Automating Geometric Proofs of Collision Avoidance with Active Corners

Nishant Kheterpal, Elanor Tang, Jean-Baptiste Jeannin

MICHIGAN ENGINEERING
UNIVERSITY OF MICHIGAN

fmcad.22

Motivation

- Automation in safety-critical systems is gaining prevalence

Motivation

- Automation in safety-critical systems is gaining prevalence
- Formal guarantees of safety are important for medical devices, aircraft, automated vehicles, etc.

Motivation

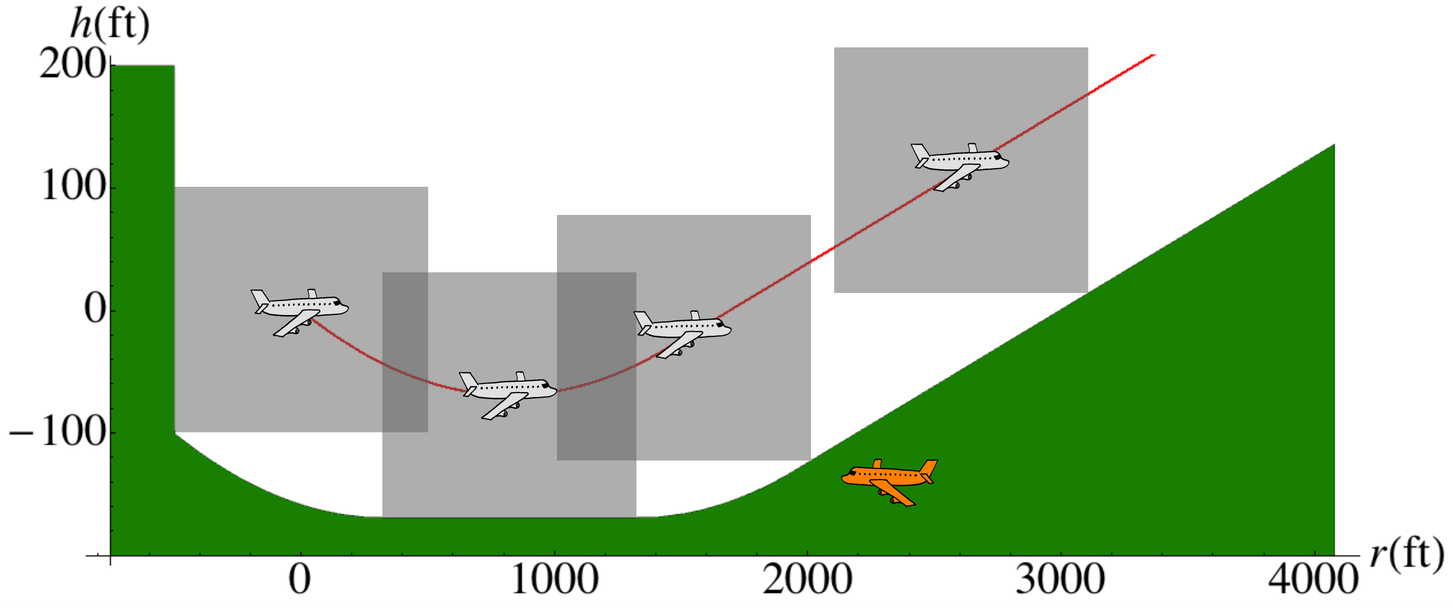
- Automation in safety-critical systems is gaining prevalence
- Formal guarantees of safety are important for medical devices, aircraft, automated vehicles, etc.
- Verification is often difficult
  - Rarely possible to verify implementations in situ
  - Theorem provers require user interaction to run

Motivation

- Automation in safety-critical systems is gaining prevalence
- Formal guarantees of safety are important for medical devices, aircraft, automated vehicles, etc.
- Verification is often difficult
  - Rarely possible to verify implementations in situ
  - Theorem provers require user interaction to run
- My work attempts to automate certain collision avoidance verification tasks

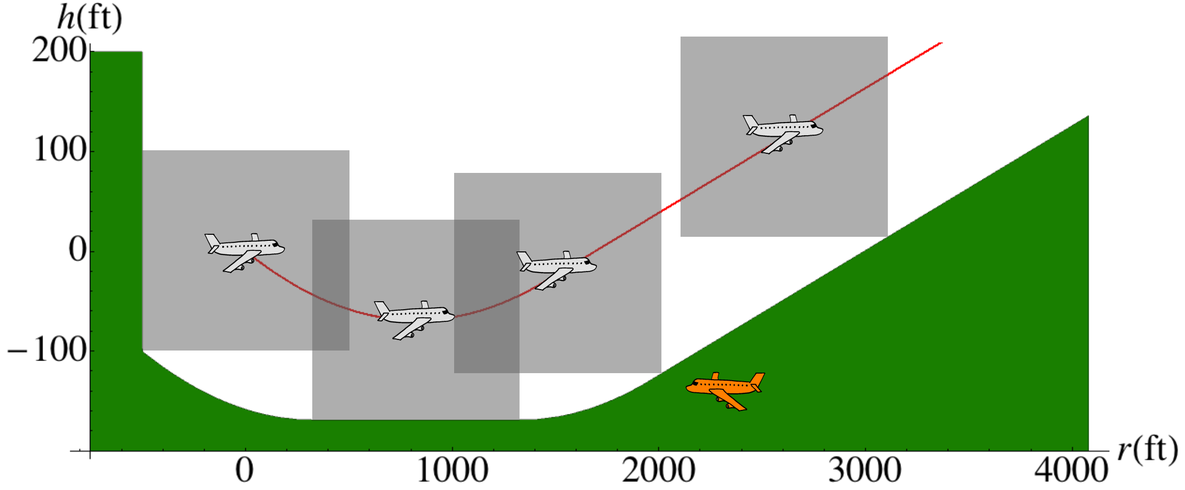
Inspiration

- Past work by Jeannin et al. verified collision avoidance safety for ACAS X
- ACAS X is an airborne collision avoidance system that sends climb/descend commands to pilots to avoid mid-air collisions with approaching aircraft



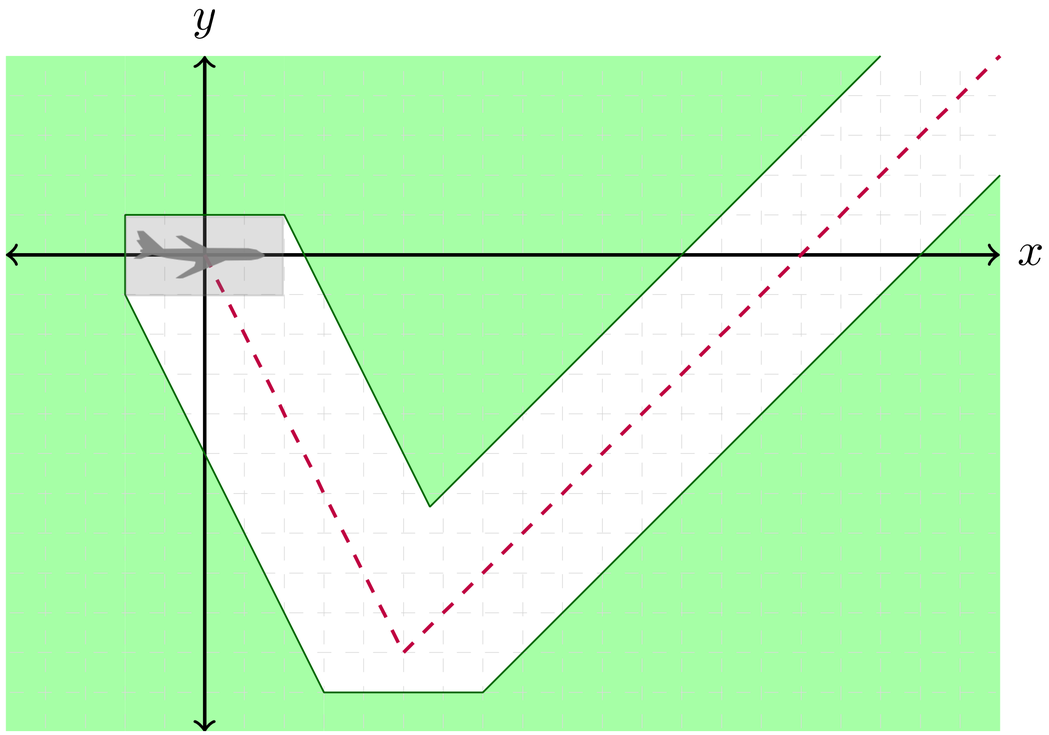
Inspiration

- Past work by Jeannin et al. verified collision avoidance safety for ACAS X
- ACAS X is an airborne collision avoidance system that sends climb/descend commands to pilots to avoid mid-air collisions with approaching aircraft
- How can we make sure a control system issues correct advisories?



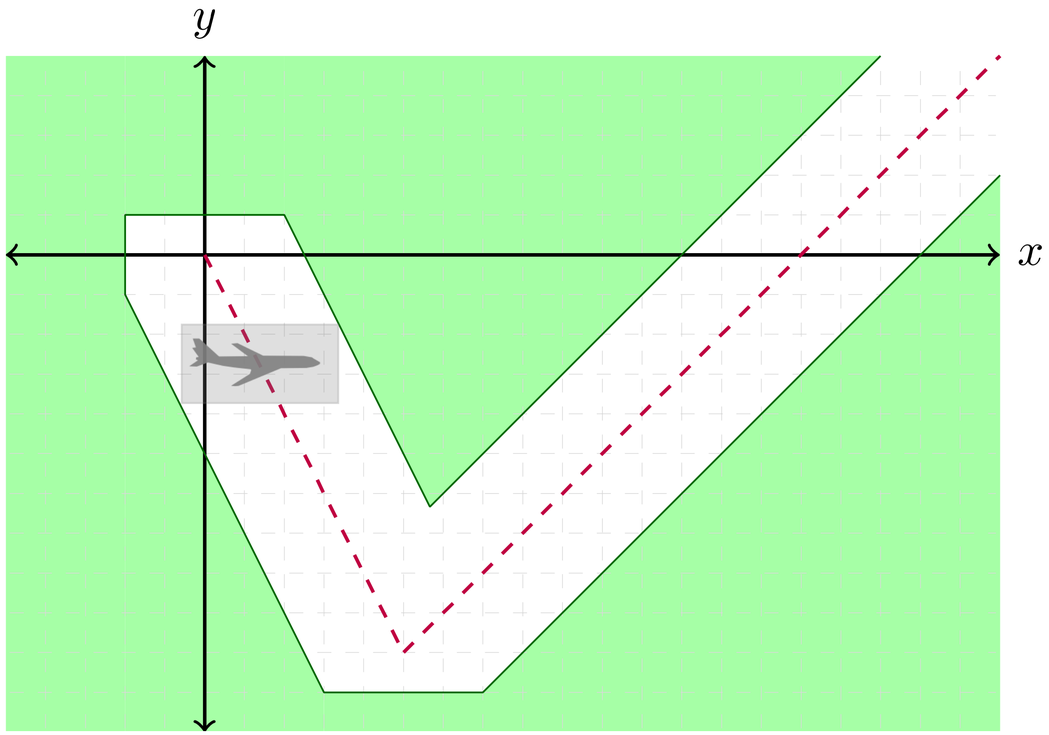
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



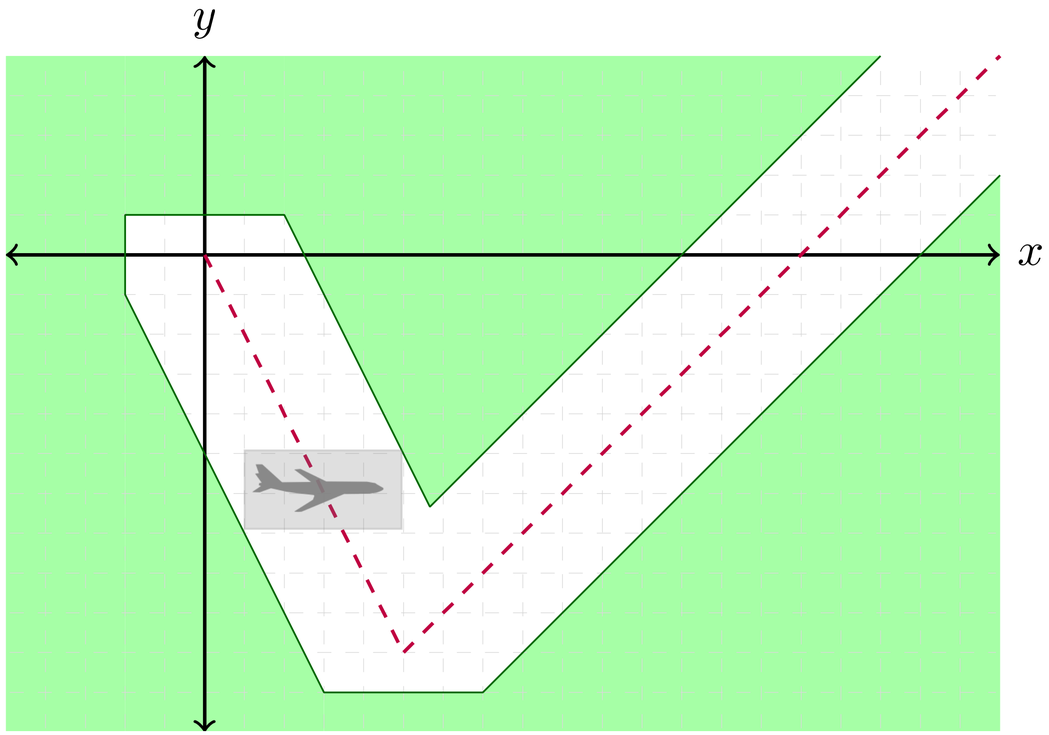
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



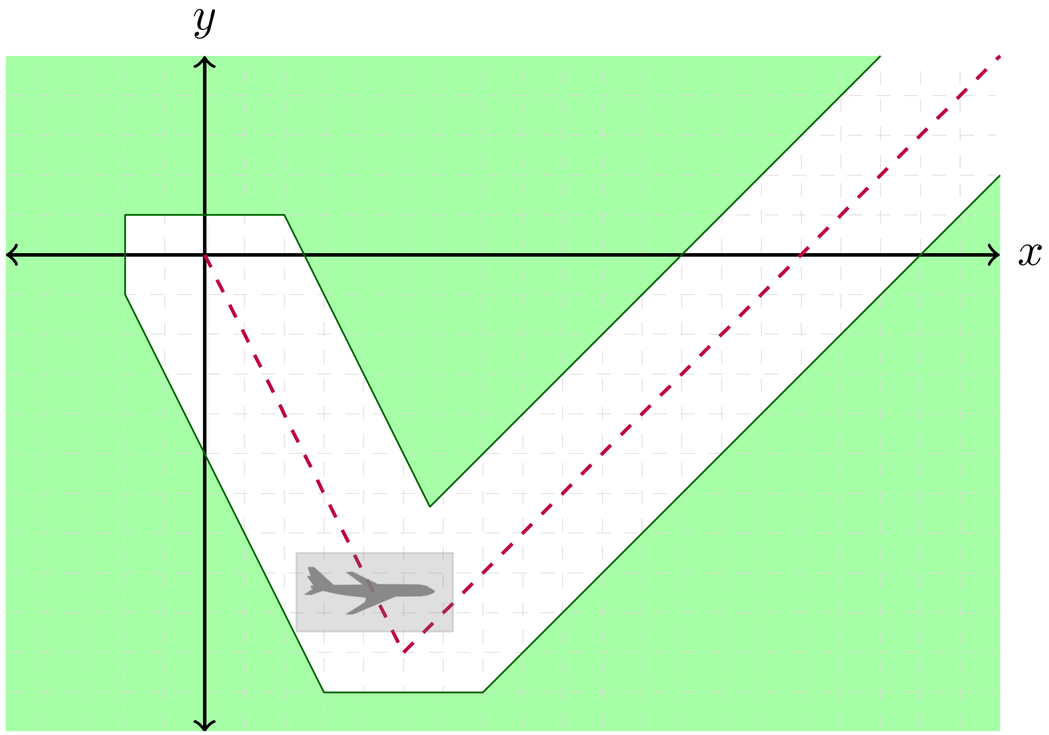
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



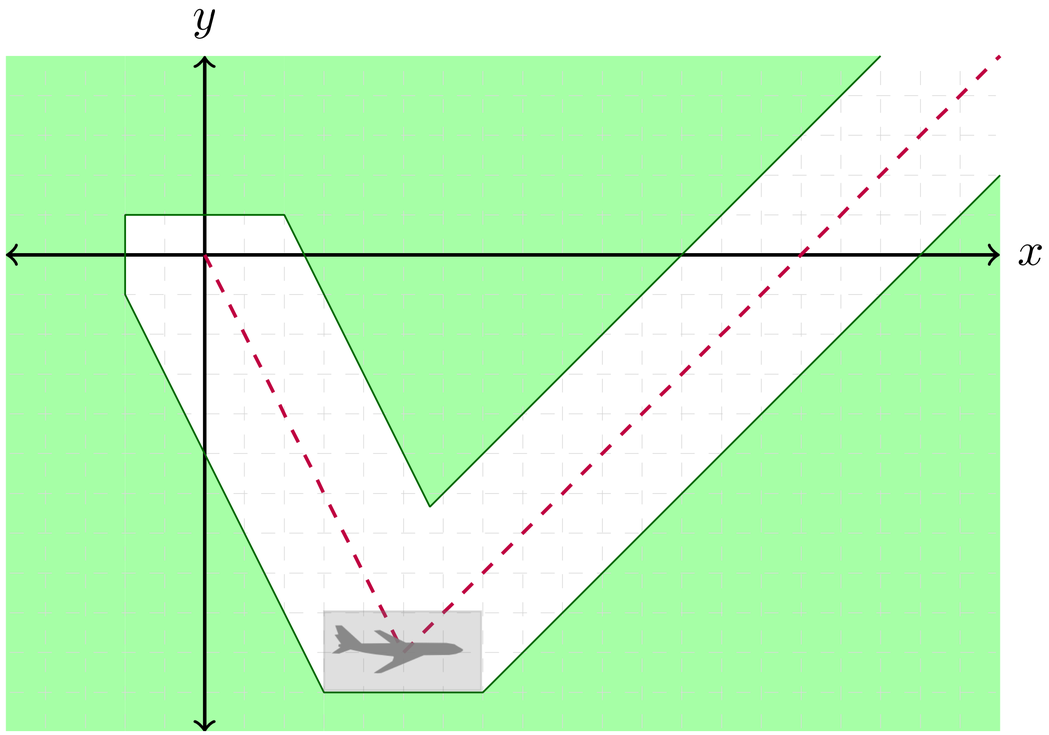
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



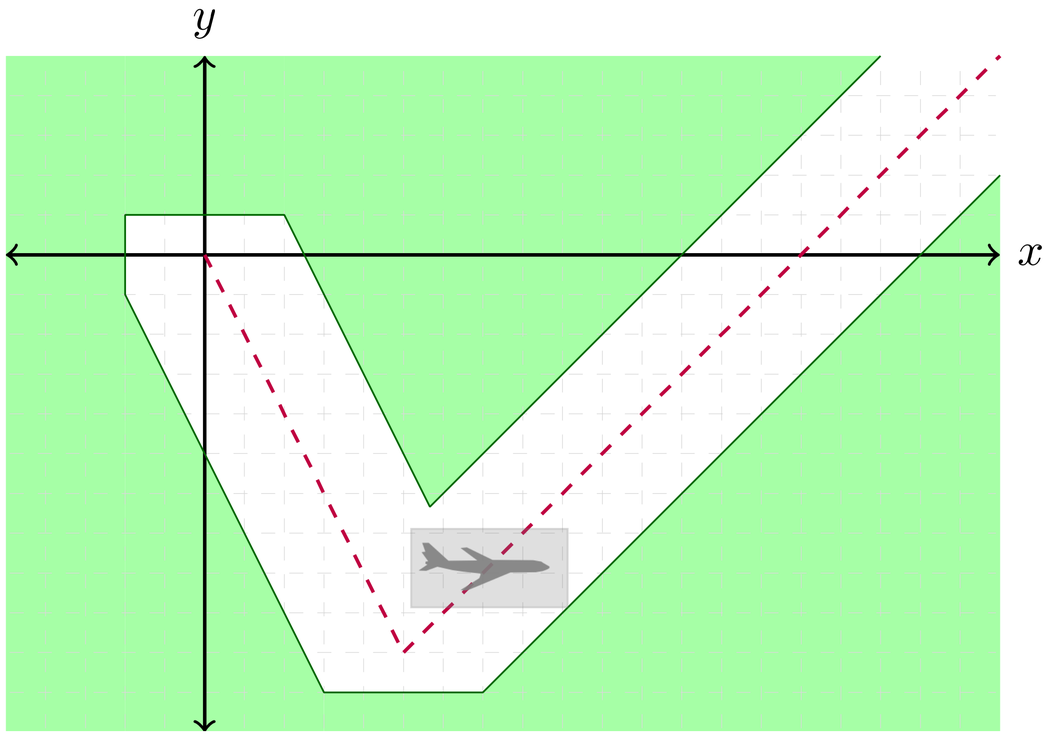
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



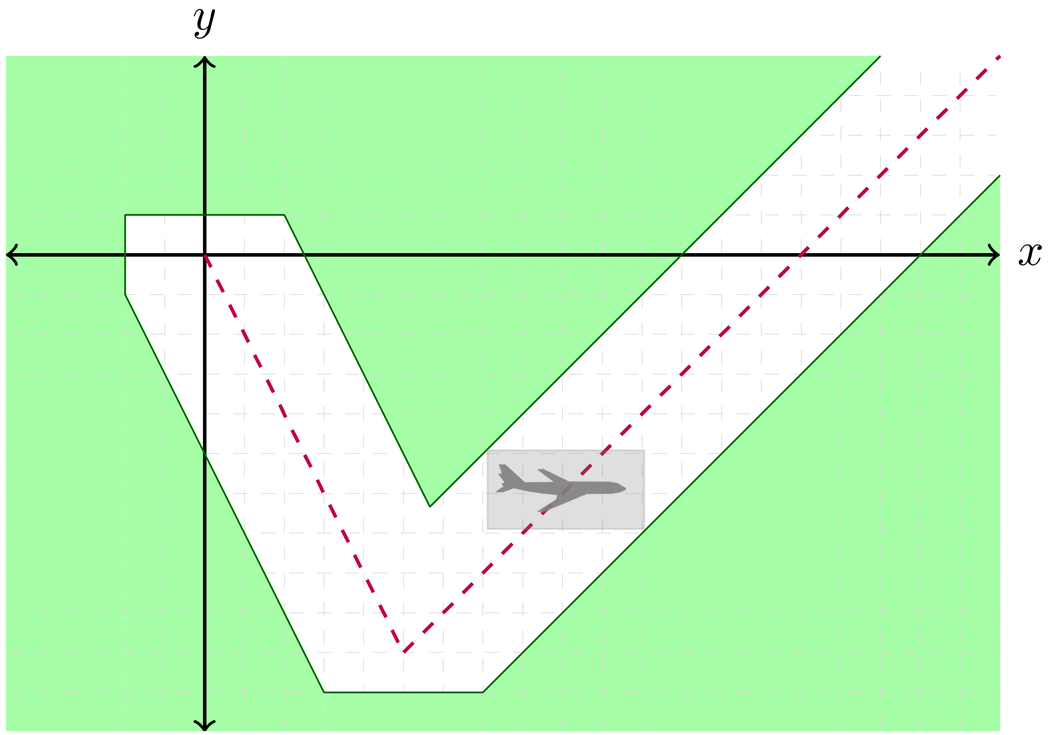
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



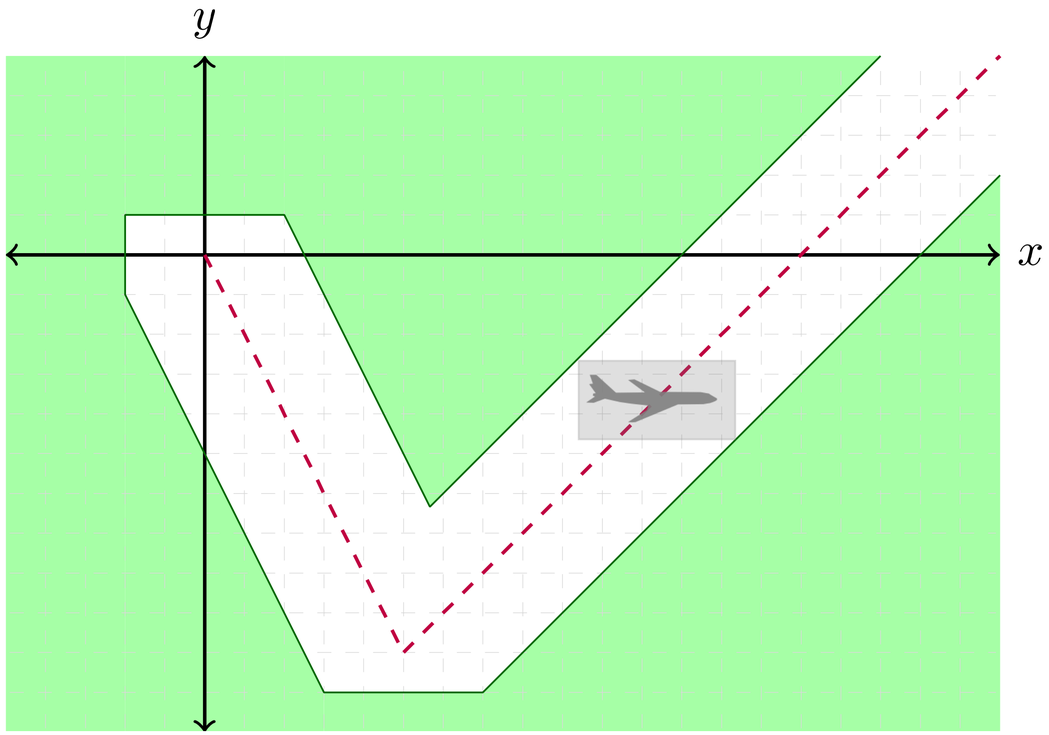
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



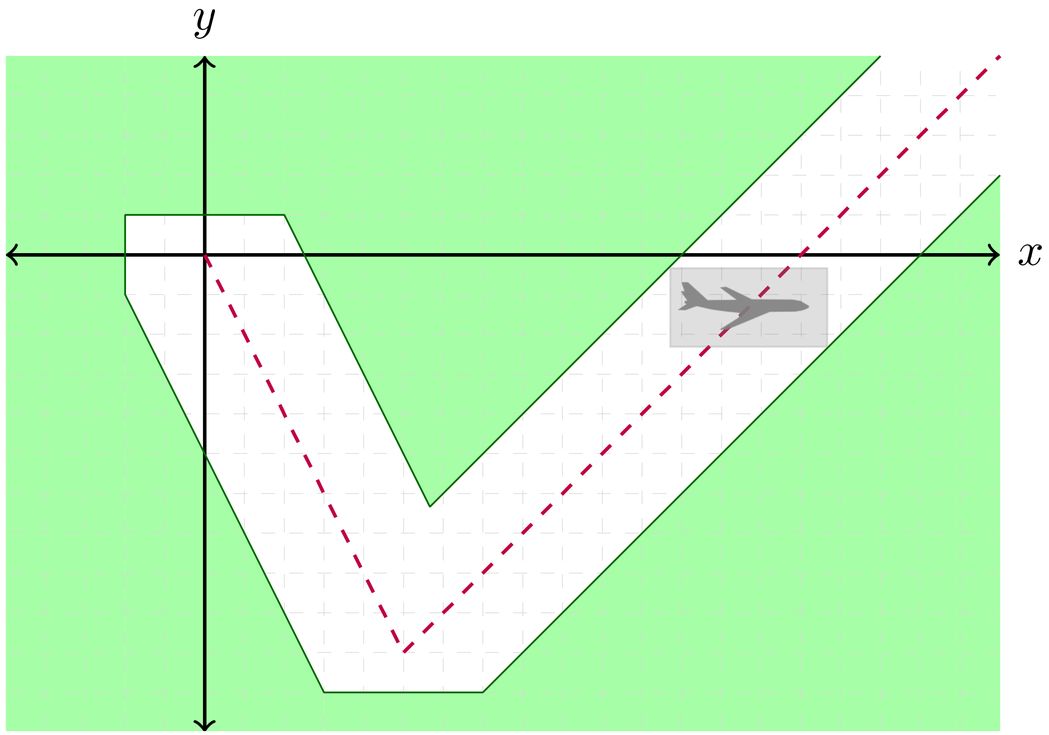
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



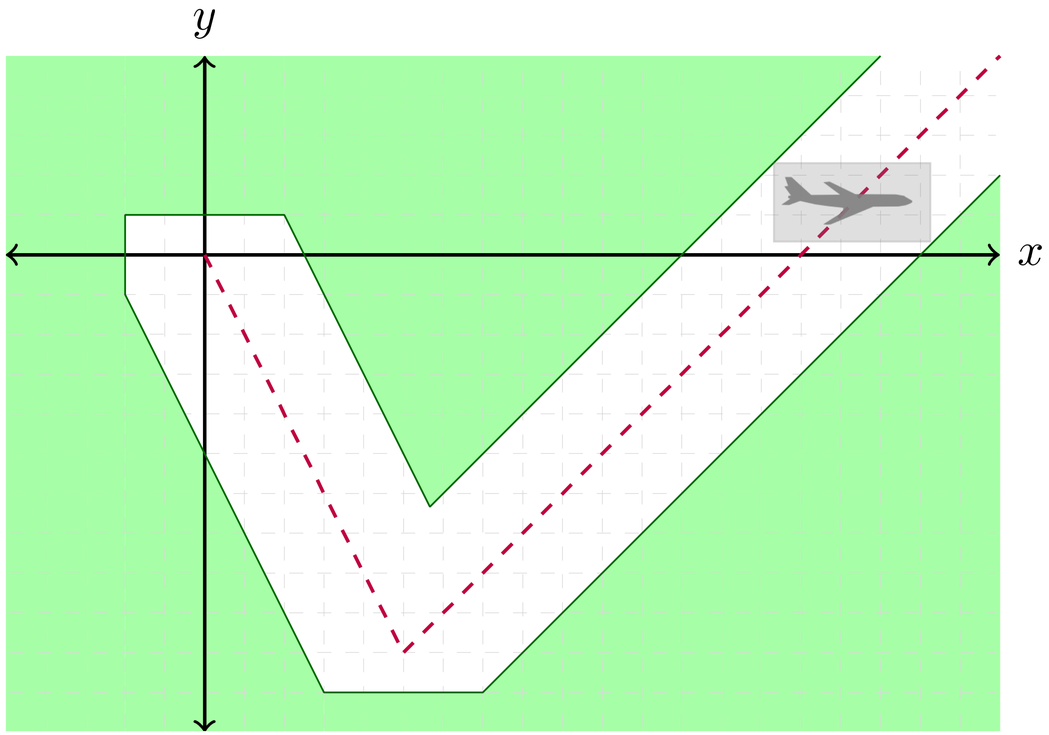
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



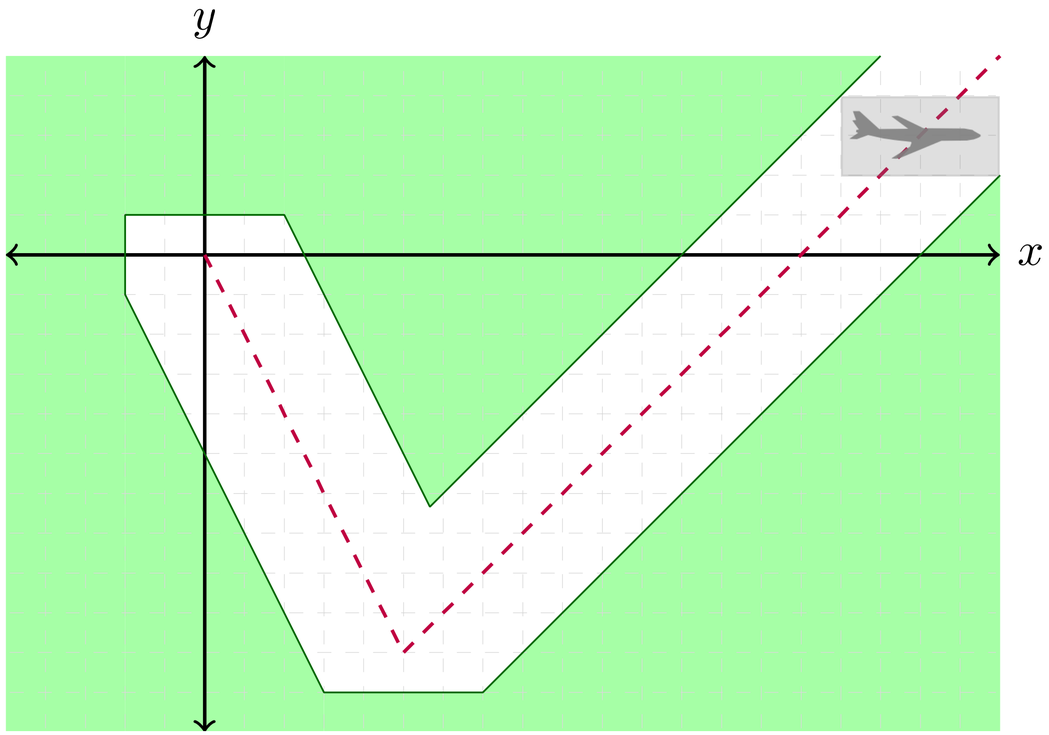
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



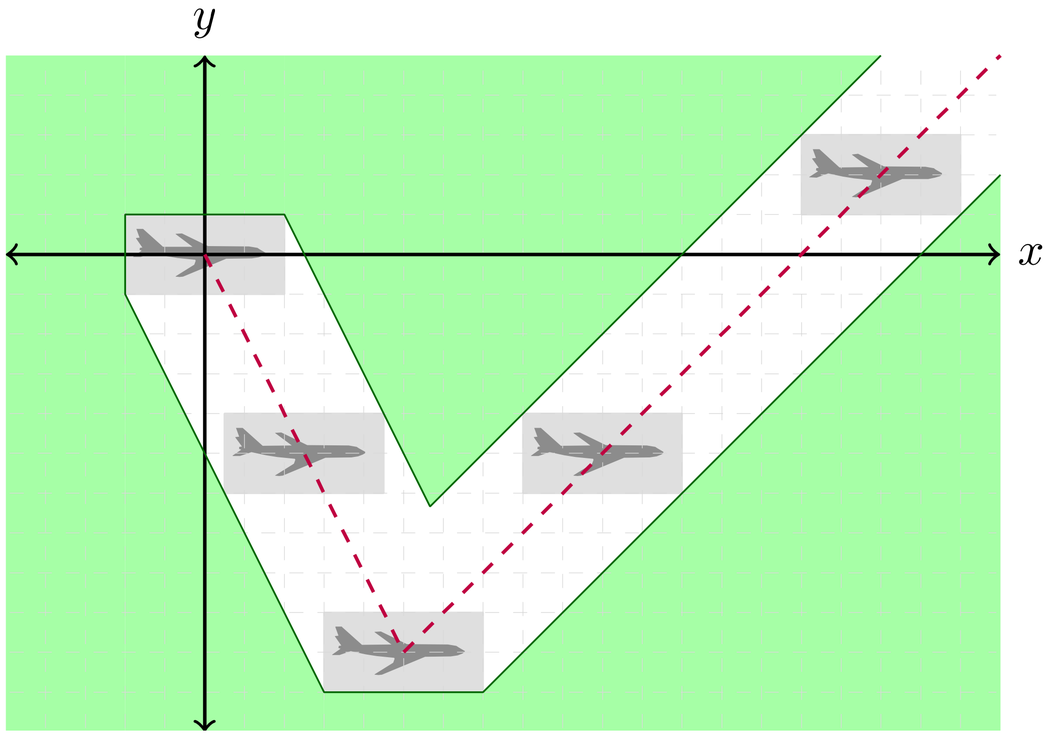
Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region



Inspiration
Problem Statement

- Inputs
  - A convex polygon
  - A known piecewise, planar trajectory
- Goal
  - Automatically generate a symbolic, provably correct, quantifier-free safe region
- **Key insight:** We must consider the trajectories of corners and more to construct a safe region



Jeannin et al., TACAS ’15; Jeannin et al., STTT ‘17

Inspiration
Past Approach

- Proofs of advisory safety used *implicit* and *explicit* safe regions

Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

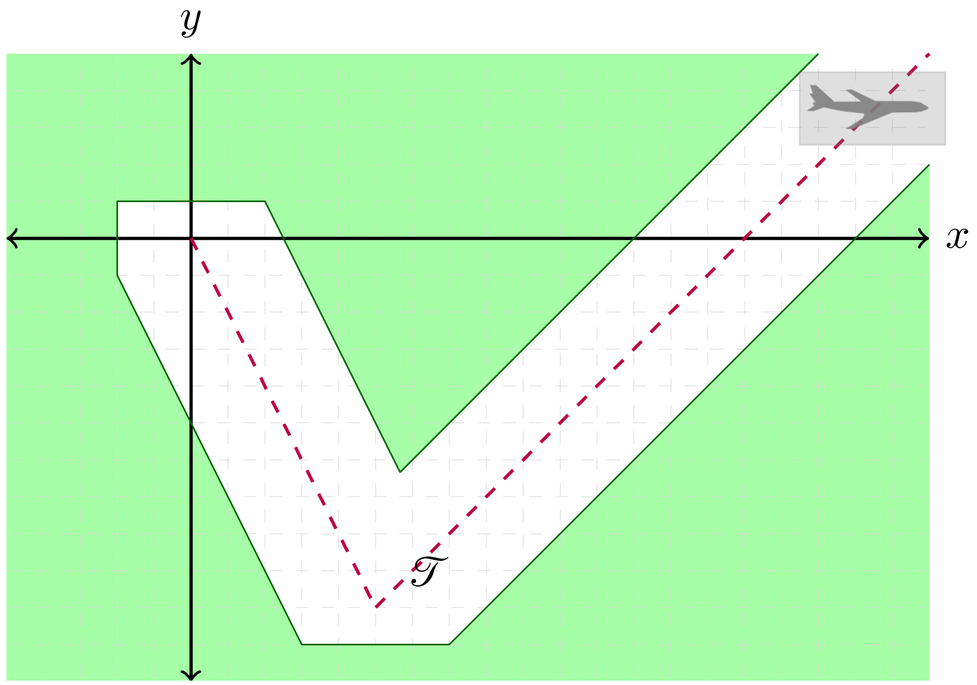
Inspiration
Past Approach

- Proofs of advisory safety used *implicit* and *explicit* safe regions
- *Implicit* safe region:
  - Along some path, obstacle at (x_O, y_O) is a safe distance away

Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Inspiration
Past Approach

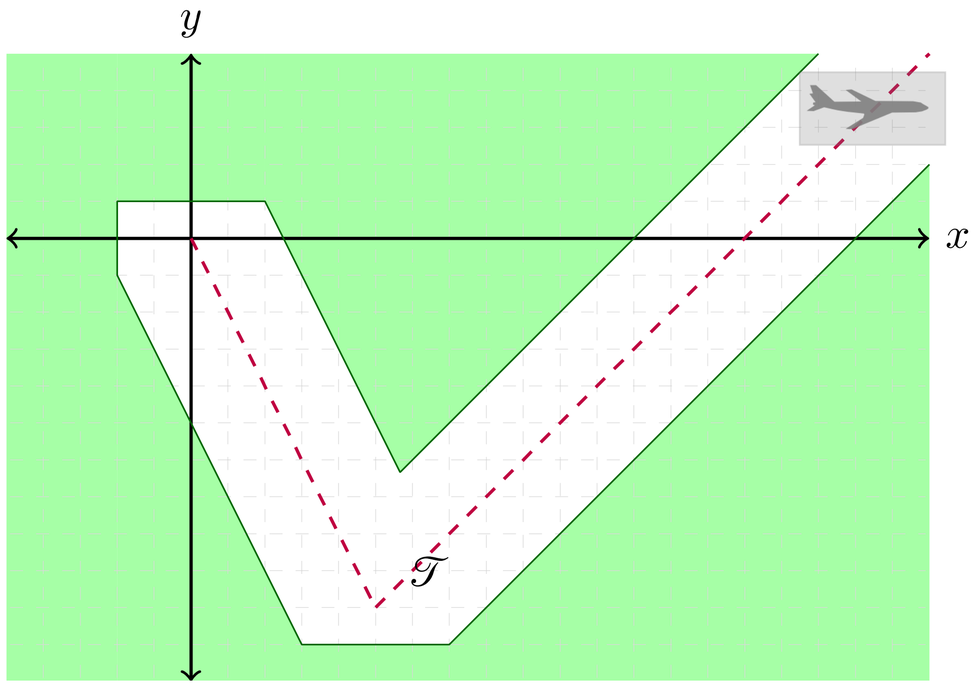
- Proofs of advisory safety used *implicit* and *explicit* safe regions
- *Implicit* safe region:
  - Along some path, obstacle at (x_O, y_O) is a safe distance away
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|\, x_O - x_{\mathcal{T}} \,| > w \vee |\, y_O - y_{\mathcal{T}} \,| > h \right)$



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Inspiration
Past Approach

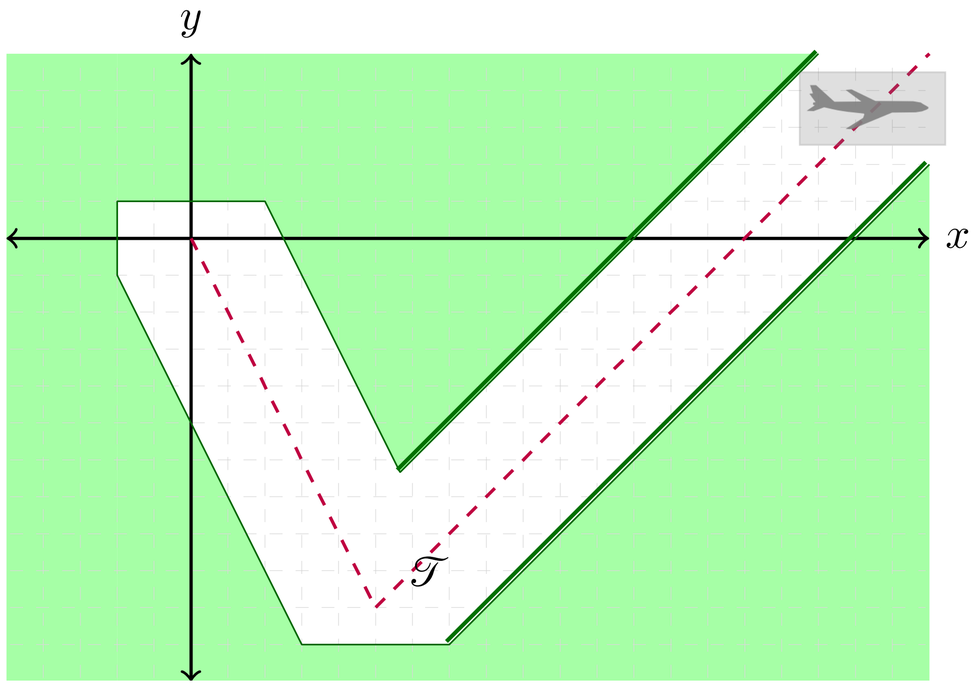
- Proofs of advisory safety used *implicit* and *explicit* safe regions
- *Implicit* safe region:
  - Along some path, obstacle at (x_O, y_O) is a safe distance away
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|x_O - x_{\mathcal{T}}| > w \vee |y_O - y_{\mathcal{T}}| > h \right)$
- *Explicit* safe region:
  - Directly encodes a region in space where the plane/car will never be



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

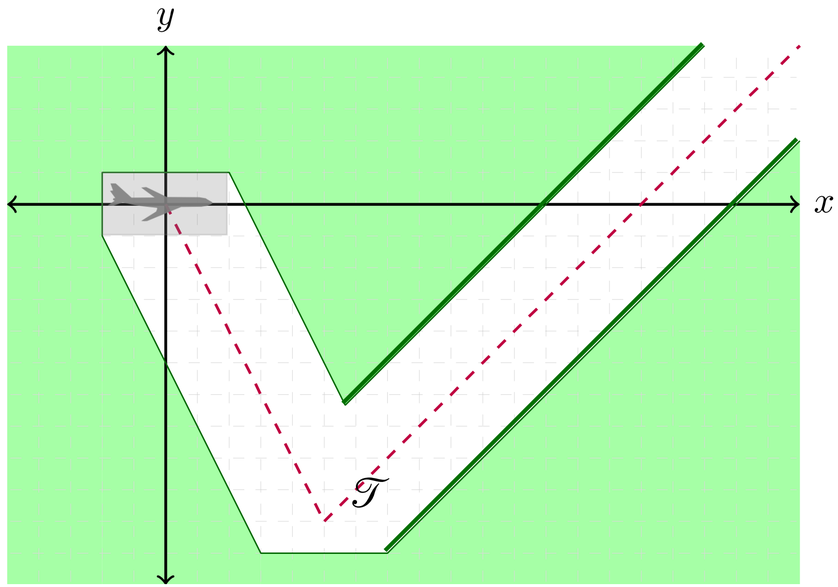
Inspiration
Past Approach

- Proofs of advisory safety used *implicit* and *explicit* safe regions
- *Implicit* safe region:
  - Along some path, obstacle at (x_O, y_O) is a safe distance away
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|x_O - x_{\mathcal{T}}| > w \vee |y_O - y_{\mathcal{T}}| > h \right)$
- *Explicit* safe region:
  - Directly encodes a region in space where the plane/car will never be
  - e.g. $\left((y_O \geq x_O + w + h - 15) \vee (y_O \leq x_O - w - h - 15) \right)$



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

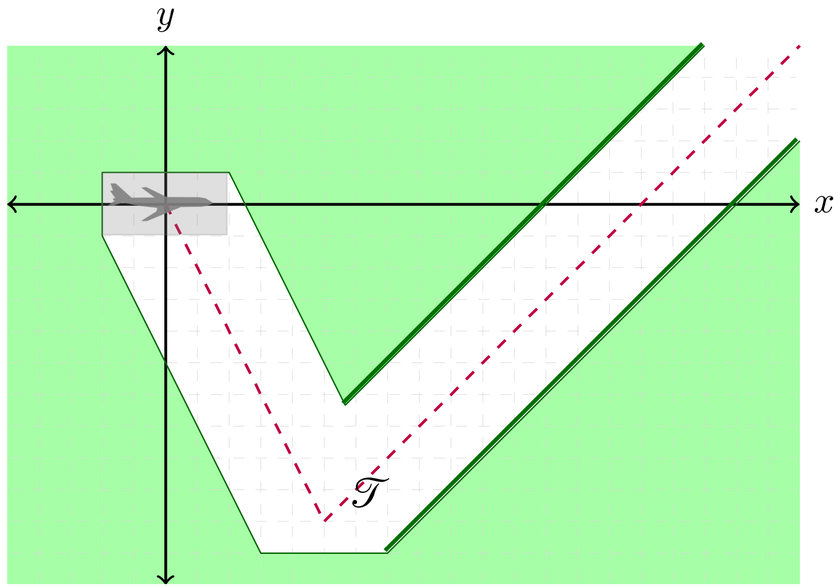
Inspiration
Shortcomings



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Inspiration
Shortcomings

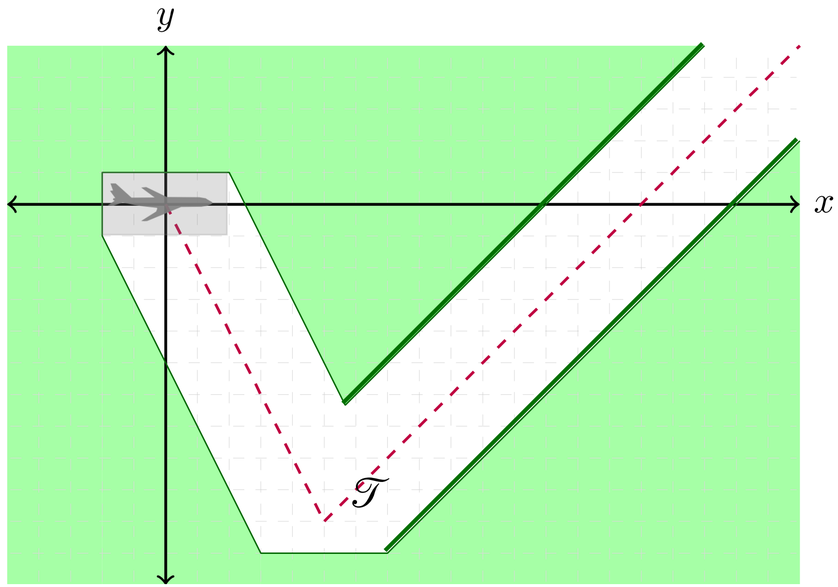
- *Implicit* regions are straightforward to prove safe but difficult to use at runtime due to quantifiers
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|\, x_O - x_{\mathcal{T}} \,| > w \vee |\, y_O - y_{\mathcal{T}} \,| > h \right)$
- *Explicit* formulations are more efficient when checking safety



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Inspiration
Shortcomings

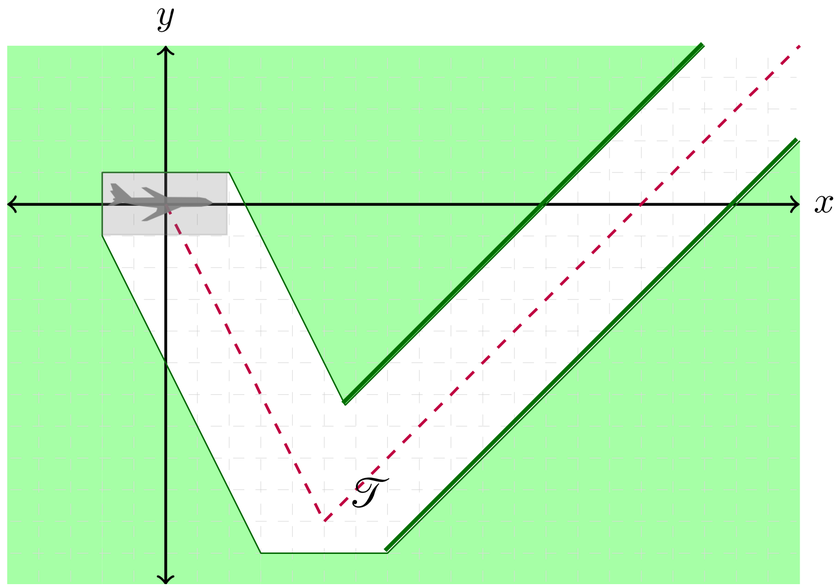
- *Implicit* regions are straightforward to prove safe but difficult to use at runtime due to quantifiers
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|\, x_O - x_{\mathcal{T}} \,| > w \vee |\, y_O - y_{\mathcal{T}} \,| > h \right)$
- *Explicit* formulations are more efficient when checking safety
  - $\left(\left(y_O \geq x_O + w + h - 15 \right) \vee \left(y_O \leq x_O - w - h - 15 \right) \right)$
  - Constructing regions and proving equivalence in a theorem prover takes hundreds of hours
- Some repeated structure in past work using this approach



Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Inspiration
Shortcomings

- *Implicit* regions are straightforward to prove safe but difficult to use at runtime due to quantifiers
  - $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|\, x_O - x_{\mathcal{T}} \,| > w \vee |\, y_O - y_{\mathcal{T}} \,| > h \right)$
- *Explicit* formulations are more efficient when checking safety
  - $\left(\left(y_O \geq x_O + w + h - 15 \right) \vee \left(y_O \leq x_O - w - h - 15 \right) \right)$
  - Constructing regions and proving equivalence in a theorem prover takes hundreds of hours
- Some repeated structure in past work using this approach
- Can we bridge the gap automatically?



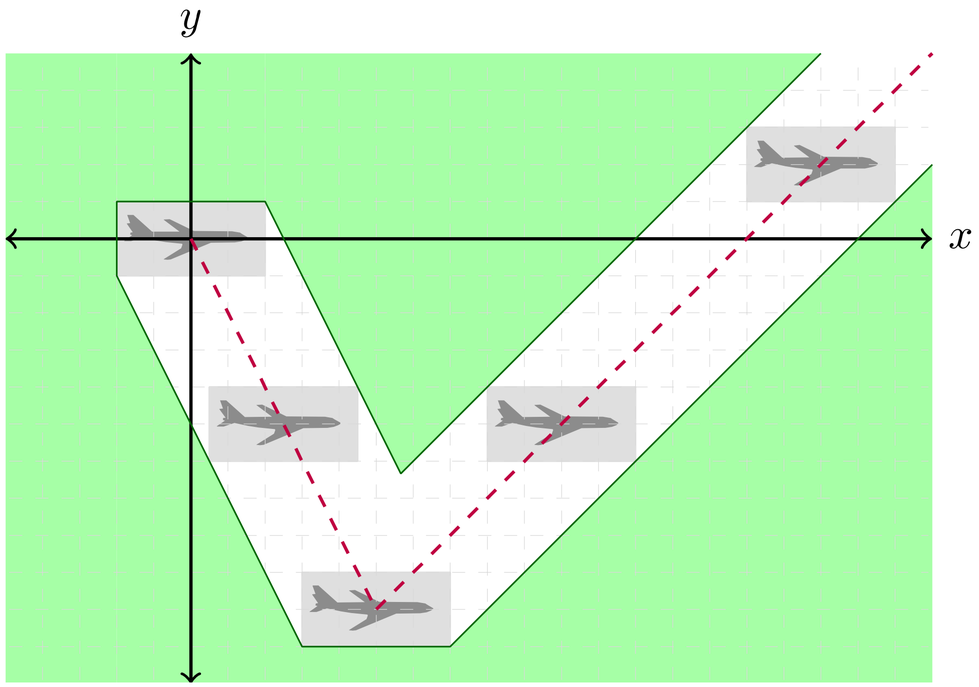
Adler et al., AIAA ’19; Abhishek et al., HSCC ‘20

Algorithm
Overview

- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$

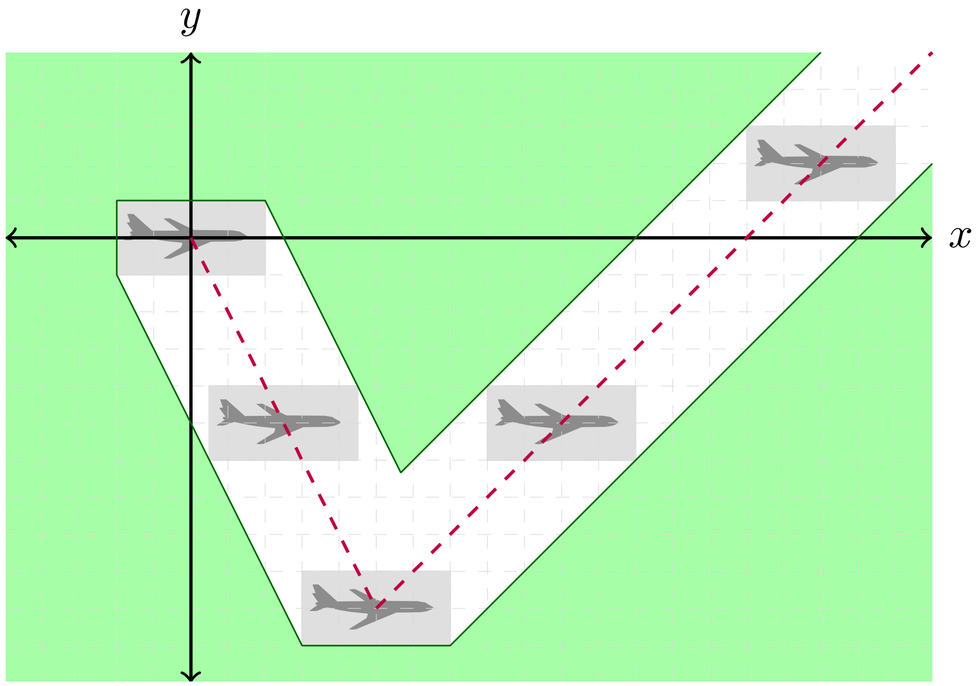
Algorithm
Overview

- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$
- $$\mathscr{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \left(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \right)$



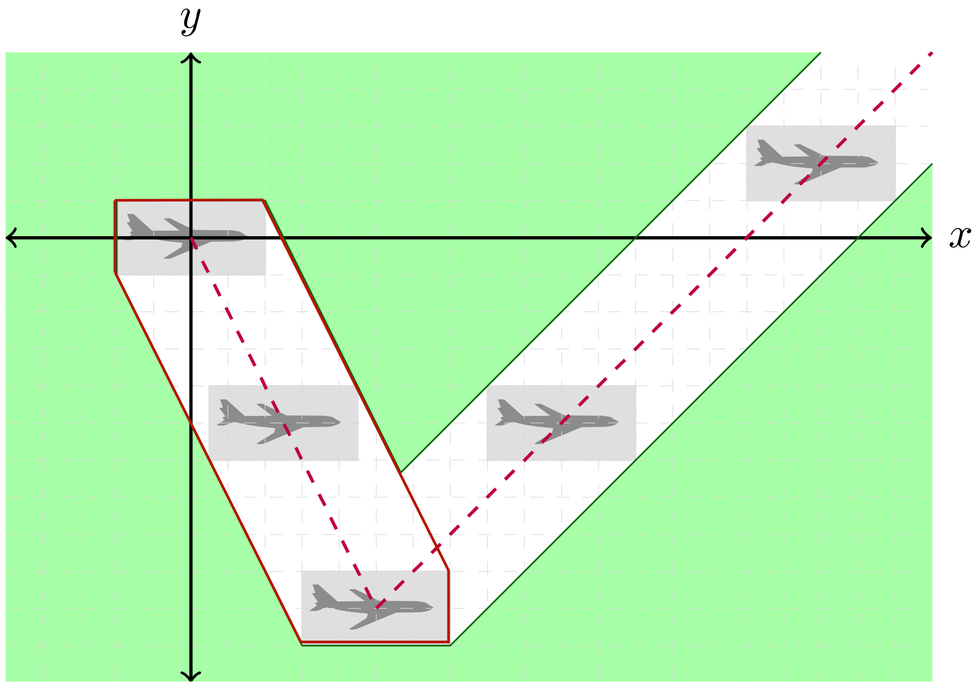
Algorithm
Overview

- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$
- $$\mathscr{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \left(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \right)$
- Automatically construct a quantifier-free, geometric representation of the unsafe region



Algorithm
Overview

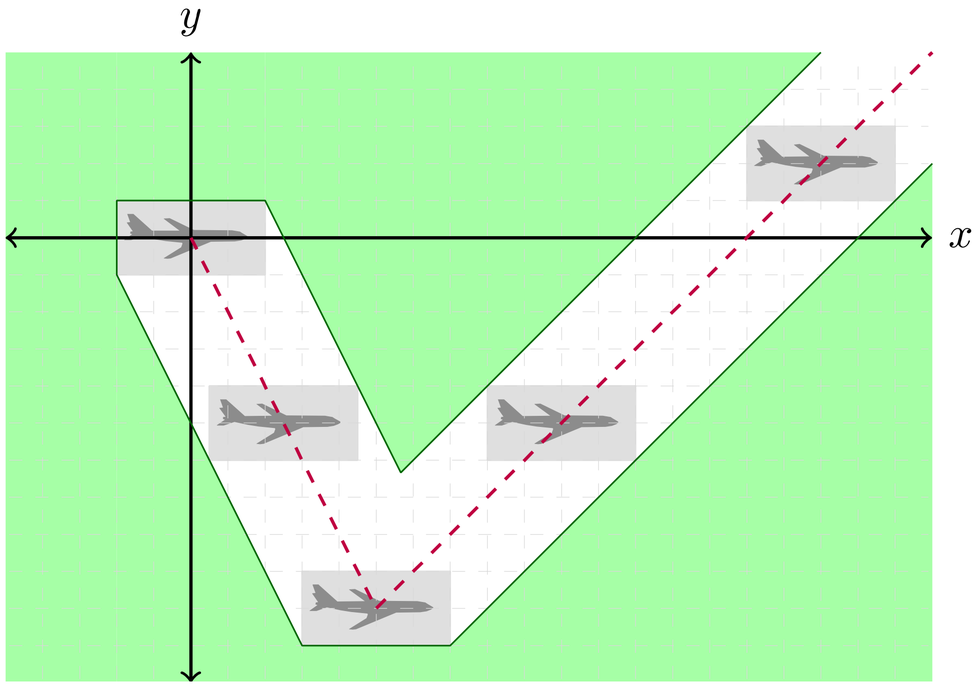
- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$
- $$\mathscr{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \left(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \right)$
- Automatically construct a quantifier-free, geometric representation of the unsafe region



$$((x_O \geq -w) \wedge (y_O \leq h) \wedge (y_O \geq -2x_O - 2w - h) \wedge (x_O \leq 5 + w) \quad \wedge (y_O \leq -2x_O + 2w + h) \wedge (y_O \geq -10 - h)) \bigvee$$

Algorithm
Overview

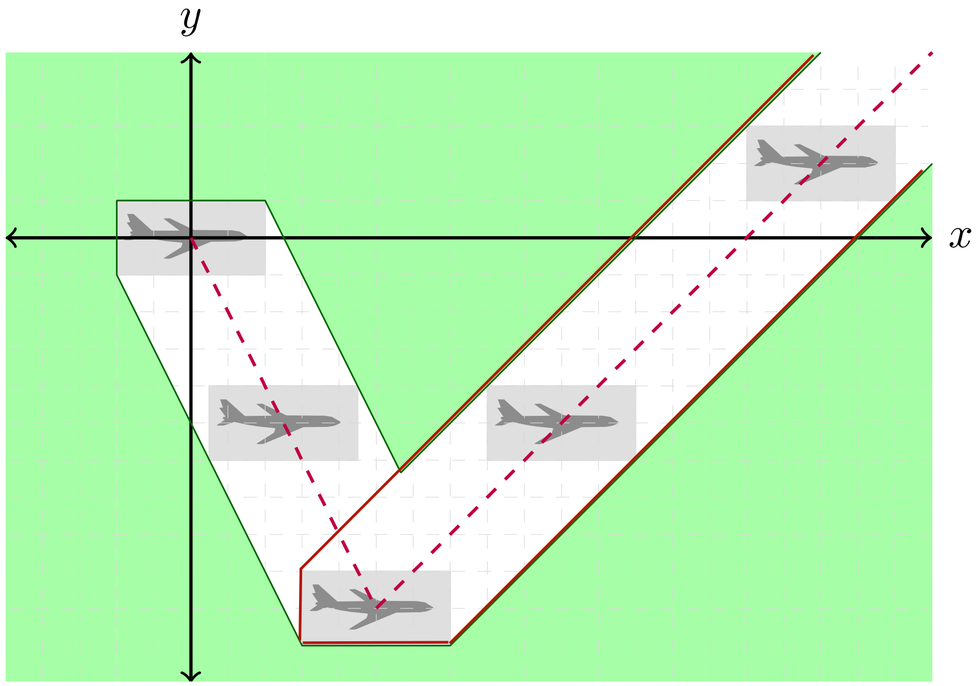
- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$
- $$\mathscr{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \left(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \right)$
- Automatically construct a quantifier-free, geometric representation of the unsafe region



$$((x_O \geq -w) \wedge (y_O \leq h) \wedge (y_O \geq -2x_O - 2w - h) \wedge (x_O \leq 5 + w) \quad \wedge (y_O \leq -2x_O + 2w + h) \wedge (y_O \geq -10 - h)) \bigvee$$

Algorithm
Overview

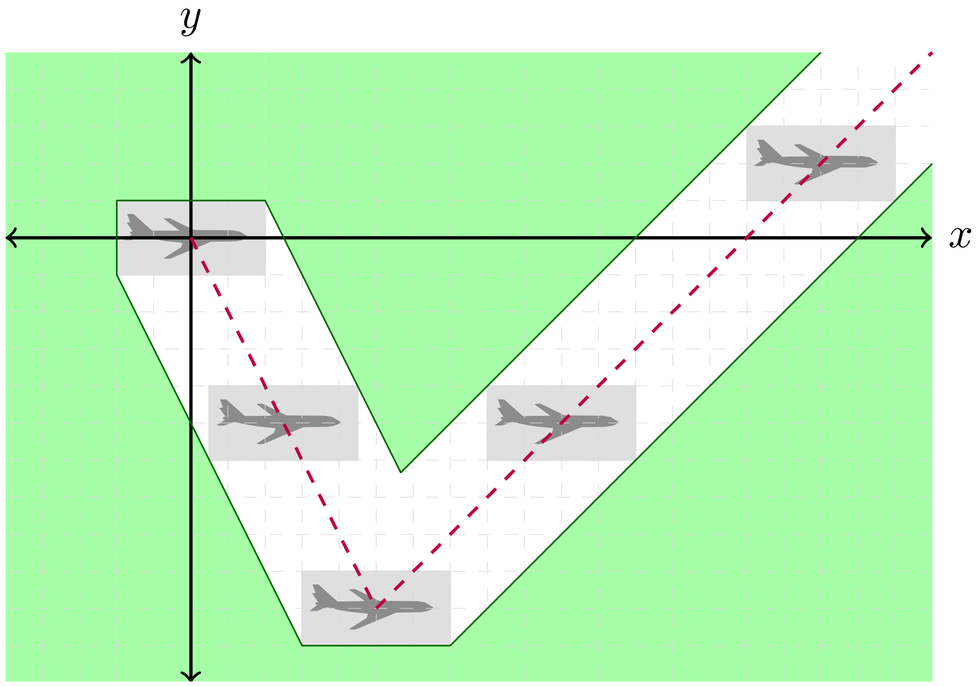
- Given a convex polygon translating on a *known* piecewise trajectory $\mathcal{T}$
- $$\mathcal{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathcal{T}}, y_{\mathcal{T}}) \in \mathcal{T}, \left(|x_O - x_{\mathcal{T}}| > w \vee |y_O - y_{\mathcal{T}}| > h \right)$
- Automatically construct a quantifier-free, geometric representation of the unsafe region



$$((x_O \geq -w) \wedge (y_O \leq h) \wedge (y_O \geq -2x_O - 2w - h) \wedge (x_O \leq 5 + w) \quad \wedge (y_O \leq -2x_O + 2w + h) \wedge (y_O \geq -10 - h)) \bigvee$$
$$((x_O \geq 5 - w) \wedge (y_O \geq -10 - h) \wedge (y_O \leq x_O + w + h - 15) \wedge (y_O \geq x_O - w - h - 15))$$

Algorithm
Overview

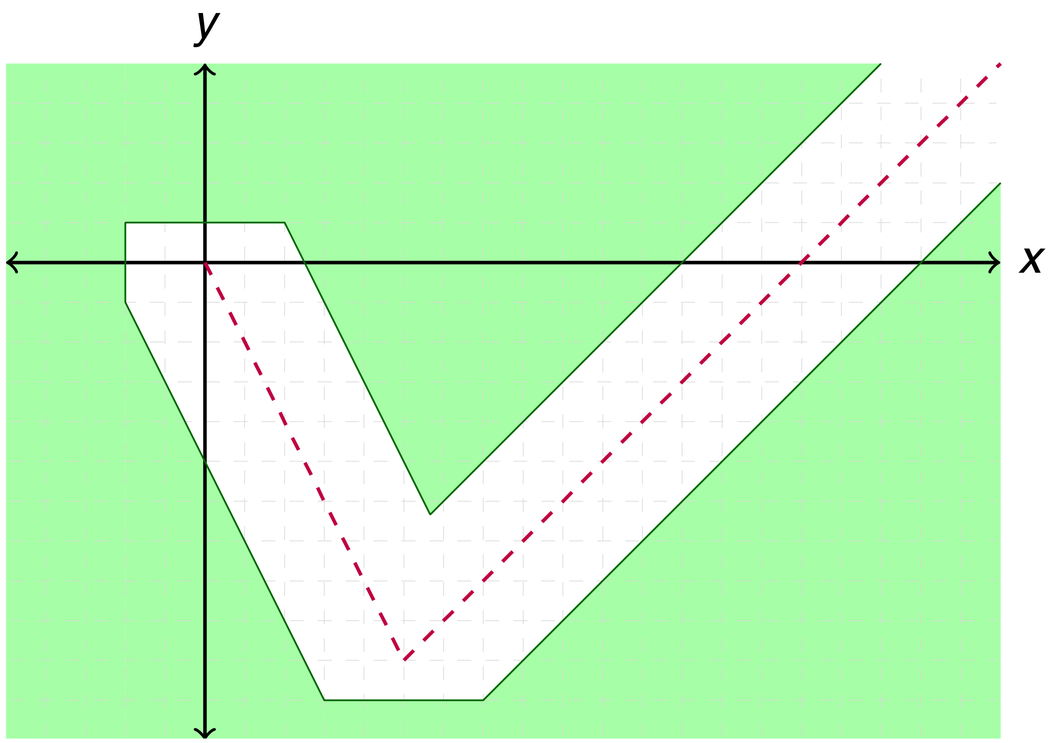
- Given a convex polygon translating on a *known* piecewise trajectory $\mathscr{T}$
- $$\mathscr{T} = \begin{cases} y = -2x, & x \in [0, 5] \\ y = x - 15, & x \in [5, \infty) \end{cases}$$
- $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \left(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \right)$
- Automatically construct a quantifier-free, geometric representation of the unsafe region



$$((x_O \geq -w) \wedge (y_O \leq h) \wedge (y_O \geq -2x_O - 2w - h) \wedge (x_O \leq 5 + w) \quad \wedge (y_O \leq -2x_O + 2w + h) \wedge (y_O \geq -10 - h)) \bigvee$$
$$((x_O \geq 5 - w) \wedge (y_O \geq -10 - h) \wedge (y_O \leq x_O + w + h - 15) \wedge (y_O \geq x_O - w - h - 15))$$

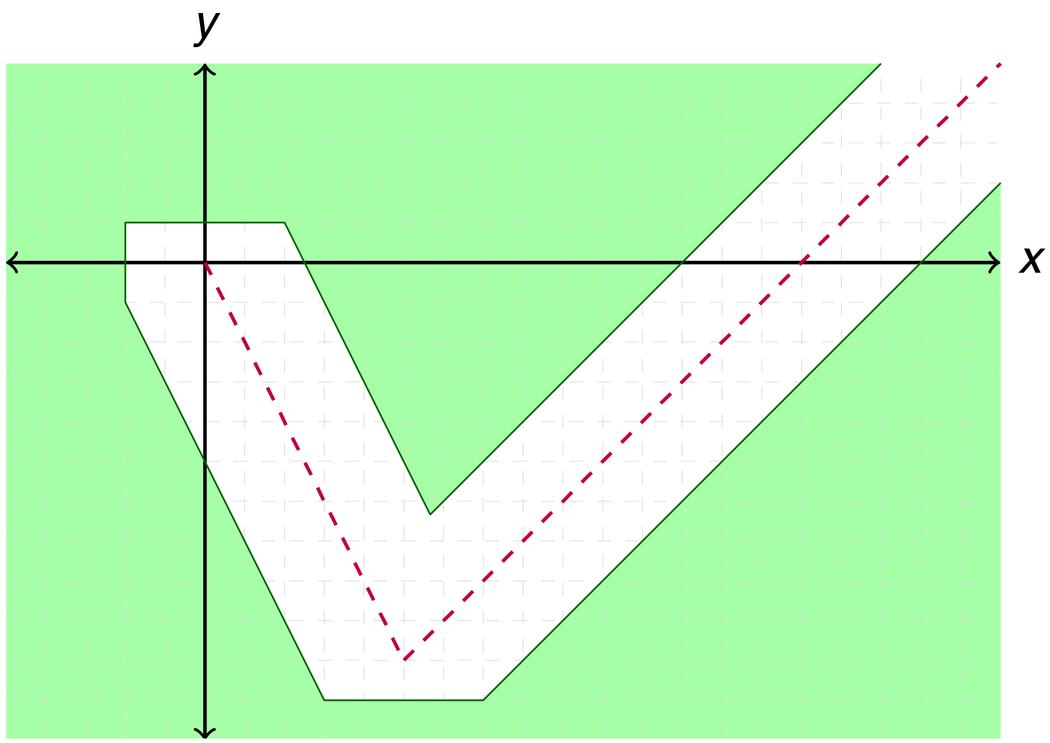
Algorithm
Intuition: Active Corners

- “Follow the corners”



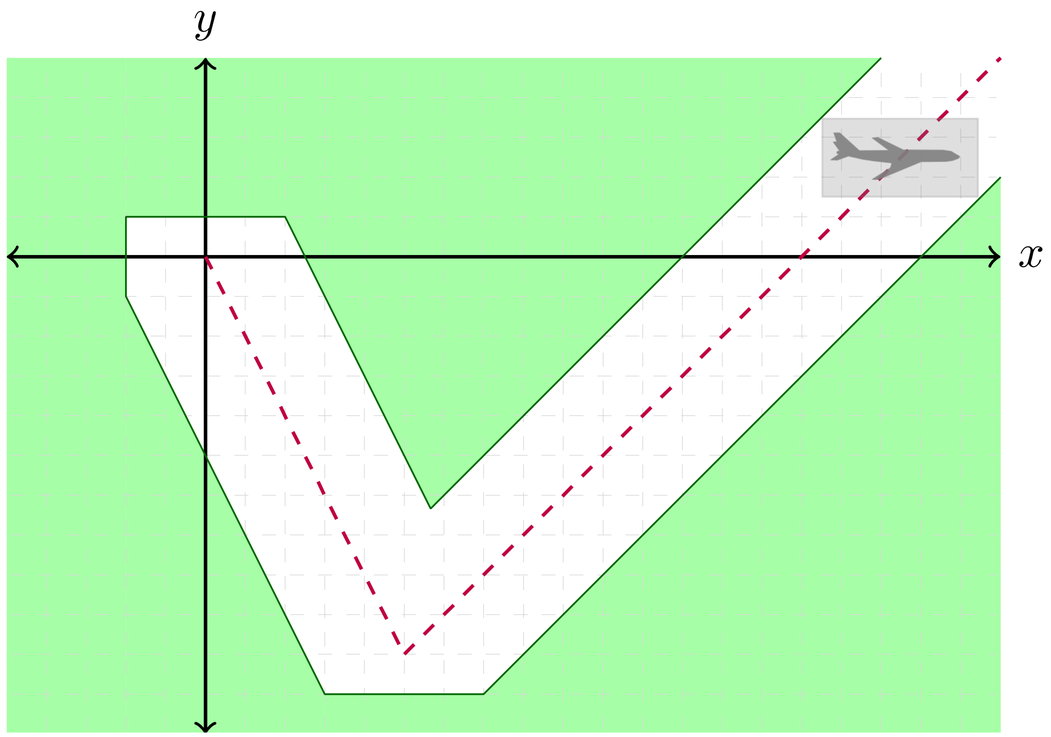
Algorithm
Intuition: Active Corners

- "Follow the corners"
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*



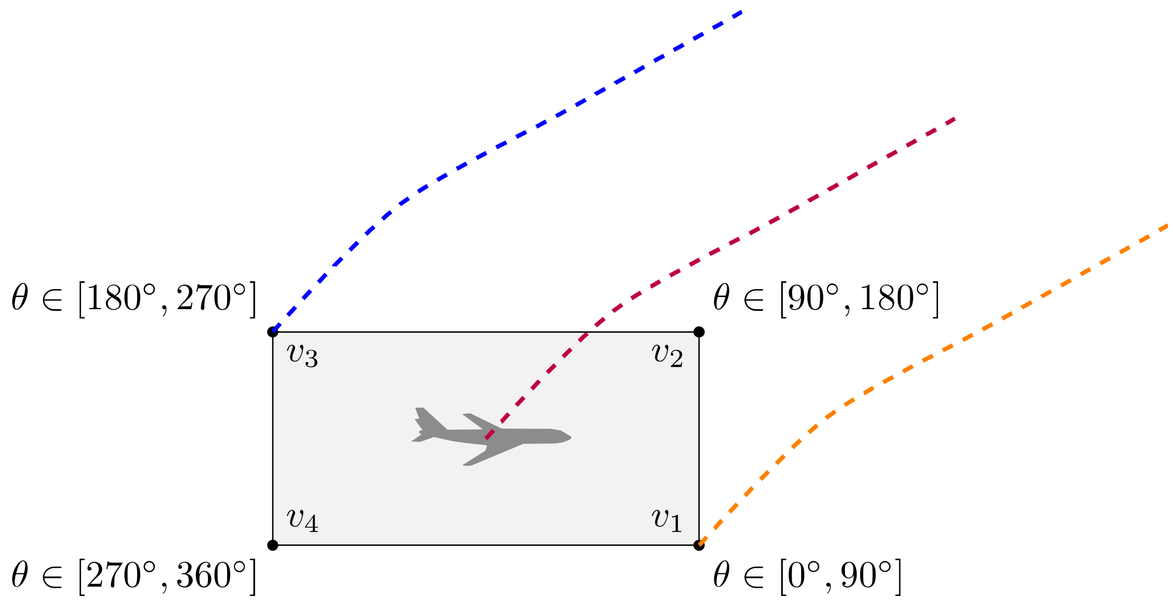
Algorithm
Intuition: Active Corners

- "Follow the corners"
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*



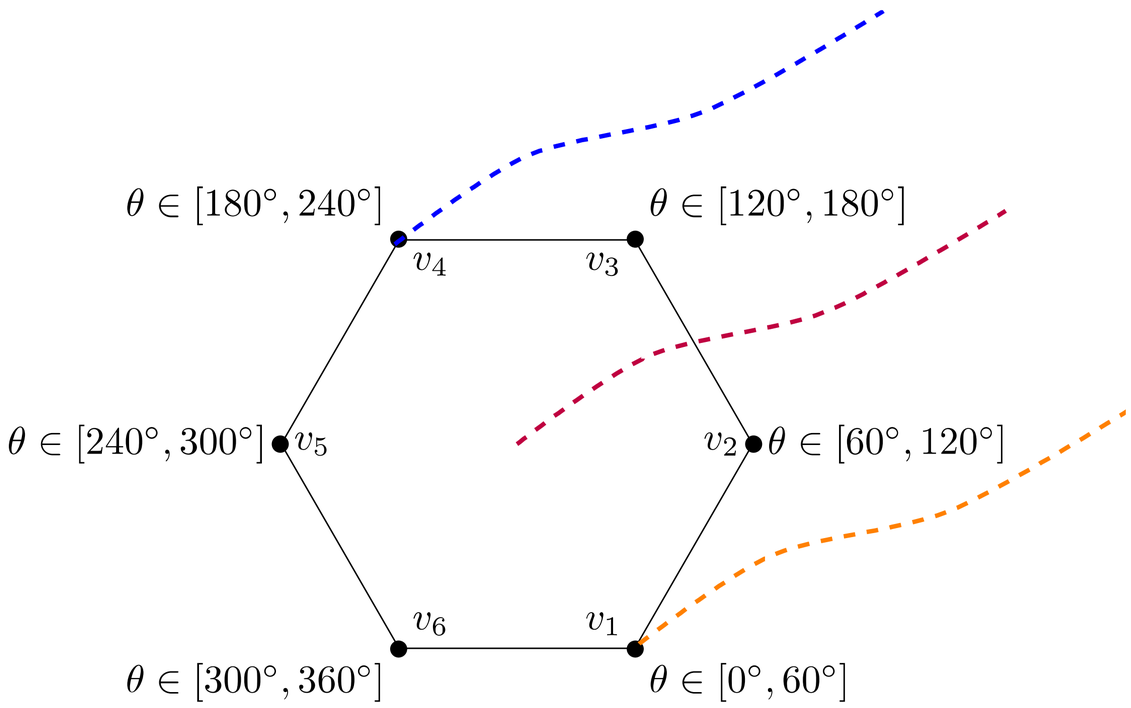
Algorithm
Intuition: Choosing active corners

- "Follow the corners"
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*
- Choose which ***active corners*** to follow based on the angle of the trajectory



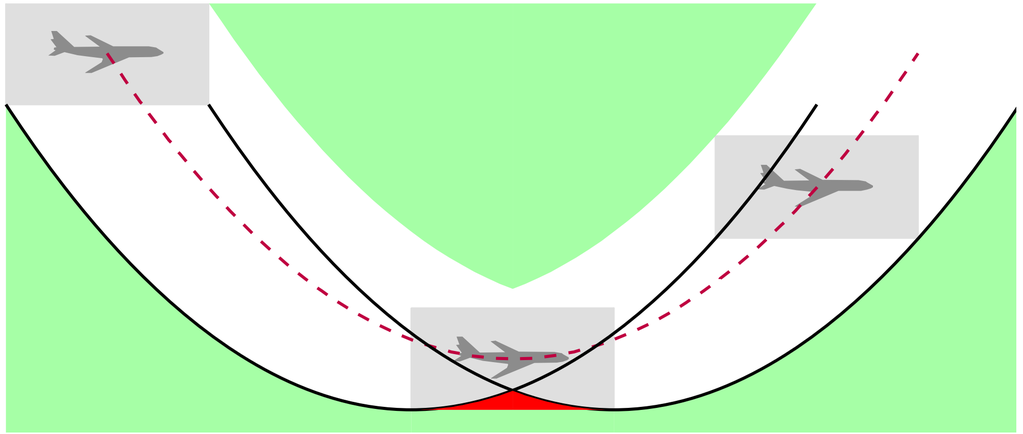
Algorithm
Intuition: Choosing active corners

- "Follow the corners"
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*
- Choose which ***active corners*** to follow based on the angle of the trajectory



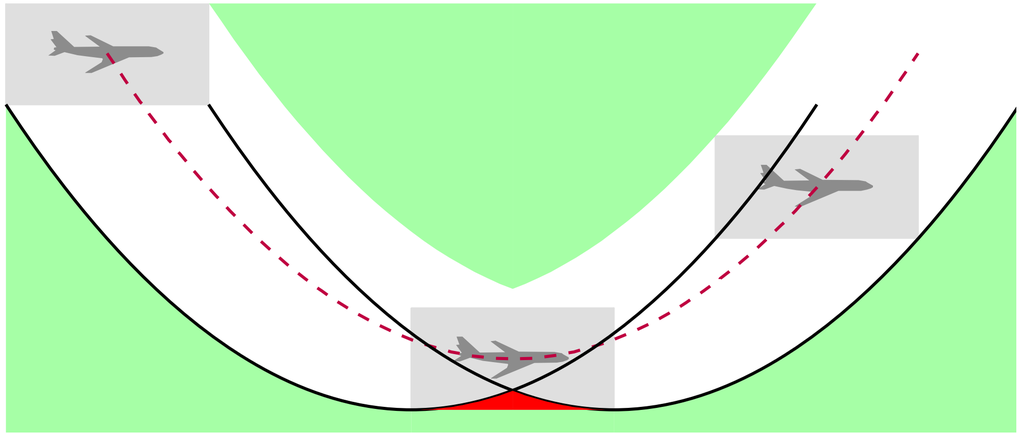
Algorithm
Intuition

- “Follow the corners”
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*
- Choose which ***active corners*** to follow based on the angle of the trajectory
- At ***transition points,*** the active corner changes: must also include polygon in the region



Algorithm
Intuition

- "Follow the corners"
- We found that the safe region boundaries were either *sides of the polygon* or *trajectories of its corners*
- Choose which ***active corners*** to follow based on the angle of the trajectory
- At ***transition points,*** the active corner changes: must also include polygon in the region
  - Using only corners misses the ***notch*** (red)

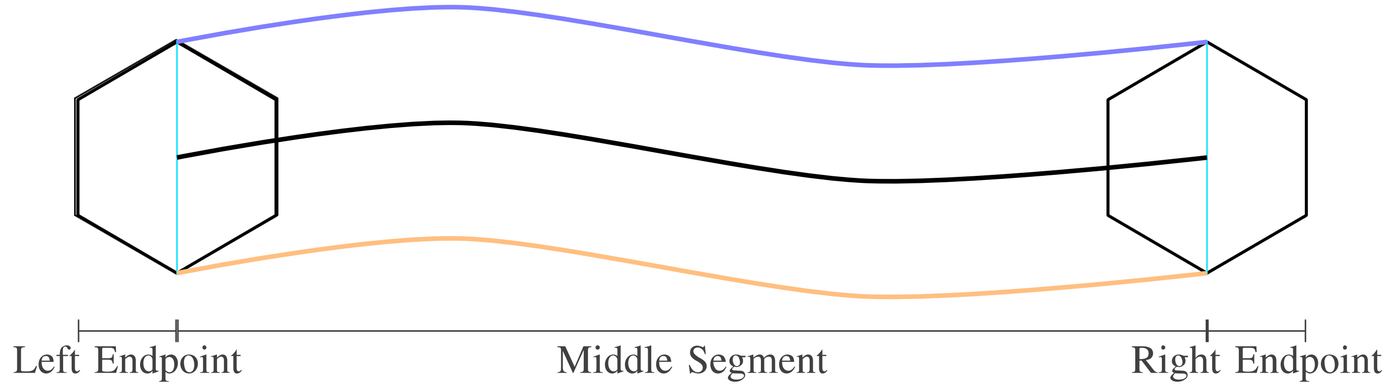


Proof Structure

- Proved equivalence of our method's output to the input *implicit region*
- Correctness implication: $safe_{expl} \implies safe_{impl}$
- $safe_{expl}$: $\Big(\big(y_O \geq x_O + w + h - 15 \big) \vee \big(y_O \leq x_O - w - h - 15 \big) \Big)$
- $safe_{impl}$: $\forall (x_{\mathscr{T}}, y_{\mathscr{T}}) \in \mathscr{T}, \big(|x_O - x_{\mathscr{T}}| > w \vee |y_O - y_{\mathscr{T}}| > h \big)$

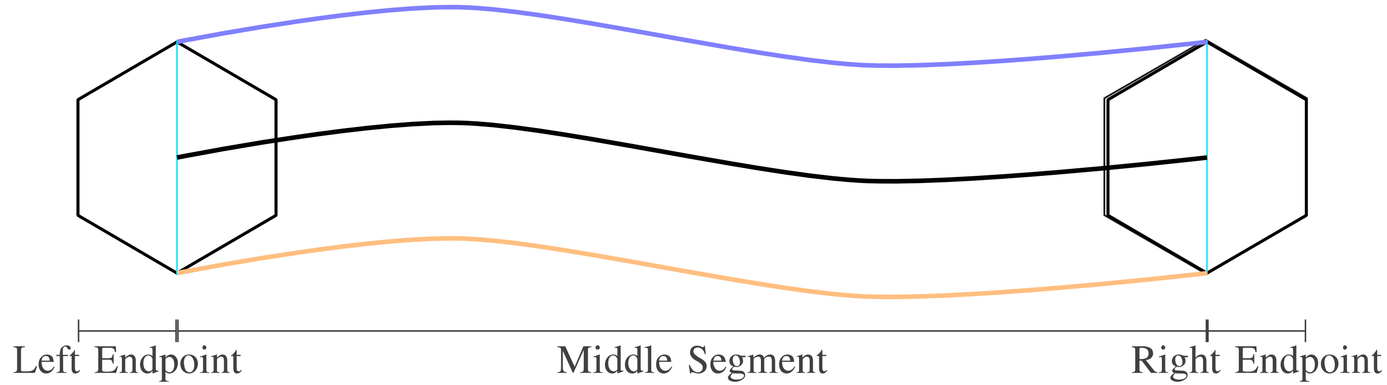
Proof Structure

- Proved equivalence of our method's output to the input *implicit region*
- Correctness: $safe_{expl} \implies safe_{impl.}$ By contradiction: $unsafe_{impl} \implies unsafe_{expl}$
- Intuitively: "any point inside any polygon along the trajectory lies either between the active corners or in a transition point"



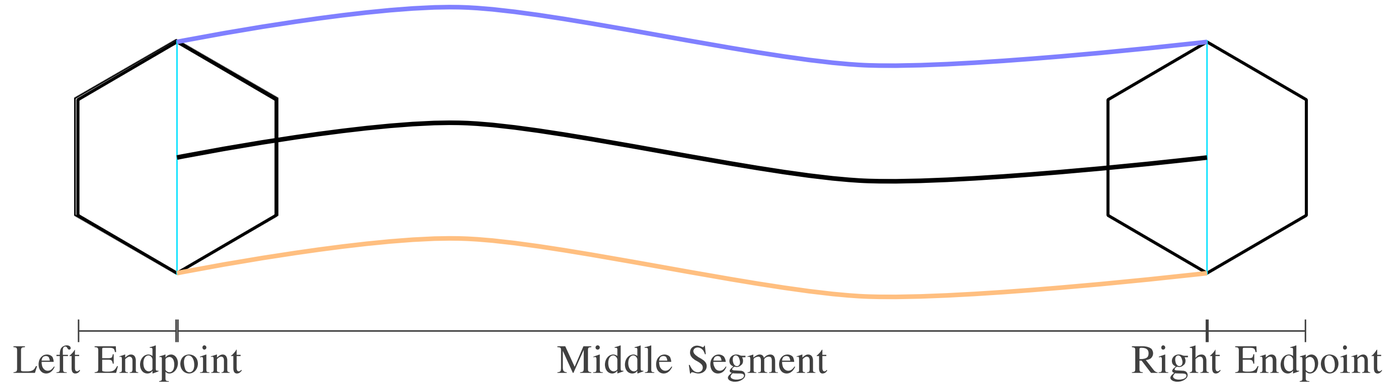
Proof Structure

- Proved equivalence of our method's output to the input *implicit region*
- Correctness: $safe_{expl} \implies safe_{impl.}$ By contradiction: $unsafe_{impl} \implies unsafe_{expl}$
- Intuitively: "any point inside any polygon along the trajectory lies either between the active corners or in a transition point"



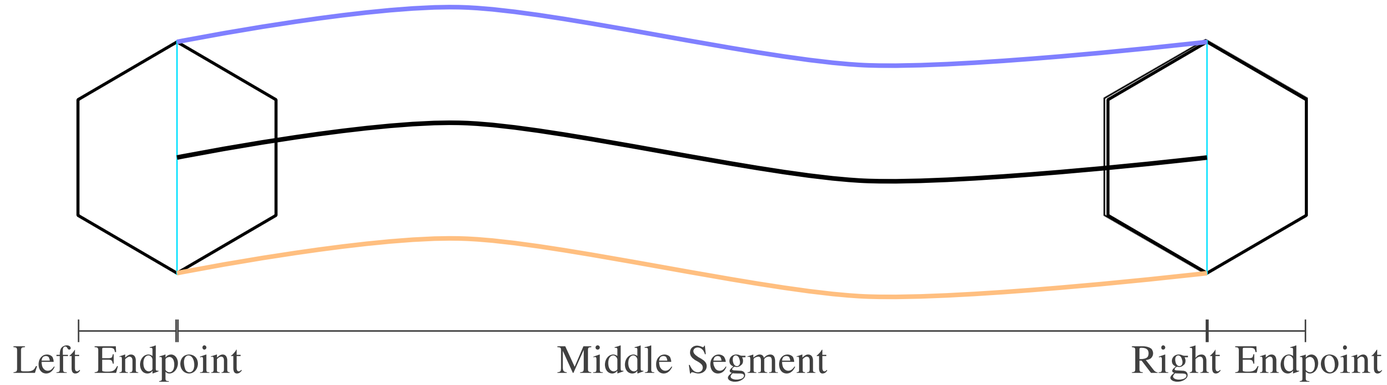
Proof Structure

- Proved equivalence of our method's output to the input *implicit region*
- Proof deals with (rotated) segments of trajectory over which there is no active corner change and includes start & end *transition points*
  - Accounts for notch handling this way



Proof Structure

- Proved equivalence of our method's output to the input *implicit region*
- Proof deals with (rotated) segments of trajectory over which there is no active corner change and includes start & end *transition points*
  - Accounts for notch handling this way



Implementation

- We have built an open-source library implementing the active corner method

Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math

Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions

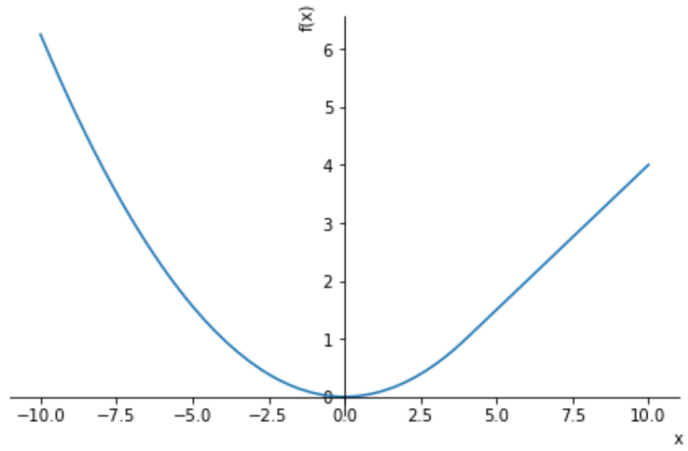
Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions
- Available on Github

Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions
- Available on Github

```
In [3]: x, y = symbols("x y")

        traj_piecewise = Piecewise(
                             (x**2 / 16, x < 4),
                             (x/2 - 1, x >= 4))
        plot(traj_piecewise)
```




```
Out[3]: <sympy.plotting.plot.Plot at 0x11e141f10>
```

Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions
- Available on Github

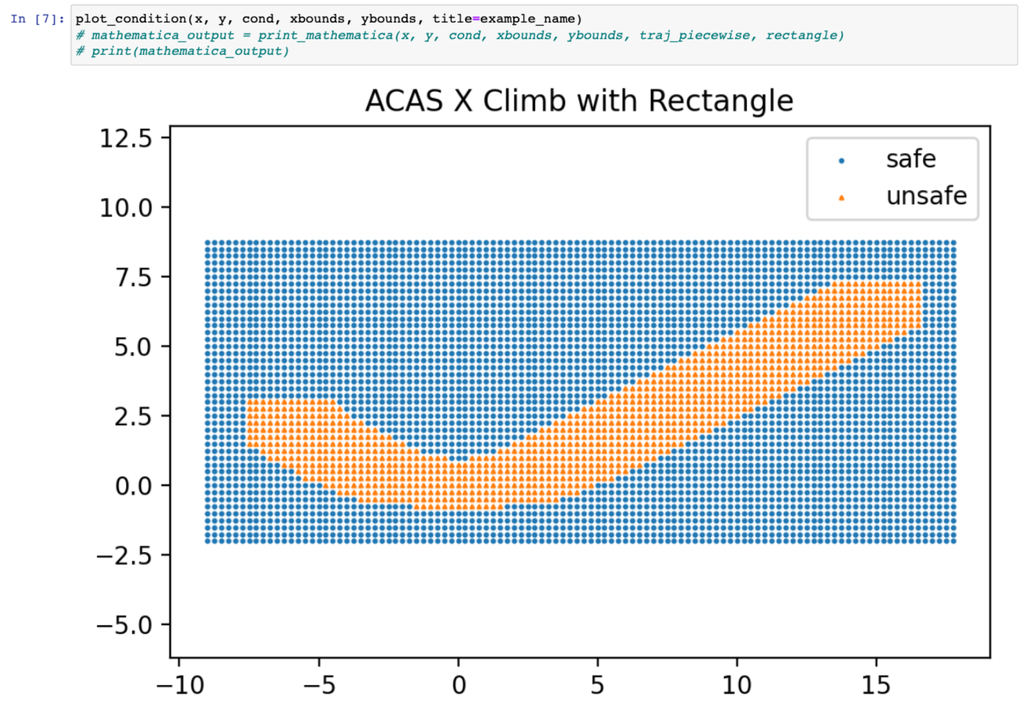
```
In [4]: domain = Interval(-6, 15)
        xbounds = (domain.inf-3, domain.sup+3)
        ybounds = (-2, 9)

In [5]: example_name = "ACAS X Climb with Rectangle"

        cond = compute_unsafe_cond(
            x, y,
            rectangle,
            traj_piecewise,
            domain = domain,
        )
        cond
```

Out[5]:

$$\left(x \geq -\frac{15}{2} \wedge x \leq \frac{3}{2} \wedge \left(\frac{3x}{4} - \frac{3y}{2}\right)\left(\frac{3x}{4} - \frac{3y}{2} + \frac{63}{8}\right) \leq 0 \wedge \left(y - \begin{cases} \frac{9}{4} & \text{for } x - \frac{3}{2} < -6 \\ \frac{\left(x-\frac{3}{2}\right)^2}{16} & \text{for } x - \frac{3}{2} \leq 0 \\ 0 & \text{otherwise} \end{cases} - \frac{3}{4}\right)\left(y - \begin{cases} \frac{9}{4} & \text{for } x + \frac{3}{2} < -6 \\ \frac{\left(x+\frac{3}{2}\right)^2}{16} & \text{for } x + \frac{3}{2} \leq 0 \\ 0 & \text{otherwise} \end{cases} + \frac{3}{4}\right) \leq 0\right)$$
$$\vee \left(x \geq -\frac{3}{2} \wedge x \leq \frac{11}{2} \wedge \left(\frac{3x}{4} + \frac{3y}{2}\right)\left(\frac{3x}{4} + \frac{3y}{2} - \frac{9}{2}\right) \leq 0 \wedge \left(y - \begin{cases} 0 & \text{for } x - \frac{3}{2} < 0 \\ \frac{\left(x-\frac{3}{2}\right)^2}{16} & \text{for } x - \frac{3}{2} \leq 4 \\ 1 & \text{otherwise} \end{cases} + \frac{3}{4}\right)\left(y - \begin{cases} 0 & \text{for } x + \frac{3}{2} < 0 \\ \frac{\left(x+\frac{3}{2}\right)^2}{16} & \text{for } x + \frac{3}{2} \leq 4 \\ 1 & \text{otherwise} \end{cases} - \frac{3}{4}\right) \leq 0\right)$$
$$\vee \left(x \geq \frac{5}{2} \wedge x \leq \frac{33}{2} \wedge \left(\frac{3x}{4} + \frac{3y}{2} - 21\right)\left(\frac{3x}{4} + \frac{3y}{2} - \frac{9}{2}\right) \leq 0 \wedge \left(y - \begin{cases} 1 & \text{for } x - \frac{3}{2} < 4 \\ \frac{x}{2} - \frac{7}{4} & \text{for } x - \frac{3}{2} \leq 15 \\ \frac{13}{2} & \text{otherwise} \end{cases} + \frac{3}{4}\right)\left(y - \begin{cases} 1 & \text{for } x + \frac{3}{2} < 4 \\ \frac{x}{2} - \frac{1}{4} & \text{for } x + \frac{3}{2} \leq 15 \\ \frac{13}{2} & \text{otherwise} \end{cases} - \frac{3}{4}\right) \leq 0\right)$$
$$\vee \left(\frac{9}{4} - \frac{3x}{2} \geq 0 \wedge \frac{9}{4} - 3y \geq 0 \wedge \frac{3x}{2} + \frac{9}{4} \geq 0 \wedge 3y + \frac{9}{4} \geq 0\right) \vee \left(\frac{21}{4} - 3y \geq 0 \wedge \frac{33}{4} - \frac{3x}{2} \geq 0 \wedge \frac{3x}{2} - \frac{15}{4} \geq 0 \wedge 3y - \frac{3}{4} \geq 0\right)$$
$$\vee \left(9 - 3y \geq 0 \wedge -\frac{3x}{2} - \frac{27}{4} \geq 0 \wedge \frac{3x}{2} + \frac{45}{4} \geq 0 \wedge 3y - \frac{9}{2} \geq 0\right) \vee \left(\frac{87}{4} - 3y \geq 0 \wedge \frac{99}{4} - \frac{3x}{2} \geq 0 \wedge \frac{3x}{2} - \frac{81}{4} \geq 0 \wedge 3y - \frac{69}{4} \geq 0\right)$$
$$\vee \left(\frac{9}{4} - \frac{3x}{2} \leq 0 \wedge \frac{9}{4} - 3y \leq 0 \wedge \frac{3x}{2} + \frac{9}{4} \leq 0 \wedge 3y + \frac{9}{4} \leq 0\right) \vee \left(\frac{21}{4} - 3y \leq 0 \wedge \frac{33}{4} - \frac{3x}{2} \leq 0 \wedge \frac{3x}{2} - \frac{15}{4} \leq 0 \wedge 3y - \frac{3}{4} \leq 0\right)$$
$$\vee \left(9 - 3y \leq 0 \wedge -\frac{3x}{2} - \frac{27}{4} \leq 0 \wedge \frac{3x}{2} + \frac{45}{4} \leq 0 \wedge 3y - \frac{9}{2} \leq 0\right) \vee \left(\frac{87}{4} - 3y \leq 0 \wedge \frac{99}{4} - \frac{3x}{2} \leq 0 \wedge \frac{3x}{2} - \frac{81}{4} \leq 0 \wedge 3y - \frac{69}{4} \leq 0\right)$$

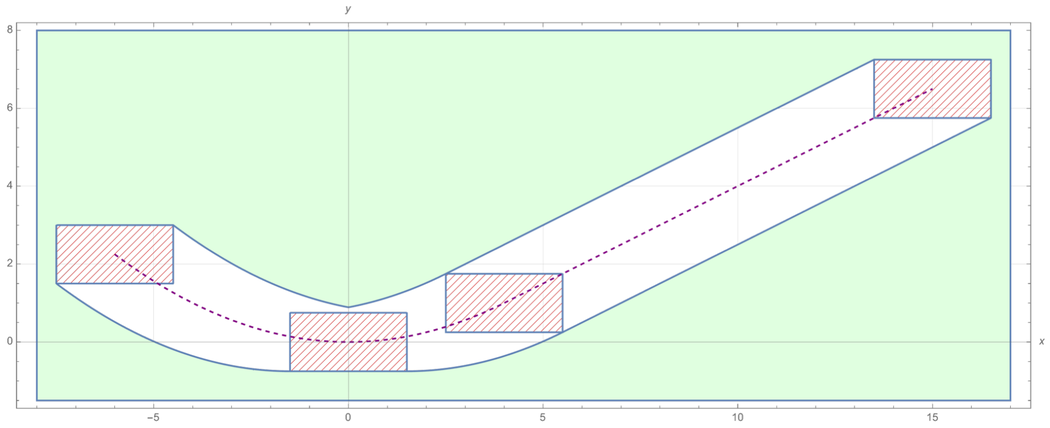
Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions
- Available on Github



Implementation

- We have built an open-source library implementing the active corner method
- Uses Python and Sympy for symbolic math
- Automatically generates fully symbolic safe or unsafe regions
- Available on Github

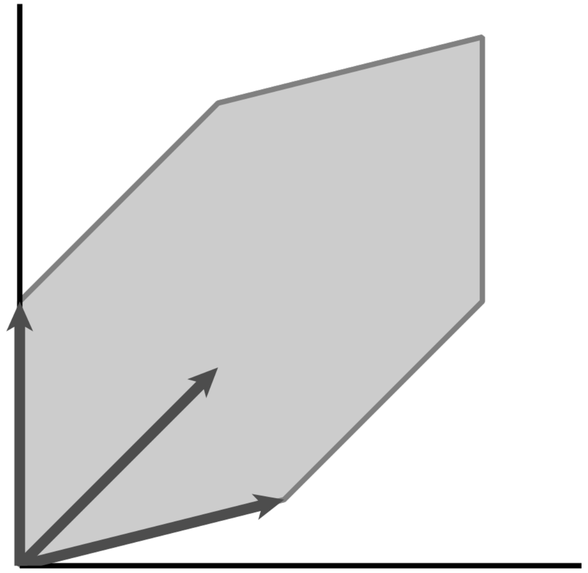


Related Work
Reachability & Safety

Collins, *ATFL* 1975. Ben-Or, Kozen, and Reif, *JCSS* 1986.

Related Work
Reachability & Safety

- Zonotope Reachability
  - Given a range of initial conditions and a differential equation
  - Propagate initial condition through differential equation and overapproximate reachable set
  - Results are not exact, unlike our method
- Quantifier Elimination by Cylindrical Algebraic Decomposition (CAD), Ben-Or, Kozen, and Reif (BKR), etc



Collins, *ATFL* 1975. Ben-Or, Kozen, and Reif, *JCSS* 1986.

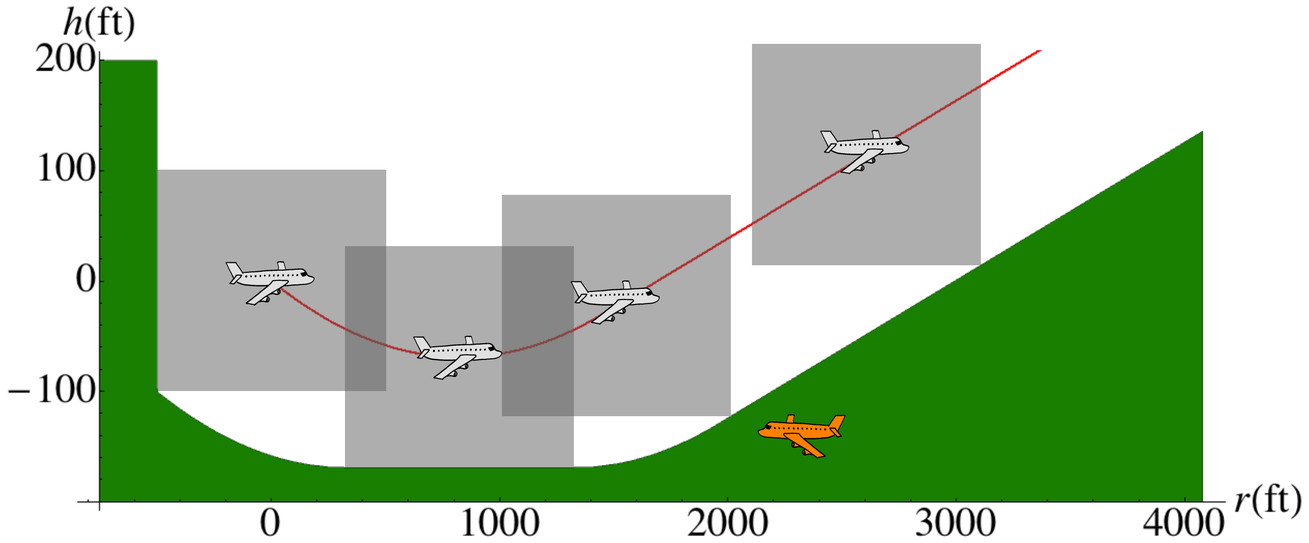
Related Work
Reachability & Safety

- Zonotope Reachability
  - Given a range of initial conditions and a differential equation
  - Propagate initial condition through differential equation and overapproximate reachable set
  - Results are not exact, unlike our method
- Quantifier Elimination by Cylindrical Algebraic Decomposition (CAD), Ben-Or, Kozen, and Reif (BKR), etc
  - $$\exists x \in \mathbf{R} \,.\, (a \neq 0 \wedge ax^2 + bx + c = 0) \iff a \neq 0 \wedge b^2 - 4ac \geq 0$$
  - Most general approach for generating an exact explicit safe region
  - Runtime is prohibitive for symbolic applications: doubly exponential

Collins, *ATFL* 1975. Ben-Or, Kozen, and Reif, *JCSS* 1986.

Applications

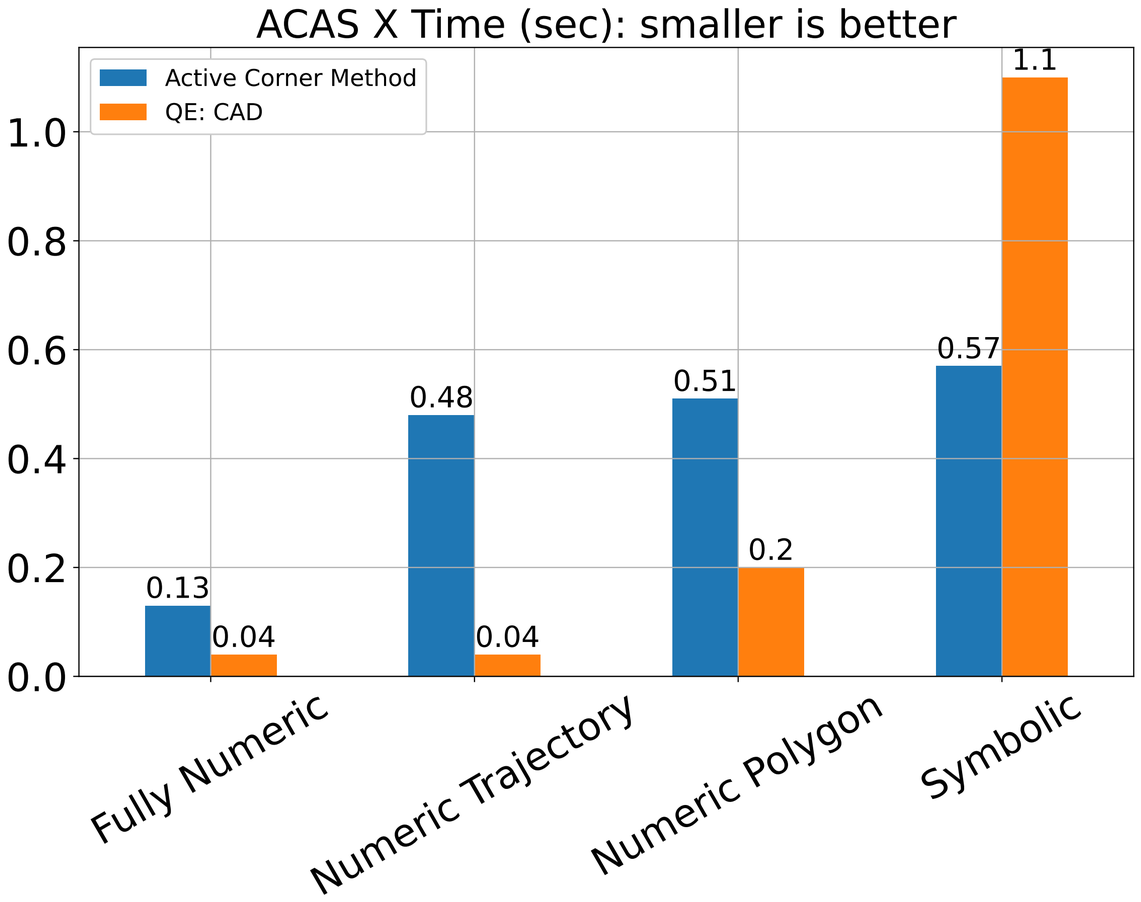
- Applied to two past papers from our group performing collision avoidance verification
- TACAS ’15 ACAS X paper verifying aircraft collision avoidance advisories
  - Rectangular object
  - Parabolic-then-linear path



Performance Evaluation

Case Study 1

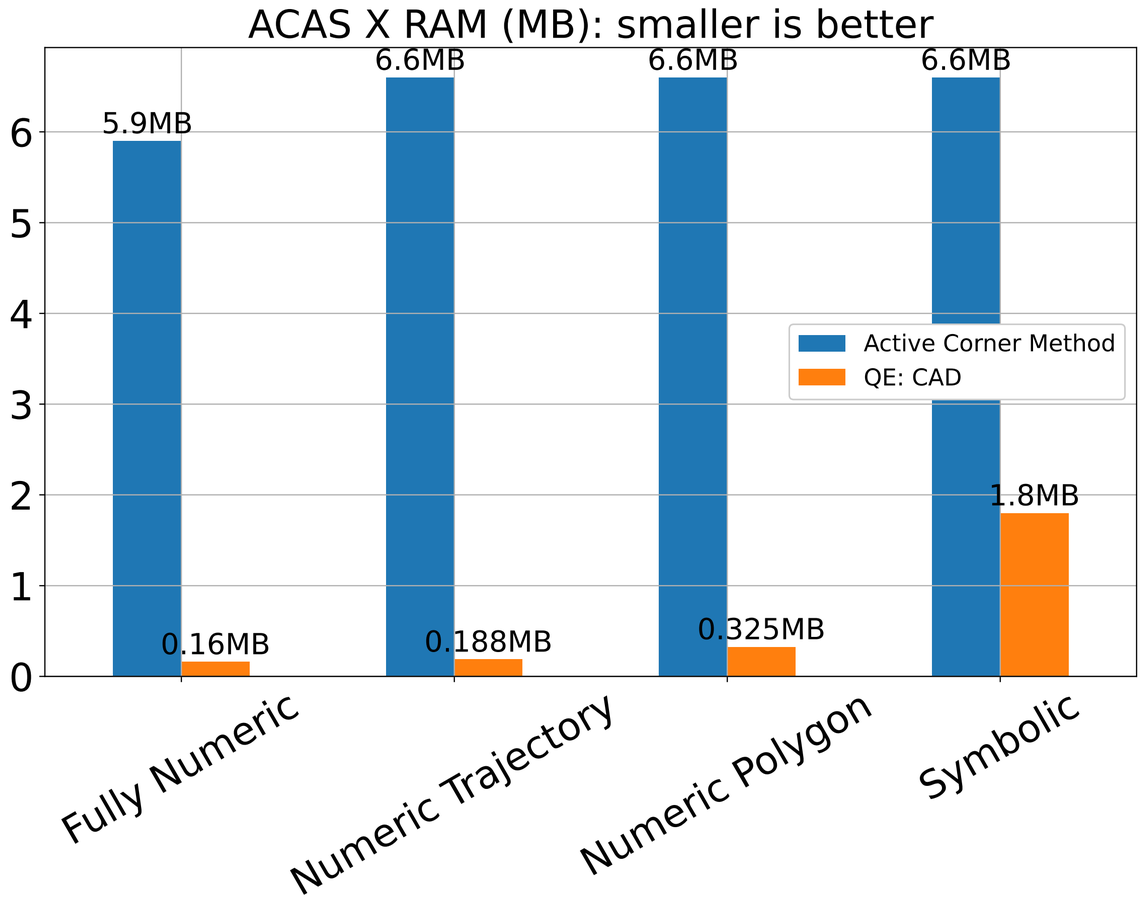
- TACAS ’15 ACAS X verification paper with side-view of aircraft collision avoidance advisories
- For this polynomial example, CAD performs **comparably** to our implementation for numeric cases
- Our implementation begins to beat CAD as the problem becomes **increasingly symbolic**



Performance Evaluation

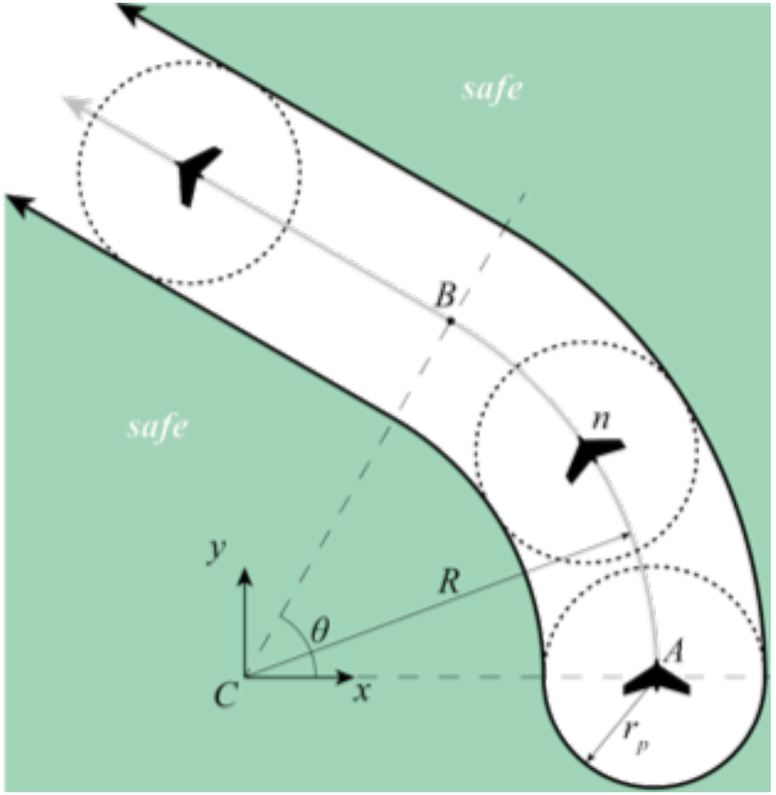
Case Study 1

- TACAS ’15 ACAS X verification paper with side-view of aircraft collision avoidance advisories
- As for memory usage, CAD uses **less memory** than our implementation
- Note that Mathematica is **purpose-built** for efficient mathematical computation



Applications
Case Study 2

- UAV top-down collision avoidance
  - Approximate circle as a hexagon
  - Circular-then-linear path

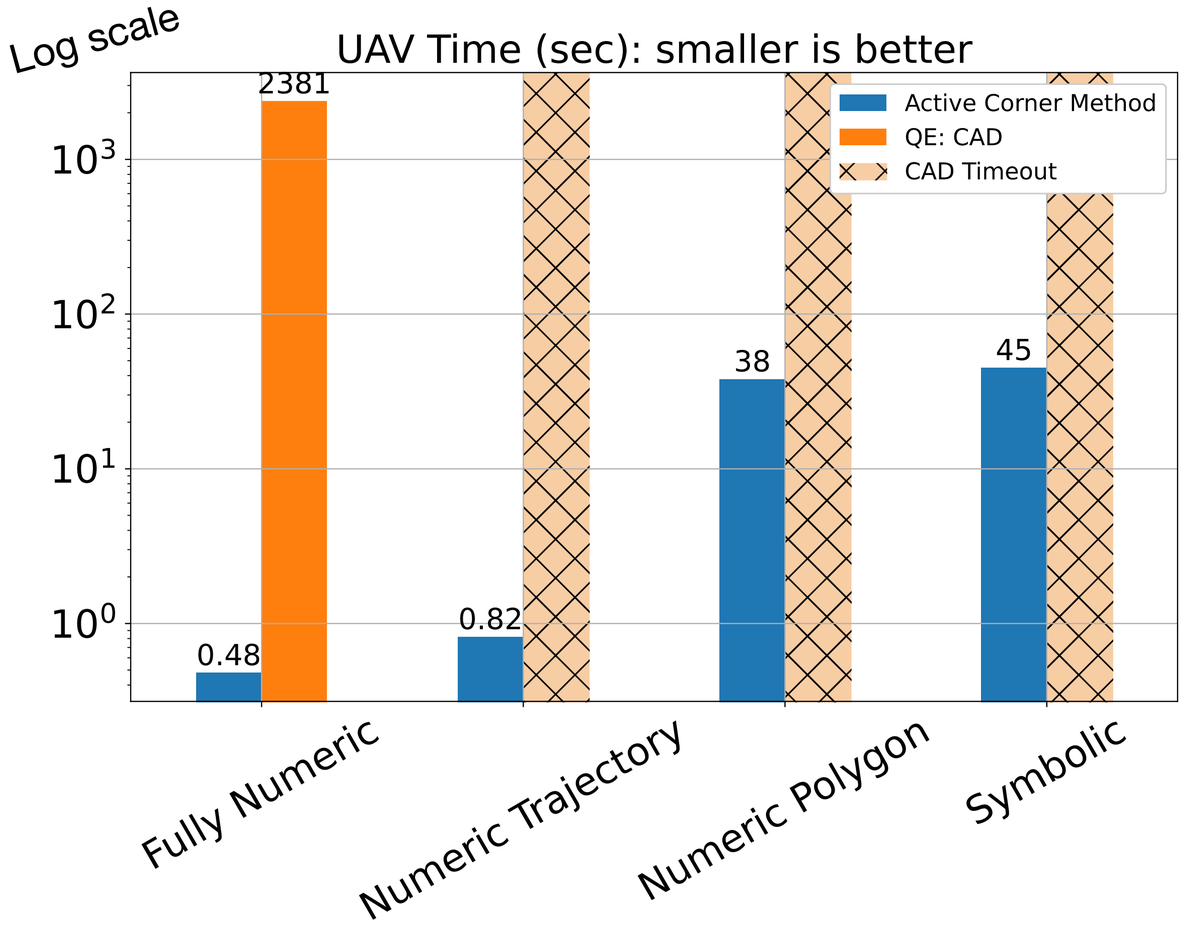


Adler et al., AIAA ’19.

Performance Evaluation

Case Study 2

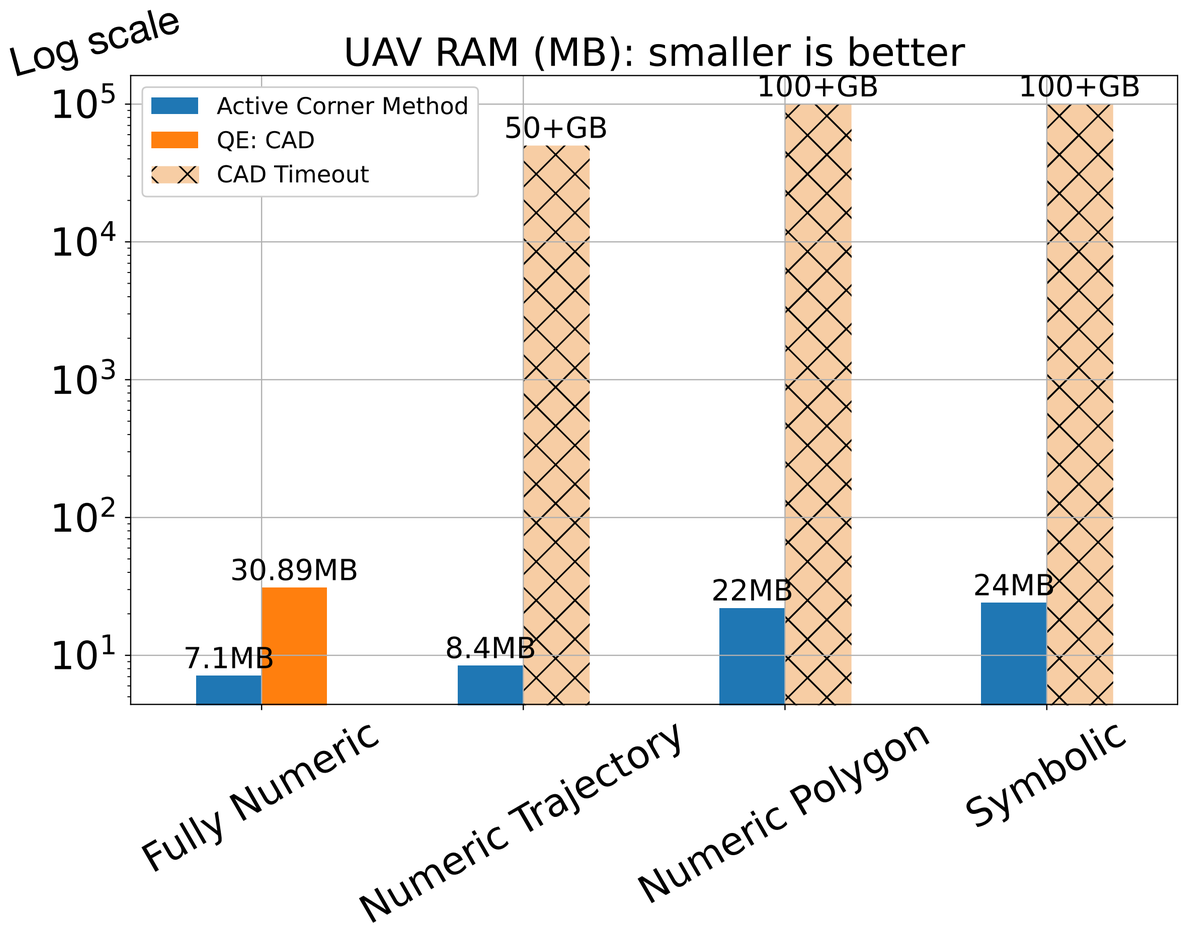
- UAV top-down collision avoidance
- CAD **does not terminate** for all but the fully numeric example
- Ran overnight on an iMac Pro with 128GB RAM
- Our implementation runs in **under a minute** at worst



Performance Evaluation

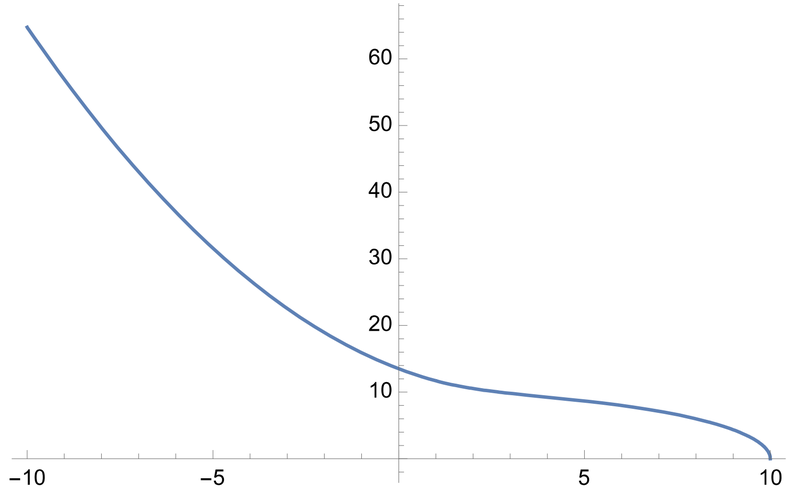
Case Study 2

- UAV top-down collision avoidance
- For all but the fully numeric case, CAD memory usage **explodes** to over 50 GB
- Our implementation keeps memory usage **under 25 MB**



Performance Evaluation
Dubins Path

- Tested the active corner method with a highly non-polynomial trajectory
- Fully symbolic Dubins path: semicircular segments connected by straight lines
- Commonly used for robot motion planning

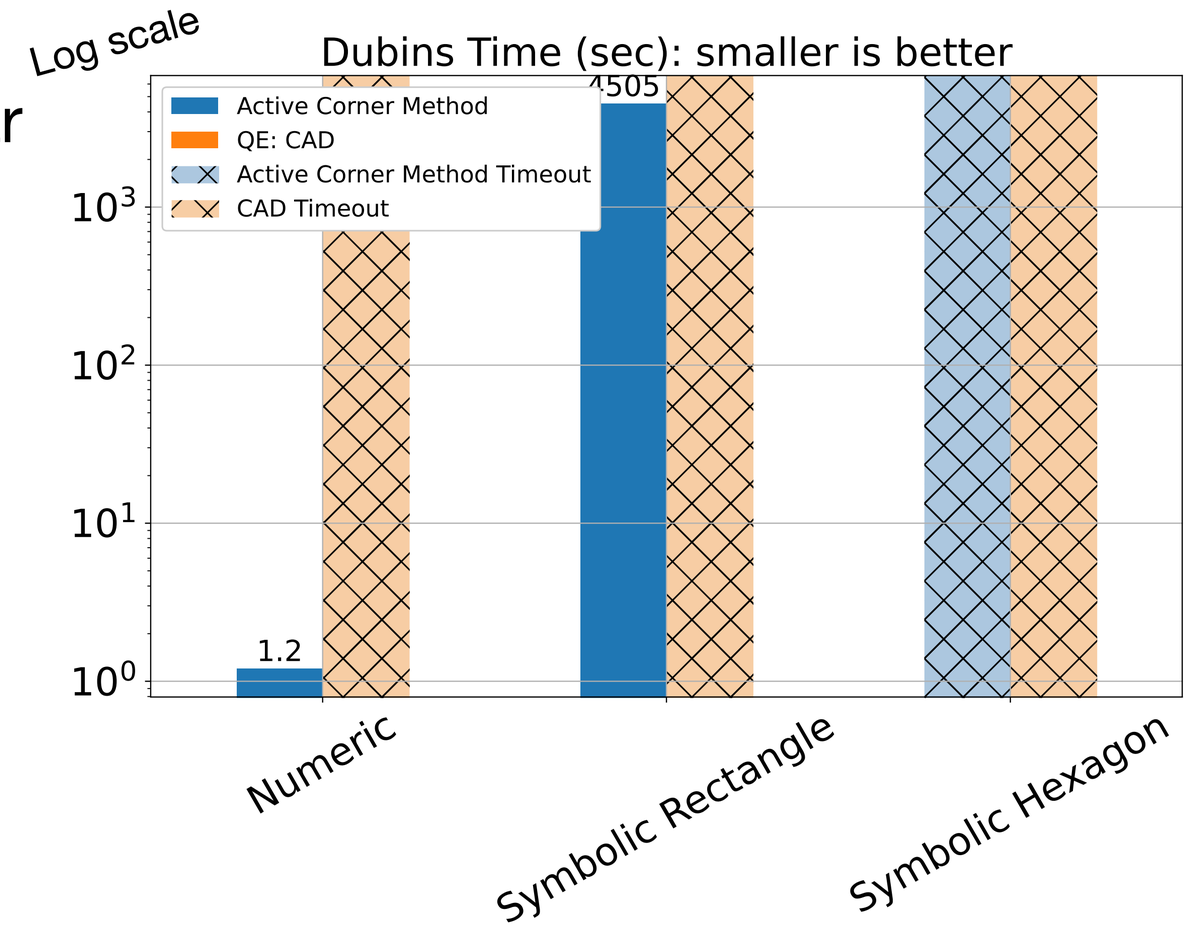


$$\begin{cases} \sqrt{R^2 - x^2} & \text{for } x > \frac{R}{\sqrt{\tan^2(\theta)+1}} \\ R\sin(\theta) - \frac{-R\cos(\theta)+x}{\tan(\theta)} & \text{for } p < x \\ \frac{R}{\sin(\theta)} - \frac{p}{\tan(\theta)} - \sqrt{\frac{p^2}{\cos^2(\theta)} - (-2p+x)^2} + |p\tan(\theta)| & \text{for } b < x \\ \frac{R}{\sin(\theta)} - \frac{p}{\tan(\theta)} + \frac{(-b+x)(-2p+x)|\cos(\theta)|}{\sqrt{p^2-(b-2p)^2\cos^2(\theta)}} - \frac{\sqrt{-b^2\cos^2(\theta)+4bp\cos^2(\theta)-4p^2\cos^2(\theta)+p^2}}{|\cos(\theta)|} + |p\tan(\theta)| & \text{otherwise} \end{cases}$$

Performance Evaluation

Dubins Path

- Fully symbolic Dubins path: semicircular segments connected by straight lines
- CAD **does not terminate** for any case, including numeric examples
- Active corner method fails to terminate for a symbolic hexagon
  - Too many transition points
- Ran overnight on an iMac Pro with 128GB RAM

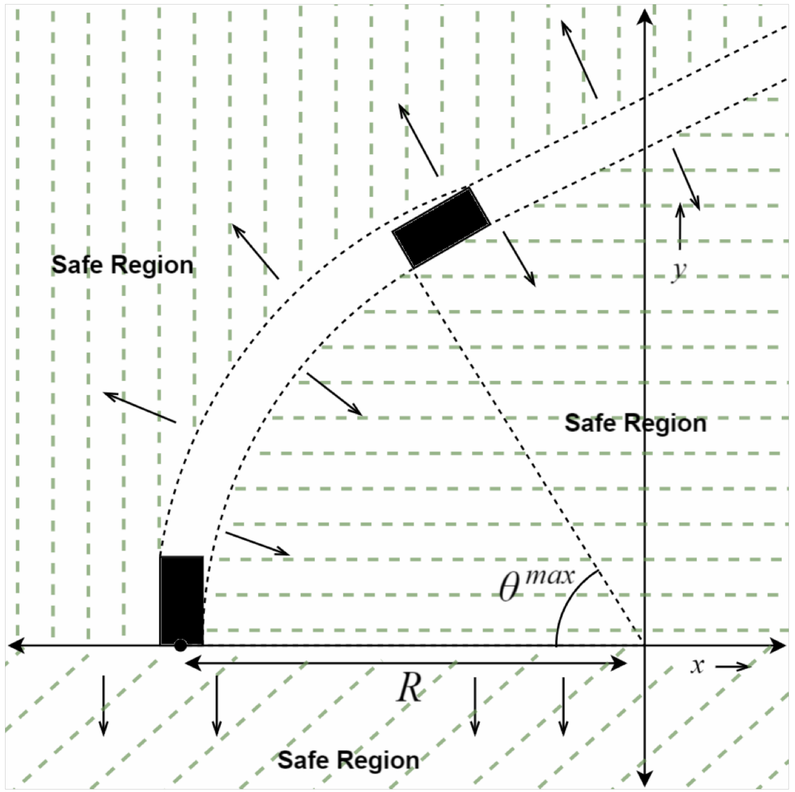


Summary

- Automated method for constructing *symbolic geometric safe regions* without quantifiers from convex polygon and piecewise trajectories
- Proof of correctness for the active corner method
- Open-source Python implementation using Sympy
- Performance study and comparison to Cylindrical Algebraic Decomposition

Future Work

- Automatic proof certificate generation using PVS[1]
- Extension of this method to rotating objects or n-dimensional settings



Thanks!

Any questions?

TOYOTA RESEARCH INSTITUTE

NASA

Thanks!

Any questions?

Backup slides next

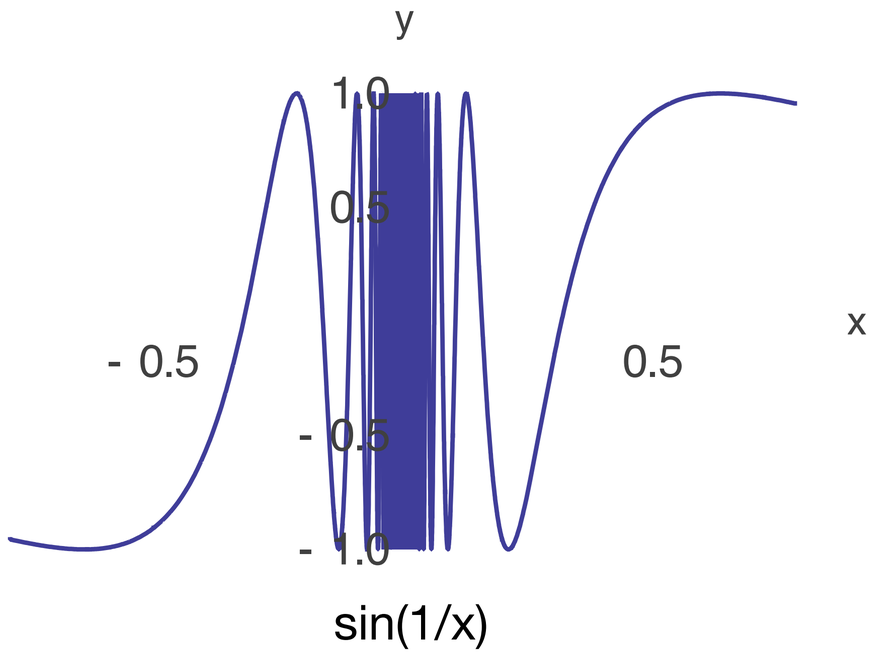
TOYOTA RESEARCH INSTITUTE

NASA

Backup

Proof Structure
(Mild) Assumptions

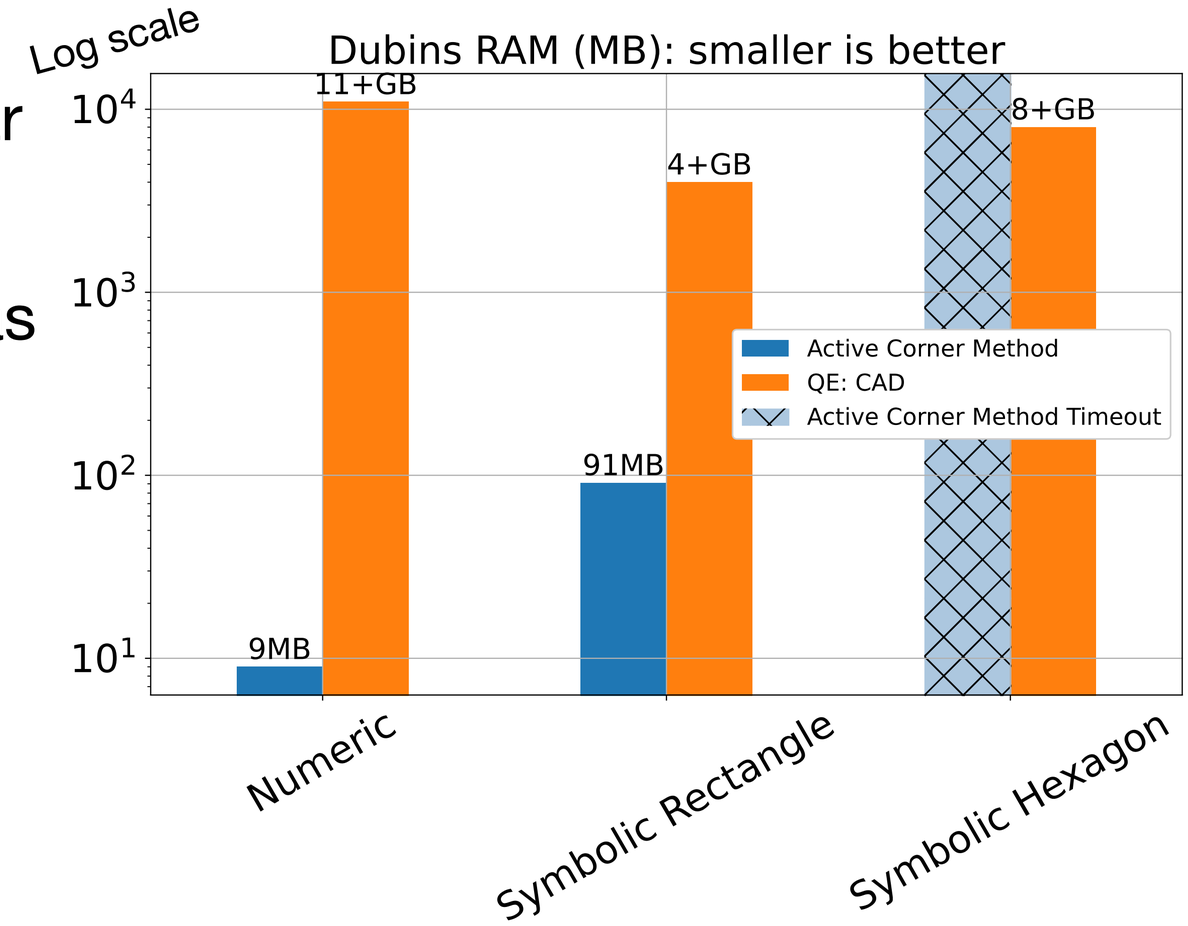
- Proof considers trajectories with a finite number of transition points over a a finite domain
- Without this, there are an infinite number of notches to account for even on finite domains
- In practice, this assumption is mild, though pathological trajectories do exist



Performance Evaluation

Dubins Path

- Fully symbolic Dubins path: semicircular segments connected by straight lines
- CAD RAM usage grows to several GB as it fails to terminate
- Active corner method memory usage stays under 100 MB if terminating



Performance Evaluation

Example	Instance	Active Corners Time	Active Corners RAM	CAD Time	CAD RAM
UAV	Fully Numeric	**0.48 sec**	**7.1 MB**	*2381* sec*	30.89 MB
UAV	Numeric Trajectory	**0.82 sec**	**8.4 MB**	DNF	50+ GB
UAV	Numeric Hexagon	**38 sec**	**22 MB**	DNF	100+ GB
UAV	Fully Symbolic	**45 sec**	**24 MB**	DNF	100+ GB
Dubins	Fully Numeric	**1.2 sec**	**9.0 MB**	DNF	11+ GB
Dubins	Fully Symbolic: Rectangle	**4505 sec**	**91 MB**	DNF	4+ GB
Dubins	Fully Symbolic: Hexagon	DNF	N/A	DNF	8+ GB
ACAS X	Fully Numeric	0.13 sec	5.9 MB	**0.04 sec**	**160 KB**
ACAS X	Numeric Trajectory	0.48 sec	6.6 MB	**0.04 sec**	**188 KB**
ACAS X	Numeric Rectangle	0.51 sec	6.6 MB	**0.2 sec**	**325 KB**
ACAS X	Fully Symbolic	**0.57 sec**	6.6 MB	1.1 sec	**1.8 MB**

Better results **bolded**. DNF: example did not finish in 8+ hours. **: incorrect answer*

Performance Evaluation

Example	Instance	Active Corners Time	Active Corners RAM	CAD Time	CAD RAM
UAV	Fully Numeric	**0.48 sec**	**7.1 MB**	*2381* sec*	30.89 MB
UAV	Numeric Trajectory	**0.82 sec**	**8.4 MB**	DNF	50+ GB
UAV	Numeric Hexagon	**38 sec**	**22 MB**	DNF	100+ GB
UAV	Fully Symbolic	**45 sec**	**24 MB**	DNF	100+ GB
Dubins	Fully Numeric	**1.2 sec**	**9.0 MB**	DNF	11+ GB
Dubins	Fully Symbolic: Rectangle	**4505 sec**	**91 MB**	DNF	4+ GB
Dubins	Fully Symbolic: Hexagon	DNF	N/A	DNF	8+ GB
ACAS X	Fully Numeric	0.13 sec	5.9 MB	**0.04 sec**	**160 KB**
ACAS X	Numeric Trajectory	0.48 sec	6.6 MB	**0.04 sec**	**188 KB**
ACAS X	Numeric Rectangle	0.51 sec	6.6 MB	**0.2 sec**	**325 KB**
ACAS X	Fully Symbolic	**0.57 sec**	6.6 MB	1.1 sec	**1.8 MB**

Better results **bolded**. DNF: example did not finish in 8+ hours. **: incorrect answer*

$$\begin{cases} cx^2 & \text{for } b > x \\ b^2c + 2bc\,(-b + x) & \text{otherwise} \end{cases}$$

ACAS X

Performance Evaluation

Example	Instance	Active Corners Time	Active Corners RAM	CAD Time	CAD RAM
UAV	Fully Numeric	**0.48 sec**	**7.1 MB**	*2381* sec*	30.89 MB
UAV	Numeric Trajectory	**0.82 sec**	**8.4 MB**	DNF	50+ GB
UAV	Numeric Hexagon	**38 sec**	**22 MB**	DNF	100+ GB
UAV	Fully Symbolic	**45 sec**	**24 MB**	DNF	100+ GB
Dubins	Fully Numeric	**1.2 sec**	**9.0 MB**	DNF	11+ GB
Dubins	Fully Symbolic: Rectangle	**4505 sec**	**91 MB**	DNF	4+ GB
Dubins	Fully Symbolic: Hexagon	DNF	N/A	DNF	8+ GB
ACAS X	Fully Numeric	0.13 sec	5.9 MB	**0.04 sec**	**160 KB**
ACAS X	Numeric Trajectory	0.48 sec	6.6 MB	**0.04 sec**	**188 KB**
ACAS X	Numeric Rectangle	0.51 sec	6.6 MB	**0.2 sec**	**325 KB**
ACAS X	Fully Symbolic	**0.57 sec**	6.6 MB	1.1 sec	**1.8 MB**

Better results **bolded**. DNF: example did not finish in 8+ hours. **: incorrect answer*

$$\begin{cases} cx^2 & \text{for } b > x \\ b^2c + 2bc\left(-b + x\right) & \text{otherwise} \end{cases}$$

ACAS X

$$\begin{cases} \sqrt{R^2 - x^2} & \text{for } x > \frac{R}{\sqrt{\tan^2(\theta)+1}} \\ R\sin(\theta) - \frac{-R\cos(\theta)+x}{\tan(\theta)} & \text{otherwise} \end{cases}$$

UAV

Performance Evaluation

Example	Instance	Active Corners Time	Active Corners RAM	CAD Time	CAD RAM
UAV	Fully Numeric	**0.48 sec**	**7.1 MB**	*2381* sec*	30.89 MB
UAV	Numeric Trajectory	**0.82 sec**	**8.4 MB**	DNF	50+ GB
UAV	Numeric Hexagon	**38 sec**	**22 MB**	DNF	100+ GB
UAV	Fully Symbolic	**45 sec**	**24 MB**	DNF	100+ GB
Dubins	Fully Numeric	**1.2 sec**	**9.0 MB**	DNF	11+ GB
Dubins	Fully Symbolic: Rectangle	**4505 sec**	**91 MB**	DNF	4+ GB
Dubins	Fully Symbolic: Hexagon	DNF	N/A	DNF	8+ GB
ACAS X	Fully Numeric	0.13 sec	5.9 MB	**0.04 sec**	**160 KB**
ACAS X	Numeric Trajectory	0.48 sec	6.6 MB	**0.04 sec**	**188 KB**
ACAS X	Numeric Rectangle	0.51 sec	6.6 MB	**0.2 sec**	**325 KB**
ACAS X	Fully Symbolic	**0.57 sec**	6.6 MB	1.1 sec	**1.8 MB**

Better results **bolded**. DNF: example did not finish in 8+ hours. **: incorrect answer*

$$\begin{cases} \sqrt{R^2 - x^2} & \text{for } x > \frac{R}{\sqrt{\tan^2(\theta)+1}} \\ R\sin(\theta) - \frac{-R\cos(\theta)+x}{\tan(\theta)} & \text{for } p < x \\ \frac{R}{\sin(\theta)} - \frac{p}{\tan(\theta)} - \sqrt{\frac{p^2}{\cos^2(\theta)} - (-2p+x)^2} + |p\tan(\theta)| & \text{for } b < x \\ \frac{R}{\sin(\theta)} - \frac{p}{\tan(\theta)} + \frac{(-b+x)(-2p+x)|\cos(\theta)|}{\sqrt{p^2-(b-2p)^2\cos^2(\theta)}} - \frac{\sqrt{-b^2\cos^2(\theta)+4bp\cos^2(\theta)-4p^2\cos^2(\theta)+p^2}}{|\cos(\theta)|} + |p\tan(\theta)| & \text{otherwise} \end{cases}$$

Dubins