CTRLVERIF. Analysis of control systems

Lecture 0. Introduction.

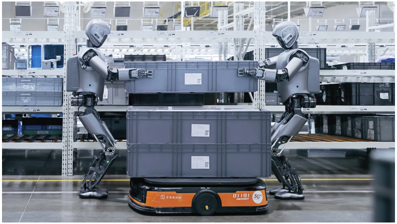
Eric Goubault and Sylvie Putot

MPRI

Motivation: trusting autonomous controlled systems

Control systems that operate autonomously in complex environments are a reality

Flight control computers



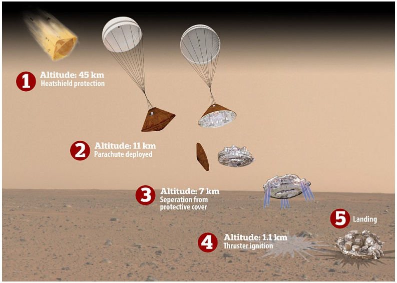
Robotics

Exploration and monitoring

Autonomous driving

But are they safe ?

Ariane 5 crash (1996)

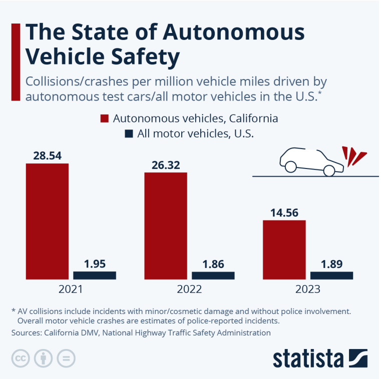


Schiaparelli's crash on Mars

Software bugs, but related to control and robustness in highly uncertain environment:

- Ariane 5: a piece of code was copied from Ariane 4. But the flight dynamics changed: overflow in a conversion from a 64 bits to a 16 bits number
- Schiaparelli: saturation of the IMU (inertial Measurement Unit): negative estimated altitude when it was still at 3.7km and premature release of the parachute...

Learning-enabled systems in particular are fragile

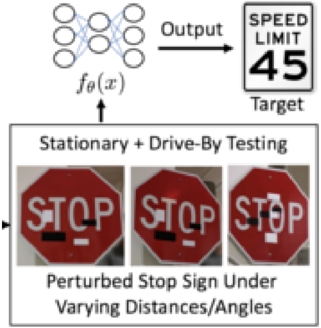


The State of Autonomous Vehicle Safety

Collisions/crashes per million vehicle miles driven by autonomous test cars/all motor vehicles in the U.S.*

* AV collisions include incidents with minor/cosmetic damage and without police involvement. Overall motor vehicle crashes are estimates of police-reported incidents.

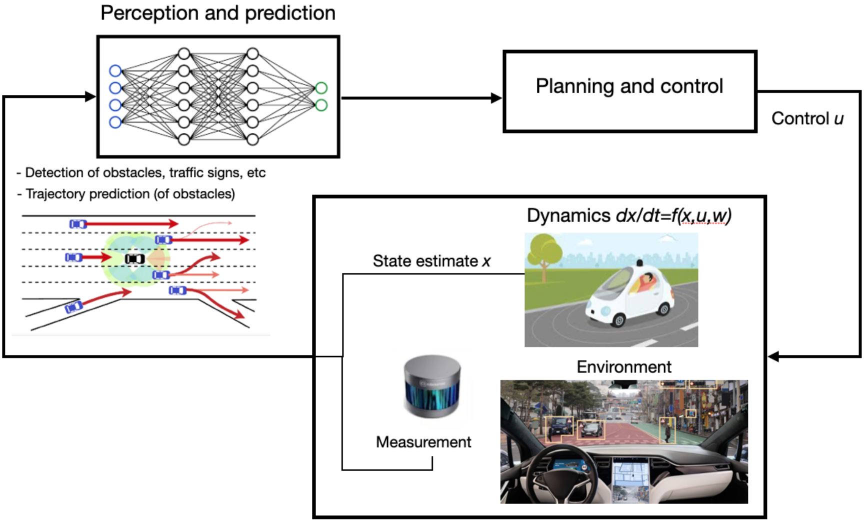
Sources: California DMV, National Highway Traffic Safety Administration



Output
SPEED LIMIT 45
Target
$f_\theta(x)$
Stationary + Drive-By Testing
STOP
Perturbed Stop Sign Under Varying Distances/Angles

Eykholt et al, Robust Physical-World Attacks on Deep Learning Visual Classification, 2018

The typical control loop



Perception and prediction

- Detection of obstacles, traffic signs, etc
- Trajectory prediction (of obstacles)

Planning and control

Control u

Dynamics $dx/dt=f(x,u,w)$

State estimate x

Environment

Measurement

- ▶ Need to operate in unknown, uncertain and dynamic environments
- ▶ Should be robust to change in lightning, and noise in general

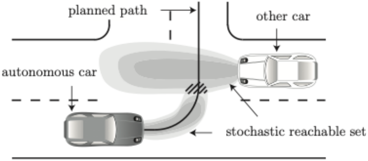
Reachability analysis

Specifying the expected behavior for control systems

- A difficulty with respect to traditional program verification: many applications lack mathematical specifications
- For example, how do we specificy that a traffic sign should recognized ?

Rechability analysis: ensure that safe envelopes of dynamics are maintained

- finite-horizon properties (robustness to disturbances, reach-avoid properties),
- infinite-horizon properties (stability, viability kernels, etc.)
- in every possible configuration for uncertain inputs, parameters and disturbances
- Input-output relationships when a specification is available

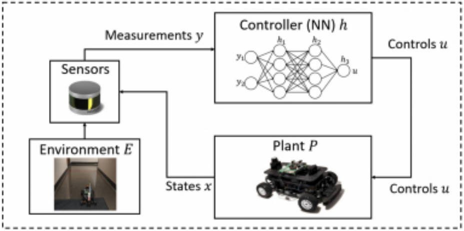


M. Althoff, Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars

The closed-loop: a time-triggered hybrid system

Given

- plant dynamic f,
- state x, control u, disturbance $w \in \mathcal{W}$
- controller h
- control period Δt_u



Time-triggered (u computed every $\Delta_u t$) dynamical system with non-linear feedback:

$$\begin{aligned}
\dot{x}(t) &= f(x(t), u(t), w(t)) \\
x(t_0) &= x_0 \in \mathcal{X}_0 \\
u(t) &= u_k = h(y(x(\tau_k))),\ \text{for } t \in [\tau_k, \tau_{k+1}),\ \text{with } \tau_k = t_0 + k\Delta t_u,\ \forall k \geq 0
\end{aligned}$$

Reachable sets of dynamical systems

Under classical hypotheses, there exists a flow $\varphi^f(s; x_0, u)$ solution of the initial value problem:

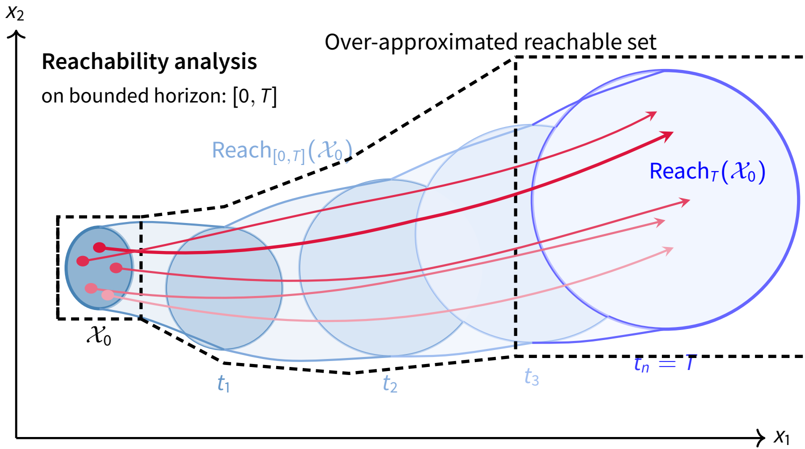
$$(S)\begin{cases}\dot{x}(t) = f(x(t), u(t)) \\ x(0) \in \mathcal{X}_0, u(t) \in \mathcal{U} \subseteq \mathbb{R}^p\end{cases}$$

Given the set of initial conditions $\mathcal{X}_0$, system dynamics $\dot{x}(t) = f(x(t), u(t))$:

- State x_f is reachable at time t if
  $$\exists x_0 \in \mathcal{X}_0, \exists u : [0, t] \to \mathcal{U}, \text{s.t. } \varphi^f(t; x_0, u) = x_f.$$
  The reachable set at time t is $\text{Reach}_t(\mathcal{X}_0, \mathcal{U}) = \{x_f | x_f \text{ is reachable at time } t.\}$
  The reachable set is $\text{Reach}(\mathcal{X}_0, \mathcal{U}) = \{x_f | \exists t, x_f \text{ is reachable at time } t\}$, but it is not often computed over infinite time.
- The reachable set or tube or flowpipe over time horizon $[0, T]$ is:

$$\text{Reach}_{[0,T]}(\mathcal{X}_0, \mathcal{U}) = \{x_f | x_f \text{ is reachable at time } t \leq T\}$$

Reachable sets of dynamical systems



Reachability analysis

on bounded horizon: $[0, T]$

Over-approximated reachable set

x_2
$\mathrm{Reach}_{[0,T]}(\mathcal{X}_0)$
$\mathrm{Reach}_T(\mathcal{X}_0)$
$\mathcal{X}_0$
t_1
t_2
t_3
$t_n = T$
x_1

Simulation:

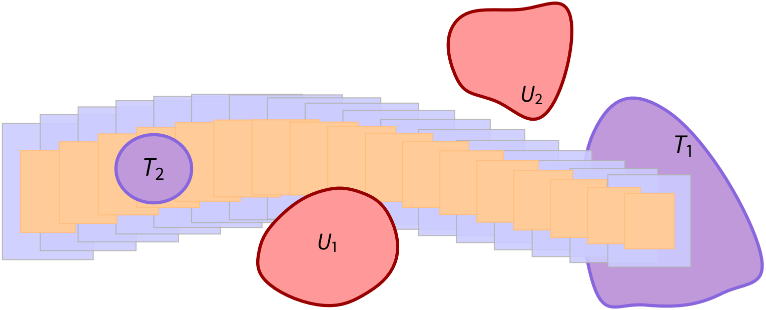
- approximate samples of behavior
- over finite time

Reachability analysis:

- covers all behaviors
- over finite or infinite time

Reachability-based verification: reach-avoid problem

System must reach goal set T1 (and T2) while avoiding unsafe sets U1 and U2



- Outer (or over)-approximation = guaranteed to include all reachable states
  - proves safety wrt to unsafe region U_2, but unconclusive for U_1
  - proves that target region T_1 is reached, but unconclusive for T_2
- Inner (or under)-approximation = all states are guaranteed to be reachable
  - proves unsafety wrt to U_1
  - proves that target region T_2 is entirely reachable ("sweep-avoid?")
  - proves that some trajectories do not reach T_2.

Outline of the course

Recent semantic abstractions for controlled and AI-based systems:

1. Set-based computation and abstract interpretation
2. Abstract interpretation-based verification of feedforward neural networks
3. Reachability verification for uncertain dynamical systems (possibly nn-controlled)
4. Inner- and outer-approximation of general quantified reachability problems
5. Beyond finite time reachability, and homotopy theory
6. The geometry of the reachability space, topological complexity and cohomology
7. The geometry of neural networks, quality of classification and persistent homology
8. The geometry of reachable states in coordination problems (if time allows)

Outline

Courses 1-4: set-based computations

- from function image to general quantified reachability problems
- efficient, but ignore in general the underlying geometry:
  - we are often relying on local solutions
  - we often convexify for tractability, or at least rely on the geometry of the approximations

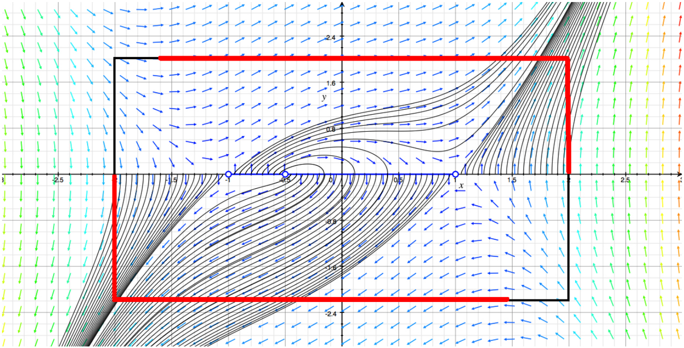
Courses 5-8 : from sets to geometry

- characterize the geometric properties of these reachable sets: connected components, holes ? homotopy type ?
- some specific applications, e.g.:
  - from bounded time reachability to unbounded time
  - measure of complexity of some control problems
  - quality of neural network classifier / complexity of classification problems

Beyond finite time reachability: invariants

Example: from bounded-time reachability to unbounded time reachability

For a continuous dynamical system (ODE), we will be able to “read" what happens within some compact set, from the exit set of the flow map on that compact set - we will see some applications of this (Wazewski’s property, Conley index theory) to the existence of local (positive) invariants within some prescribed compact set.



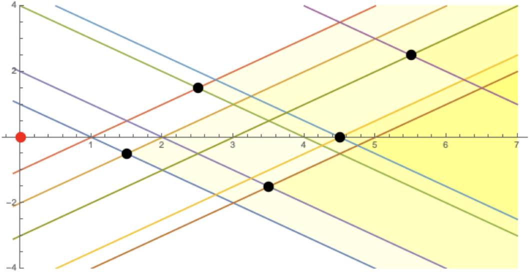
$$\dot{x} = y, \dot{y} = y + (x^2 - 1)\left(x + \frac{1}{2}\right)$$

The geometry of the reachability space

Example: backward reachability

Find controls that will reach some target (sub-)space after some amount of time, and will not run into some other subspaces (e.g. obstacles, as for reach-avoid problems):

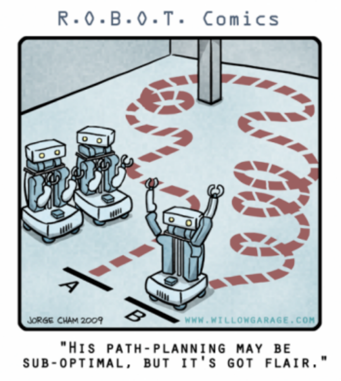
- connected components lead to at least one discrete control (switched controller)
- the finer topology will lead to at least know the necessary dimension, or number, of the continuous controls



(set of backward reachable states/controls with all these obstacles: 11 components)

Similarly, but for measuring the “complexity" of potential controller

Topological complexity



Motion planning algorithm : how do I go from point *A* to point *B*?

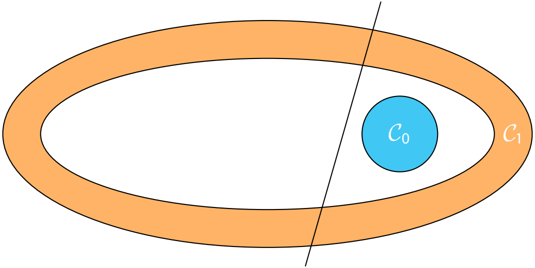
Its complexity : how can I partition the space of points A, B so that to apply on each part a “simple” (e.g. continuous in that case) formula?

The number of discontinuities (similarly to the last slide) is a measure of that complexity.

The geometry of neural networks

Example: the “shape of classes" in neural network classification

A neural network used as a classifier, trained over a set of labelled data, is going to be of quality if it “recognizes" the shapes of the different input classes.



Here: two nested classes with a linear separator; about 80% of the samples will be classified correctly, but we clearly missed the point here.

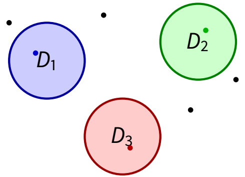
Example: complexity of classification problems

(already a glimpse on that in the last example)

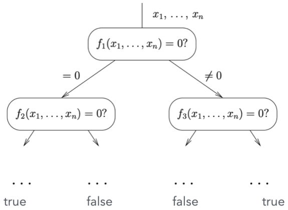
Example: complexity of data classification

When the classes labeled data has “complex" shapes, this has to reflect into the complexity of any classifier (e.g. neural network).

Suppose for now the data forms a “real" topological space (not just a discrete set of points). Then having *m* connected components means we need at least an algorithm with complexity $O(log\ m)$ to classify them (“set membership problem" using algebraic decision trees)



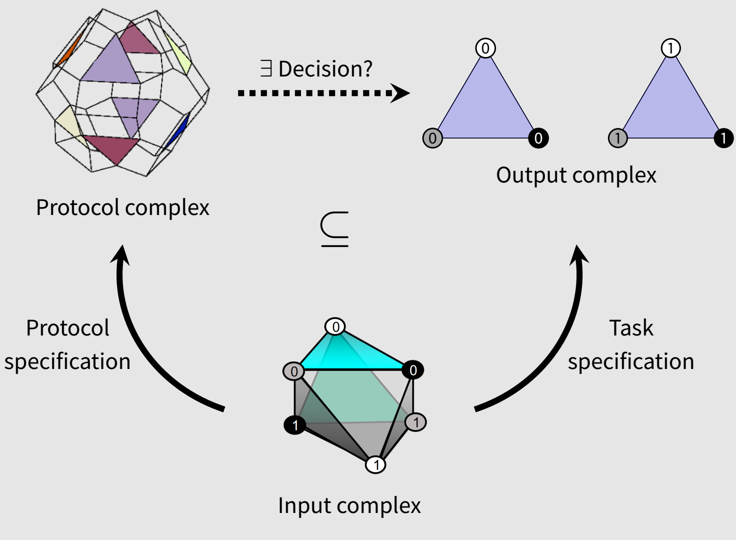
X together with points sampled in the ambient space


$x_1, \ldots, x_n$
$f_1(x_1, \ldots, x_n) = 0?$
$= 0$
$\neq 0$
$f_2(x_1, \ldots, x_n) = 0?$
$f_3(x_1, \ldots, x_n) = 0?$
...
...
...
...
true
false
false
true

Corresponding decision tree

Example: discrete dynamical systems/distributed systems

Task specification (the protocol complex is a depiction of the set of reachable states)



∃ Decision?

Protocol complex

Output complex

⊆

Protocol specification

Task specification

Input complex