Hybrid System Falsification and Reinforcement Learning

Formal Method for Cyber-Physical Systems

Clovis Eberhart David Sprunger

National Institute of Technology, Japan

SOKENDAI lesson, July 1, 8, and 22

Lecture structure

Lectures

- 1st: falsification (problem, framework, logics...), by me
- 2nd: deep learning for falsification (learning, reinforcement learning, application to falsification...), by David
- 3rd: advanced techniques in falsification and reinforcement learning, by David and me

Evaluation

Easy practical assignment (in Python).

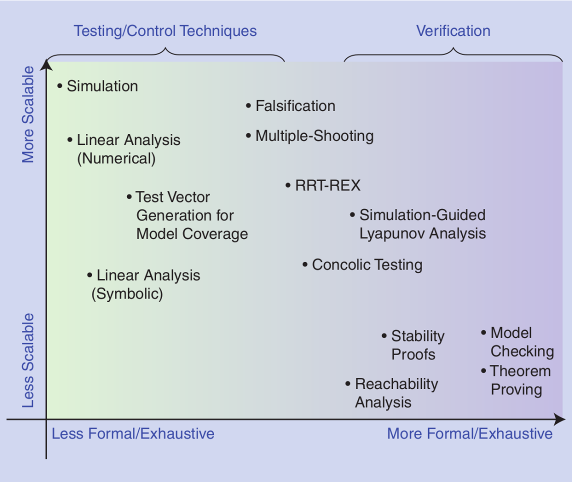
Questions?

- Ask them during the lesson.
- Find me at my desk (Palaceside building).
- `clovis.eberhart@gmail.com`

First lecture overview

Table of Contents

Formal method landscape



Kapinski, Deshmukh, Jin, Ito, Butts, *Simulation-Based Approaches for Verification of Embedded Control Systems*, IEEE Control Magazine, 2010

Testing

Testing

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a test suite $\{t_i\}_{i \in I}$.

Testing

Testing

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a test suite $\{t_i\}_{i \in I}$.

Characteristics

- simple (run tests)
- black-box (unknown systems)
- versatile (guarantees, explainable failures. . .)
- no formal guarantee
- too general

Verification

Verification

Given: a model $\mathcal{M}$, a property φ.
Goal: automatically prove that $\mathcal{M} \models \varphi$.

Verification

Verification

Given: a model $\mathcal{M}$, a property φ.
Goal: automatically prove that $\mathcal{M} \vDash \varphi$.

Characteristics

- complex (design model, use specific techniques, so typically not used by engineers)
- white-box (known systems only)
- formal proof (strong guarantee)
- ill-suited to CPS (continuous systems)

Verification

Verification

Given: a model $\mathcal{M}$, a property φ.
Goal: automatically prove that $\mathcal{M} \models \varphi$.

Characteristics

- complex (design model, use specific techniques, so typically not used by engineers)
- white-box (known systems only)
- formal proof (strong guarantee)
- ill-suited to CPS (continuous systems)

Theorem proving

Given: a model $\mathcal{M}$, a property φ.
Goal: prove that $\mathcal{M} \models \varphi$.

Falsification

Falsification

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Falsification

Falsification

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Characteristics

- particular case of testing
- black-box (unknown systems)
- relatively simple
- no proof (no formal guarantee)

Verification versus falsification

- Verification:
  - finds a proof: system verifies property,
  - finds nothing:
- Falsification:
  - finds a counterexample: system violates property,
  - finds nothing:

Verification versus falsification

- Verification:
  - finds a proof: system verifies property,
  - finds nothing: nothing can be said.
- Falsification:
  - finds a counterexample: system violates property,
  - finds nothing: nothing can be said.

Verification versus falsification

- Verification:
  - finds a proof: system verifies property,
  - finds nothing: nothing can be said.
- Falsification:
  - finds a counterexample: system violates property,
  - finds nothing: nothing can be said.

Interaction:

- verification for falsification: constraining state space by reachability analysis,
- falsification for verification: coverage-based falsification techniques.

Table of Contents

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system?

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system? $\rightsquigarrow$ hybrid system

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system? $\rightsquigarrow$ hybrid system
- What is a property?

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system? $\rightsquigarrow$ hybrid system
- What is a property? $\rightsquigarrow$ logical formula

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system? $\rightsquigarrow$ hybrid system
- What is a property? $\rightsquigarrow$ logical formula
- What is a counterexample?

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.
Goal: generate a counterexample to $\mathcal{S} \models \varphi$.

Questions

- What is a system? $\rightsquigarrow$ hybrid system
- What is a property? $\rightsquigarrow$ logical formula
- What is a counterexample? $\rightsquigarrow$ an input (and output) signal to the system that violates the property

General framework

Reminder

Given: a system $\mathcal{S}$, a property φ.

Goal: generate a counterexample to $\mathcal{S} \vDash \varphi$.

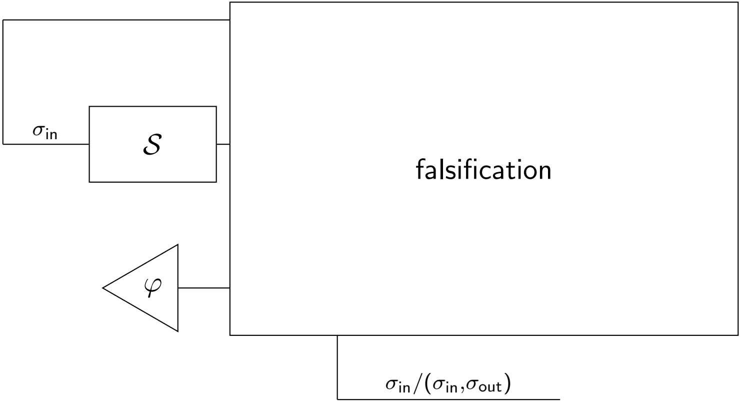
Questions

- What is a system? $\rightsquigarrow$ hybrid system
- What is a property? $\rightsquigarrow$ logical formula
- What is a counterexample? $\rightsquigarrow$ an input (and output) signal to the system that violates the property

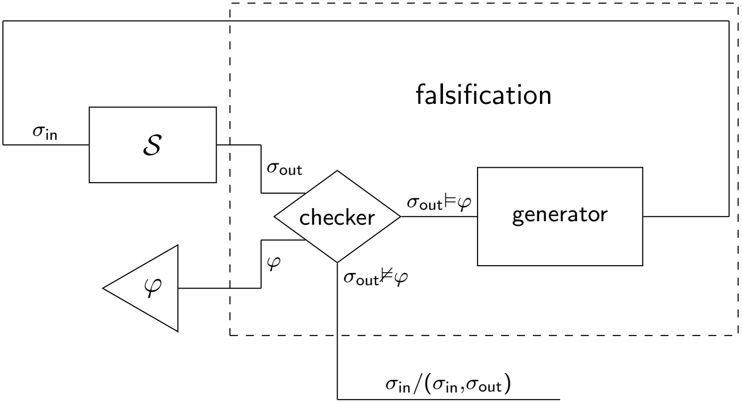
Challenges

- infinite (and high-dimensional) search space
- non-linear dynamics

The falsification loop



The falsification loop

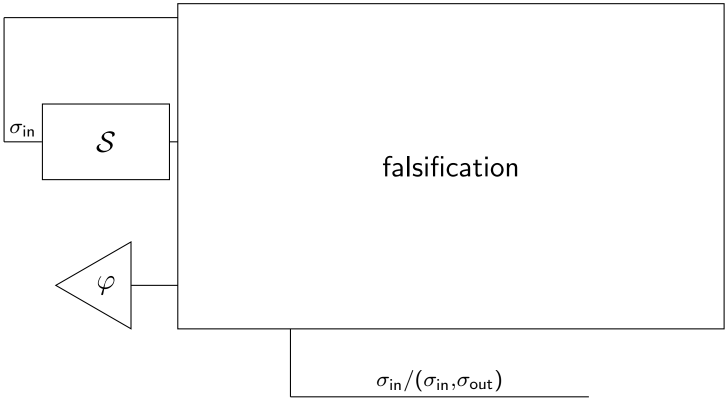


The falsification algorithm

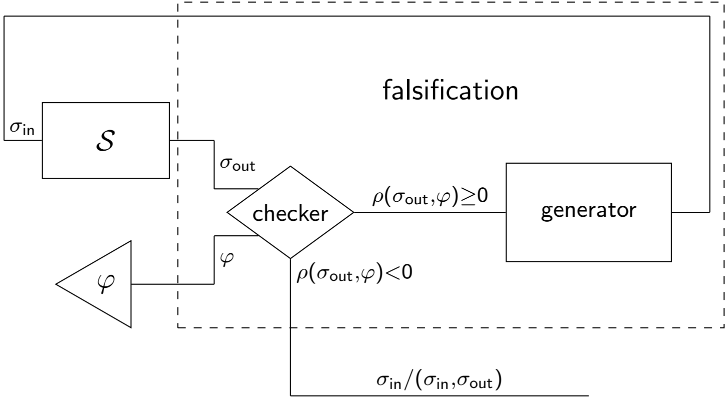
Input: A system $\mathcal{S}$, a formula φ, a satisfaction predicate $\models$, and a timeout t_{max}
Output: A signal σ_{in} such that $\mathcal{S}(\sigma_{\mathsf{in}}) \not\models \varphi$

found = false;
while *not(* found *) and* $t < t_{max}$ **do**
    $\sigma_{\mathsf{in}} =$ generate();
    $\sigma_{\mathsf{out}} = \mathcal{S}(\sigma_{\mathsf{in}})$;
    found $= \sigma_{\mathsf{out}} \not\models \varphi$;
end
if found **then**
    return σ_{in}
else
    return *"timeout"*
end

Optimisation-based falsification



Optimisation-based falsification



The optimisation-based falsification algorithm

Input: A system $\mathcal{S}$, a formula φ, a robustness function ρ, and a timeout t_{max}
Output: A signal σ_{in} such that $\mathcal{S}(\sigma_{\mathsf{in}}) \not\models \varphi$

found = false;
while *not(*found*) and* $t < t_{max}$ **do**
  $\sigma_{\mathsf{in}} =$ search_minimum(ρ);
  $\sigma_{\mathsf{out}} = \mathcal{S}(\sigma_{\mathsf{in}})$;
  found $= \rho(\sigma_{\mathsf{out}}, \varphi) < 0$;
end
if found **then**
  return σ_{in}
else
  return *"timeout"*
end

Required: $\rho(\sigma, \varphi) \geq 0 \iff \sigma \models \varphi$

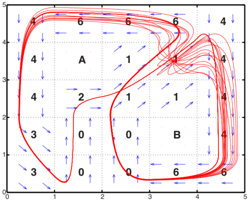
Table of Contents

Hybrid systems

Definition

A hybrid system is a dynamical system that exhibits both continuous and discrete dynamic behavior – a system that can both *flow* (described by a differential equation) and *jump* (described by a state machine or automaton).

Wikipedia



Fehnker, Ivančić, *Benchmarks for Hybrid Systems Verification*, Hybrid Systems: Computation and Control, pp 326–341

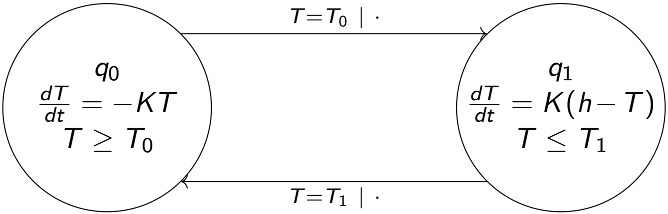
Hybrid system: definition

Hybrid system

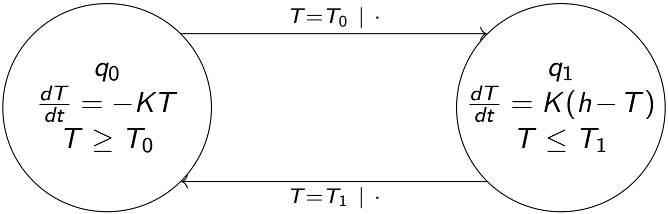
A Hybrid system is a tuple $\mathcal{H} = (Q, X, \text{GUARD}, \text{JUMP}, U, \text{FLOW})$ of:

- a finite set of modes Q,
- a family of continuous state spaces $X = \{X_q \subseteq \mathbb{R}^{n_q} \mid q \in Q\}$,
- $\text{GUARD}_{q,q'} \subseteq X_q$ is the set of states in X_q that can transition to mode q',
- $\text{JUMP}_{q,q'} : X_q \to X_{q'}$ describes the transition from q to q',
- U is the input space,
- FLOW_q, is a set of differential equations in X_q and U, seen as a function $X_q \times U \times \mathbb{R}_{\geq 0} \to X_q$,

Example of hybrid system: thermostat

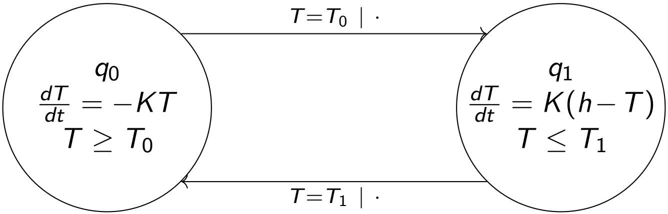


Example of hybrid system: thermostat



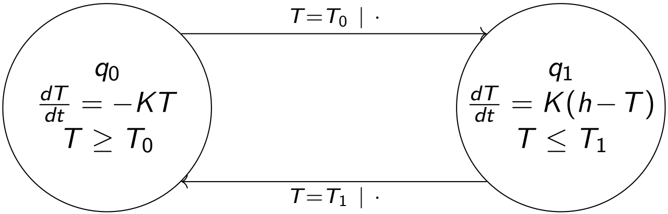
- $Q = \{q_0, q_1\}$,

Example of hybrid system: thermostat



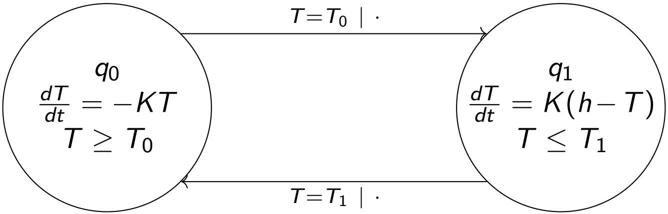
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,

Example of hybrid system: thermostat



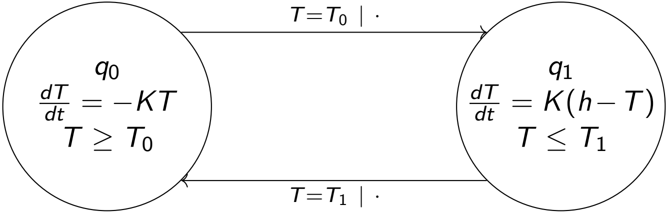
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,

Example of hybrid system: thermostat



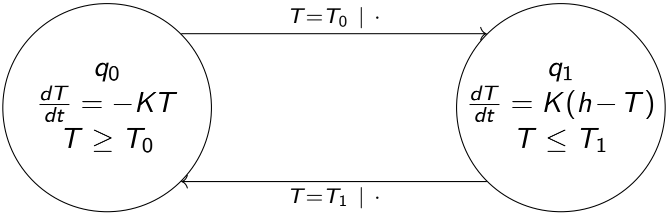
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \{m\}$

Example of hybrid system: thermostat



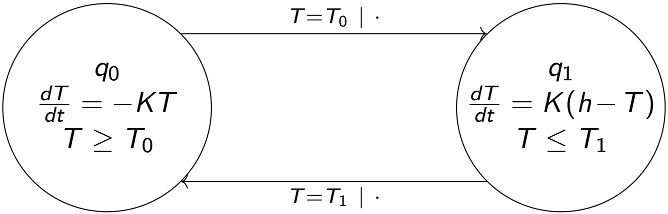
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\text{GUARD}_{q_0,q_1} = \{m\}$
- $\text{GUARD}_{q_1,q_0} = \{M\}$

Example of hybrid system: thermostat



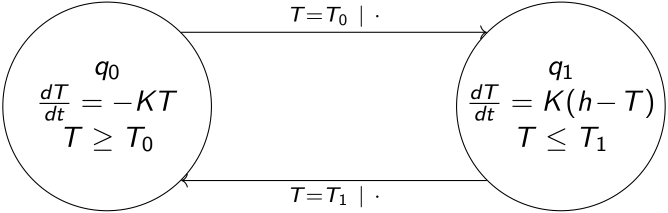
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \{m\}$
- $\mathrm{GUARD}_{q_1,q_0} = \{M\}$
- $\mathrm{JUMP}_{q,q'}(T) = T$

Example of hybrid system: thermostat



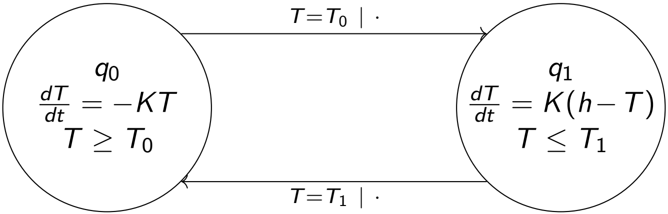
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \{m\}$
- $\mathrm{GUARD}_{q_1,q_0} = \{M\}$
- $\mathrm{JUMP}_{q,q'}(T) = T$
- $U = \emptyset$

Example of hybrid system: thermostat



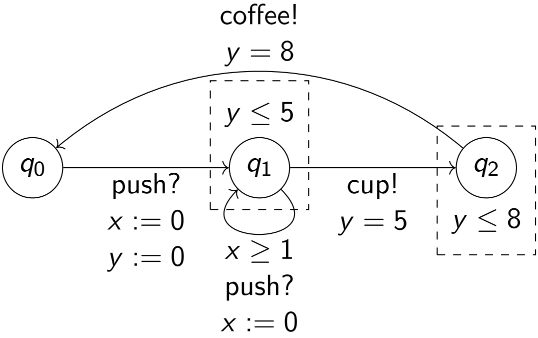
- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \{m\}$
- $\mathrm{GUARD}_{q_1,q_0} = \{M\}$
- $\mathrm{JUMP}_{q,q'}(T) = T$
- $U = \emptyset$
- $\mathrm{FLOW}_{q_0} = (\frac{dT}{dt} = -KT)$

Example of hybrid system: thermostat

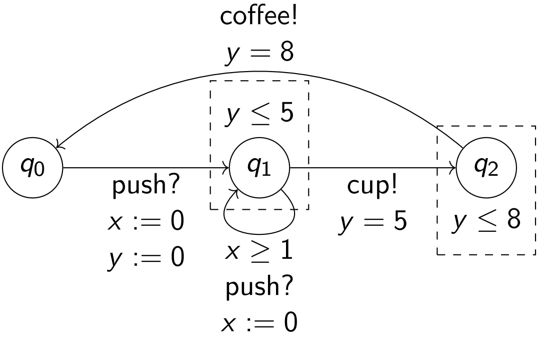


- $Q = \{q_0, q_1\}$,
- $X_{q_0} = \{T \in \mathbb{R} \mid T \geq T_0\}$,
- $X_{q_1} = \{T \in \mathbb{R} \mid T \leq T_1\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \{m\}$
- $\mathrm{GUARD}_{q_1,q_0} = \{M\}$
- $\mathrm{JUMP}_{q,q'}(T) = T$
- $U = \emptyset$
- $\mathrm{FLOW}_{q_0} = (\frac{dT}{dt} = -KT)$
- $\mathrm{FLOW}_{q_1} = (\frac{dT}{dt} = K(h - T))$

Example of hybrid system: timed automata

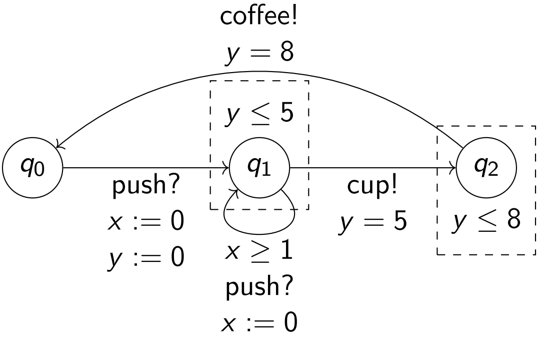


Example of hybrid system: timed automata



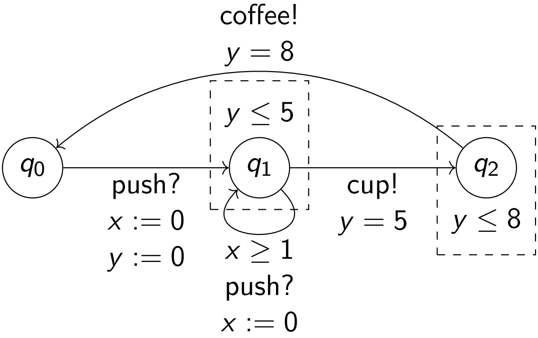
- $Q = \{q_0, q_1, q_2\}$,

Example of hybrid system: timed automata



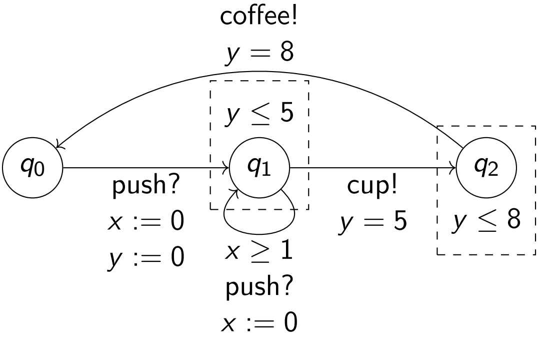
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,

Example of hybrid system: timed automata



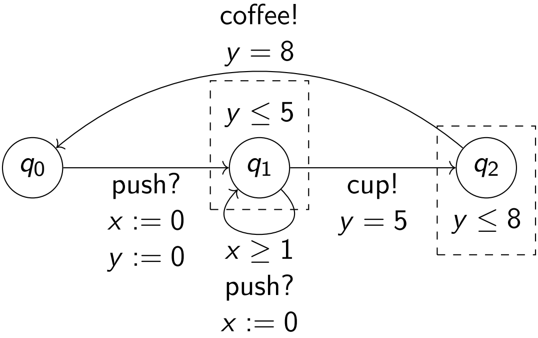
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} =$
  $\{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,

Example of hybrid system: timed automata



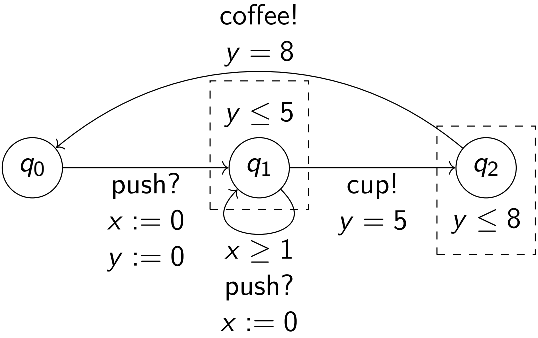
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \mathbb{R}^2$

Example of hybrid system: timed automata



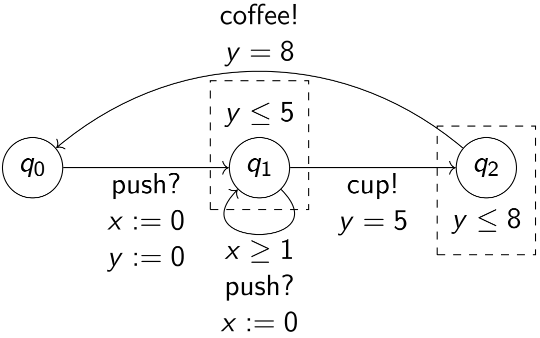
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \mathbb{R}^2$
- $\mathrm{GUARD}_{q_1,q_2} = \mathbb{R} \times \{5\}$

Example of hybrid system: timed automata



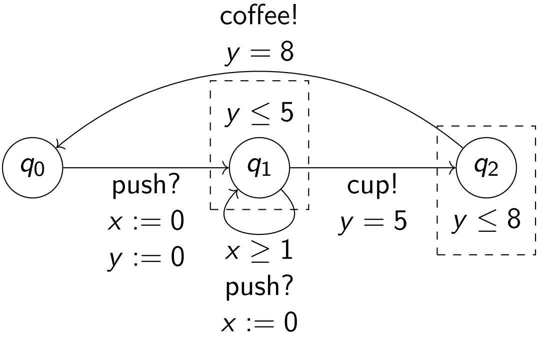
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\text{GUARD}_{q_0,q_1} = \mathbb{R}^2$
- $\text{GUARD}_{q_1,q_2} = \mathbb{R} \times \{5\}$
- $\text{JUMP}_{q_0,q_1}(x, y) = (0, 0)$

Example of hybrid system: timed automata



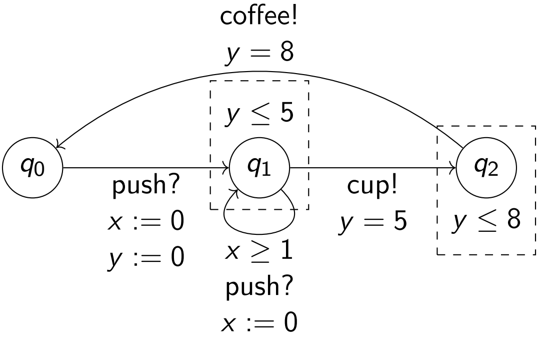
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \mathbb{R}^2$
- $\mathrm{GUARD}_{q_1,q_2} = \mathbb{R} \times \{5\}$
- $\mathrm{JUMP}_{q_0,q_1}(x, y) = (0, 0)$
- $\mathrm{JUMP}_{q_1,q_1}(x, y) = (0, y)$

Example of hybrid system: timed automata



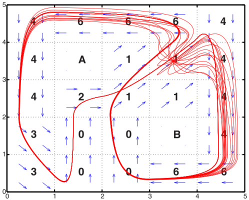
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\text{GUARD}_{q_0,q_1} = \mathbb{R}^2$
- $\text{GUARD}_{q_1,q_2} = \mathbb{R} \times \{5\}$
- $\text{JUMP}_{q_0,q_1}(x, y) = (0, 0)$
- $\text{JUMP}_{q_1,q_1}(x, y) = (0, y)$
- $U = \emptyset$

Example of hybrid system: timed automata



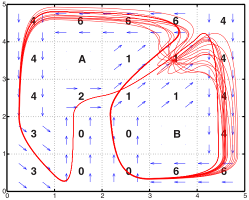
- $Q = \{q_0, q_1, q_2\}$,
- $X_{q_0} = \mathbb{R}^2$,
- $X_{q_1} = \{(x, y) \in \mathbb{R}^2 \mid y \leq 5\}$,
- $\mathrm{GUARD}_{q_0,q_1} = \mathbb{R}^2$
- $\mathrm{GUARD}_{q_1,q_2} = \mathbb{R} \times \{5\}$
- $\mathrm{JUMP}_{q_0,q_1}(x, y) = (0, 0)$
- $\mathrm{JUMP}_{q_1,q_1}(x, y) = (0, y)$
- $U = \emptyset$
- $\mathrm{FLOW}_q = (\frac{dx}{dt} = \frac{dy}{dt} = 1)$

Example of hybrid system: navigation



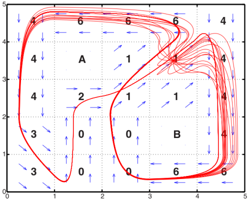
- $Q = \{(i, j) \mid i, j \in [0..4]\}$
- X:
  - $X_{(i,j)} = [0, 1] \times [0, 1] \times \mathbb{R} \times \mathbb{R}$ $(i, j \in [1..3])$
  - $X_{(0,j)} = (-\infty, 1] \times [0, 1] \times \mathbb{R} \times \mathbb{R}$ $(j \in [1..3])$
  - $X_{(4,j)} = [0, \infty) \times [0, 1] \times \mathbb{R} \times \mathbb{R}$ $(j \in [1..3])$
  - $X_{(i,0)} = [0, 1] \times (-\infty, 1] \times \mathbb{R} \times \mathbb{R}$ $(i \in [1..3])$
  - $X_{(i,4)} = [0, 1] \times [0, \infty) \times \mathbb{R} \times \mathbb{R}$ $(i \in [1..3])$
  - $X_{(0,0)} = (-\infty, 1] \times (-\infty, 1] \times \mathbb{R} \times \mathbb{R}$
  - ...

Example of hybrid system: navigation



- GUARD:
  - $\text{GUARD}_{(i,j),(i-1,j)} = \{(0, y, v_x, v_y)\}$
  - $\text{GUARD}_{(i,j),(i+1,j)} = \{(1, y, v_x, v_y)\}$
  - $\text{GUARD}_{(i,j),(i,j-1)} = \{(x, 0, v_x, v_y)\}$
  - $\text{GUARD}_{(i,j),(i,j+1)} = \{(x, 1, v_x, v_y)\}$
- JUMP:
  - $\text{JUMP}_{(i,j),(i-1,j)}(x, y, v_x, v_y) = (x + 1, y, v_x, v_y)$
  - $\text{JUMP}_{(i,j),(i+1,j)}(x, y, v_x, v_y) = (x - 1, y, v_x, v_y)$
  - $\text{JUMP}_{(i,j),(i,j-1)}(x, y, v_x, v_y) = (x, y + 1, v_x, v_y)$
  - $\text{JUMP}_{(i,j),(i,j+1)}(x, y, v_x, v_y) = (x, y - 1, v_x, v_y)$

Example of hybrid system: navigation



- $U = ([-0.1, 0.1]^{\mathbb{R}_{\geq 0}})^2$
- $\mathrm{FLOW}_{i,j}$:
  - $\frac{dx}{dt} = v_x + u_x(t)$,
  - $\frac{dy}{dt} = v_y + u_y(t)$,
  - $\frac{dv_x}{dt} = 0.1(v_y - v_y^{(i,j)}) - 1.2(v_x - v_x^{(i,j)})$,
  - $\frac{dv_x}{dt} = 0.1(v_x - v_x^{(i,j)}) - 1.2(v_y - v_y^{(i,j)})$,

  where
  - $(u_x, u_y) \in U$,
  - $v_x^{(i,j)}$ and $v_y^{(i,j)}$ are constants.

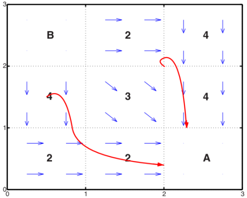
Run of a hybrid automaton

Run

Finite or infinite sequence $(q_0, x_0) \rightarrow_{t_0} (q_1, x_1) \rightarrow_{t_1} \ldots$ such that, for each $i \in [1..n-1]$:

- $y_i \in \mathrm{GUARD}(q_i, q_{i+1})$,
- $\mathrm{JUMP}_{q_i, q_{i+1}}(y_i) = x_{i+1}$,

where $y_i = \mathrm{FLOW}_{q_i}(x_i, u, t_i - t_{i-1})$.



- $((1,2), x_0) \rightarrow_{t_0} ((2,2), x_1) \rightarrow_{t_1} ((2,1), x_2) \rightarrow_{t_2} ((2,0), x_3)$
- $((0,1), x'_0) \rightarrow_{t'_0} ((0,0), x'_1) \rightarrow_{t'_1} ((1,0), x'_2) \rightarrow_{t'_2} ((2,0), x'_3)$

Runs as signals

It is often more practical to consider runs of a hybrid system as timed signals.

Signal

A signal on a set of variables $\{x_i \mid i \in [1..n]\}$, where each x_i takes values in X_i, is a function $\mathbb{R}_{\geq 0} \to \prod_{i=1}^{n} X_i$.

Translation

In the case where all X_q's are subspaces of a given X, the following signal corresponds to the run $\rho = (q_0, x_0) \to_{t_0} (q_1, x_1) \to_{t_1} \ldots$ of a hybrid automaton under input signal u:

$$\sigma_\rho : \left| \begin{array}{ccc} [0, \sum_{i=1}^{n} t_i) & \to & Q \times X \\ \left(\sum_{i=1}^{k} t_i\right) + t & \mapsto & (q_k, \mathsf{FLOW}_{q_k}(x_k, u, t)), \end{array} \right.$$

where $t < t_{k+1}$.

Signals in practice

Time-boundedness

Since simulation cannot be run for an infinite amount of time, so all considered signals are time-bounded: they are not functions $\mathbb{R}_{\geq 0} \to U$ but $[0, T) \to U$ for some $T \in \mathbb{R}$.

Finite representation of input signals

We must also stick to classes of signals that can be represented by finite means, e.g., piecewise constant signals, piecewise affine signals, spline (piecewise polynomial) signals...

Control points (those points between which the function is interpolated) can be chosen equidistant, or according to other policies.

Discretisation of signals

Discretisation of output signals

Output signals may not be finitely representable, so they are discretised.

Attention

Discretisation of signals can lead to:

- false positives: e.g., for formula $\mathsf{F}\,\varphi$, if $\sigma(t_i) \not\models \varphi$, but $\sigma(t) \models \varphi$ for some $t_i < t < t_{i+1}$, the algorithm will return σ as a (wrong) counterexample,
- false negatives: for dual reasons.

Initial position and signals

Two kinds of falsifications:

- falsification on initial position,
- falsification on signals.

Differences:

- An initial position can be seen as a constant signal, so falsification on initial positions is easier.
- Many falsification methods incrementally modify the shape of the signal (learning approaches, Monte-Carlo Tree Search...), so they are ill-suited to falsification on initial positions.

Table of Contents

Signal temporal logic

Syntax

$$\varphi ::= \top \mid f \sim 0 \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \, \mathsf{U}_I \, \varphi$$

where $I = [a, \infty)$ or $[a, b]$ for $a < b$, $f : S \to \mathbb{R}$, and $\sim \in \{>, =\}$ is a comparison operator.

Syntactic sugar

- $\bot = \neg\top$, $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi) \ldots$
- $f \geq 0 = f > 0 \vee f = 0$, $f \leq 0 = \neg f > 0 \ldots$
- $\mathsf{F}_I \, \varphi = \top \, \mathsf{U}_I \, \varphi$, $\mathsf{G}_I \, \varphi = \neg \, \mathsf{F}_I \, \neg\varphi$
- $\mathsf{U} \, \varphi = \mathsf{U}_{[0,\infty)} \, \varphi \ldots$

STL examples

To avoid awkward notations, $f \sim 0$ is never written as a function, but as a computation on states. Thus, if $f : (q, x, y, v_x, v_y) \mapsto x^2 + y^2 - 1$, $f < 0$ would be written $x^2 + y^2 - 1 < 0$ or $x^2 + y^2 < 1$.

Examples

- $q \neq (3,1) \, \mathsf{U} \, q = (1,3)$
- $\mathsf{G}\,(\text{gear} = 2 \rightarrow \neg\, \mathsf{F}_{[0,\varepsilon]}\,(\text{gear} \neq 2 \wedge \mathsf{F}_{[0,\tau]}\,\text{gear} = 2))$
- $(q = q_i \wedge x \in [x_i^-, x_i^+]) \wedge \mathsf{F}_{[0,T]}\,(q = q_f \wedge x \in [x_f^-, x_f^+])$
- $(q = q_i \wedge x \in [x_i^-, x_i^+]) \wedge \mathsf{G}_{[0,T]}\,\neg(q = q_f \wedge x \in [x_f^-, x_f^+])$
- $\mathsf{G}\,(\text{danger} \rightarrow \mathsf{F}_{[0,t]}\,\neg\text{danger})$ with $\text{danger} = (q = q_d \wedge x \in [x_d^-, x_d^+])$

Boolean semantics of STL

The semantics of STL formulas is defined over timed signals:

Boolean semantics

$$
\begin{aligned}
\sigma, t \models \top &\iff \top \\
\sigma, t \models f \sim 0 &\iff f(\sigma(t)) \sim 0 \\
\sigma, t \models \neg\varphi &\iff \sigma, t \not\models \varphi \\
\sigma, t \models \varphi \wedge \psi &\iff \sigma, t \models \varphi \text{ and } \sigma, t \models \psi \\
\sigma, t \models \varphi \, \mathsf{U}_{[a,b]} \, \psi &\iff \text{there is } t' \in [a, b] \text{ such that } \sigma, t' \models \psi \\
& \text{ and for all } t'' \in [a, t'),\ \sigma, t'' \models \varphi
\end{aligned}
$$

$\sigma \models \varphi$ stands for $\sigma, 0 \models \varphi$.

Robustness semantics of STL

We can define a robustness semantics, whose value is not a boolean, but a real:

Robustness semantics

$$
\begin{aligned}
\rho(\sigma, \top, t) &= \infty \\
\rho(\sigma, f > 0, t) &= f(\sigma(t)) \\
\rho(\sigma, f = 0, t) &= -|f(\sigma(t))| \\
\rho(\sigma, \neg\varphi, t) &= -\rho(\sigma, \varphi, t) \\
\rho(\sigma, \varphi \wedge \psi, t) &= \min(\rho(\sigma, \varphi, t), \rho(\sigma, \psi, t)) \\
\rho(\sigma, \varphi \, \mathsf{U}_{[a,b]} \, \psi, t) &= \sup_{t' \in [a,b]} \min\left(\rho(\sigma, \psi, t'), \inf_{t'' \in [a,t')} \rho(\sigma, \varphi, t'')\right)
\end{aligned}
$$

$\rho(\sigma, \varphi)$ stands for $\rho(\sigma, \varphi, 0)$.

Soundness of robustness semantics

Theorem

If $\rho(\sigma, \varphi) > 0$, *then* $\sigma \models \varphi$.
If $\rho(\sigma, \varphi) < 0$, *then* $\sigma \not\models \varphi$.

Proof.

Structural induction. □

Remark

If $\rho(\sigma, \varphi) = 0$, nothing can be said: if $\sigma(x)(0) = x_0$ for $\varphi = (x = x_0)$:

- $\rho(\sigma, \varphi) = \rho(\sigma, \neg\varphi) = 0$, but
- $\sigma \models \varphi$, $\sigma \not\models \neg\varphi$.

Advantage: if functions f are continuous or smooth, we can use optimisation techniques to find the minimal robustness.

Computing the robustness semantics

Computing ρ

Given: piecewise-constant σ (because discretised).
Goal: compute $\rho(\sigma, \varphi)$.

Naive idea: compute $\rho(\sigma, \varphi, t)$ inductively.

Problem

Complexity to compute $\rho(\sigma, \varphi \, \mathsf{U}_{[a,b]} \, \psi, -)$ is
$\mathcal{O}$(number of control points $\times$ number of control points in $[a, b]$)

Efficiently computing $\rho(\sigma, \varphi, -)$

First step: get rid of U_I :

- $\varphi \, \mathsf{U}_{[a,b]} \, \psi \equiv \mathsf{F}_{[a,b]} \, \varphi \wedge \mathsf{G}_{[0,a]} \, (\varphi \, \mathsf{U} \, \psi)$
- $\varphi \, \mathsf{U}_{[a,\infty)} \, \psi \equiv \mathsf{G}_{[0,a]} \, (\varphi \, \mathsf{U} \, \psi)$

Goal

The function $t \mapsto \rho(\sigma, \varphi, t)$ is a signal $\mathbb{R}_{\geq 0} \to \mathbb{R}$ denoted $\rho(\sigma, \varphi, -)$. Our goal is to compute it from $\rho(\sigma, \psi, -)$ for subformulas ψ of φ.

Need to know how to recursively compute signals for $\top$, $f \sim 0$, $\neg\varphi$, $\varphi \vee \psi$, $\varphi \, \mathsf{U} \, \psi$, and $\mathsf{F}_{[a,b]} \, \varphi$ in $\mathcal{O}$(number of control points).

Computing $\rho(\sigma, \varphi \, \mathsf{U} \, \psi, -)$

Without loss of generality, signals y and y' representing $\rho(\sigma, \varphi, -)$ and $\rho(\sigma, \psi, -)$ have the same control points t_i (otherwise, take the union of control points).
Let z be the signal corresponding to $\rho(\sigma, \varphi \, \mathsf{U} \, \psi, -)$, then:

$$\begin{aligned}
z(t_i) &= \sup_{t \in [t_i, \infty)} \min\{y'(t), \inf_{[t_i, t)} y\} \\
&= \max\{\min\{y'(t_i), y(t_i)\}, \sup_{t \in [t_{i+1}, \infty)} \min\{y'(t), \inf_{[t_i, t)} y\}\} \\
&= \max\{\min\{y'(t_i), y(t_i)\}, \min\{y(t_i), \sup_{t \in [t_{i+1}, \infty)} \min\{y'(t), \inf_{[t_{i+1}, t)} y\}\}\} \\
&= \min\{y(t_i), \max\{y'(t_i), z(t_{i+1})\}\}
\end{aligned}$$

Thus, there is an algorithm to compute $\rho(\sigma, \varphi \, \mathsf{U} \, \psi, -)$ from $\rho(\sigma, \varphi, -)$ and $\rho(\sigma, \varphi, -)$ whose complexity is linear in the number of control points.

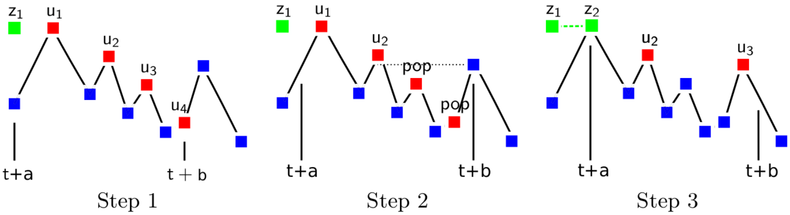
Computing $\rho(\sigma, \mathsf{F}_{[a,b]}\,\varphi, -)$

We need to compute $z(t) = \sup_{t+[a,b]} y = \max_{t_i \in t+[a,b]}\{y(t_i)\}$.

Idea

Compute M such that $i \in M$ iff $t_i \in t + [a, b]$ and for all $t_j \in t + [a, b]$, $y(t_j) < y(t_i)$.

Thus: $y(t_{\min M}) = \max_{t_i \in t+[a,b]}\{y(t_i)\}$.



Alexandre Donzé, Thomas Ferrère, Oded Maler, *Efficient Robust Monitoring for STL*, International Conference on Computer Aided Verification 2013

Analysis

- M can be implemented to have all operations in $\mathcal{O}(1)$ (doubly-linked list),
- all control points are popped at most once,
- in total, the number of comparisons (done when searching for elements to pop) is at most $2n$,

therefore, computing $\rho(\sigma, \mathsf{F}_{[a,b]}\,\varphi, -)$ can be done in time linear with respect to the number of control points.

Overall complexity

The number of control points is at most $d^{h(\varphi)}|\sigma|$, so the whole complexity is $\mathcal{O}(|\varphi|d^{h(\varphi)}|\sigma|)$.

Generalisation

The same argument (but more complex) applies to more general signals (say, piecewise affine).

Table of Contents

Optimisation techniques

Falsification is based on a number of optimisation techniques:

- ant colony,
- CMA-ES,
- cross-entropy,
- gradient descent,
- hill-climbing,
- Nelder-Mead,
- simulated annealing,
- ...

Gradient descent

Goal: find a *local* minimum to a function f.

$i = 0$;
while *continue* **do**
    $x_{i+1} = x_i - \gamma_i \cdot \nabla(f)(x_i)$;
    best $= x_{i+1}$;
    if $f(x_{i+1}) > f(x_i)$ **then**
        continue $=$ false;
        best $= x_i$;
    end
    $i++$;
end
return *best*

Finds a local minimum: may be useful in certain cases.

Hill climbing

Goal: find a *local* maximum to a function f.

```
i = 0;
while continue do
    for x' ∈ neighbours(x_i) do
        if f(x') > f(x_i) then
            x_{i+1} = x';
            break;
        end
    end
    continue = x_{i+1} ≠ x_i;
    i++;
end
return x_{i-1}
```

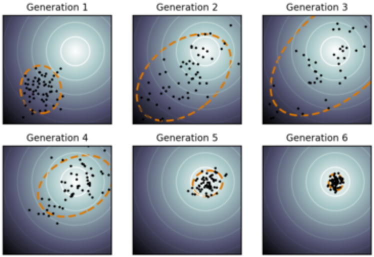
- Finds a local maximum: may be useful in certain cases.
- Simpler than gradient descent (no derivatives to compute), but less efficient.

CMA-ES

CMA-ES: Covariance Matrix Adaptation Evolution Strategy

Goal: find the maximum of a function f on a space X.

while *continue* **do**
$x_1, \ldots, x_n = \text{sample-multivariate-normal}(m, \sigma^2 C);$
$x_1, \ldots, x_n = \text{sort}(x_1, \ldots, x_n, f);$
$m, \sigma, C = \text{update}(x_1, \ldots, x_n, m, \sigma, C);$
end
return m



Wikipedia

CMA-ES for falsification

Characteristics

- evolutionary algorithm (the parameters evolve towards a better sampler)
- sampling-based $\rightsquigarrow$ no need to compute derivatives

Adaptation

Take X to be the space of signals σ and f to be $-\rho(\sigma, \varphi)$.

Remark

This is a way to adapt sampling-based optimisation to falsification, so it works for other such algorithms, such as the cross-entropy method.

Nelder-Mead method

Goal: find the maximum of a function f on a space X of dimension n.

```
1 x_1, ..., x_{n+1} = simplex();
2 while continue do
3     x_i = worst-vertex(x_1, ..., x_{n+1}, f);
4     x'_i = reflect(x_i, (x_j)_{j≠i});
  end
5 return best-vertex(f)
```

- 1: the structure is just a simplex.
- 3: when reflecting a vertex, if the new value is much better than the previous one, we keep stretching, otherwise, we shrink.

Adaptation to falsification

Take X a subspace of signals of finite dimension (e.g., fix a shape and control points), and $f = -\rho(\sigma, \varphi)$.

Ant colony

Goal: find a path in a graph that maximises performance.

1 **while** *continue* **do**
2   **foreach** *"ant" i in colony* **do**
3         $p_i =$ construct-solution(i);
4   local-update-pheromones(i,f);
    end
5   global-update-pheromones();
6     $p =$ argmax(p_i,f)
end
7 **return** p

- 3: ants walk randomly on the graph, choosing neighbours with more pheromone more often.
- 4: ants put pheromone on their chosen edge, making it more attractive; the better the ant's solution (i.e., the smaller $f(p_i)$), the more pheromone she puts.
- 5: pheromone gets put on all edges taken by the best solution.

Ant colony for falsification

Adaptation

Take G to be the graph:

- whose vertices are input signals σ,
- there is an edge between σ and τ when they are "close enough", say, $\|\sigma - \tau\|_\infty < C$, i.e., for all $t < t_{\max}$ and component x of U, $|\sigma(t)(x) - \tau(t)(x)| < C$,

and $f(\sigma_1 \ldots \sigma_n)$ to be $-\rho(\sigma_n, \varphi)$.

The ant colony algorithm tries to find a path that maximises f, i.e., minimises $\rho(\sigma, \varphi)$.

Remark

This is a way to adapt optimisation of a function on a graph to falsification, so it works for other optimisation algorithms that work on graphs, such as simulated annealing.

Simulated annealing

Goal: find a global maximum of a function f on a space X.

$T = T_0$;
$i = 0$;
while $T(i) > 0$ **do**
    $x_{i+1} = \text{neighbour}(x_i)$;
    if $E(x_{i+1}) < E(x_i)$ *and* $rand() < P(E(x_{i+1}), E(x_i), T)$ **then**
        $x_{i+1} = x_i$;
    end
    $i{+}{+}$;
end
return x_{i-1}

- Avoids local minima by "cooling" the system down progressively.
- Many parameters $\rightsquigarrow$ adaptative, but tricky.

Conclusion

- Falsification:
  - method to find counterexamples to a property,
  - useful in the world of formal methods,
  - black-box method,
  - relies on optimisation algorithms.
- Hybrid system:
  - continuous and discrete parameters,
  - non-linear behaviour,
  - very expressive.
- Formulas:
  - expressed in a temporal logic,
  - boolean and robustness semantics.