Verifying the uncountable: LTL verification in continuous-time

Iman Shames

CIICADA Lab, College of Engineering and Computer Science, ANU

Foundations of Computing/Computer Science

D Selvaratnam, M Cantoni, J M Davoren, I Shames, "Sampling polynomial trajectories for LTL verification", Theoretical Computer Science, Volume 897, 2022, Pages 135-163

Outline

Formal Verification – Trusting an Autonomous System

Temporal Logic – Writing Task Specifications

`PolyTrace` Algorithm – Checking Task Specifications

Examples and Conclusions

Why verify?

- Path planning is extremely hard
- Approximations make it tractable
- Some requirements must be relaxed for planning
- Uncertainty: plan for expected value, but verify against worst case?

Formal verification for checking a plan against requirements that path planner may not be able to guarantee

$\overbrace{\textbf{Trusted}}^{\text{formal verification}}\ \underbrace{\textbf{Autonomous}}_{\text{path planning}}\ \textbf{Systems}$

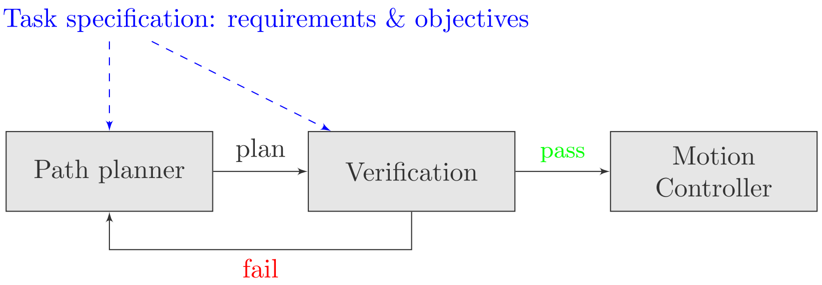
A disclaimer from the 1662 revision of the Book of Common Prayer[1]

> And having thus endeavoured to discharge duties in this weighty affair, as in the sight of God, and to approve our sincerity therein (so far as lay in us) to the consciences of all men; although we know it impossible (in such variety of apprehension, humours and interests, as are in the world) to please all; nor can expect the men of factious, peevish, and perverse spirits should be satisfied with anything that can be done in this kind by other than themselves...

- One might find the lack of optimism disheartening, but one cannot fault the aforementioned preface's authors' realism.
- I feel the same as these authors.

[1] I have found this in the outstanding book *Surpassing Wonder: The Invention of the Bible and the Talmuds by Donald Harman Akenson*

Envisaged architecture



Interaction between modules

Some interdependence between planning and verification
Temporal logic: mathematical language enabling precise Task specifications

Formal Verification – Trusting an Autonomous System

Temporal Logic – Writing Task Specifications

`PolyTrace` Algorithm – Checking Task Specifications

Examples and Conclusions

Temporal Logic

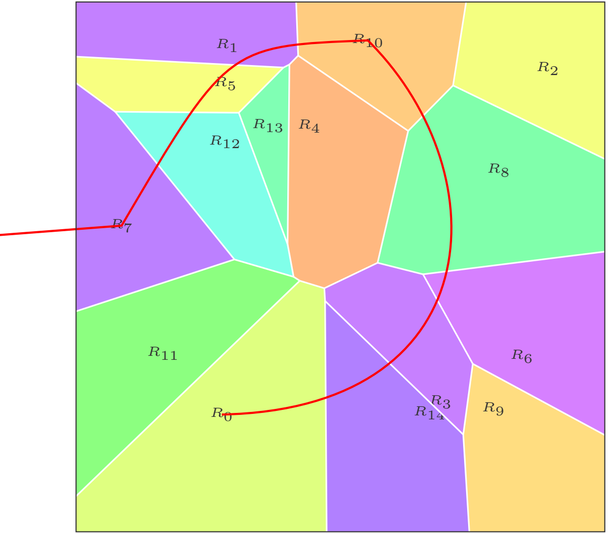
Formal mathematical language

- Linear temporal logic (LTL)
- Signal Temporal Logic (STL) $\leftarrow$ explicit timing bounds

Used in computer science to specify complex software & hardware requirements
Captures discrete objectives with logical dependencies that may change over time
{`and`, `or`, `not`, `implies`, `always`, `eventually`, `until` }
Verification output is binary: pass or fail (robustness notions do exist)

Seek + Avoid:

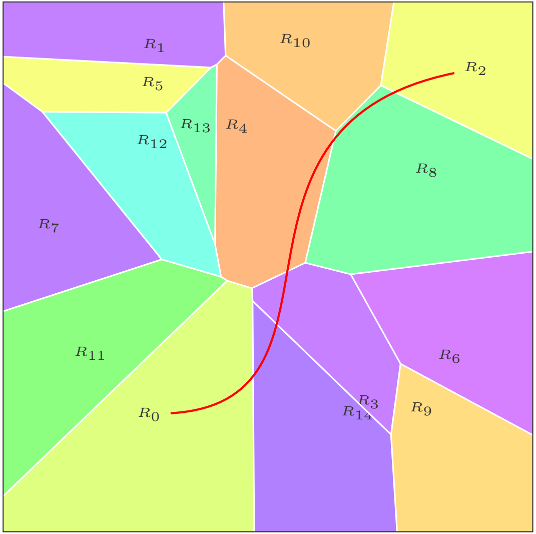
$\phi := R_0 \wedge \Diamond R_7 \wedge \Diamond R_{10} \wedge \Box \neg R_4$



$\wedge$ = and
$\Diamond$ = eventually
$\Box$ = always
$\neg$ = not

Path planning:

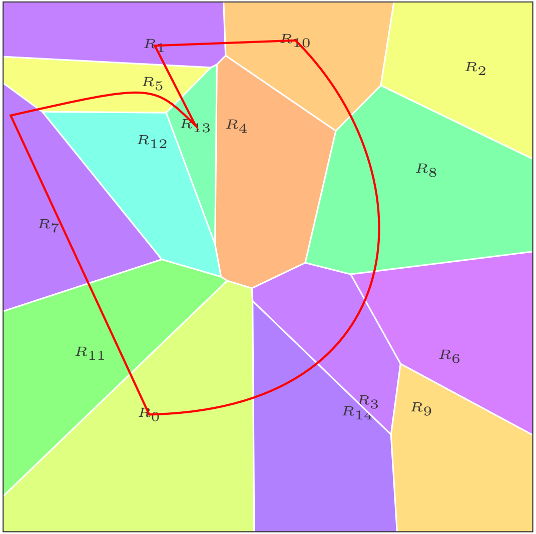
$\phi := ((((R_0 \mathcal{U} R_{14}) \mathcal{U} R_3) \mathcal{U} R_4) \mathcal{U} R_8) \mathcal{U} \square R_2$



$\mathcal{U}$ = until
$\square$ = always

Persistent Surveillance + Avoid:

$\phi := \Box(\Diamond R_0 \wedge \Diamond R_6 \wedge \Diamond R_{13} \wedge \Diamond R_{11}) \wedge \Box\neg(R_4 \vee R_{12})$



$\Box$ = always
$\Diamond$ = eventually
$\wedge$ = and
$\neg$ = not
$\vee$ = or

Paths

State: $\mathbf{x} := (x, y, z, v, \varphi, \rho) \in \mathbb{R}^6$ ←state space
x, y, z = position coordinates
v = speed
φ = fuel level
ρ = "risk"
Path: $r : [0, L] \to \mathbb{R}^6$,

$$r(s) = \begin{bmatrix} x(s) & y(s) & z(s) & v(s) & \varphi(s) & \rho(s) \end{bmatrix}^\top$$

Describes continuous evolution of key system variables

For more info on STL + Risk see:

S. Safaoui, L. Lindemann, D. V. Dimarogonas, I. Shames and T. H. Summers, "Control Design for Risk-Based Signal Temporal Logic Specifications," in IEEE Control Systems Letters, vol. 4, no. 4, pp. 1000-1005, Oct. 2020, doi: 10.1109/LCSYS.2020.2998543.

Encoding objectives and constraints

Constraints and **discrete objectives** are well suited to temporal logic verification.

$$\mathbf{x} := (x, y, z, v, \varphi, \rho)$$

Home base: $R_{home} = \{\mathbf{x} \in \mathbb{R}^6 \mid (x, y) \in [0, 1]^2\}$
Full tank: $R_{fuel} = \{\mathbf{x} \in \mathbb{R}^6 \mid \varphi \geq 0.95\}$
Low-risk Mode: $R_{low-risk} = \{\mathbf{x} \in \mathbb{R}^6 \mid \rho \leq \epsilon\}$
Over-speed: $R_{overspeed} = \{\mathbf{x} \in \mathbb{R}^6 \mid |v| > V_{max}\}$
Unsafe territory: $R_{unsafe} = \{\mathbf{x} \in \mathbb{R}^6 \mid (x, y) \in [-2, 3] \times [7, 9]\}$
Specification:

$$R_{fuel} \wedge \Box(\neg R_{overspeed} \wedge (R_{unsafe} \rightarrow R_{low-risk})) \wedge \Diamond\Box R_{home}$$

LTL Semantics

Alphabet: $A = \{R_1, ..., R_N\}$
$R_1, R_2, ...R_N \subset \mathbb{R}^n$ may overlap and need not partition the space
An infinite word α maps discrete time steps to regions

$$\alpha : \mathbb{N} \rightarrow 2^A$$

$\alpha(k) = \{R_1, R_3\} \leftarrow$ agent in regions R_1 and R_3 at time k
$\alpha(k+1) = \emptyset \leftarrow$ agent not in any regions at time $k+1$
Satisfaction:

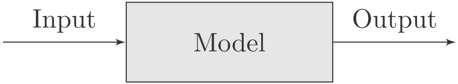
$$\alpha \models \phi \leftarrow \text{LTL formula}$$

Model Checking and **Path Checking** are two approaches to testing satisfacion

Model Checking

A **model** serves as a finite representation of its set of output words

state machines, transition systems, state-space, transfer functions



Model checking verifies whether **every** output satisfies ϕ

- exhaustive certificate
- finite transitions systems are decidable (software models)

Downsides:

- computationally expensive
- infinite state models usually undecidable
- often overkill

Path Checking

Focus on checking a **single word**: $\alpha \models \phi$

Need a finite representation of the words we want to check

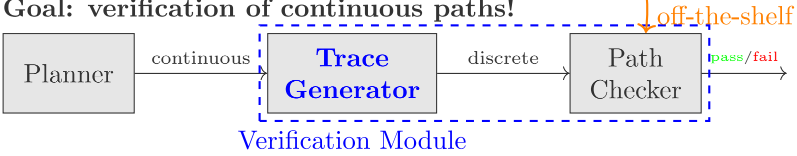
- variant of LTL for finite words
- **lasso words**: $\alpha\beta\beta\beta\ldots$

Advantages:

- Cheaper than model checking
- Possible for all paths of practical interest
- Often all you need for online decision making

Problem Statement

Model checking the path planner is intractable
Path checking algorithms for discrete words exist
$\alpha : \mathbb{N} \to 2^A$ $\leftarrow$ *countable set*
A robot residing in the physical universe has trajectories that are continuous paths
$r : [0, L] \to \mathbb{R}^n$ $\leftarrow$ *uncountable set*
Goal: verification of continuous paths!



trace is a **discrete** word capturing all transitions taken by continuous path

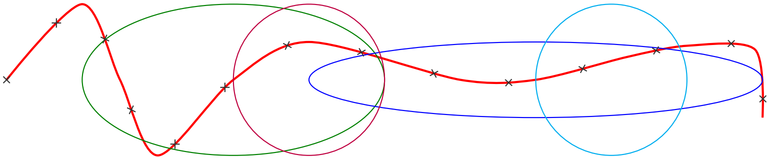
Formal Verification – Trusting an Autonomous System

Temporal Logic – Writing Task Specifications

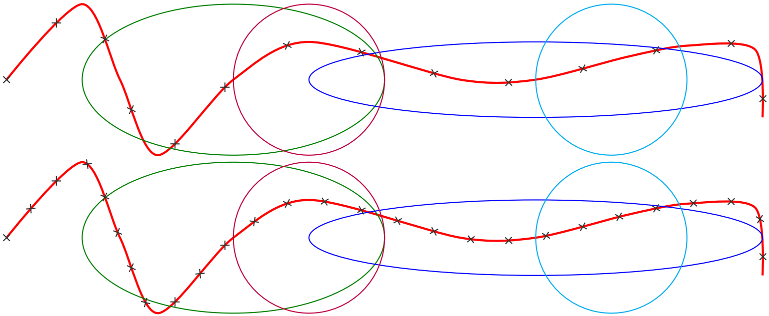
`PolyTrace` Algorithm – Checking Task Specifications

Examples and Conclusions

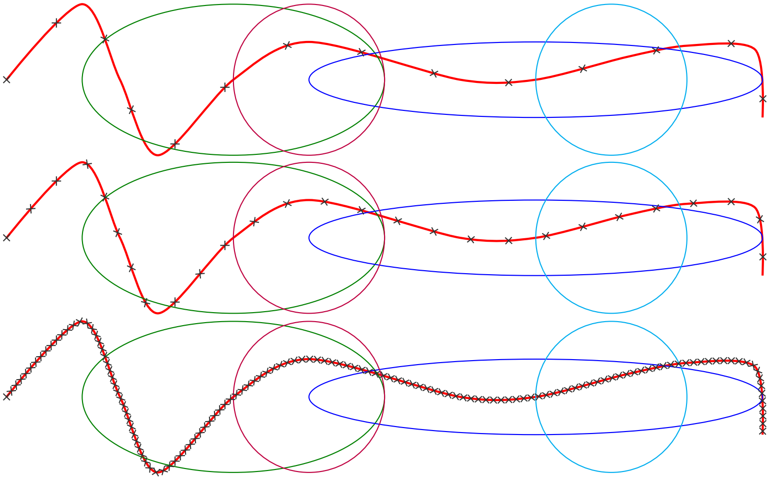
How to generate a trace?



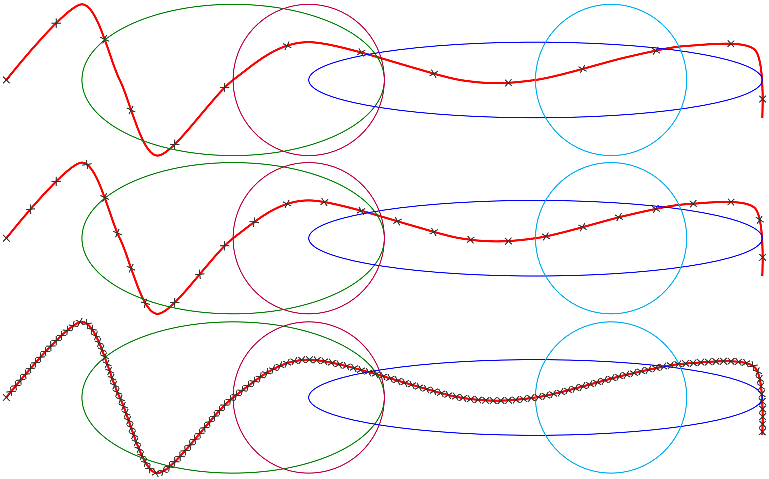
How to generate a trace?



How to generate a trace?



How to generate a trace?



Uniform Sampling: Inefficient. No guarantees. **Is there a better way to sample?**

Key objective: we don't want to keep sampling until the cows come home...

Geometry & Algebra of the Path

Path $r : [0, L] \to \mathbb{R}^2$
Between two waypoints $s \in [s_n, s_{n+1}]$,

$$\text{Example:} \quad r(s) = \begin{bmatrix} r_1(s) \\ r_2(s) \end{bmatrix} = \begin{bmatrix} a_3 s^3 + a_2 s^2 + a_1 s + a_0 \\ b_3 s^3 + b_2 s^2 + b_1 s + b_0 \end{bmatrix} \in \mathbb{R}^2$$

Semi-algebraic Region : $R_i = \{x \in \mathbb{R}^2 \mid g_i(x) \leq 0\}$,

$$g_i(x, y) = c_1 x^2 + c_2 y^2 + c_3 xy + c_4 x + c_5 y + c_6$$

- circles, ellipses, hyperbolae + intersections & unions
- straight lines, walls, grid cells

Composition:

$$g_i \circ r(s) = g_i(r_1(s), r_2(s))$$

is a 6th order univariate polynomial

Roots of $g_i \circ r$ correspond to **boundary crossings!**

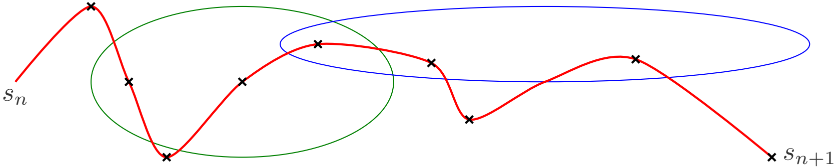
Strategy

Every boundary crossing is a root of

$$P(s) = \prod_i g_i \circ r(s).$$

For each segment $[s_n, s_{n+1}]$, ← between two waypoints

1. Find the roots of P in $[s_n, s_{n+1}]$ to get the boundary crossings
2. Sample on either side of each root (check sign of each $g_i \circ r$)

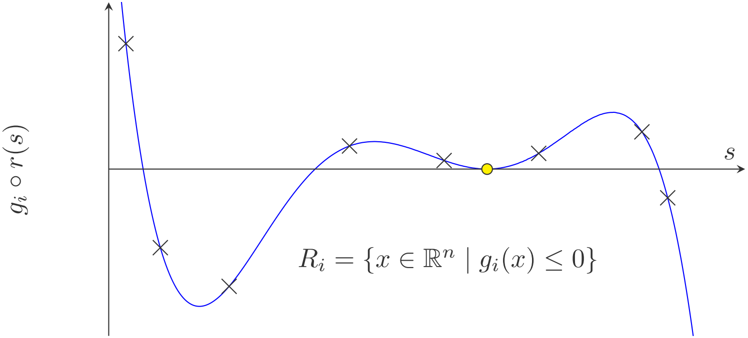


Abel-Ruffini Theorem: no general algebraic expression for roots of polynomials of degree greater than 4!

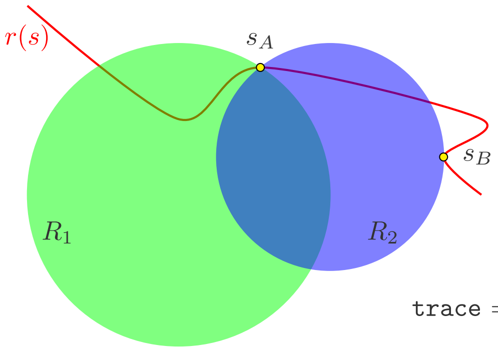
Solution Component: Root Isolation Algorithms

Given any univariate polynomial $p(s)$, a root isolation algorithm generates a set of intervals such that

- each interval contains exactly one root of p
- every root of p is contained in an interval



Isolated Points



The trace information is needed at isolated points! We don't know where they are and approximate roots miss transitions.

$$\texttt{trace} = \big(\emptyset, \{R_1\}, \underbrace{\{R_1, R_2\}}_{s_A}, \{R_2\}, \emptyset, \underbrace{\{R_2\}}_{s_B}, \emptyset\big)$$

Types of isolated points:

- double crossing: $g_1 \circ r(s_A) = g_2 \circ r(s_A) = 0$
- bouncing: $(g_2 \circ r)'(s_B) = g_2 \circ r(s_B) = 0$

All isolated points are repeated roots of $P = \prod_{i=1}^{M} g_i \circ r$.

Isolated Point Conundrum

Observation function $h : \mathbb{R}^n \to 2^{\{R_1,\ldots,R_M\}}$

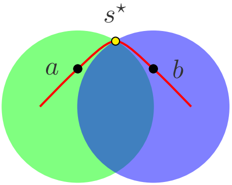
$$h(x) = \{R_i \mid x \in R_i\}$$

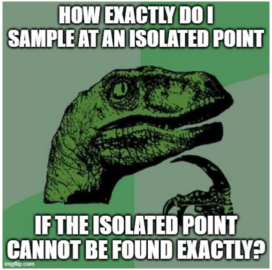
$h \circ r(s)$ samples path $r : [0, L] \to \mathbb{R}^n$ at point $s \in [0, L]$

Let $P(s^\star) = 0$. If (a, b) is an isolating interval for $s^\star$, then

$$h \circ r(s^\star) = h \circ r(a) \cup \{R_i \mid \exists s \in (a, b),\ g_i \circ r(s) = 0\}$$

$h \circ r(s^\star) = \{R_1\} \cup \{R_1, R_2\}$





Isolated Point Conundrum

Observation function $h : \mathbb{R}^n \to 2^{\{R_1,\ldots,R_M\}}$

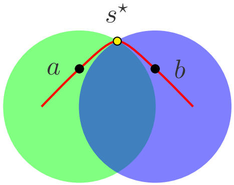
$$h(x) = \{R_i \mid x \in R_i\}$$

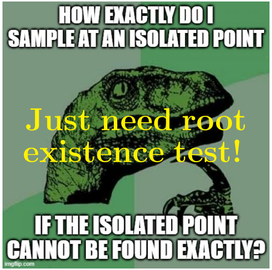
$h \circ r(s)$ samples path $r : [0, L] \to \mathbb{R}^n$ at point $s \in [0, L]$

Let $P(s^\star) = 0$. If (a, b) is an isolating interval for $s^\star$, then

$$h \circ r(s^\star) = h \circ r(a) \cup \{R_i \mid \exists s \in (a, b),\ g_i \circ r(s) = 0\}$$

$h \circ r(s^\star) = \{R_1\} \cup \{R_1, R_2\}$





Descartes' Rule of Signs

Let $p(s)$ be a non-zero univariate polynomial, and $n := \deg(p)$. Then p has no roots in (a, b) **if and only if** all the coefficients of

$$q(s) := (s+1)^n p\left(\frac{as+b}{s+1}\right)$$

have the same sign.

Computationally robust root existence test derived from Descartes' rule of signs
Arithmetic operations on the coefficients of p.
All the pieces are now in place.

Encyclopædia Britannica: *René Descartes*, National Library of Medicine

PolyTrace Algorithm

Extracts repeated roots of P

All **boundary points** are roots of

$$P(s) = \prod_{i:g_i \circ r \neq 0} g_i \circ r(s).$$

All **isolated points** are roots of $V = \gcd(P, P')$
For each segment $[s_n, s_{n+1}]$:

1. Use root isolation algorithm to sample between each root of P in $[s_n, s_{n+1}]$
2. If V has roots in $[s_n, s_{n+1}]$, "sample at" isolated points via root existence method

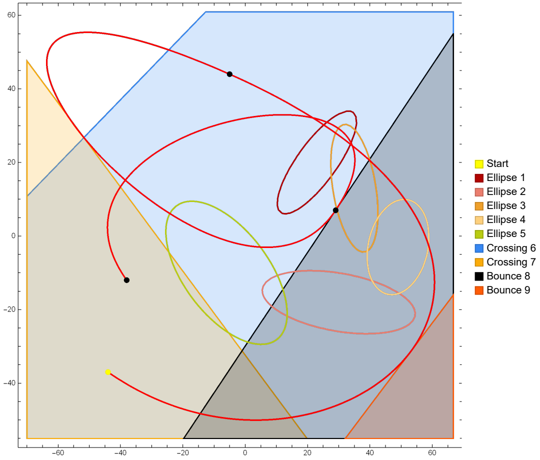
Formal Verification – Trusting an Autonomous System

Temporal Logic – Writing Task Specifications

`PolyTrace` Algorithm – Checking Task Specifications

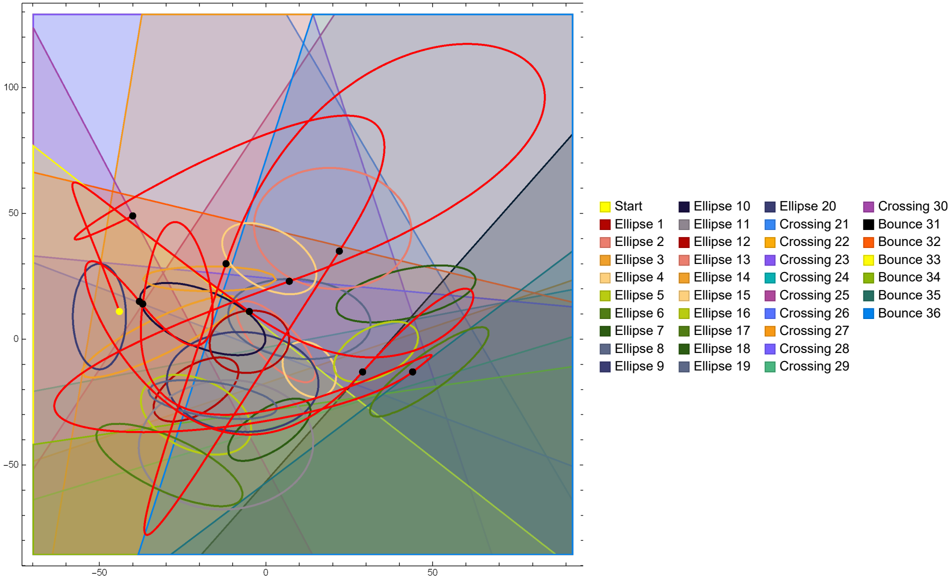
Examples and Conclusions

Simulation: eyeball



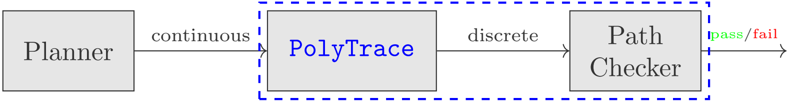
output=$\{6,7\}, \{6,7,8\}, \{6,8\}, \{6,8,9\}, \{6,8\}, \{4,6,8\}, \{6,8\}, \{3,6,8\}, \{3,6\}, \{1,3,6\}, \{1,6\}, \{6\}, \{\}, \{7\}, \{6,7\}, \{6\}, \{5,6\}, \{6\}, \{6,8\}, \{6\}, \{3,6\}, \{1,3,6\}, \{1,6\}, \{6\}, \{6,7\}$

Simulation: nightmare



Simulation: nightmare trace

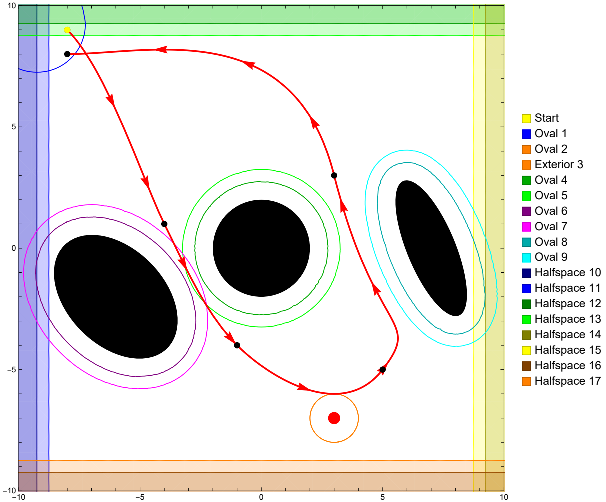
```
Out[489]= {{20, 21, 23, 26, 27, 28, 30, 32, 33}, {20, 21, 23, 27, 28, 30, 32, 33}, {21, 23, 27, 28, 30, 32, 33}, {21, 23, 30, 32, 33}, {21, 23, 27, 30, 32, 33}, {21, 23, 27, 32, 33}, {21, 23, 27, 32}, {21, 23, 25, 27, 32}, {14, 21, 23, 25, 27, 32}, {14, 21, 23, 25, 27, 28, 32},
{21, 23, 25, 27, 28, 32}, {21, 23, 25, 27, 28, 32, 36}, {3, 21, 23, 25, 27, 28, 32, 36}, {3, 8, 21, 23, 25, 27, 28, 32, 36}, {8, 21, 23, 25, 27, 28, 32, 36}, {2, 8, 21, 23, 25, 27, 28, 32, 36}, {2, 8, 12, 21, 23, 25, 27, 28, 32, 36}, {2, 8, 21, 23, 25, 27, 28, 32, 36},
{2, 8, 21, 23, 24, 25, 27, 28, 32, 36}, {2, 4, 8, 21, 23, 24, 25, 27, 28, 32, 36}, {4, 8, 21, 23, 24, 25, 27, 28, 32, 36}, {8, 21, 23, 24, 25, 27, 28, 32, 36}, {8, 16, 21, 23, 24, 25, 27, 28, 32, 36}, {16, 21, 23, 24, 25, 27, 28, 32, 36},
{16, 21, 22, 23, 24, 25, 27, 28, 32, 36}, {16, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {21, 22, 24, 25, 27, 28, 31, 32, 36}, {24, 25, 27, 28, 31, 32, 36}, {25, 27, 28, 31, 32, 36}, {25, 27, 31, 32, 36},
{7, 25, 27, 31, 32, 36}, {7, 21, 25, 27, 31, 32, 36}, {7, 21, 25, 27, 28, 31, 32, 36}, {21, 25, 27, 28, 31, 32, 36}, {21, 23, 24, 25, 27, 28, 31, 32, 36}, {16, 21, 23, 24, 25, 27, 28, 31, 32, 36}, {16, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36},
{16, 21, 22, 23, 24, 25, 27, 28, 32, 36}, {21, 22, 23, 24, 25, 27, 28, 32, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 36},
{18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 36}, {11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 36}, {5, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 32, 33, 34, 36}, {5, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36},
{5, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36}, {1, 5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36},
{1, 5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {5, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33},
{5, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 20, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 20, 21, 23, 26, 27, 28, 30, 32, 33},
{20, 21, 23, 26, 27, 28, 30, 32, 33}, {20, 21, 23, 26, 28, 30, 32, 33}, {20, 21, 23, 28, 30, 32, 33}, {21, 23, 28, 30, 32, 33}, {21, 23, 30, 32, 33}, {21, 23, 28, 30, 32, 33}, {20, 21, 23, 28, 30, 32, 33}, {20, 21, 23, 27, 28, 30, 32, 33},
{21, 23, 27, 28, 30, 32, 33}, {10, 21, 23, 27, 28, 30, 32, 33}, {10, 21, 23, 26, 27, 28, 30, 32, 33}, {10, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 10, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 25, 26, 27, 28, 30, 32, 33},
{9, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {9, 10, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {9, 10, 21, 23, 25, 26, 27, 28, 32, 33}, {10, 21, 23, 25, 26, 27, 28, 32, 33}, {10, 21, 23, 25, 26, 27, 28, 32, 33, 36}, {10, 21, 23, 25, 26, 27, 28, 32, 33},
{10, 21, 23, 25, 27, 28, 32, 33}, {3, 10, 21, 23, 25, 27, 28, 32, 33}, {10, 21, 23, 25, 27, 28, 32, 33}, {21, 23, 25, 27, 28, 32, 33}, {14, 21, 23, 25, 27, 28, 32, 33}, {14, 21, 23, 25, 27, 28, 32}, {14, 21, 23, 25, 27, 32}, {21, 23, 25, 27, 32},
{21, 23, 27, 32}, {21, 23, 27, 32, 33}, {21, 23, 27, 30, 32, 33}, {21, 23, 27, 28, 30, 32, 33}, {14, 21, 23, 27, 28, 30, 32, 33}, {21, 23, 27, 28, 30, 32, 33}, {10, 21, 23, 27, 28, 30, 32, 33}, {10, 21, 23, 25, 26, 27, 28, 30, 32, 33},
{21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {5, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {5, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33},
{1, 5, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 9, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 9, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36},
{1, 5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36}, {5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36}, {5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36},
{9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 11, 18, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36},
{9, 18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {4, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36},
{21, 22, 23, 24, 25, 26, 27, 28, 32, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 36}, {21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {16, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {6, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36},
{6, 21, 22, 24, 25, 27, 28, 31, 32, 36}, {6, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {6, 21, 22, 23, 24, 25, 27, 28, 31, 32, 35, 36}, {6, 21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {21, 22, 23, 24, 25, 27, 28, 31, 32, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 36},
{21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 34, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 31, 32, 33, 34, 36}, {21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36},
{9, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {5, 9, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 34, 36}, {5, 9, 11, 18, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 9, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36},
{5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {5, 11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34},
{17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {17, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {21, 23, 24, 25, 26, 28, 30, 32, 33}, {21, 23, 24, 26, 28, 30, 32, 33},
{21, 23, 26, 28, 30, 32, 33}, {3, 21, 23, 26, 28, 30, 32, 33}, {3, 21, 23, 26, 27, 28, 30, 32, 33}, {3, 20, 21, 23, 26, 27, 28, 30, 32, 33}, {3, 20, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {3, 10, 21, 23, 25, 26, 27, 28, 30, 32, 33},
{3, 10, 21, 23, 25, 27, 28, 30, 32, 33}, {3, 10, 21, 23, 25, 27, 28, 32, 33}, {3, 10, 21, 23, 25, 27, 28, 32, 33, 36}, {3, 8, 10, 21, 23, 25, 27, 28, 32, 33, 36}, {3, 8, 21, 23, 25, 27, 28, 32, 33, 36}, {3, 8, 21, 23, 25, 27, 28, 32, 36},
{8, 21, 23, 25, 27, 28, 32, 36}, {8, 15, 21, 23, 25, 27, 28, 32, 36}, {8, 15, 21, 23, 25, 27, 32, 36}, {8, 13, 15, 21, 23, 25, 27, 32, 36}, {13, 15, 21, 23, 25, 27, 32, 36}, {13, 21, 23, 25, 27, 32, 36}, {13, 21, 23, 25, 27, 36}, {13, 21, 25, 27, 36},
{21, 25, 27, 36}, {25, 27, 36}, {23, 25, 27, 36}, {21, 23, 25, 27, 36}, {21, 23, 25, 27}, {21, 23, 25, 27, 32}, {15, 21, 23, 25, 27, 32}, {21, 23, 25, 27, 32}, {14, 21, 23, 25, 27, 32}, {14, 21, 23, 25, 27, 28, 32}, {14, 21, 23, 25, 27, 28, 32, 33},
{21, 23, 25, 27, 28, 32, 33}, {10, 21, 23, 25, 27, 28, 32, 33}, {3, 10, 21, 23, 25, 27, 28, 32, 33}, {3, 10, 21, 23, 25, 26, 27, 28, 32, 33}, {10, 21, 23, 25, 26, 27, 28, 32, 33}, {9, 10, 21, 23, 25, 26, 27, 28, 32, 33}, {9, 10, 21, 23, 25, 26, 27, 28, 30, 32, 33},
{9, 21, 23, 25, 26, 27, 28, 30, 32, 33}, {9, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 9, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 9, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 9, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33},
{1, 5, 9, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 19, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 21, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {1, 5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33},
{5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {5, 11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34}, {11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34},
{17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34}, {21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34}, {21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36}, {11, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36},
{11, 17, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36}, {11, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 36}, {5, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36},
{5, 9, 11, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 34, 36}, {5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 30, 32, 33, 36}, {5, 9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36},
{9, 11, 19, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 11, 21, 22, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 11, 21, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 21, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {9, 12, 21, 23, 24, 25, 26, 27, 28, 32, 33, 36},
{2, 9, 12, 21, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {2, 8, 9, 12, 21, 23, 24, 25, 26, 27, 28, 32, 33, 36}, {2, 8, 9, 12, 21, 23, 24, 25, 27, 28, 32, 33, 36}, {2, 8, 9, 12, 21, 23, 25, 27, 28, 32, 33, 36}, {2, 8, 9, 12, 21, 23, 25, 27, 28, 32, 36},
{2, 8, 12, 21, 23, 25, 27, 28, 32, 36}, {2, 8, 21, 23, 25, 27, 28, 32, 36}, {8, 21, 23, 25, 27, 28, 32, 36}, {8, 13, 21, 23, 25, 27, 28, 32, 36}, {8, 13, 21, 23, 25, 27, 32, 36}, {13, 21, 23, 25, 27, 32, 36}, {13, 21, 23, 25, 27, 36}, {13, 21, 25, 27, 36},
{21, 25, 27, 36}, {25, 27, 36}, {23, 25, 27, 36}, {21, 23, 25, 27, 36}, {21, 23, 25, 27}, {21, 23, 27}, {21, 23, 27, 32}, {21, 23, 27, 30, 32}, {21, 23, 27, 30, 32, 33}, {21, 23, 30, 32, 33}, {21, 23, 27, 30, 32, 33}, {21, 23, 27, 30, 32}}
```



A concrete example: chemical material drum recovery by a robot

Specification: $\varphi := \varphi_C \wedge \varphi_G \wedge \varphi_I \wedge \varphi_D$

- φ_C: permits the robot to make contact with the surface of the drum
- φ_G: requires the robot to eventually make contact with the drum, before entering the target zone and remaining there.
- φ_I: prohibits the robot from colliding with the inner boundaries of the obstacles and 'unsafe' zones
- φ_D: prohibits the robot from colliding with the outer boundaries of the obstacles and 'unsafe' zones from the point of contact and recovery of the drum



Contributions

Theoretical:

- extend path checking to continuous paths
- topological conditions for lossless sampling

Algorithm:

- path checks arbitrary 2D/3D polynomial spline paths (e.g., minimum jerk, snap, ...)
- no approximations
- general LTL (without next) requirements
- versatile region description: semi-algebraic sets
- polynomial time complexity: $O(NM^6)$

A "robust" treatment: D Selvaratnam, M Cantoni, M Davoren, I Shames, "MITL verification under timing uncertainty." FORMATS, pp. 136-152, 2022.

Acknowledgement: Trusted Autonomous System D-CRC, and all the smart people that I have learned from through our interactions and collaborations.