperative human-robot interaction
- Robots must be made **adaptable** and **safe**
- Focus is on
  - checking requirements of **control software**
  - **learning to interact** with the environment
  - using **formal models** and techniques



To what extent reliability and safety of (the control software in) **adaptive** CPSs can be **analyzed automatically**?

Automating analysis: Why?



'o appear, AAAI-94

The First Law of Robotics
(a call to arms)

Daniel Weld Oren Etzioni*
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{weld, etzioni}@cs.washington.edu

Abstract

Even before the advent of Artificial Intelligence, science fiction writer Isaac Asimov recognized that an agent must place the protection of humans from harm at a higher priority than obeying human orders. Inspired by Asimov, we pose the following fundamental questions: (1) How should one formalize the rich, but informal, notion of "harm"? (2) How can an agent avoid performing harmful actions, and do so in a computationally tractable manner? (3) How should an agent resolve conflict between its goals and the need to avoid harm? (4) When should an agent prevent a human from harming herself? While we address some of these questions in technical detail, the primary goal of this paper is to focus attention on Asimov's concern: society will reject autonomous agents unless we have some credible means of making them safe!

The Three Laws of Robotics:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Isaac Asimov (Asimov 1942):

Motivation

In 1940, Isaac Asimov stated the First Law of Robotics, capturing an essential insight: an intelligent agent[1] should not slavishly obey human commands — its foremost goal should be to avoid harming humans. Consider the following scenarios:

- A construction robot is instructed to fill a pothole in the road. Although the robot repairs the cavity, it leaves the steam roller, chunks of tar, and an oil slick in the middle of a busy highway.
- A softbot (software robot) is instructed to reduce disk utilization below 90%. It succeeds, but inspection reveals that the agent deleted irreplaceable LaTeX files without backing them up to tape.

While less dramatic than Asimov's stories, the scenarios illustrate his point: not all ways of satisfying a human order are equally good; in fact, sometimes it is better not to satisfy the order at all. As we begin to deploy agents in environments where they can do some real damage, the time has come to revisit Asimov's Laws. This paper explores the following fundamental questions:

- **How should one formalize the notion of "harm"?** We define `dont-disturb` and `restore`— two domain-independent primitives that capture aspects of Asimov's rich but informal notion of harm within the classical planning framework.
- **How can an agent avoid performing harmful actions, and do so in a computationally tractable manner?** We leverage and extend the familiar mechanisms of planning with subgoal interactions (Tate 1977; Chapman 1987; McAllester & Rosenblitt 1991; Penberthy & Weld 1992) to detect potential harm in polynomial time. In addition, we explain how the agent can avoid harm using tactics such as *confrontation* and *evasion* (executing subplans to defuse the threat of harm).
- **How should an agent resolve conflict between its goals and the need to avoid harm?** We impose a strict hierarchy where `dont-disturb` constraints override planners goals, but `restore` constraints do not.
- **When should an agent prevent a human from harming herself?** At the end of the paper, we show how our framework could be extended to partially address this question.

*We thank Steve Hanks, Nick Kushmerick, Neal Lesh, Kevin Sullivan, and Mike Williamson for helpful discussions. This research was funded in part by the University of Washington Royalty Research Fund, by Office of Naval Research Grants 90-J-1904 and 92-J-1946, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772.

[1] Since the field of robotics now concerns itself primarily with kinematics, dynamics, path planning, and low level control issues, this paper might be better titled "The First Law of Agenthood." However, we keep the reference to "Robotics" as a historical tribute to Asimov.

The First Law of Robotics
[Asimov, 1940]
"A robot may not injure a human being, or, through inaction, allow a human being to come to harm."

Automating analysis: Why?

'o appear, AAAI-94

The First Law of Robotics
(a call to arms)

Daniel Weld **Oren Etzioni***
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{weld, etzioni}@cs.washington.edu

Abstract

Even before the advent of Artificial Intelligence, science fiction writer Isaac Asimov recognized that an agent must place the protection of humans from harm at a higher priority than obeying human orders. Inspired by Asimov, we pose the following fundamental questions: (1) How should one formalize the rich, but informal, notion of "harm"? (2) How can an agent avoid performing harmful actions, and do so in a computationally tractable manner? (3) How should an agent resolve conflict between its goals and the need to avoid harm? (4) When should an agent prevent a human from harming herself? While we address some of these questions in technical detail, the primary goal of this paper is to focus attention on Asimov's concern: society will reject autonomous agents unless we have some credible means of making them safe!

The Three Laws of Robotics:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Isaac Asimov (Asimov 1942):

Motivation

In 1940, Isaac Asimov stated the First Law of Robotics, capturing an essential insight: an intelligent agent[1] should not slavishly obey human commands — its foremost goal should be to avoid harming humans. Consider the following scenarios:

- A construction robot is instructed to fill a pothole in the road. Although the robot repairs the cavity, it leaves the steam roller, chunks of tar, and an oil slick in the middle of a busy highway.
- A softbot (software robot) is instructed to reduce disk utilization below 90%. It succeeds, but inspection reveals that the agent deleted irreplaceable LaTeX files without backing them up to tape.

While less dramatic than Asimov's stories, the scenarios illustrate his point: not all ways of satisfying a human order are equally good; in fact, sometimes it is better not to satisfy the order at all. As we begin to deploy agents in environments where they can do some real damage, the time has come to revisit Asimov's Laws. This paper explores the following fundamental questions:

- **How should one formalize the notion of "harm"?** We define `dont-disturb` and `restore`— two domain-independent primitives that capture aspects of Asimov's rich but informal notion of harm within the classical planning framework.
- **How can an agent avoid performing harmful actions, and do so in a computationally tractable manner?** We leverage and extend the familiar mechanisms of planning with subgoal interactions (Tate 1977; Chapman 1987; McAllester & Rosenblitt 1991; Penberthy & Weld 1992) to detect potential harm in polynomial time. In addition, we explain how the agent can avoid harm using tactics such as *confrontation* and *evasion* (executing subplans to defuse the threat of harm).
- **How should an agent resolve conflict between its goals and the need to avoid harm?** We impose a strict hierarchy where `dont-disturb` constraints override planners goals, but `restore` constraints do not.
- **When should an agent prevent a human from harming herself?** At the end of the paper, we show how our framework could be extended to partially address this question.

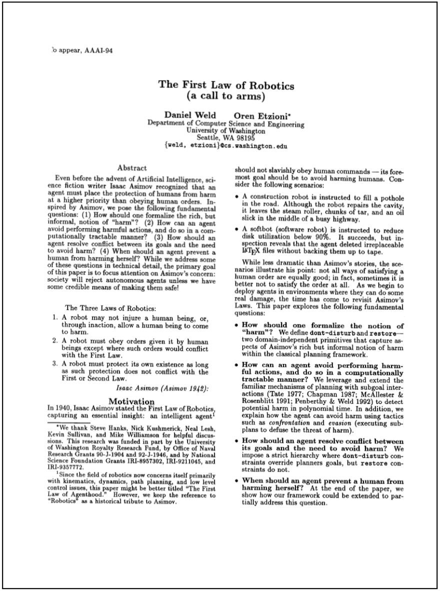
*We thank Steve Hanks, Nick Kushmerick, Neal Lesh, Kevin Sullivan, and Mike Williamson for helpful discussions. This research was funded in part by the University of Washington Royalty Research Fund, by Office of Naval Research Grants 90-J-1904 and 92-J-1946, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772.

[1] Since the field of robotics now concerns itself primarily with kinematics, dynamics, path planning, and low level control issues, this paper might be better titled "The First Law of Agenthood." However, we keep the reference to "Robotics" as a historical tribute to Asimov.

The First Law of Robotics
[Asimov, 1940]
"A robot may not injure a human being, or, through inaction, allow a human being to come to harm."

"...before we release autonomous agents into real-world environments, we need some credible and computationally tractable means of making them obey Asimov's First Law."

Automating analysis: Why?



`o appear, AAAI-94

The First Law of Robotics
(a call to arms)

Daniel Weld **Oren Etzioni***
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{weld, etzioni}@cs.washington.edu

Abstract

Even before the advent of Artificial Intelligence, science fiction writer Isaac Asimov recognized that an agent must place the protection of humans from harm at a higher priority than obeying human orders. Inspired by Asimov, we pose the following fundamental questions: (1) How should one formalize the rich, but informal, notion of "harm"? (2) How can an agent avoid performing harmful actions, and do so in a computationally tractable manner? (3) How should an agent resolve conflict between its goals and the need to avoid harm? (4) When should an agent prevent a human from harming herself? While we address some of these questions in technical detail, the primary goal of this paper is to focus attention on Asimov's concern: society will reject autonomous agents unless we have some credible means of making them safe!

The Three Laws of Robotics:

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Isaac Asimov (Asimov 1942):

Motivation

In 1940, Isaac Asimov stated the First Law of Robotics, capturing an essential insight: an intelligent agent[1] should not slavishly obey human commands — its foremost goal should be to avoid harming humans. Consider the following scenarios:

- A construction robot is instructed to fill a pothole in the road. Although the robot repairs the cavity, it leaves the steam roller, chunks of tar, and an oil slick in the middle of a busy highway.
- A softbot (software robot) is instructed to reduce disk utilization below 90%. It succeeds, but inspection reveals that the agent deleted irreplaceable LaTeX files without backing them up to tape.

While less dramatic than Asimov's stories, the scenarios illustrate his point: not all ways of satisfying a human order are equally good; in fact, sometimes it is better not to satisfy the order at all. As we begin to deploy agents in environments where they can do some real damage, the time has come to revisit Asimov's Laws. This paper explores the following fundamental questions:

- **How should one formalize the notion of "harm"?** We define `dont-disturb` and `restore`— two domain-independent primitives that capture aspects of Asimov's rich but informal notion of harm within the classical planning framework.
- **How can an agent avoid performing harmful actions, and do so in a computationally tractable manner?** We leverage and extend the familiar mechanisms of planning with subgoal interactions (Tate 1977; Chapman 1987; McAllester & Rosenblitt 1991; Penberthy & Weld 1992) to detect potential harm in polynomial time. In addition, we explain how the agent can avoid harm using tactics such as *confrontation* and *evasion* (executing subplans to defuse the threat of harm).
- **How should an agent resolve conflict between its goals and the need to avoid harm?** We impose a strict hierarchy where `dont-disturb` constraints override planners goals, but `restore` constraints do not.
- **When should an agent prevent a human from harming herself?** At the end of the paper, we show how our framework could be extended to partially address this question.

*We thank Steve Hanks, Nick Kushmerick, Neal Lesh, Kevin Sullivan, and Mike Williamson for helpful discussions. This research was funded in part by the University of Washington Royalty Research Fund, by Office of Naval Research Grants 90-J-1904 and 92-J-1946, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772.

[1]Since the field of robotics now concerns itself primarily with kinematics, dynamics, path planning, and low level control issues, this paper might be better titled "The First Law of Agenthood." However, we keep the reference to "Robotics" as a historical tribute to Asimov.

The First Law of Robotics
[Asimov, 1940]
"A robot may not injure a human being, or, through inaction, allow a human being to come to harm."

"...before we release autonomous agents into real-world environments, we need some credible and computationally tractable means of making them obey Asimov's First Law."

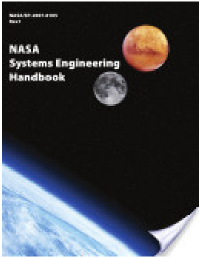
*"Given a complex world where the agent does not have complete information, **any** attempt to formalize the second half of Asimov's First Law is fraught with difficulties."*

The bigger picture: RAMSS

- **Reliability**: ability to perform required functions under stated conditions for a specified period of time
- **Availability**: proportion of time a system is in a functioning condition
- **Maintainability**: probability that a system will be retained in or restored to a specified condition within a given period of time
- **Safety**: ability to control recognized hazards to achieve acceptable level of risk
- **Security**: degree of resistance to, or protection from system damage

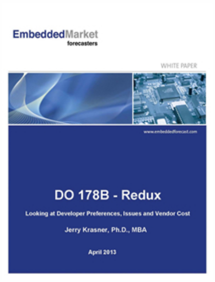
What about “off-the-shelf” engineering?

Safety is widely recognized as a design objective in complex systems



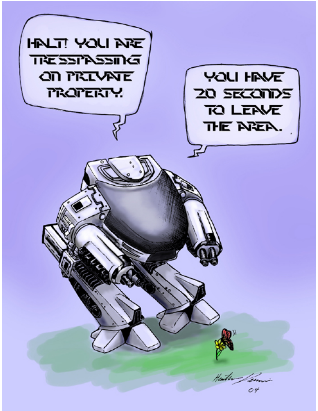
Methodologies

Standards



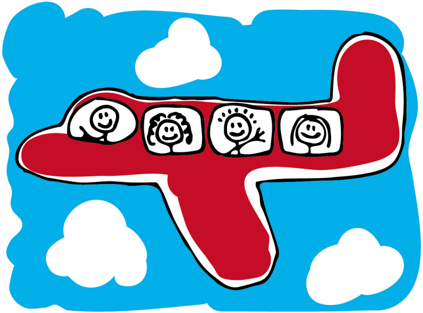
Guidelines

Adaptive robots are not, e.g., planes...



ED 209 shows a reliability defect, leading to potential safety defects

vs.



Planes are dependable, but we do not expect them to operate autonomously (if they did, they would be UAVs)

... still, they need to be certified

DRAFT INTERNATIONAL STANDARD ISO/DIS 13482

ISO/TC **184**/SC **2** Secretariat: **SIS**

Voting begins on **2011-09-08** Voting terminates on **2012-02-08**

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

Robots and robotic devices — Safety requirements for non-industrial robots — Non-medical personal care robot

Robots et composants robotiques — Exigences de sécurité — Robots non médicaux pour les soins personnels

ICS 25.040.30

ISO/CEN PARALLEL PROCESSING

This draft has been developed within the International Organization for Standardization (ISO), and processed under the **ISO-lead** mode of collaboration as defined in the Vienna Agreement.

This draft is hereby submitted to the ISO member bodies and to the CEN member bodies for a parallel five-month enquiry.

Should this draft be accepted, a final draft, established on the basis of comments received, will be submitted to a parallel two-month approval vote in ISO and formal vote in CEN.

To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.

Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

- ISO 13482:2014
- Safety requirements for Non-industrial robots
- Non-medical personal care robots
- Makes provision for safe autonomous actions
- Autonomy = adaptivity: autonomous **evaluative decisions** taken by the robot that might use **cognitive models not built in** at factory.

Safe (and reliable) agents: How?

Safe (and reliable) agents: How?

- **Intrinsic safety**: it is not possible to model an unsafe agent (Unlikely)

Safe (and reliable) agents: How?

- **Intrinsic safety**: it is not possible to model an unsafe agent (Unlikely)
- **Safety by construction**: the agent will be safe as long as specific design guidelines are strictly observed (Staple method in engineering)

Safe (and reliable) agents: How?

- **Intrinsic safety**: it is not possible to model an unsafe agent (Unlikely)
- **Safety by construction**: the agent will be safe as long as specific design guidelines are strictly observed (Staple method in engineering)
- **Demonstrable safety**: it can be proved that the agent design reduces undesirable events to an acceptable level (This tutorial!)

Safe (and reliable) agents: How?

- **Intrinsic safety**: it is not possible to model an unsafe agent (Unlikely)
- **Safety by construction**: the agent will be safe as long as specific design guidelines are strictly observed (Staple method in engineering)
- **Demonstrable safety**: it can be proved that the agent design reduces undesirable events to an acceptable level (This tutorial!)
- **Monitorable safety**: it can be ensured that the agent recognizes actions leading to undesirable events (Hardly disposable, will touch upon it)

Agenda

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Our contribution

Given a (specific kind of) neural network ν and a (safety) specification s

Network Abstraction

1. Find an abstraction α
2. If $\nu \models_\alpha s$ then STOP: ν is safe
3. Otherwise, refine α and go back to step (2)

Challenge: Find/refine α

Our contribution

Given a (specific kind of) neural network ν and a (safety) specification *s*

Network Abstraction

1. Find an abstraction α
2. If $\nu \models_\alpha s$ then STOP: ν is safe
3. Otherwise, refine α and go back to step (2)

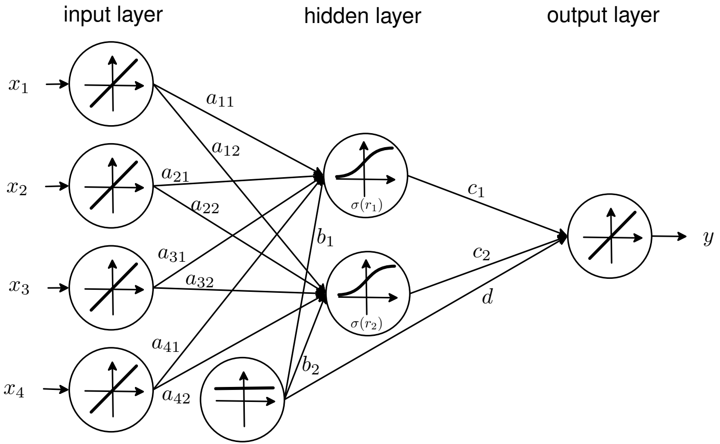
Challenge: Find/refine α

Network Repair

1. Given an abstraction α
2. If $\nu \models_\alpha s$ then STOP: ν is safe
3. Otherwise, modify ν and go back to step (2)

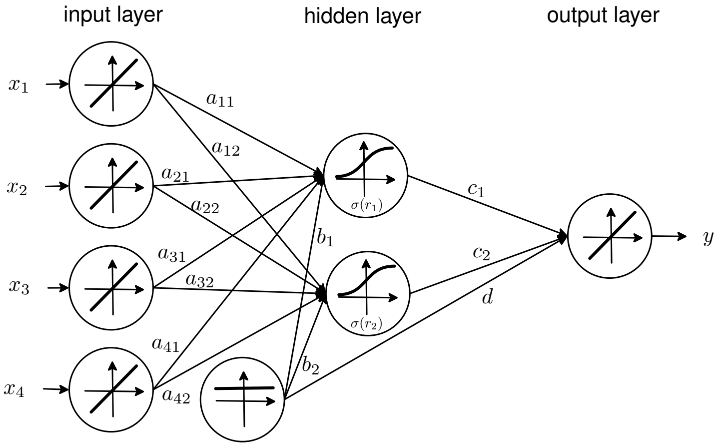
Challenge: Modify ν automatically

Single hidden-layer MLP



- Input to the j-th hidden neuron (n inputs): $r_j = \sum_{i=1}^{n} a_{ji}x_i + b_j$
- Hidden neurons driven by **logistic function**: $\sigma(r) = \frac{1}{1+\exp(-r)}$
- Output (m hidden neurons): $y = \sum_{j=1}^{m} c_j\sigma(r_j) + d$

Single hidden-layer MLP



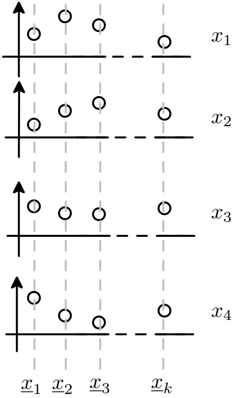
Universal approximation theorem

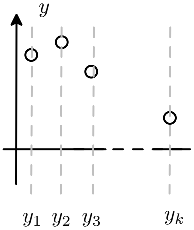
Single hidden-layer MLPs featuring "smooth" hidden-neuron functions can in principle approximate any function $f : \mathbb{R}^n \to \mathbb{R}$.

MLPs are (straight line) programs

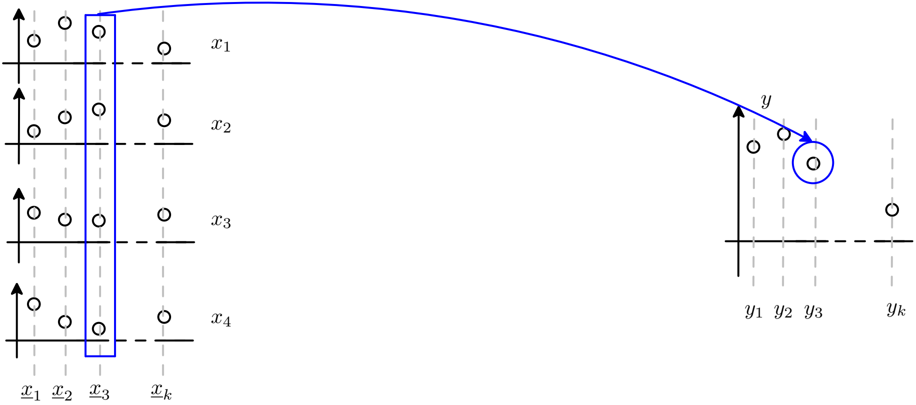
```
const int n = ... // input signals
const int m = ... // hidden nodes (single layer)

const real a[n][m] = { ... }; // weights for input connections
const real b[m] = { ... }; // weights for bias node
const real c[m] = { ... }; // weights for output connections
const real d = ... ;

real network(real x[n]) {
  i = 1; j = 1; y = 0;
  while (j <= m) {
     real r = 0;
     while (i <= n) {
        r = r + a[i][j] * x[i] + b[j];
        ++i;
      }
      y = y + c[j] * (1 / (1 + exp(-r)));
      ++j;
   }
   y = y + d;
   return y;
}
```

The task of MLP synthesis

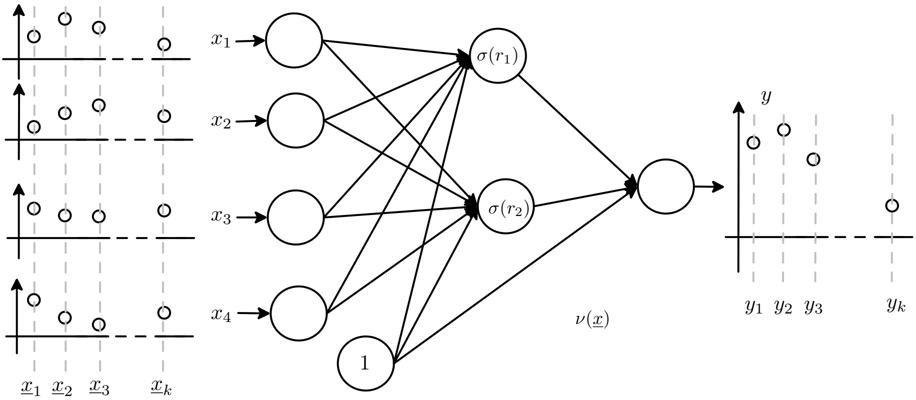




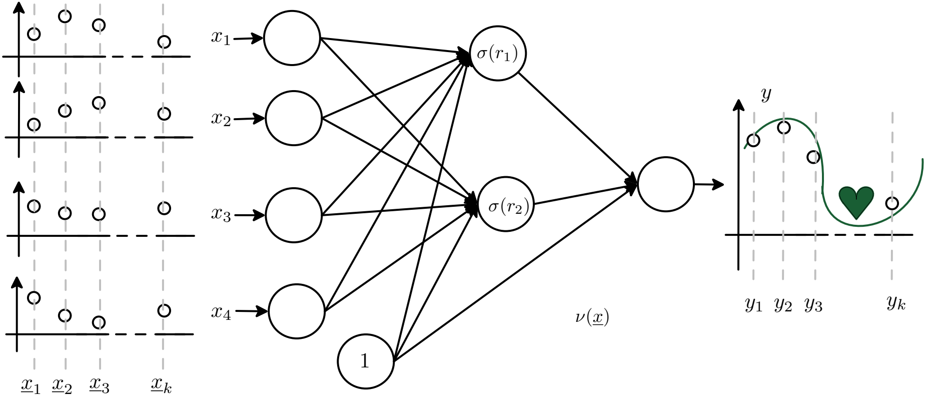
The task of MLP synthesis



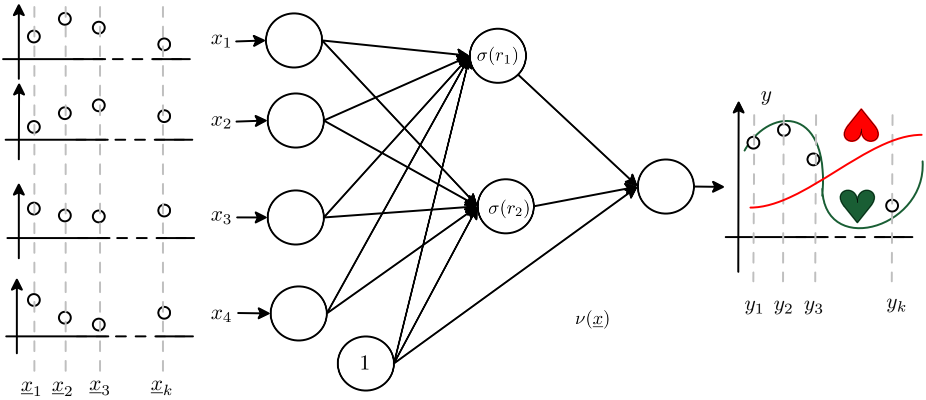
The task of MLP synthesis



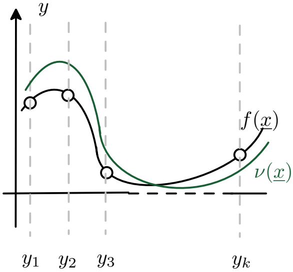
The task of MLP synthesis



The task of MLP synthesis



How good is your MLP?

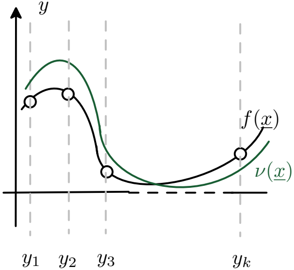


- Easy to know on the dataset, e.g.,

$$\hat{\epsilon} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y_i - \nu(\underline{x}_i))^2} \quad \text{RMSE}$$

- How good is ν in generalizing to f, e.g., $\epsilon = ||f(\underline{x}) - \nu(\underline{x})||$? $\Rightarrow$ f is **unknown**!

How good is your MLP?



- Easy to know on the dataset, e.g.,

$$\hat{\epsilon} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y_i - \nu(\underline{x}_i))^2} \quad \text{RMSE}$$

- How good is ν in generalizing to f, e.g., $\epsilon = ||f(\underline{x}) - \nu(\underline{x})||$? $\Rightarrow$ f is **unknown**!

Leave-one-out estimation of generalization error

1. Given input patterns X and labels Y, we synthesize the MLP $\nu_{(i)}$ considering $X_{(i)} = \{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_k\}$ and corresponding $Y_{(i)}$.
2. Repeat (2) for k times, to obtain k different MLPs.
3. Compute RMSE as follows

$$\hat{\epsilon} = \sqrt{\frac{1}{k}\sum_{i=1}^{k}(y_i - \nu_{(i)}(\underline{x}_i))^2}$$

Safety for MLPs: a proposal

Network ν as a function $\nu : \mathcal{I} \to \mathcal{O}$ where

- $\mathcal{I} = D_1 \times \ldots \times D_n$ is the **input domain** and each $D_i = [a_i, b_i]$ is a closed interval with $a_i, b_i \in \mathbb{R}$ and $a_i \leq b_i$.
- $\mathcal{O}$ is the **output domain**, a closed interval in $\mathbb{R}$.
- Define **safety thresholds** $l, h \in \mathcal{O}$ with $l < h$.
- Require output of ν to range within $[l, h]$ for all acceptable inputs.

Safety for MLPs: a proposal

Network ν as a function $\nu : \mathcal{I} \rightarrow \mathcal{O}$ where

- $\mathcal{I} = D_1 \times \ldots \times D_n$ is the **input domain** and each $D_i = [a_i, b_i]$ is a closed interval with $a_i, b_i \in \mathbb{R}$ and $a_i \leq b_i$.
- $\mathcal{O}$ is the **output domain**, a closed interval in $\mathbb{R}$.
- Define **safety thresholds** $l, h \in \mathcal{O}$ with $l < h$.
- Require output of ν to range within $[l, h]$ for all acceptable inputs.

A network $\nu : \mathcal{I} \rightarrow \mathcal{O}$ is **safe** when it satisfies the property

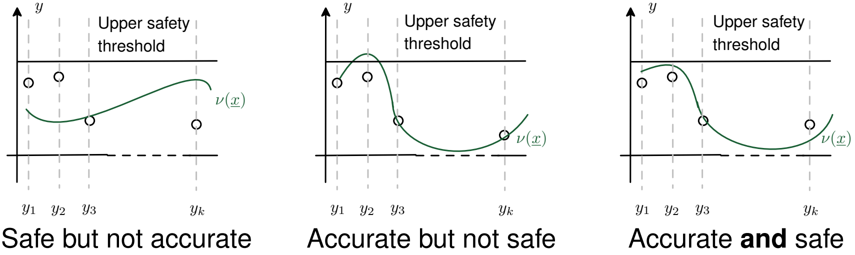
$$\forall \underline{x} \in \mathcal{I} : \nu(\underline{x}) \in [l, h] \text{ with } l, h \in \mathcal{O}$$

Safety vs. accuracy

- Training and validation methods assume **i.i.d. samples**
- In practice, we do not know whether this is the case
  $\Rightarrow$ we may loose even **statistical guarantees**
- MLPs are fairly robust w.r.t. failure of i.i.d. assumption
  $\Rightarrow$ we still need to **avoid misbehaviors**

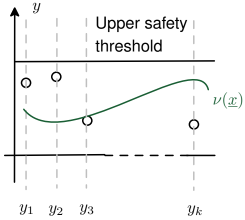
Safety vs. accuracy

- Training and validation methods assume **i.i.d. samples**
- In practice, we do not know whether this is the case
  $\Rightarrow$ we may loose even **statistical guarantees**
- MLPs are fairly robust w.r.t. failure of i.i.d. assumption
  $\Rightarrow$ we still need to **avoid misbehaviors**

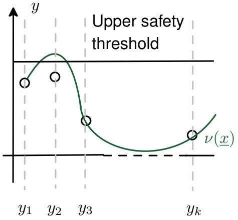


Safety vs. accuracy

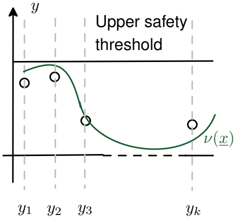
- Training and validation methods assume **i.i.d. samples**
- In practice, we do not know whether this is the case
  $\Rightarrow$ we may loose even **statistical guarantees**
- MLPs are fairly robust w.r.t. failure of i.i.d. assumption
  $\Rightarrow$ we still need to **avoid misbehaviors**



Safe but not accurate



Accurate but not safe



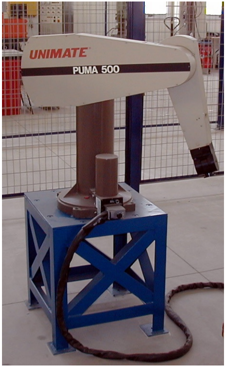
Accurate **and** safe

Estimated accuracy and safety do not imply each other!

Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Learning forward kynematics of a PUMA 500



PUMA 500
Industrial 6 DoF
manipulator

Task

Learn to control the end-effector position along a straight line using the motor angles as input.

- Dataset (141 patterns)
  - input vectors $\underline{x} = \langle \theta_1, \ldots, \theta_6 \rangle$ encoding 6 joint angles (in radians)
  - output labels y corresponding to end-effector coordinates (in meters)
- **Safe range** for y is $[-0.35, 0.35]$
- Synthesis summary
  - **training**: 0.64s; **error**: $\hat{\epsilon} = 0.024$m (RMSE)
  - **error distribution**: ranges from 3.2×10^{-5}m (min) to 0.123m (max), median value of 0.020m.

Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

How to check safety?

- Testing exhaustively all the input vectors? **Untenable!**

How to check safety?

- Testing exhaustively all the input vectors? **Untenable!**
- Sampling input vectors? **Only probabilistic guarantees.**

How to check safety?

- Testing exhaustively all the input vectors? **Untenable!**
- Sampling input vectors? **Only probabilistic guarantees.**
- From a **formal methods** standpoint:
  - Neural networks are combination of real-valued non-linear and trascendental functions $\Rightarrow$ undecidable theories!
  - Rational approximations of real numbers? $\Rightarrow$ still too cumbersome!

How to check safety?

- Testing exhaustively all the input vectors? **Untenable!**
- Sampling input vectors? **Only probabilistic guarantees.**
- From a **formal methods** standpoint:
  - Neural networks are combination of real-valued non-linear and trascendental functions $\Rightarrow$ undecidable theories!
  - Rational approximations of real numbers? $\Rightarrow$ still too cumbersome!

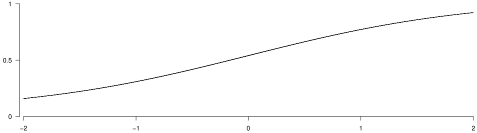
An approach based on abstract interpretation

- A concrete network ν is a function $\nu : \mathbb{R}^n \to \mathbb{R}$
- **Sound abstractions** can be obtained via **interval arithmetics**
- Abstract networks are functions $\tilde{\nu} : [\mathbb{R}]^n \to [\mathbb{R}]$ encoded as **Boolean combinations** of **linear** constraints

$\Rightarrow$ **Key point**: abstracting hidden layer neurons!

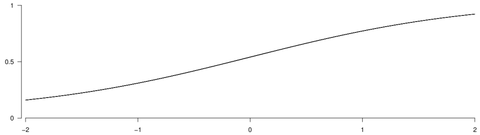
Abstracting hidden-layer neurons

Logistic function $\sigma : \mathbb{R} \to (0, 1)$

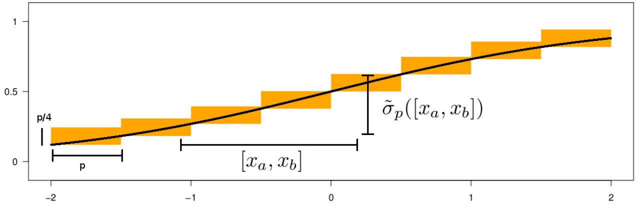


Abstracting hidden-layer neurons

Logistic function $\sigma : \mathbb{R} \to (0,1)$

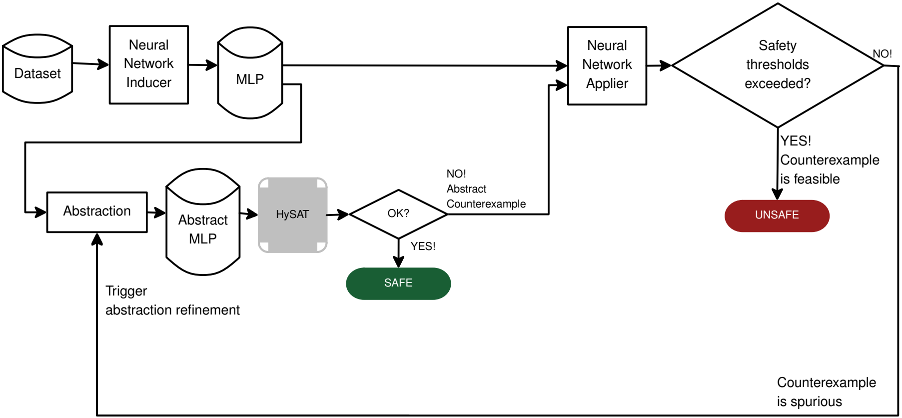


Abstract logistic function $\tilde{\sigma}_p : [\mathbb{R}] \to [[0,1]]$ ($p \in \mathbb{R}^+$)



Height of "staircase steps" $\Rightarrow$ maximum slope of tangent to σ ($p/4$)

Abstraction/Refinement loop



Abstraction is refined by using smaller and smaller values of p
Counterexample **Triggered** Abstraction Refinement (CETAR)

Results on the PUMA case study

l	h	RESULT	# CETAR	TIME (S)	
				TOTAL	HYSAT
-0.350	0.350	UNSAFE	8	1.95	1.01
-0.450	0.450	UNSAFE	9	3.15	2.10
-0.550	0.550	UNSAFE	12	6.87	5.66
-0.575	0.575	SAFE	11	6.16	4.99
-0.600	0.600	SAFE	1	0.79	0.12
-0.650	0.650	SAFE	1	0.80	0.13

- "l" and "h"lower and upper safety thresholds, resp.
- "# CETAR" indicates number of abstraction-refinement loops.
- "TIME" is total CPU time and the time spent by HYSAT.

Why repair?

- The bounds in which we guarantee safety are not satisfactory: 64% **larger** than the desired ones.
- Can we do better?

Why repair?

- The bounds in which we guarantee safety are not satisfactory: 64% **larger** than the desired ones.
- Can we do better?
- **Observation**: spurious counterexamples are weak points in the abstract network, close-to-weak points in the concrete one.

Why repair?

- The bounds in which we guarantee safety are not satisfactory: 64% **larger** than the desired ones.
- Can we do better?
- **Observation**: spurious counterexamples are weak points in the abstract network, close-to-weak points in the concrete one.
- **Idea**: repair the network by adding spurious counterexamples to the dataset and retraining.

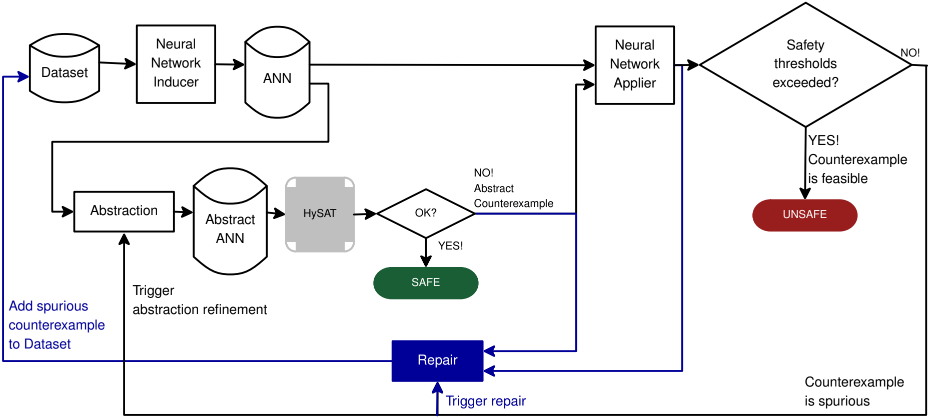
Why repair?

- The bounds in which we guarantee safety are not satisfactory: 64% **larger** than the desired ones.
- Can we do better?
- **Observation**: spurious counterexamples are weak points in the abstract network, close-to-weak points in the concrete one.
- **Idea**: repair the network by adding spurious counterexamples to the dataset and retraining.

Main points

- In practice, we do not have access to the **true** response corresponding to spurious counterexamples **inputs.**
- We use the **concrete network** response as an **approximation.**
- In our experiments, overfit is not an issue.

Abstraction/Refinement and Repair

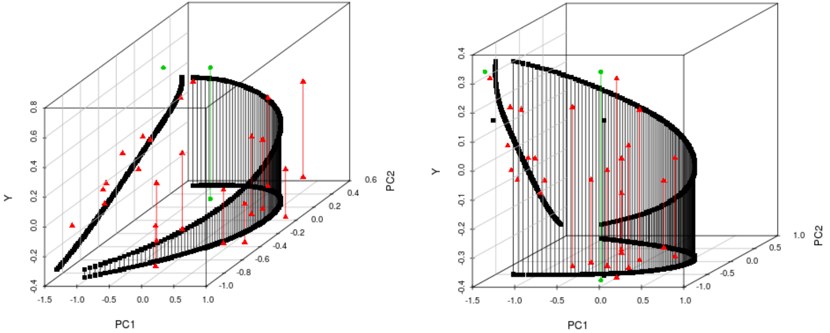


Results adding repair on the PUMA dataset

l	h	RESULT	# CETAR	TIME (S)		
				TOTAL	MLP	HYSAT
-0.350	0.350	UNSAFE	11	9.50	7.31	1.65
-0.400	0.400	UNSAFE	7	6.74	4.68	1.81
-0.425	0.425	UNSAFE	13	24.93	8.74	1.52
-0.450	0.450	SAFE	3	3.11	1.92	1.10

- "l" and "h"lower and upper safety thresholds, resp.
- "# CETAR" indicates number of abstraction-refinement loops.
- "TIME" is total CPU time including time spent to retrain the network (MLP), and to invoke HYSAT.

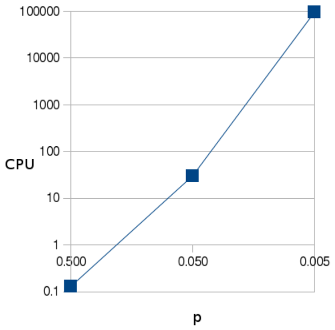
Why repair works?

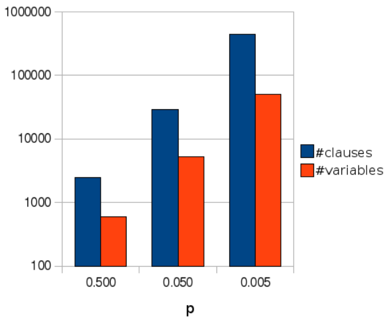


- Start from tightest SAFE interval $[-0.575, 0.575]$
- Use **true responses** on spurious inputs $\Rightarrow$ **Manual repair**
- First spurious cex (left) enables us to close at $[-0.4, 0.4]$.
- Second spurious cex (right) enables us to reach $[-0.355, 0.355]$!
- Random input vectors (control) $\Rightarrow$ no consistent improvements.

Why not using the most precise abstraction up front?

- Consider the range $[-0.65, 0.65]$
- **Baseline**: $p = 0.5$, network declared SAFE in 0.13s
- $10\times$ **decrease** in p (more and more precise abstractions)





- At least $100\times$ **increase** in CPU time (and growing)
- Size of the encoding grows proportionately

Will a retrained MLP maintain safety?

Only if MLP is retrained adding “right” patterns

- Spurious counterexamples ⇒ **improvement**!
- Randomly generated input patterns ⇒ **mixed results**

#	l	h
1	-0.46	0.46
2	-0.51	0.51
3	-0.50	0.50
4	-0.46	0.46
5	-0.48	0.48
6	-0.54	0.54
7	-0.55	0.55
8	-0.53	0.53
9	-0.59	0.59
10	-0.54	0.54

Manual repair - 1st round (was [-0.575, 0.575])

#	l	h
1	-0.43	0.43
2	-0.55	0.55
3	-0.46	0.46
4	-0.40	0.40
5	-0.39	0.39
6	-0.39	0.39
7	-0.40	0.40
8	-0.48	0.48
9	-0.51	0.51
10	-0.44	0.44

Manual repair - 2nd round (was [-0.4, 0.4])

Further extensions

- Are we limited to checking

$$\forall \underline{x} \in \mathcal{I} : \nu(\underline{x}) \in [l, h] \text{ with } l, h \in \mathcal{O}?$$

- Are we limited to (single-layer) MLPs?

More interesting (and challenging) properties

MLP $\nu : \mathcal{I} \rightarrow \mathcal{O}$ trained on a dataset *R* of *t* patterns

More interesting (and challenging) properties

MLP $\nu : \mathcal{I} \rightarrow \mathcal{O}$ trained on a dataset R of t patterns

Local safety

Given an input pattern $\underline{x}^* \neq \underline{x}$ for all $(\underline{x}, \underline{y}) \in R$ is it the case that $\nu(\underline{x}^*)$ is "close" to $\underline{y}_j$ as long as $\underline{x}^*$ is "close" to $\underline{x}_j$ and $(\underline{x}_j, \underline{y}_j) \in R$ for some $j \in \{1, \ldots, t\}$?

More interesting (and challenging) properties

MLP $\nu : \mathcal{I} \rightarrow \mathcal{O}$ trained on a dataset R of t patterns

Local safety

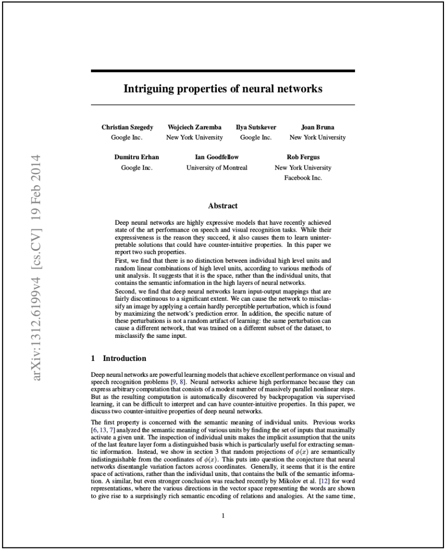
Given an input pattern $\underline{x}^* \neq \underline{x}$ for all $(\underline{x}, \underline{y}) \in R$ is it the case that $\nu(\underline{x}^*)$ is "close" to $\underline{y}_j$ as long as $\underline{x}^*$ is "close" to $\underline{x}_j$ and $(\underline{x}_j, \underline{y}_j) \in R$ for some $j \in \{1, \ldots, t\}$?

Sensitivity

Given thresholds $\delta, \epsilon \in \mathbb{R}^+$ is it the case that

$$\forall \underline{x}_1, \underline{x}_2 \in \mathcal{I} : ||\underline{x}_1 - \underline{x}_2|| \leq \delta \rightarrow ||\nu(\underline{x}_1) - \nu(\underline{x}_2)|| \leq \epsilon?$$

Are these questions interesting for ML people?



arXiv:1312.6199v4 [cs.CV] 19 Feb 2014

Intriguing properties of neural networks

Christian Szegedy Google Inc.
Wojciech Zaremba New York University
Ilya Sutskever Google Inc.
Joan Bruna New York University
Dumitru Erhan Google Inc.
Ian Goodfellow University of Montreal
Rob Fergus New York University, Facebook Inc.

Abstract

Deep neural networks are highly expressive models that have recently achieved state of the art performance on speech and visual recognition tasks. While their expressiveness is the reason they succeed, it also causes them to learn uninterpretable solutions that could have counter-intuitive properties. In this paper we report two such properties.

First, we find that there is no distinction between individual high level units and random linear combinations of high level units, according to various methods of unit analysis. It suggests that it is the space, rather than the individual units, that contains the semantic information in the high layers of neural networks.

Second, we find that deep neural networks learn input-output mappings that are fairly discontinuous to a significant extent. We can cause the network to misclassify an image by applying a certain hardly perceptible perturbation, which is found by maximizing the network's prediction error. In addition, the specific nature of these perturbations is not a random artifact of learning: the same perturbation can cause a different network, that was trained on a different subset of the dataset, to misclassify the same input.

1 Introduction

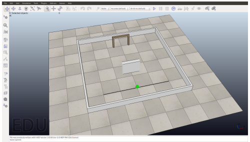
Deep neural networks are powerful learning models that achieve excellent performance on visual and speech recognition problems [9, 8]. Neural networks achieve high performance because they can express arbitrary computation that consists of a modest number of massively parallel nonlinear steps. But as the resulting computation is automatically discovered by backpropagation via supervised learning, it can be difficult to interpret and can have counter-intuitive properties. In this paper, we discuss two counter-intuitive properties of deep neural networks.

The first property is concerned with the semantic meaning of individual units. Previous works [6, 13, 7] analyzed the semantic meaning of various units by finding the set of inputs that maximally activate a given unit. The inspection of individual units makes the implicit assumption that the units of the last feature layer form a distinguished basis which is particularly useful for extracting semantic information. Instead, we show in section 3 that random projections of $\phi(x)$ are semantically indistinguishable from the coordinates of $\phi(x)$. This puts into question the conjecture that neural networks disentangle variation factors across coordinates. Generally, it seems that it is the entire space of activations, rather than the individual units, that contains the bulk of the semantic information. A similar, but even stronger conclusion was reached recently by Mikolov et al. [12] for word representations, where the various directions in the vector space representing the words are shown to give rise to a surprisingly rich semantic encoding of relations and analogies. At the same time,

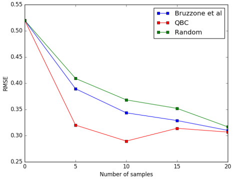
1

- **Yes!** (Somewhat surprisingly...)
- Deep networks can have **large output deviations** given **limited input noise**
- Noise is **physically realizable** and **does not** disturb humans!

Different learning machines



$\Longrightarrow$

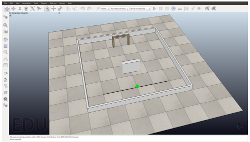


From domain interaction...

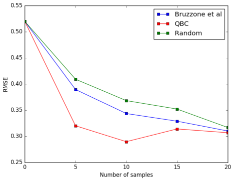
... infer automatically ... (learn)

... models as **kernel machines**.

Different learning machines



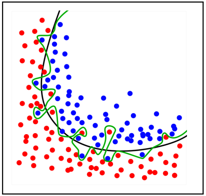
$\Rightarrow$



From domain interaction...

... infer automatically ... (learn)

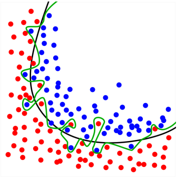
... models as **kernel machines**.



Kernel machines are funny beasts!

- Statistical guarantees only (at best)
- $\mathbb{R} \to \mathbb{R}$ functions $\Rightarrow$ no (easy) verification algos

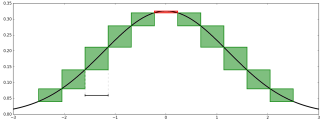
Different learning machines (cont.d)



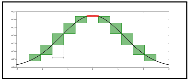
From concrete machines...

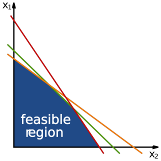
$\Longrightarrow$

... extract ... (automatically)



... conservative abstractions.





Abstractions can be model checked!

- Quantifier-Free Linear Arithmetic over $\mathbb{R}$
- Concrete machine is safe if abstract one is safe

Critiques and recent related works

CETAR approach of Pulina-Tacchella [CAV 2010]

- **Pros**: widely applicable, sound, effective (repair)
- **Cons**: hardly scalable to “monster” networks

Critiques and recent related works

CETAR approach of Pulina-Tacchella [CAV 2010]

- **Pros**: widely applicable, sound, effective (repair)
- **Cons**: hardly scalable to “monster” networks

Recent attempts to tackle “deep” networks

- X. Huang, M. Kwiatkowska, S. Wang, M. Wu - *Safety Verification of Deep Neural Networks* - Invited paper at CAV 2017
- G. Katz, C. Barrett, D. Dill, K. Julian, M. Kochederfer - *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks* - CAV 2017
- R. Ehlers - *Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks* - Published on arXiv

If you want to know more...

**Automated Verification of Neural Networks:
Advances, Challenges and Perspectives**

Francesco Leofante[1,4], **Nina Narodytska**[2], **Luca Pulina**[3], **Armando Tacchella**[1]
[1] University of Genoa , [2] VMware Research
[3] University of Sassari, [4] RWTH Aachen University
leofante@cs.rwth-aachen.de, n.narodytska@vmware.com, lpulina@uniss.it , armando.tacchella@unige.it

Abstract

Neural networks are one of the most investigated and widely used techniques in Machine Learning. In spite of their success, they still find limited application in safety- and security-related contexts, wherein assurance about networks' performances must be provided. In the recent past, automated reasoning techniques have been proposed by several researchers to close the gap between neural networks and applications requiring formal guarantees about their behavior. In this work, we propose a primer of such techniques and a comprehensive categorization of existing approaches for the automated verification of neural networks. A discussion about current limitations and directions for future investigation is provided to foster research on this topic at the crossroads of Machine Learning and Automated Reasoning.

1 Introduction

Neural Networks (NNs) are powerful learning models that can achieve impressive results in many applications, such as image classification [Taigman *et al.*, 2014] or speech recognition [Yu *et al.*, 2012], with some architectures even claimed to be matching the cognitive abilities of humans [LeCun *et al.*, 2015]. In spite of some exceptions — see, *e.g.*, [Jorgensen, 1997] and more recently [Bojarski *et al.*, 2016; Julian *et al.*, 2016] — traditional applications of NNs have been mostly confined to systems without safety or security requirements, due to the absence of effective methods to guarantee the correct behavior of such models.

There has long been an interest in the rigorous verification of NNs, with first attempts made in the early 2000s [Zakrzewski, 2001; Pullum *et al.*, 2007], mostly motivated by applications in avionic systems. This line of research was recently refueled by critical discoveries made in [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2015]: machine learning models, including state-of-the-art Deep Neural Networks (DNNs), can be unstable with respect to *adversarial perturbations*. Such perturbations represent minimal changes to correctly classified input data that can cause a network to respond in unexpected and incorrect ways. These discoveries confirmed the worthiness of efforts to develop techniques to provide guarantees about the behavior of NNs and other learning models.

Among potential approaches to ensure correct behavior of NNs, those based on Automated Reasoning show some promise. Since NNs are complex implements, it is unlikely that their performances can be checked and corrected manually. Techniques such as Adversarial Training [Goodfellow *et al.*, 2015] have been proposed with the intent to steer learning in the direction of making resulting networks more robust to adversarial attacks. However, recent results [Carlini and Wagner, 2017] have shown that existing methods still lack thorough evaluations and often they are even unable to detect adversarial examples. On the other hand, automated reasoning tools can be applied to NNs "out of the box" to perform verification of desired properties, *e.g.*, robustness, safety, and equivalence. As with any algorithmic technique, the challenge shifts towards the computational needs of automated verification, and the problem of scaling to networks of relevant size arises.

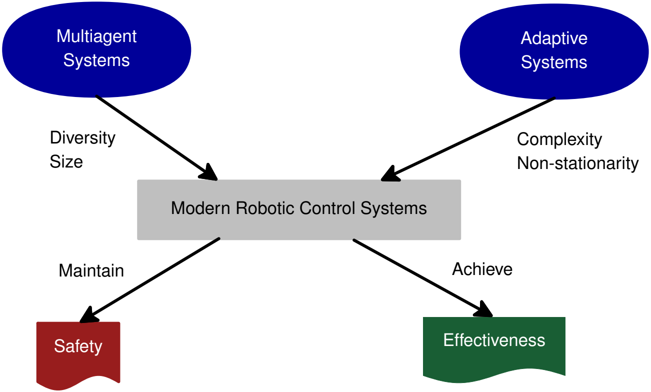
Starting from the seminal contribution of [Pulina and Tacchella, 2010], verification of NNs is not just a theoretical possibility, but it has witnessed diverse proposals based on a variety of automated reasoning techniques, including Boolean satisfiability (SAT) solvers, Satisfiability Modulo Theories (SMT) solvers and Mixed Integer Programming (MIP) solvers. The contributions to be found in the literature consider verification of diverse models, from conventional NNs, to networks apt for representation learning, *i.e.*, those "*...allowing a machine to be fed with raw data and automatically discover the representations needed for detection or classification*" [LeCun *et al.*, 2015]. Following the common usage found in the literature, we associate the term *deep* to networks apt for representation learning; by contrast, we use the term *shallow* to denote networks designed within a conventional learning framework. As a matter of fact, while all NNs are arranged in layers of elementary computation units, conventional networks are indeed shallow since they rarely consist of several layers beyond input and output ones. From the initial challenges and limitations presented in [Pulina and Tacchella, 2012], mostly related to the application of SMT solvers to prove properties of shallow NNs, several contributions have focused on the challenge of scaling SMT, as well as SAT and MIP techniques to deep networks. In this work, we present a survey of such literature, and we contribute a

Available at `https://arxiv.org/abs/1805.09938`

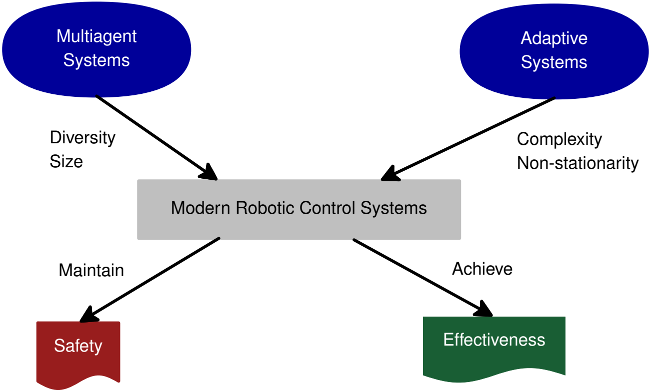
Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Motivation


Multiagent Systems
Adaptive Systems
Diversity
Size
Complexity
Non-stationarity
Modern Robotic Control Systems
Maintain
Achieve
Safety
Effectiveness

Motivation



Safety-Efficiency tradeoff

Inaction is **trivially safe**, whereas **efficient** action can be **unsafe.**

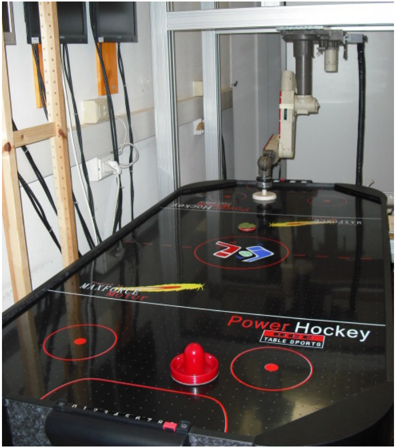
Outline

1 Stateless models
- Safety of multilayer perceptrons (MLPs)
- The PUMA manipulator case study
- Counterexample-based verification and repair

2 Hybrid modal models
- Safety in (adaptive) hybrid systems
- The Air-Hockey setup
- Modeling and experimental results

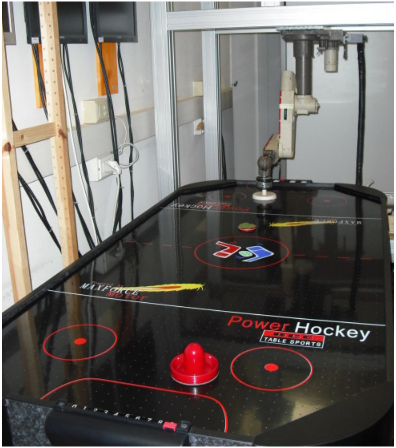
3 Probabilistic modal models
- Safety in sequential decision making (with uncertainty)
- Bioloid's standing-up task
- Learning, verification and repair

A case for Air Hockey



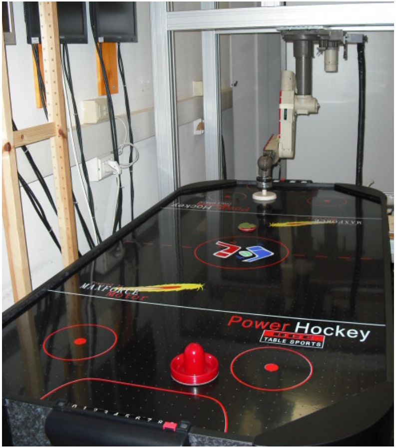
- **Fast**: rapid perception, thinking and movements.

A case for Air Hockey



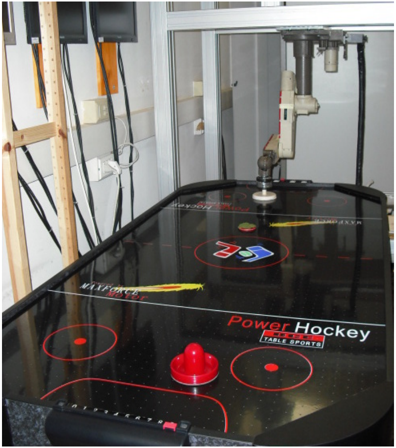
- **Fast**: rapid perception, thinking and movements.
- **Demanding**: movement must be accurate.

A case for Air Hockey



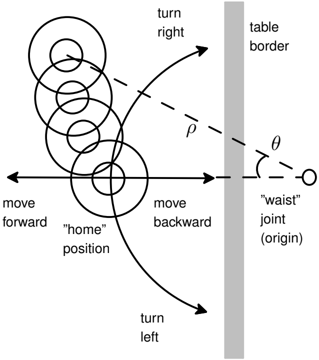
- **Fast**: rapid perception, thinking and movements.
- **Demanding**: movement must be accurate.
- **Complex**: time delays, board placement and conditions.

A case for Air Hockey



- **Fast**: rapid perception, thinking and movements.
- **Demanding**: movement must be accurate.
- **Complex**: time delays, board placement and conditions.
- **Potentially unsafe**: fast moving industrial manipulator!

Air Hockey setup: Motion control



- Polar coordinates on a **plane** with origin in the PUMA "waist" joint.
- Motion control based on **primitives**

move	forward (increase ρ), backward (decrease ρ)
turn	right (increase θ), left (decrease θ)
home	reset to $\rho = \rho_h$, $\theta = 0$

- Given (ρ, θ) **combine primitives** to reach target position.
- Always execute "turn" first.

Air Hockey setup: Learning and adaptation

- Predict (ρ, θ) in order to **intercept puck** (defense play)

Air Hockey setup: Learning and adaptation

- Predict (ρ, θ) in order to **intercept puck** (defense play)
- Working hypotheses:
  - No previous knowledge of table **size and placement**
  - No modeling of **puck motion**

Air Hockey setup: Learning and adaptation

- Predict (ρ, θ) in order to **intercept puck** (defense play)
- Working hypotheses:
  - No previous knowledge of table **size and placement**
  - No modeling of **puck motion**
- Linear model for prediction

$$
\begin{aligned}
\rho_{ee} &= p_1 + p_2\rho_1 + p_3\theta_1 + p_4\rho_2 + p_5\theta_2 \\
\theta_{ee} &= p_6 + p_7\rho_1 + p_8\theta_1 + p_9\rho_2 + p_{10}\theta_2
\end{aligned}
$$

where

- (ρ_{ee}, θ_{ee}) are end-effector coordinates
- (ρ_1, θ_1) and (ρ_2, θ_2) are two different puck positions, and
- $\mathbf{p} = \{p_1, p_2, \ldots, p_{10}\}$ is learned using **LMS optimization**.

Air Hockey setup: Learning and adaptation

- Predict (ρ, θ) in order to **intercept puck** (defense play)
- Working hypotheses:
  - No previous knowledge of table **size and placement**
  - No modeling of **puck motion**
- Linear model for prediction

$$
\begin{aligned}
\rho_{ee} &= p_1 + p_2\rho_1 + p_3\theta_1 + p_4\rho_2 + p_5\theta_2 \\
\theta_{ee} &= p_6 + p_7\rho_1 + p_8\theta_1 + p_9\rho_2 + p_{10}\theta_2
\end{aligned}
$$

  where
  - (ρ_{ee}, θ_{ee}) are end-effector coordinates
  - (ρ_1, θ_1) and (ρ_2, θ_2) are two different puck positions, and
  - $\mathbf{p} = \{p_1, p_2, \ldots, p_{10}\}$ is learned using **LMS optimization**.
- Adaptation: accumulate new samples and recompute **p**.

Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Modeling: Hybrid automata distilled

Hybrid Automaton	=	Discrete Control Modes	+	Continuos Dynamics

Modeling: Hybrid automata distilled

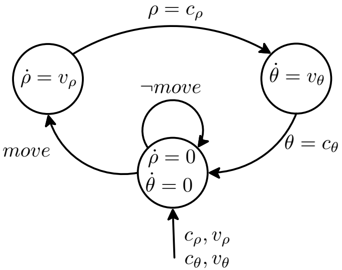
Hybrid Automaton	=	Discrete Control Modes	+	Continuos Dynamics

- Example: a simple straight-then-turn strategy to reach a reference position in polar coordinates (ρ_c, θ_c)

Modeling: Hybrid automata distilled

Hybrid Automaton = Discrete Control Modes + Continuos Dynamics

- Example: a simple straight-then-turn strategy to reach a reference position in polar coordinates (ρ_c, θ_c)
- Three control modes with linear dynamics

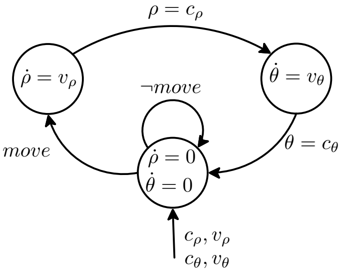


1. Stand still ($\dot{\rho} = \dot{\theta} = 0$)
2. Change ρ at constant velocity v_ρ
3. Change θ at constant velocity v_θ

Modeling: Hybrid automata distilled

Hybrid Automaton = Discrete Control Modes + Continuos Dynamics

- Example: a simple straight-then-turn strategy to reach a reference position in polar coordinates (ρ_c, θ_c)
- Three control modes with linear dynamics



1. Stand still ($\dot{\rho} = \dot{\theta} = 0$)
2. Change ρ at constant velocity v_ρ
3. Change θ at constant velocity v_θ

- Transitions on boolean events (e.g., *move*) or when reaching boundary conditions (e.g., $\rho = c_\rho$).

Modeling: dealing with multiple adaptive agents

Multiple agents

- Model each agent as a **hybrid automaton**
- Use **global variables** to handle communications between agents (a shared memory model)
- Check **asynchronous composition** of the automata

Modeling: dealing with multiple adaptive agents

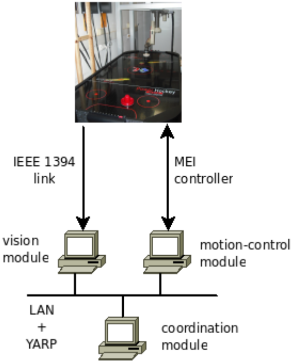
Multiple agents

- Model each agent as a **hybrid automaton**
- Use **global variables** to handle communications between agents (a shared memory model)
- Check **asynchronous composition** of the automata

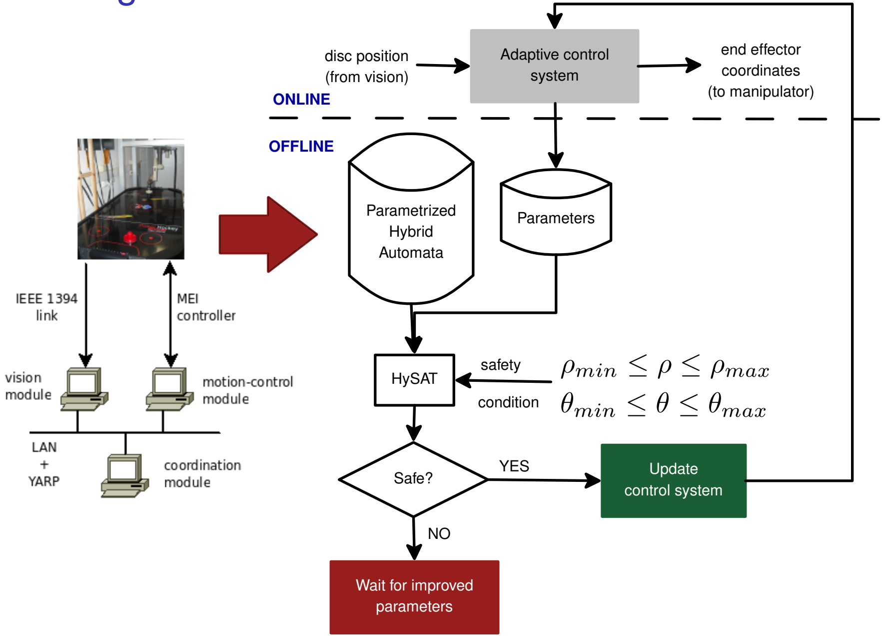
Adaptive agents

- Adaptation can change **structure** and **parameters**
- We keep **structure fixed**, only parameters change
- A “scheleton” automata encodes structure
- Once parameters are available, we have a **complete automaton** that we can check for safety.

Modeling: overview



Modeling: overview



Experimental results: setup

- Robot plays games against ten different human players.

Experimental results: setup

- Robot plays games against ten different human players.
- Three different settings of the coordination module

  Off-line: parameters are learned off-line using 50 straight and 100 single-bounce shots; **no safety check.**

Experimental results: setup

- Robot plays games against ten different human players.
- Three different settings of the coordination module

  Off-line parameters are learned off-line using 50 straight and 100 single-bounce shots; **no safety check.**

  On-line parameters $\mathbf{p}$ are learned on-line; bootstrap parameters $\mathbf{p}_0$ correspond to a **hand-made setting** checked for safety.

Experimental results: setup

- Robot plays games against ten different human players.
- Three different settings of the coordination module

  Off-line
  : parameters are learned off-line using 50 straight and 100 single-bounce shots; **no safety check.**

  On-line
  : parameters $\mathbf{p}$ are learned on-line; bootstrap parameters $\mathbf{p}_0$ correspond to a **hand-made setting** checked for safety.

  Safe on-line
  : each time a new set of parameters is learned, it is checked for safety and, if safe, it is plugged in.

Experimental results: setup

- Robot plays games against ten different human players.
- Three different settings of the coordination module
  - Off-line: parameters are learned off-line using 50 straight and 100 single-bounce shots; **no safety check.**
  - On-line: parameters **p** are learned on-line; bootstrap parameters $\mathbf{p}_0$ correspond to a **hand-made setting** checked for safety.
  - Safe on-line: each time a new set of parameters is learned, it is checked for safety and, if safe, it is plugged in.
- On-line settings **keep learning** across different players, so the more games are played, the more effective the robot becomes.
- New parameters are considered safe if HySAT cannot find a safety violation within 30 CPU seconds.

Experimental results: looking for unsafe states

PLAYER	OFF-LINE		ON-LINE	
	SHOTS	UNSAFE	SHOTS	UNSAFE
# 1	59	–	55	1
# 2	56	2	72	3
# 3	46	1	39	–
# 4	61	–	46	–
# 5	58	–	80	–
# 6	48	–	69	–
# 7	84	6	76	1
# 8	44	2	84	–
# 9	103	–	112	–
# 10	99	8	86	–

Experimental results: effectiveness?

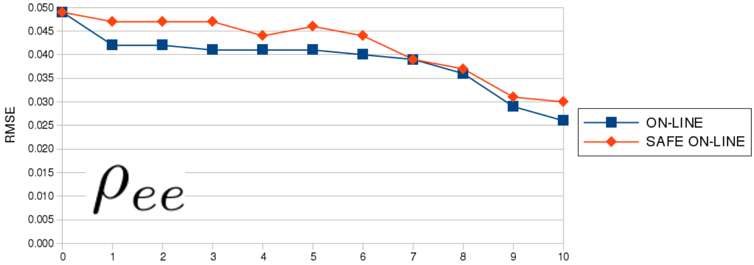
Does checking for safety hinder effectiveness?

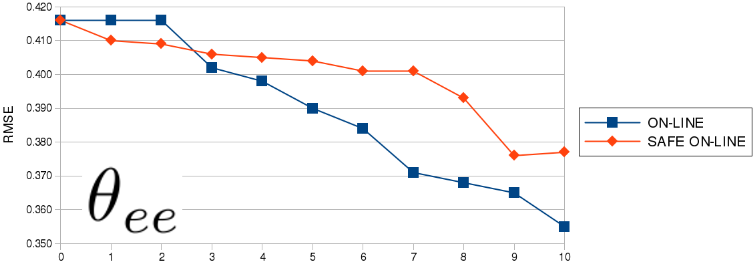
Experimental results: effectiveness?

Does checking for safety hinder effectiveness?

- Extract input coordinates and reference target positions from off-line training set
- Compute RMSE between
  - Reference target positions, and
  - output of adaptive system using linear regression
- Compare the evolution of **on-line** and **safe on-line** settings.

Experimental results: On-line vs. safe on-line learning





Summing up...

- Modelling multiagent adaptive control systems using **parametrized** hybrid automata.
- Combining offline checking and online learning to **maintain safety** without **compromising effectiveness**.
- Showcasing formal methods in robotics using a **real** and **challenging** task.

Acknowledgements
EU Information and Communication Technologies 7th Framework Programme [FP7/2007-2013] grant N. 215805, the "CHRIS" project

Critiques and recent related works

MC of hybrid-adaptive models
Metta-Natale-Pathak-Pulina-Tacchella [ICRA 2010]

- **Pros**: widely applicable, sound, effective
- **Cons**: no repair, cannot handle non-linear models, hardly scalable to multi-robot setups

Critiques and recent related works

MC of hybrid-adaptive models
Metta-Natale-Pathak-Pulina-Tacchella [ICRA 2010]

- **Pros**: widely applicable, sound, effective
- **Cons**: no repair, cannot handle non-linear models, hardly scalable to multi-robot setups

Recent attempts

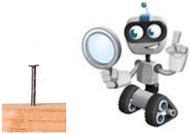
Too many to cite them in a slide!

- **Data driven** verification and synthesis
- Formal **synthesis of controllers**
- AI-Planning for hybrid systems: **build**, **execute**, **repair**, **monitor**

Outline

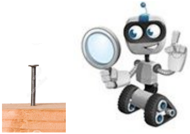
1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

How it works



Observe s_0

How it works

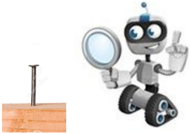


Observe s_0

$\Rightarrow$

Perform a_0

How it works



Observe s_0

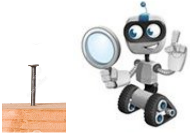
$\Rightarrow$

Perform a_0

$\Rightarrow$

Receive r_1

How it works



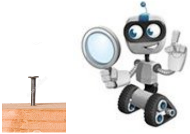
Observe s_0

$\Rightarrow$

Perform a_0

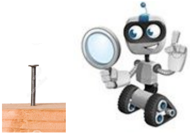
$\Rightarrow$

Receive r_1



Observe s_1

How it works



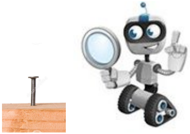
Observe s_0

$\Rightarrow$

Perform a_0

$\Rightarrow$

Receive r_1

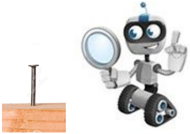


Observe s_1

$\Rightarrow$

Perform a_1

How it works

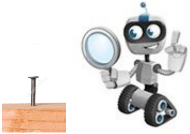

Observe s_0

$\Rightarrow$


Perform a_0

$\Rightarrow$

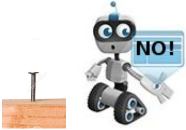

Receive r_1


Observe s_1

$\Rightarrow$

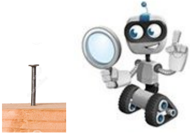

Perform a_1

$\Rightarrow$



Receive r_2

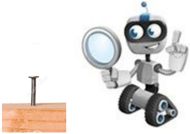
How it works



Observe s_0 $\Rightarrow$

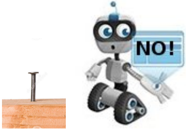
Perform a_0 $\Rightarrow$

Receive r_1

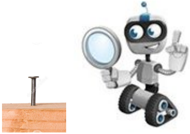


Observe s_1 $\Rightarrow$

Perform a_1 $\Rightarrow$



Receive r_2

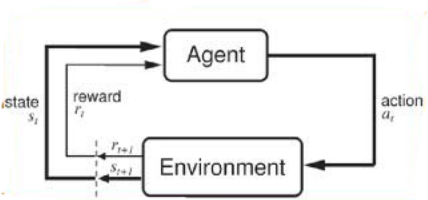


Observe s_2 $\Rightarrow$

Perform a_2 $\Rightarrow$

Receive r_3

Reinforcement Learning (RL)



Set of states S, set of actions A

- Agent can sense current state $s_t \in S$
- Agent peforms action $a_t \in A$ in state s_t
- Environment "moves" to state s_{t+1}
- Agent receives reward $r_{t+1} = \rho(s_t, a_t)$

Fact

δ and ρ are **not known** (but assumed to be **stationary**)

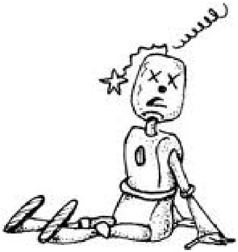
Goal

Learn **policy** $\pi : S \rightarrow A$

Safety in RL

Safety can be defined in negative terms. An agent's behavior is unsafe, if it leads to:

- **Fatal states**, e.g., injury to environment or robot, unrecoverable posture
- **Undesirable states**, e.g., singular posture requiring reset of manipulator



Exploitation vs. Exploration

I learned to ride with RL...

Safety while learning

- Steep challenge!
- RL acquires knowledge by **trial-and-error**!

Safety after learning

1. **Learn safely** (e.g., simulator)
2. **Verify** that policy π is safe
3. Possibly **fix** π
4. Deploy and **monitor**

Mathematical model

Environment is a Markovian Decision Process (MDP)

- S: Set of all possible states the system could be in
- A: Set of all possible actions
- $\rho : S \times A \rightarrow \mathbb{R}$: Rewards or utility of state(-action)
- $\delta : S \times A \rightarrow S$: Transition function such that $P(s_{t+1}|a_t, s_t, s_{t-1}, \ldots s_0) = P(s_{t+1}|a_t, s_t)$

Agents provides stochastic policy (maximizing returns)

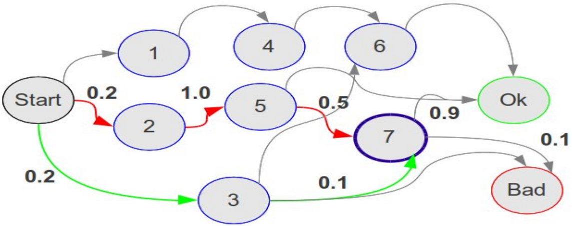
For all states $s \in S$ and actions $a \in A$, $\pi(s, a)$ is the probability of taking action a in state s.

Environment + Policy = (Discrete Time) Markov Chain

DTMC

Given a set of propositions AP, a DTMC is a tuple $(W, \overline{w}, \mathbf{P}, L)$ where

- W is a finite set of **states**
- $\overline{w} \in W$ is the **initial state**;
- $\mathbf{P} : W \times W \to [0, 1]$ is the *transition probability matrix*
- $L : W \to 2^{AP}$ is the *labeling function*.



Safety of agent = Reachability of “bad” states

Key element 1: Probabilistic Temporal Logic (PCTL)

A logic language to express probability of behaviors in DTMCs

$$\mathcal{M}, w_0 \models \mathcal{P}_{<\sigma}[\mathcal{F}\ bad]$$

a.k.a. “Given DTMC $\mathcal{M}$, is the probability of reaching some state labelled *bad* from state w_0 less than σ?”

Key element 2: Probabilistic Model Checking

- Algorithms that can decide queries in PCTL
- Tools (e.g., PRISM, STORM) that implement such algorithms

Outline

1 Stateless models
- Safety of multilayer perceptrons (MLPs)
- The PUMA manipulator case study
- Counterexample-based verification and repair

2 Hybrid modal models
- Safety in (adaptive) hybrid systems
- The Air-Hockey setup
- Modeling and experimental results

3 Probabilistic modal models
- Safety in sequential decision making (with uncertainty)
- Bioloid's standing-up task
- Learning, verification and repair

Context and motivation

- Bipedal locomotion is a challenging task for a humanoid robot
- Reliable standing-up routines are fundamental in case of a fall
- Conventional motion-planning is difficult to apply

Context and motivation

- Bipedal locomotion is a challenging task for a humanoid robot
- Reliable standing-up routines are fundamental in case of a fall
- Conventional motion-planning is difficult to apply
- Scripted strategies are often used:
  - lack flexibility (by definition)
  - reliability and robustness issues
  - daunting task

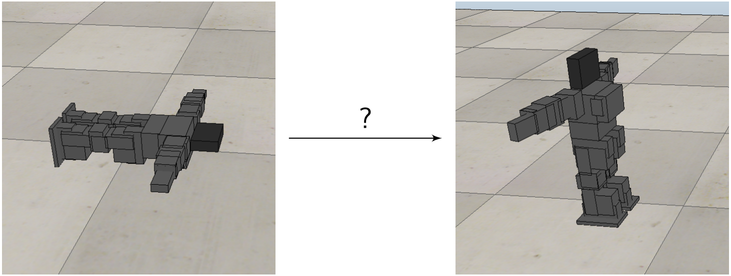
Context and motivation

- Bipedal locomotion is a challenging task for a humanoid robot
- Reliable standing-up routines are fundamental in case of a fall
- Conventional motion-planning is difficult to apply
- Scripted strategies are often used:
  - lack flexibility (by definition)
  - reliability and robustness issues
  - daunting task

Learning offers an elegant solution

Objectives

Problem: Synthesize a standing-up procedure that minimizes the expected number of falls, self-collisions and actions.



Simulated Bioloid humanoid in V-REP

Reinforcement learning

Reinforcement learning

- Goal: Learn an optimal strategy for a non-deterministic probabilistic system

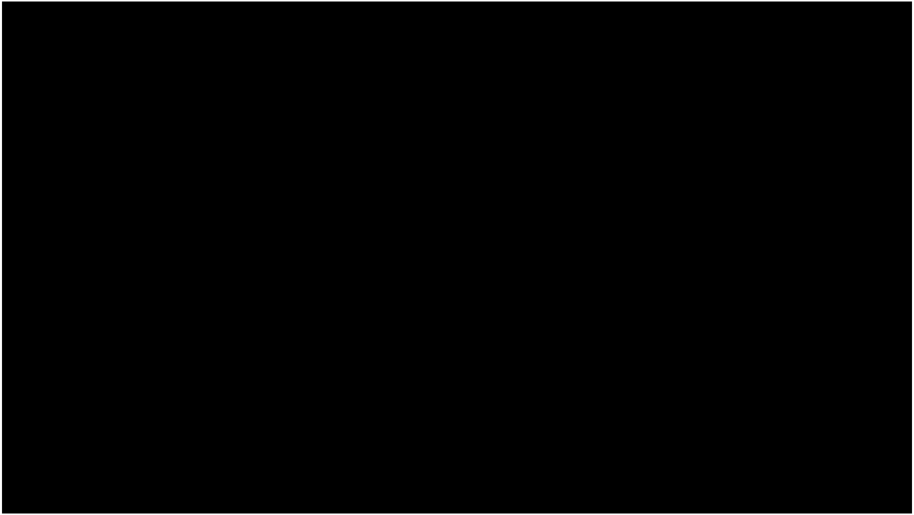
Reinforcement learning

- Goal: Learn an optimal strategy for a non-deterministic probabilistic system
- Given:
  - state set S, initial state s^{init}
  - action set Act
  - a possibility to observe the successor state when executing a given action in a given state
  - a reward function $R\colon S \times Act \times S \to \mathbb{R}$

Reinforcement learning

- Goal: Learn an optimal strategy for a non-deterministic probabilistic system
- Given:
  - state set S, initial state s^{init}
  - action set Act
  - a possibility to observe the successor state when executing a given action in a given state
  - a reward function $R \colon S \times Act \times S \to \mathbb{R}$
- Method: Q-learning

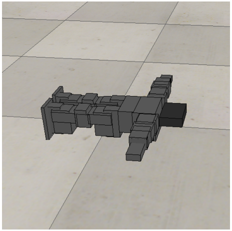
Q-learning: Learning through simulation



Q-learning on an example

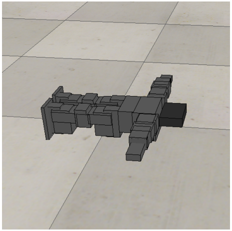
Q-learning on an example

s_0



Q-learning on an example

s_0

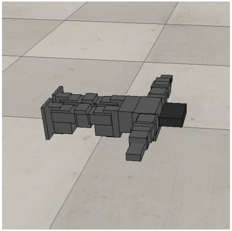


Rewards:

R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-learning on an example

s_0



Rewards:

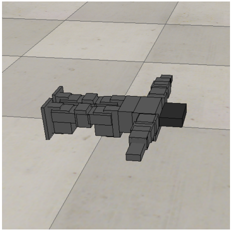
R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	0	0	
s_1	0	0	0	
s_2	0	0	0	
...				

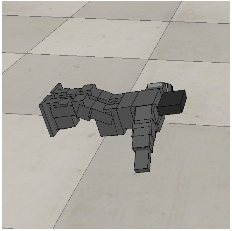
Q-learning on an example

s_0



$\downarrow a_1$

s_1



Rewards:

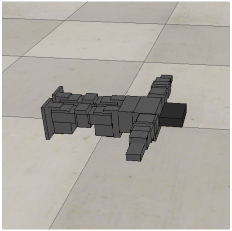
R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	0	0	
s_1	0	0	0	
s_2	0	0	0	
...				

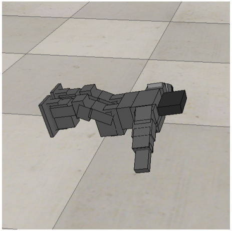
Q-learning on an example

s_0



$\downarrow a_1$

s_1



Rewards:

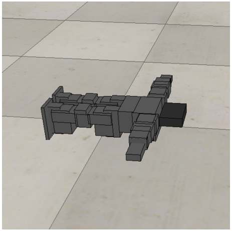
R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	0	0	
s_1	0	0	0	
s_2	0	0	0	
...				

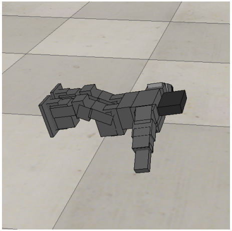
Q-learning on an example

s_0



$\downarrow a_1$

s_1



Rewards:

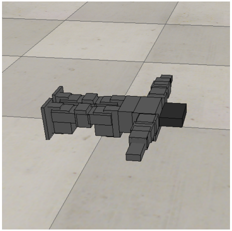
R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	0	0	
s_1	0	0	0	
s_2	0	0	0	
...				

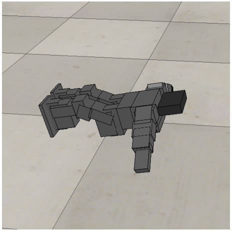
Q-learning on an example

s_0



$\downarrow a_1$

s_1



Rewards:

R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

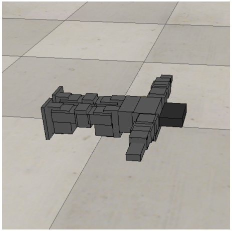
Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	0	0	
s_1	0	0	0	
s_2	0	0	0	
...				

$$Q_{k+1}(s_0, a_1) = 0.5 \cdot Q_k(s_0, a_1) + 0.5 \cdot (100 + 1 \cdot max_{a_i \in Act} Q_k(s_1, a_i))$$

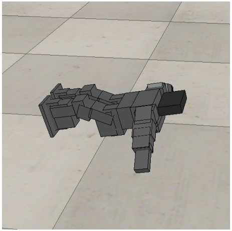
Q-learning on an example

s_0



$\downarrow a_1$

s_1



Rewards:

R	s_0	s_1	s_2	...
(s_0, a_0)	-10	100	-50	
(s_0, a_1)	-10	100	-50	
...				
(s_1, a_0)	-50	-10	100	
(s_1, a_1)	-50	-10	100	
...				

Q-matrix:

Q	a_0	a_1	a_2	...
s_0	0	50	0	
s_1	0	0	0	
s_2	0	0	0	
...				

$$Q_{k+1}(s_0, a_1) = 0.5 \cdot Q_k(s_0, a_1) + 0.5 \cdot (100 + 1 \cdot max_{a_i \in Act} Q_k(s_1, a_i))$$

Outline

1. Stateless models
   - Safety of multilayer perceptrons (MLPs)
   - The PUMA manipulator case study
   - Counterexample-based verification and repair
2. Hybrid modal models
   - Safety in (adaptive) hybrid systems
   - The Air-Hockey setup
   - Modeling and experimental results
3. Probabilistic modal models
   - Safety in sequential decision making (with uncertainty)
   - Bioloid's standing-up task
   - Learning, verification and repair

Q-learning: The action space

The robot has 18 joints $\rightarrow$ intractable action space

Simplifying assumptions:

- some joints are inhibited
- joints operate symmetrically
- action space is discretized

We end up with 730 actions:

- 3 upper limbs, 3 lower limbs, 3 actions each $\rightarrow$ action space $\{-1, 0, 1\}^6$
- additional action $a^{restart}$ for safe restart

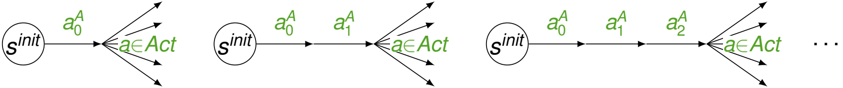
Q-learning: The state space

- Robot states: $\mathbf{s} = (x, y, z, q_0, q_1, q_2, q_3, \rho_1, \ldots, \rho_{18}) \in \mathbb{R}^{25}$
- Infinite state space!
- Full grid discretization is infeasible

Q-learning: The state space

- Robot states: $\mathbf{s} = (x, y, z, q_0, q_1, q_2, q_3, \rho_1, \ldots, \rho_{18}) \in \mathbb{R}^{25}$
- Infinite state space!
- Full grid discretization is infeasible

- Input: scripted trace $A = (a_0^A, \ldots, a_k^A)$ for standing-up
- Explore states in a "tube" around A



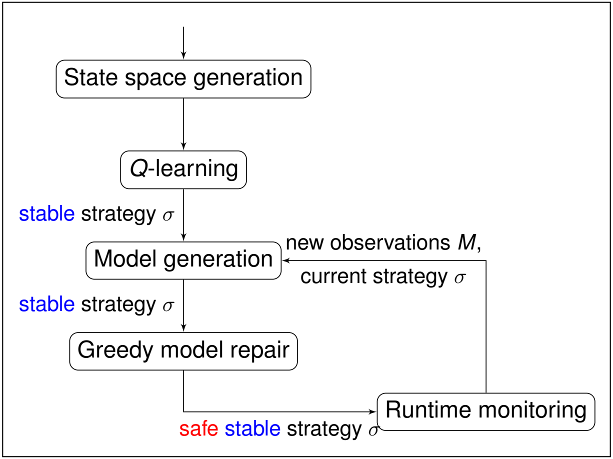
- Discretize the so reachable states $\rightarrow$ 17614 states
- **Still, several adaptation of Q-learning were needed to achieve convergence**
- Several additional paths to the goal could be identified (even shorter)

Static and runtime methods: Our framework

Wait but... how to guarantee that our properties of interest are satisfied?

Static and runtime methods: Our framework

Wait but... how to guarantee that our properties of interest are satisfied?



That's why we combine it with static analysis and runtime monitoring

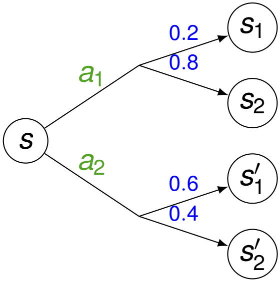
Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

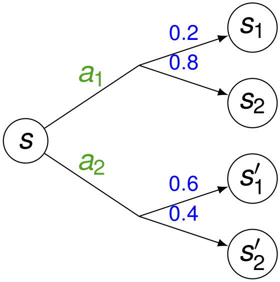
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

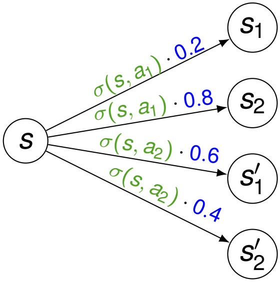
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

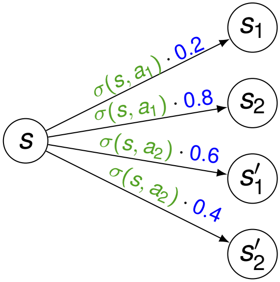
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

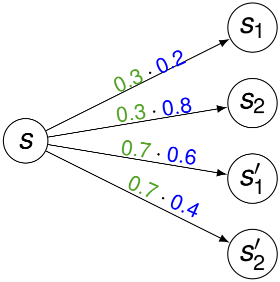
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

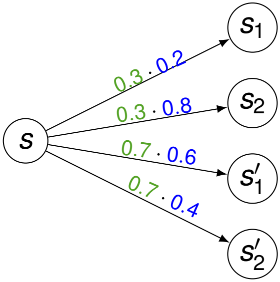
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning

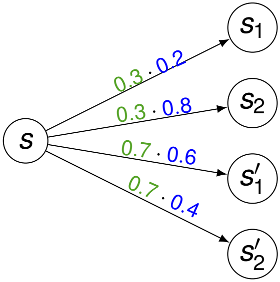


	s^{fall}	s^{coll}	s^{far}
Reach.prob. in model	0.001	0.005	0.048
Reach.prob. in simulation	0	0.003	0.046

Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

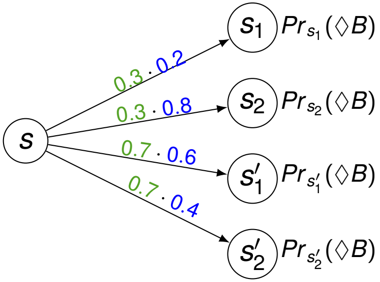
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

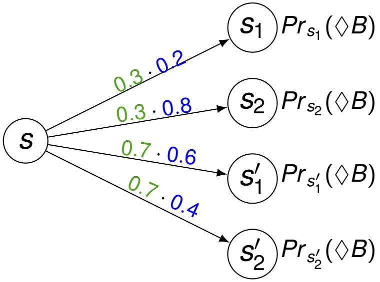
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

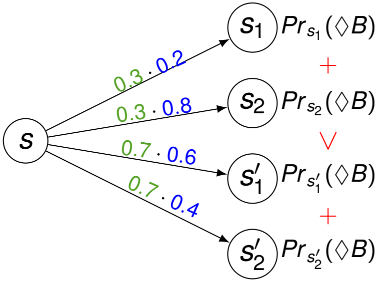
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking
- Repair the scheduler if unsafe



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

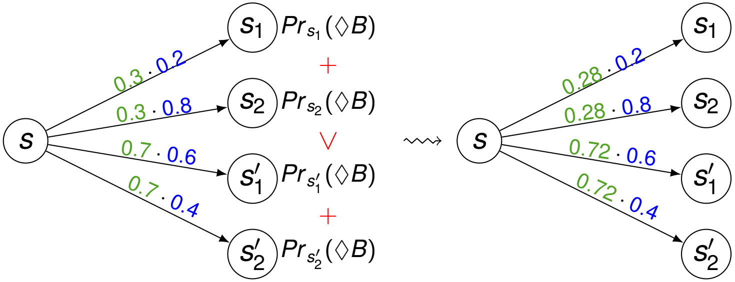
- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking
- Repair the scheduler if unsafe



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking
- Repair the scheduler if unsafe



Model repair: Idea

How can we adapt schedulers to satisfy certain safety requirements?

- Collect statistical information during Q-learning
- Compute a Markov decision process (MDP) model of the robot
- Abstract scheduler $\rightarrow$ parametric DTMC
- Instantiate parametric DTMC model by the scheduler from Q-learning
- Check safety by probabilistic model checking
- Repair the scheduler if unsafe

	s^{fall}	s^{coll}	s^{far}
Reach.prob. in model *before repair*	0.001	0.005	0.048
Reach.prob. in simulation *before repair*	0	0.003	0.046
Reach.prob. in model *after repair*	0.0003	$6.8 \cdot 10^{-6}$	0.02
Reach.prob. in simulation *after repair*	0	0	0

Runtime monitoring

So now we deploy our safe, repaired strategy on the real robot and everything should be fine right?

Runtime monitoring

So now we deploy our safe, repaired strategy on the real robot and everything should be fine right?

WRONG

Runtime monitoring

So now we deploy our safe, repaired strategy on the real robot and everything should be fine right?

WRONG

What if the assumptions on which the model was built change?

$\leadsto$ environmental changes, robot failures ...

Runtime monitoring

So now we deploy our safe, repaired strategy on the real robot and everything should be fine right?

WRONG

What if the assumptions on which the model was built change?

$\rightsquigarrow$ environmental changes, robot failures ...

Looks like this is a problem we could solve using...

Runtime monitoring

So now we deploy our safe, repaired strategy on the real robot and everything should be fine right?

WRONG

What if the assumptions on which the model was built change?

$\rightsquigarrow$ environmental changes, robot failures . . .

Looks like this is a problem we could solve using. . .

runtime monitoring

Runtime monitoring

- We collect statistical observations during deployment
- From time to time, we update the MDP model with the new observations
- Model check and repair the scheduler if needed

Runtime monitoring

- We collect statistical observations during deployment
- From time to time, we update the MDP model with the new observations
- Model check and repair the scheduler if needed

- We simulated that a part of the robot was broken
- Out of 300 simulation episodes only 2 reached the goal state
- After a feedback loop, in further 300 episodes, 197 reached the goal

Critiques and related works

Probabilistic model-checking and repair approach of Leofante-Vuotto-Abraham-Tacchella-Jansen [ISOLA 2016]

- **Pros**: manageable state and action space representations for complex systems, smooth application of formal methods
- **Cons**: time-consuming simulation

Other attempts

- Probabilistic model checking of emergent behaviors in robot swarms (C. Dixon et al.)
- Integration between learning and verification (N. Jansen et al.)

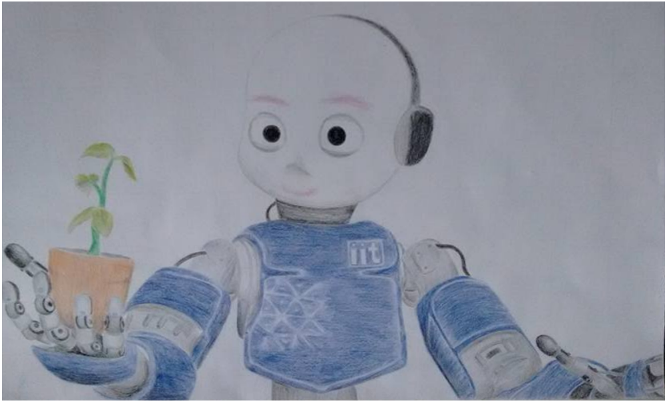
Acknowledgements

Substantial parts of the research described in this seminar has been carried out with the help of many fellow researchers:

- Luca Pulina (University of Genoa, University of Sassari)
- Giorgio Metta, Lorenzo Natale (Italian Institute of Technology)
- Erika Abraham, Joost-Pieter Katoen (RWTH-Aachen)
- Nils Jansen (RWTH Aachen, Radboud Univ. Nijmegen)
- Shashank Pathak (University of Genoa, Visteon),
- Francesco Leofante (University of Genoa and RWTH Aachen)
- Simone Vuotto (University of Genoa, University of Sassari)
- Dario Guidotti (University of Genoa)
- Claudio Castellini (DLR)

Thank you for your attention!

Questions or comments?



Drawing courtesy of Francesco Tacchella