University of Virginia | School of Engineering & Applied Science

Blending Kinematic and Software Models for Tighter Reachability Analysis

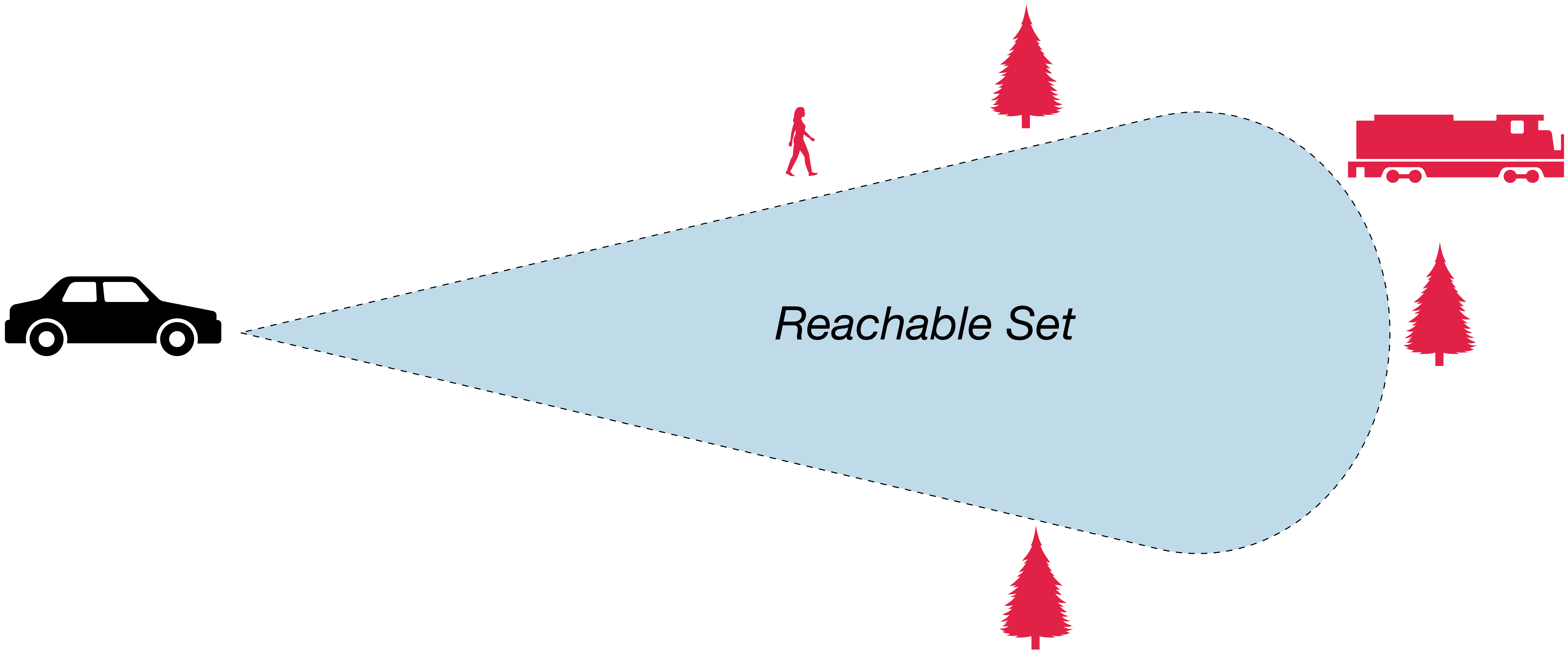
Carl Hildebrandt, Sebastian Elbaum, Nicola Bezzo

Contact: hildebrandt.carl@virginia.edu

This work was funded in part by the NSF

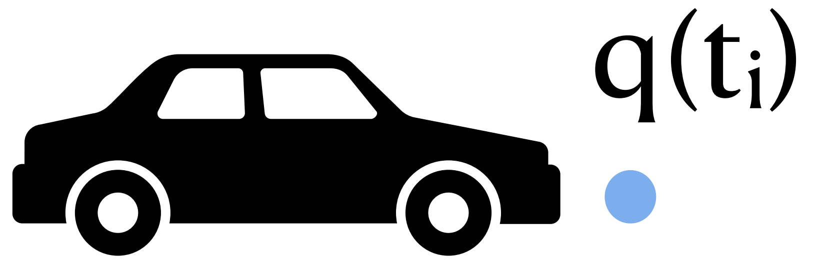
Motivation

Reachable sets are critical for path planning and navigation of mobile autonomous systems.



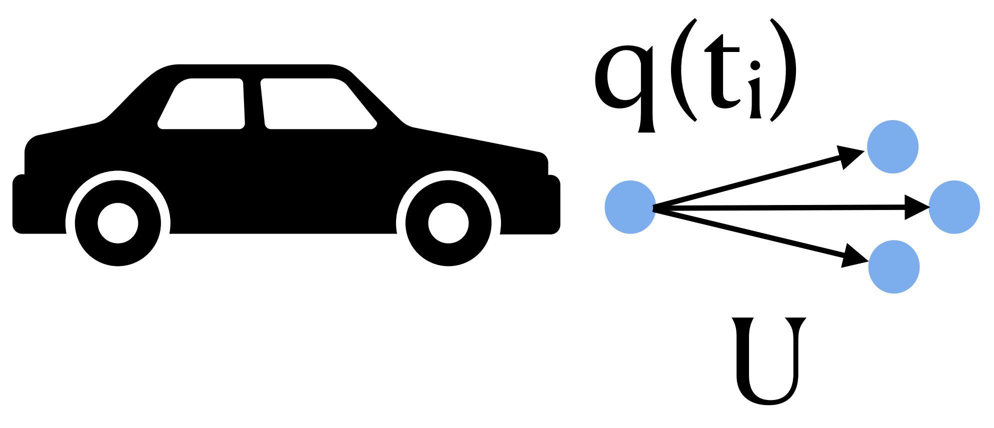
Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



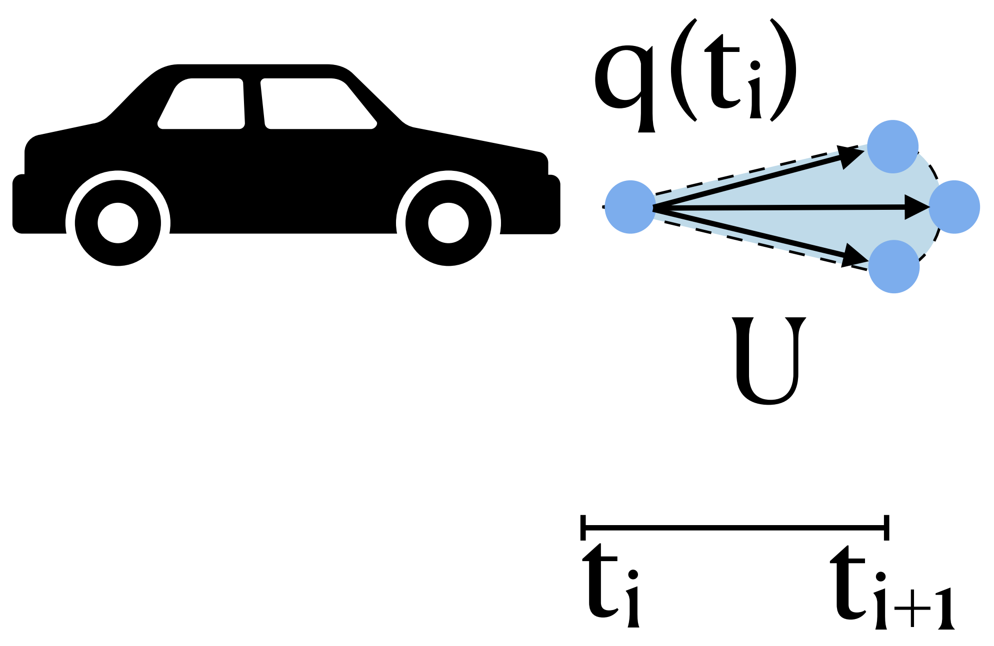
Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



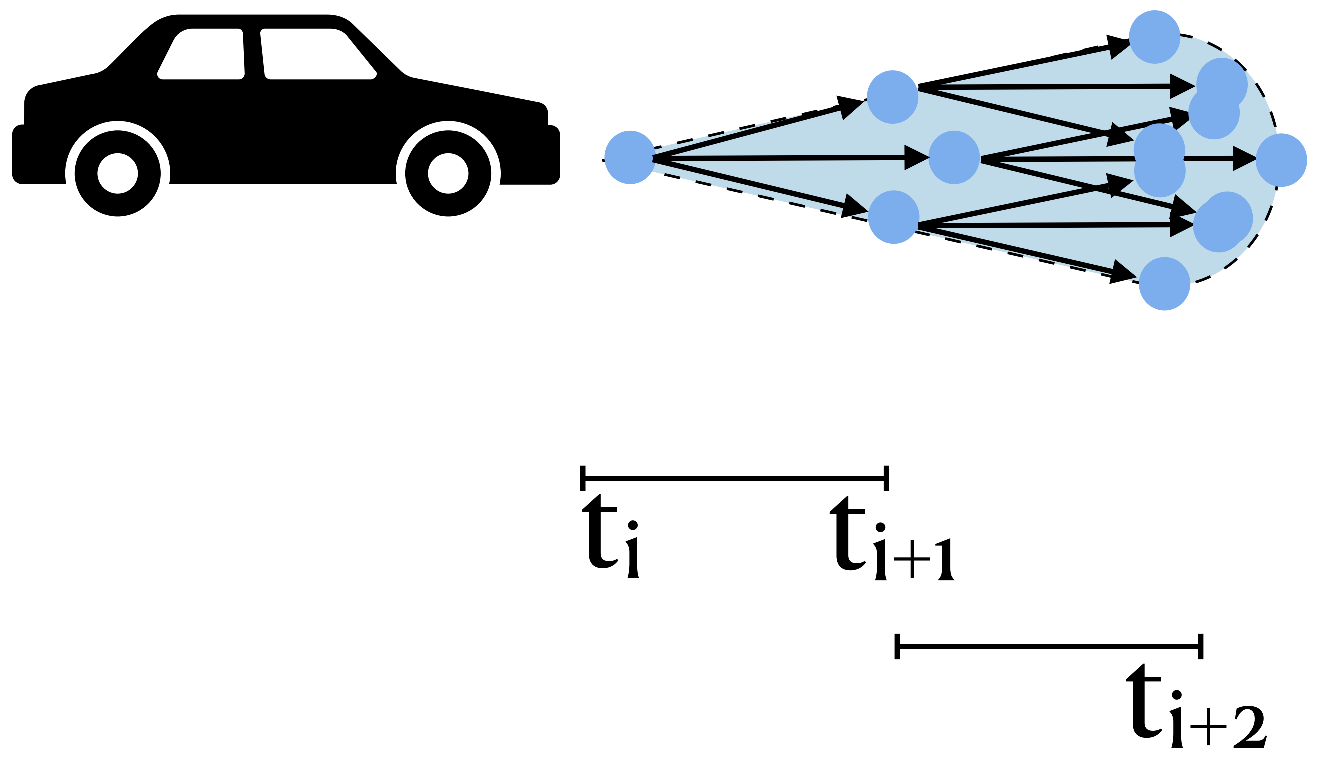
Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



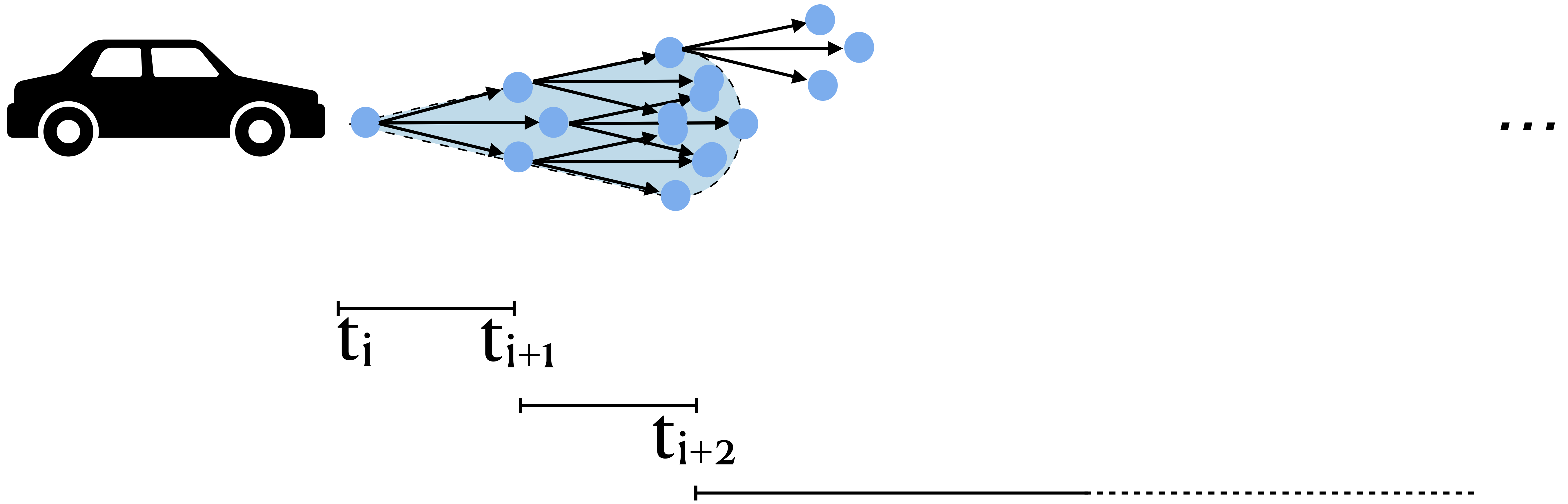
Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



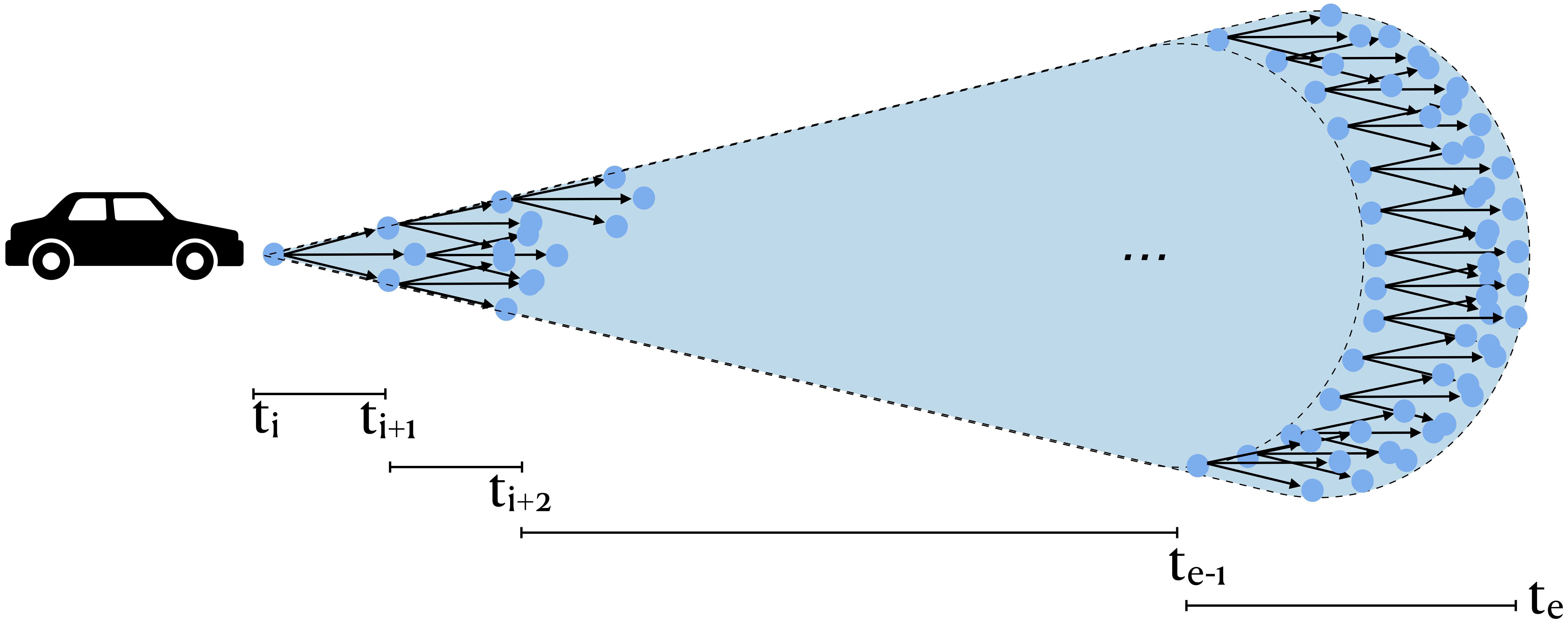
Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



Reachable Sets

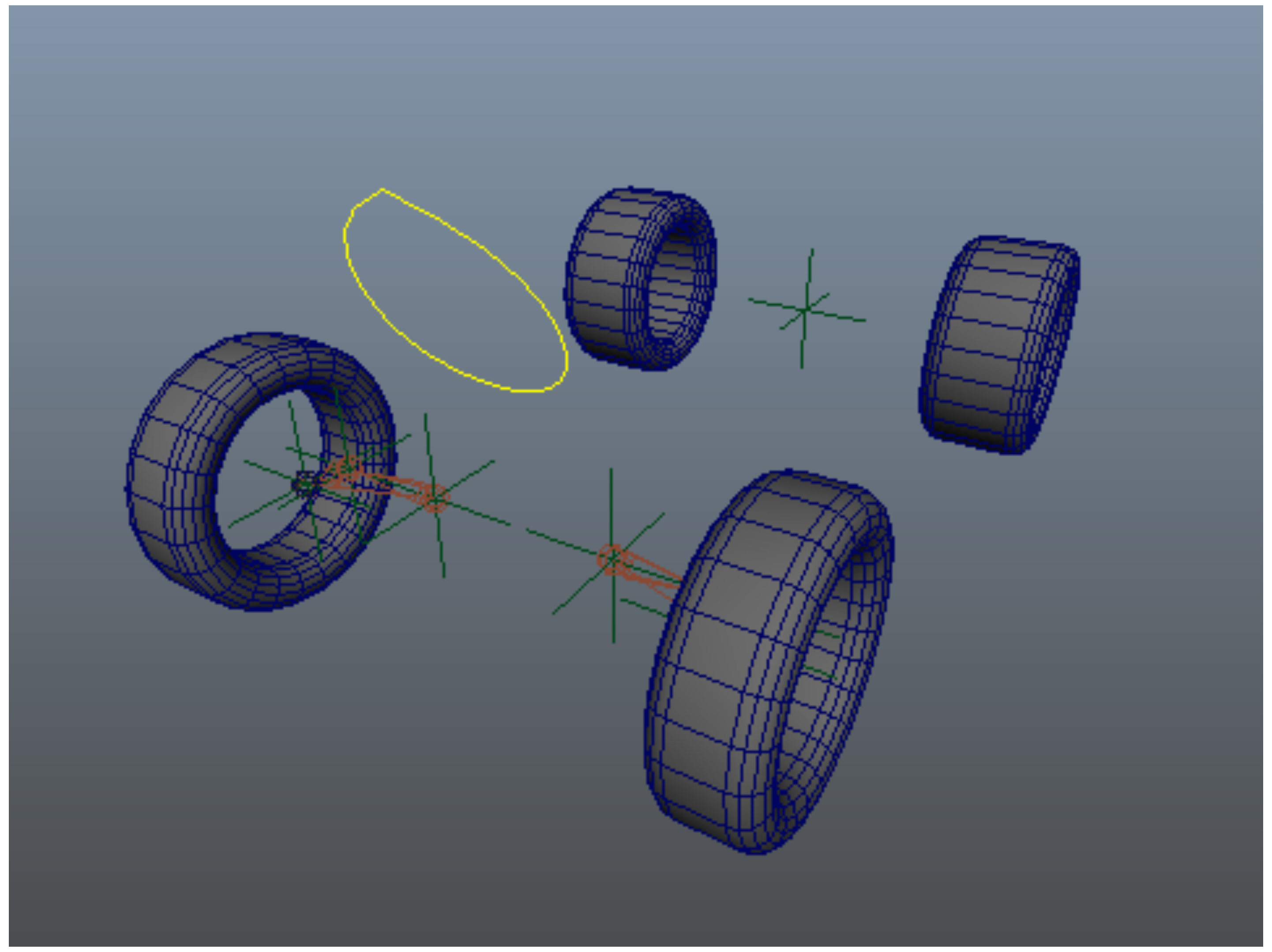
$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), \boldsymbol{U})dt$$



Traditional Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), U)dt$$

Set of Physical Inputs



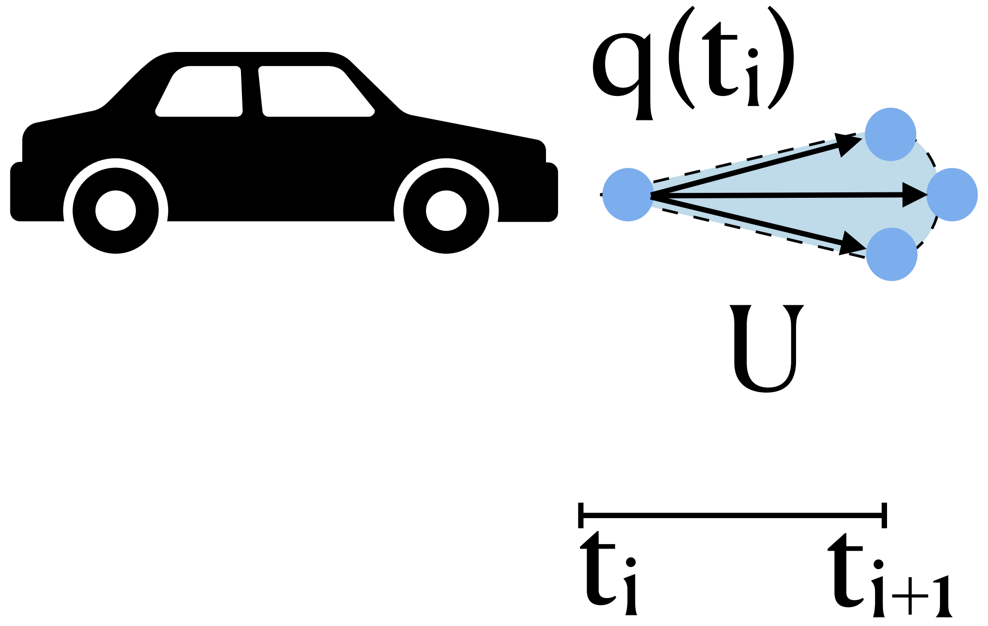


Figure reference) http://techartandstuff.blogspot.com/2013/07/how-to-create-accurate-car-steering-rig.html

Problem

If you bought a premium-class automobile recently, “it probably contains close to 100 million lines of software code.”

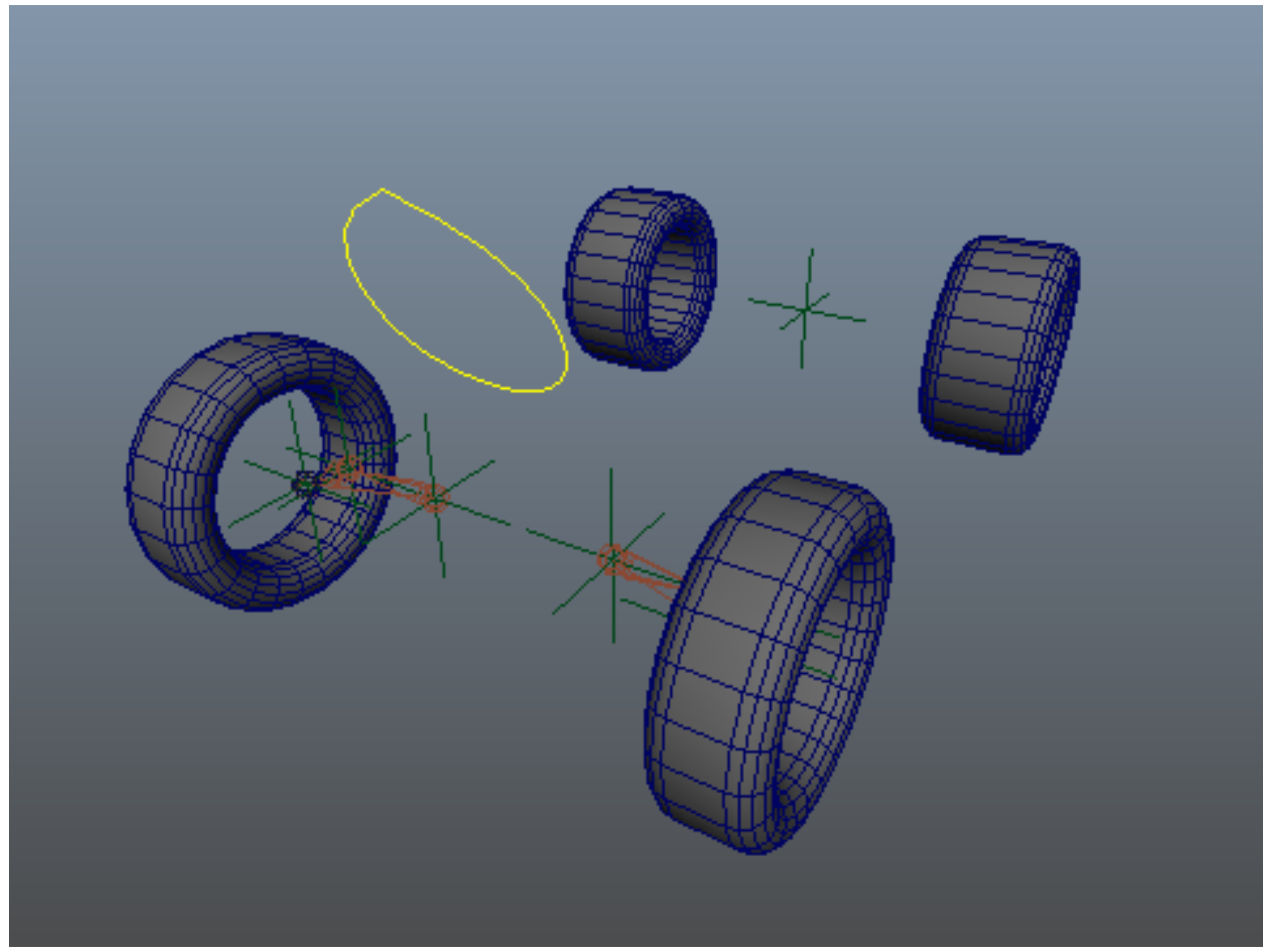
- Manfred Broy, professor of informatics at Technical University

Problem

Traditional Reachable Sets

$$R_i = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), U)dt$$

Set of Physical Inputs



Proposed Reachable Sets

$$R_i' = \boldsymbol{q}(t_i) \oplus \int_{t_0}^{t_e} f(\boldsymbol{q}(t_i), C(U))dt$$

Software Constraints

```
…
steer_angle = min(user.cmd , 30) #deg
publish(steer_angle)
```

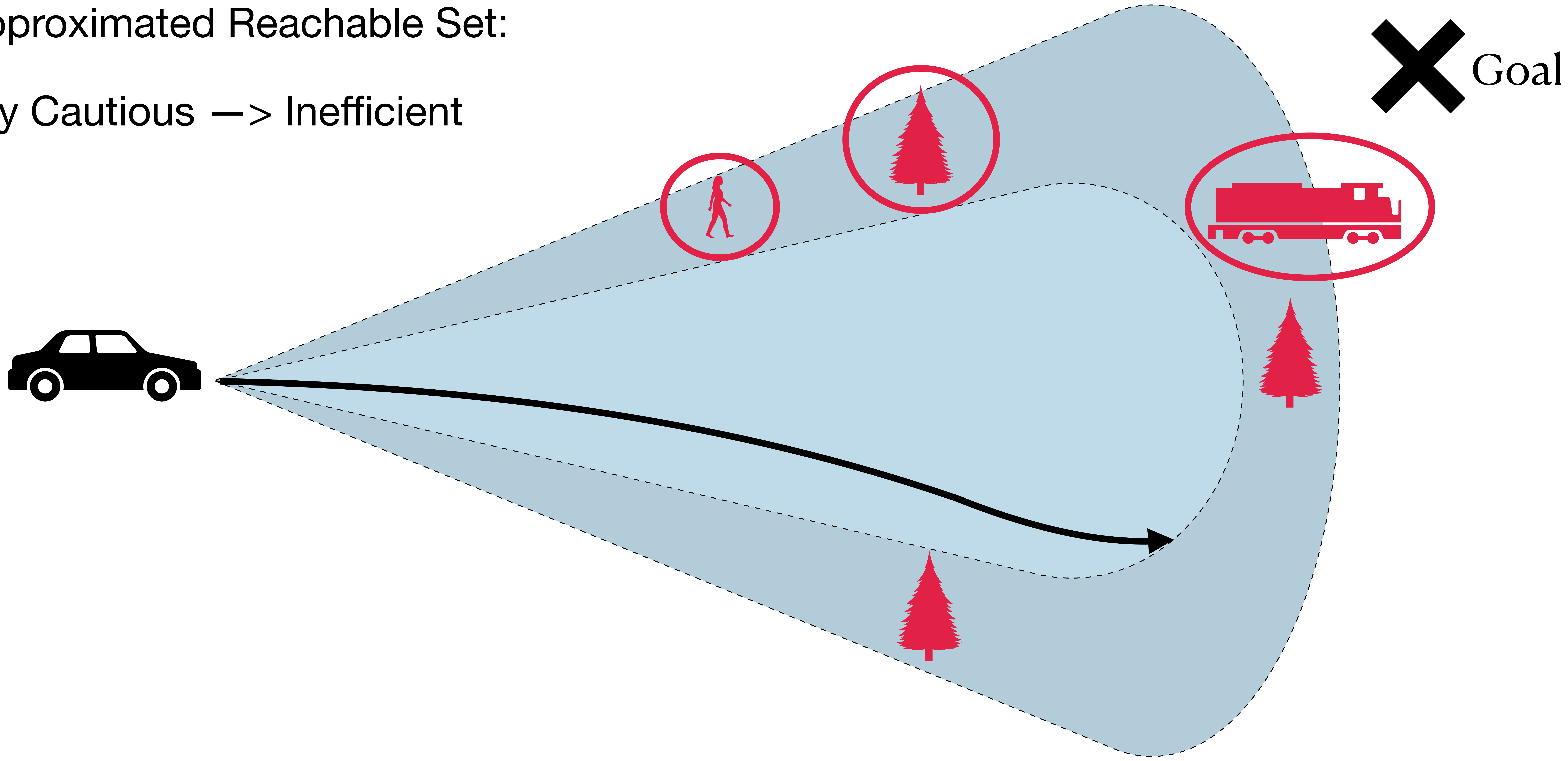
```
…
cmd_vel = min(cruise.vel , 120) #km/h
publish(cmd_vel)
```


Figure) http://techartandstuff.blogspot.com/2013/07/how-to-create-accurate-car-steering-rig.html

Reachability without Software Constraints

Over Approximated Reachable Set:

1) Overly Cautious —> Inefficient

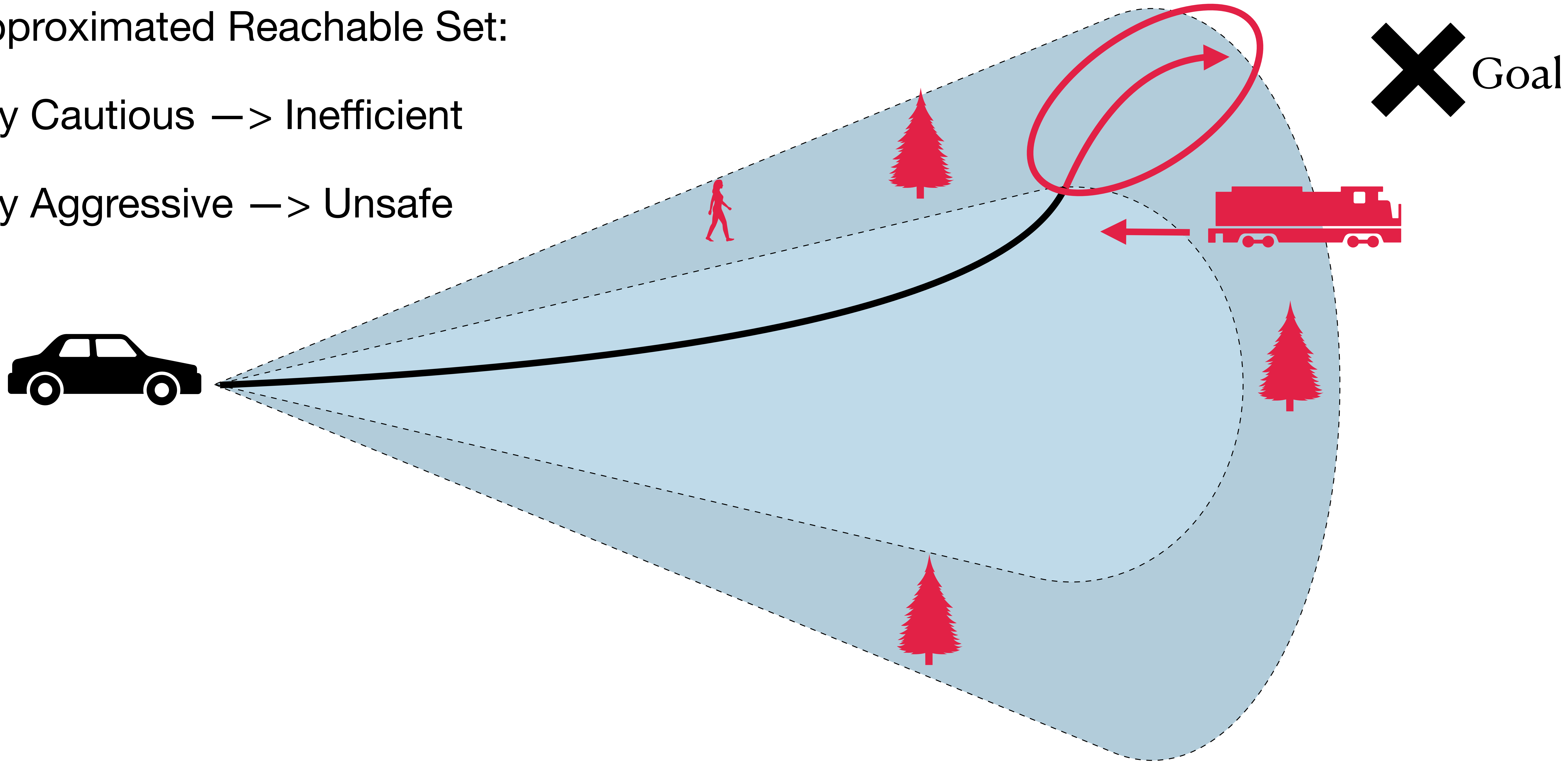


Reachability without Software Constraints

Over Approximated Reachable Set:

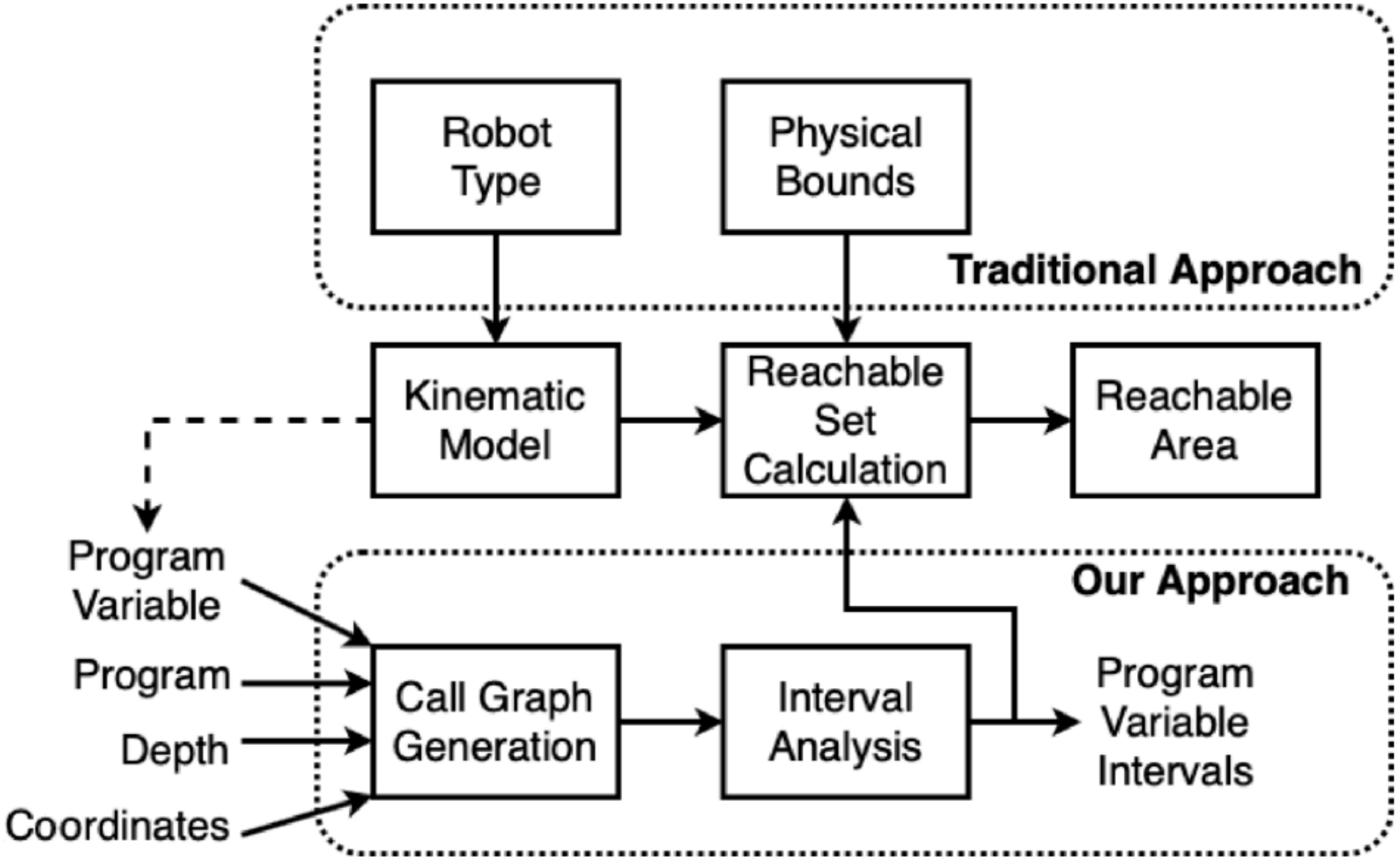
1) Overly Cautious —> Inefficient

2) Overly Aggressive —> Unsafe

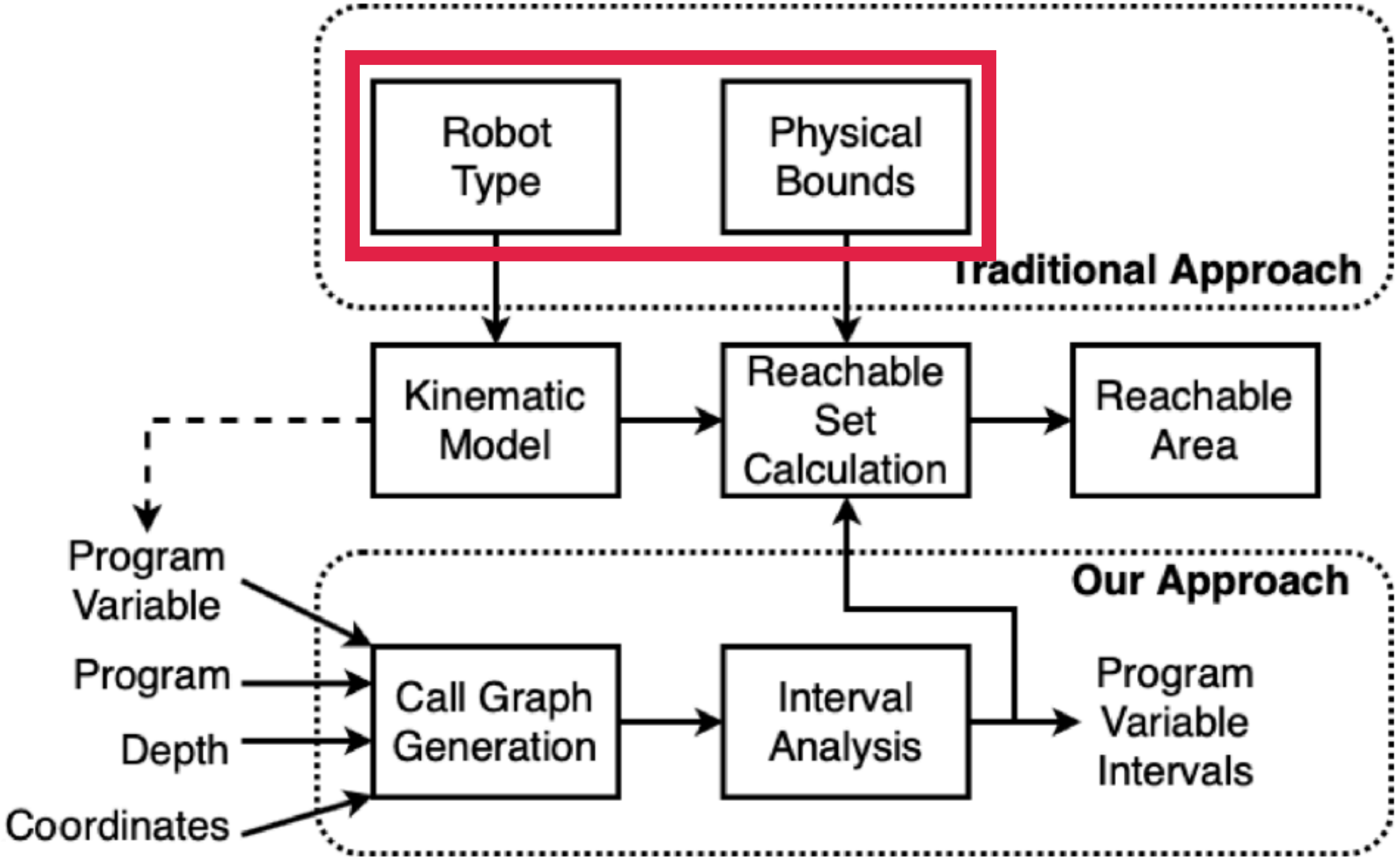


Solution

UNIVERSITY of VIRGINIA | SCHOOL of ENGINEERING & APPLIED SCIENCE



Solution



Physical Bounds

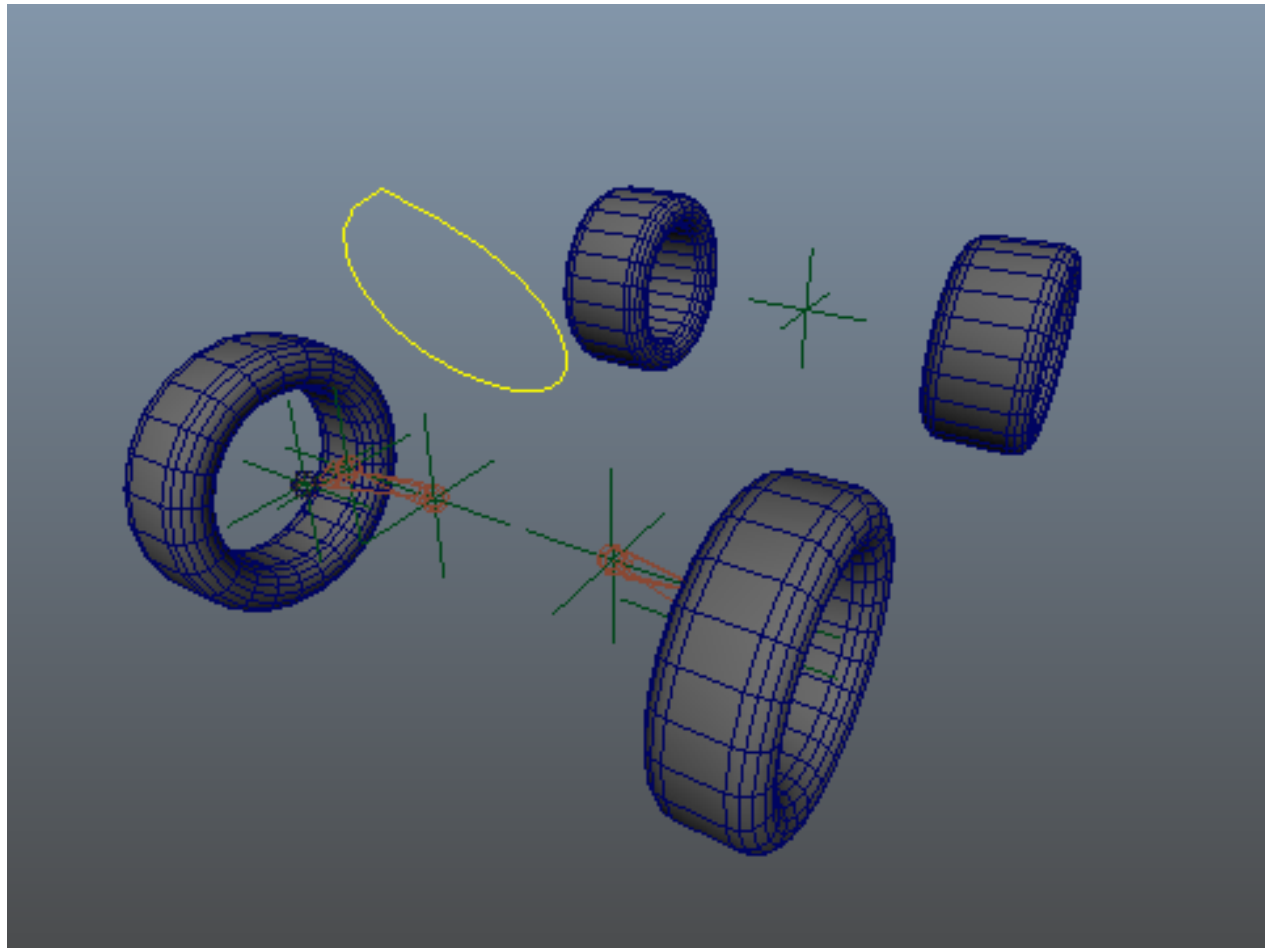
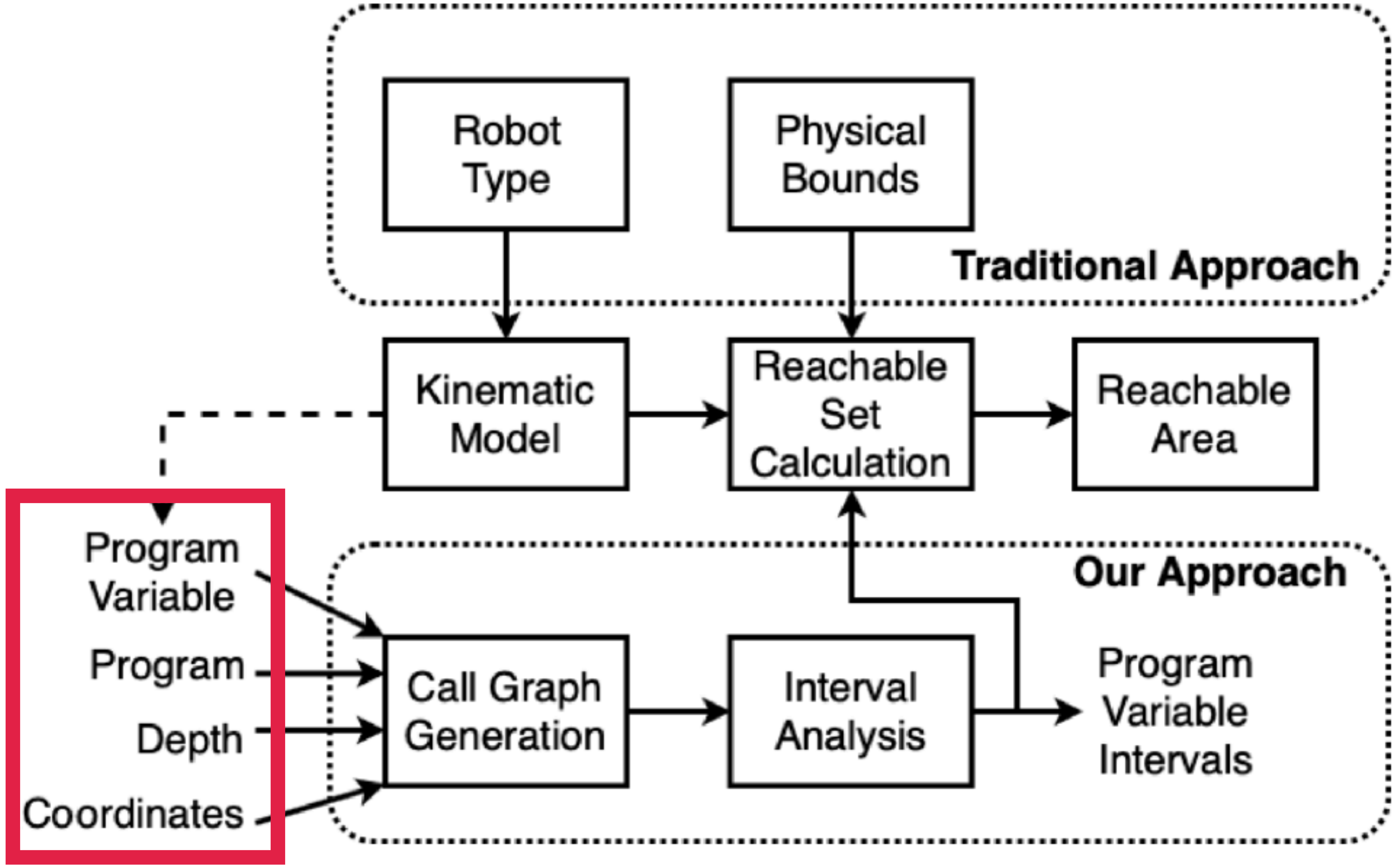


Figure) http://techartandstuff.blogspot.com/2013/07/how-to-create-accurate-car-steering-rig.html

Solution



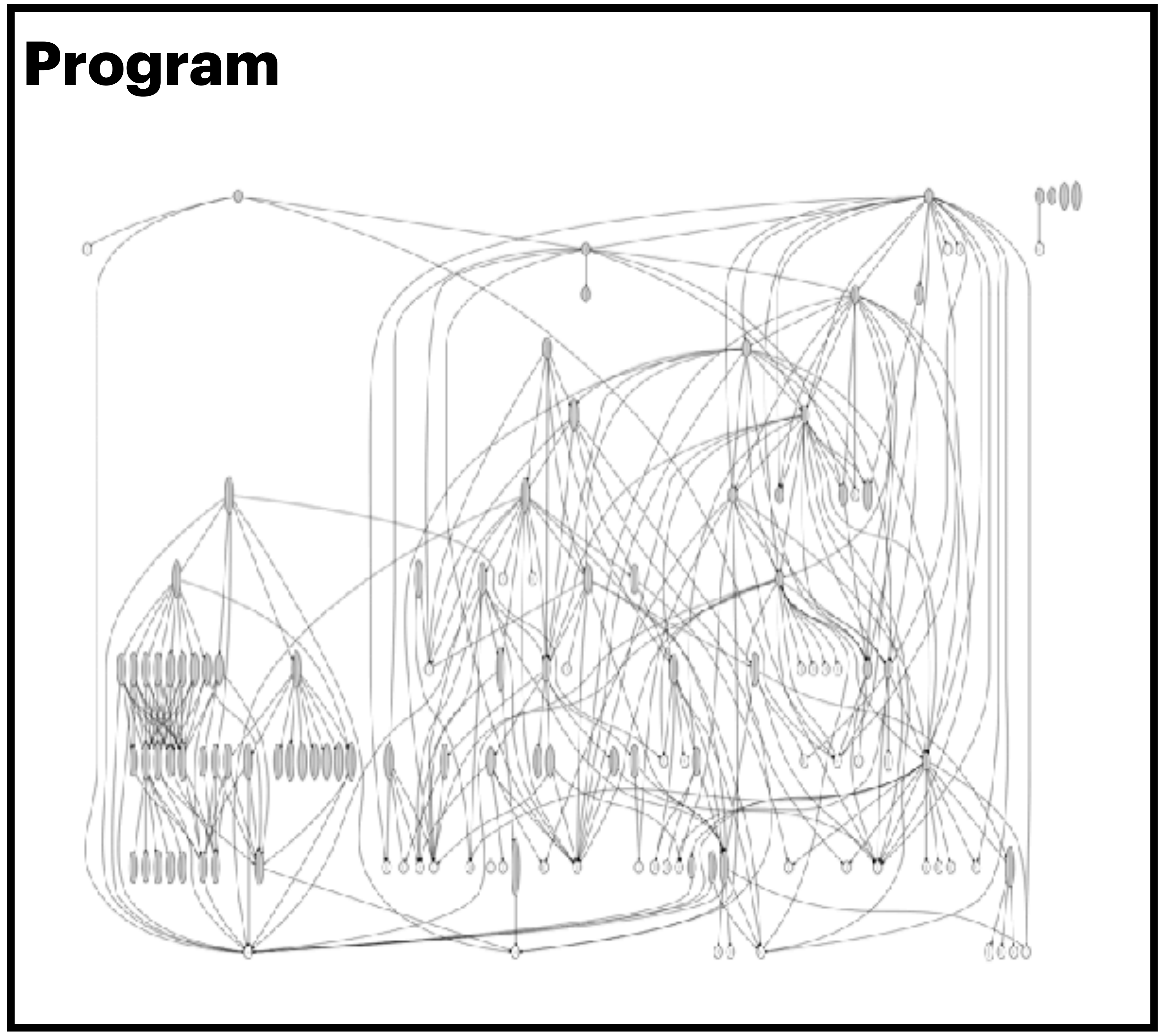
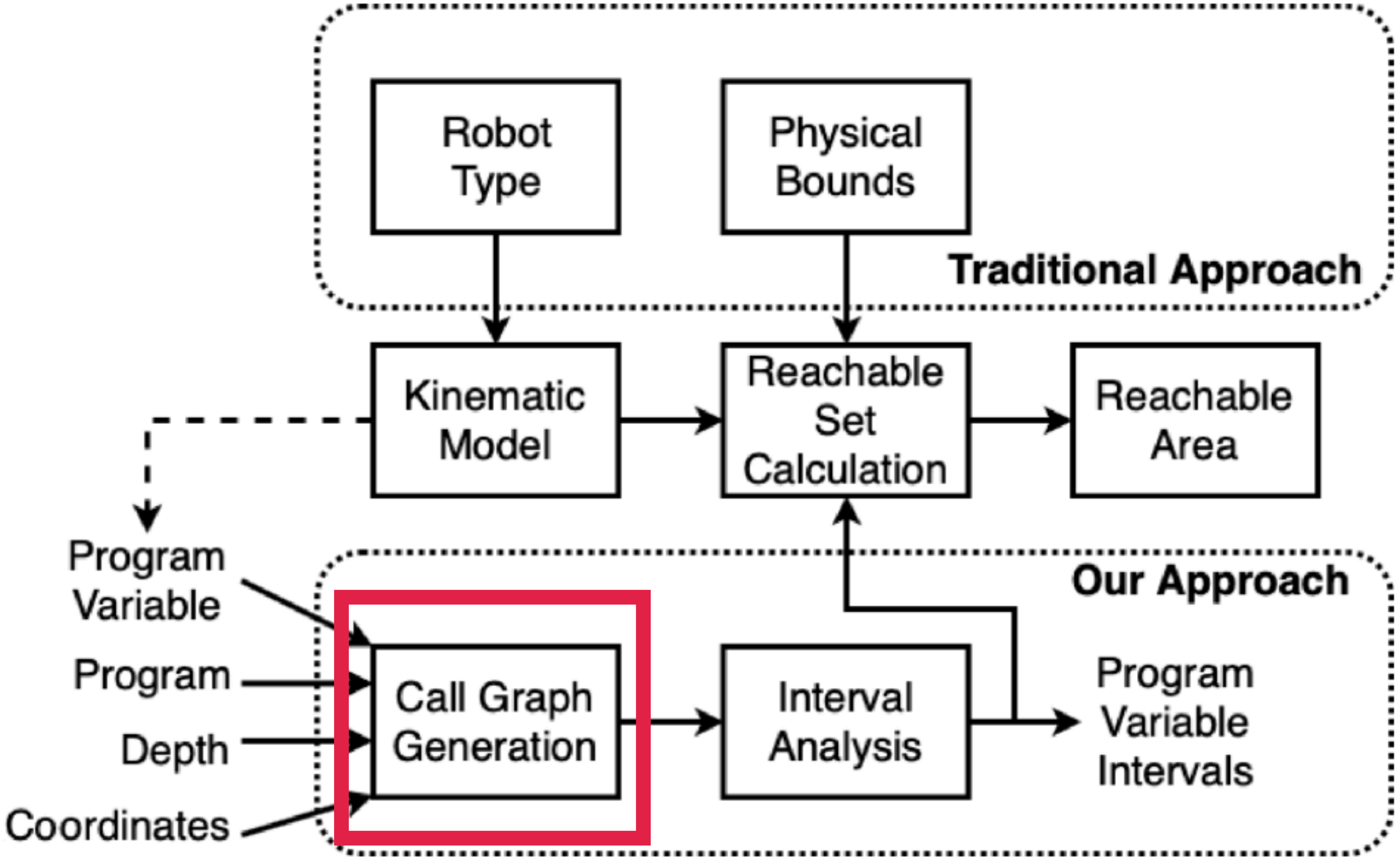
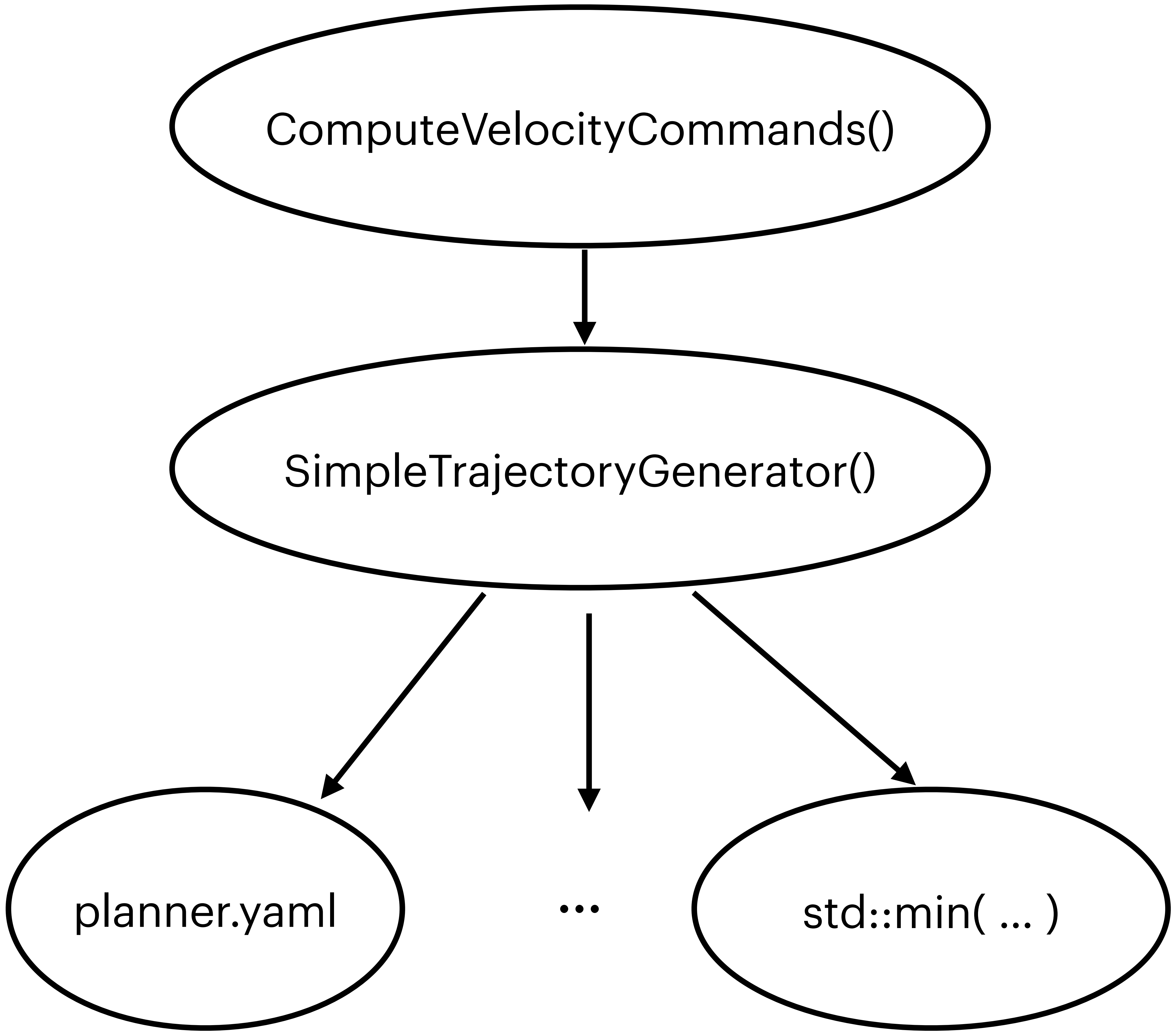


Figure) http://www.grokcode.com/864/snakefooding-python-code-for-complexity-visualization/

Solution

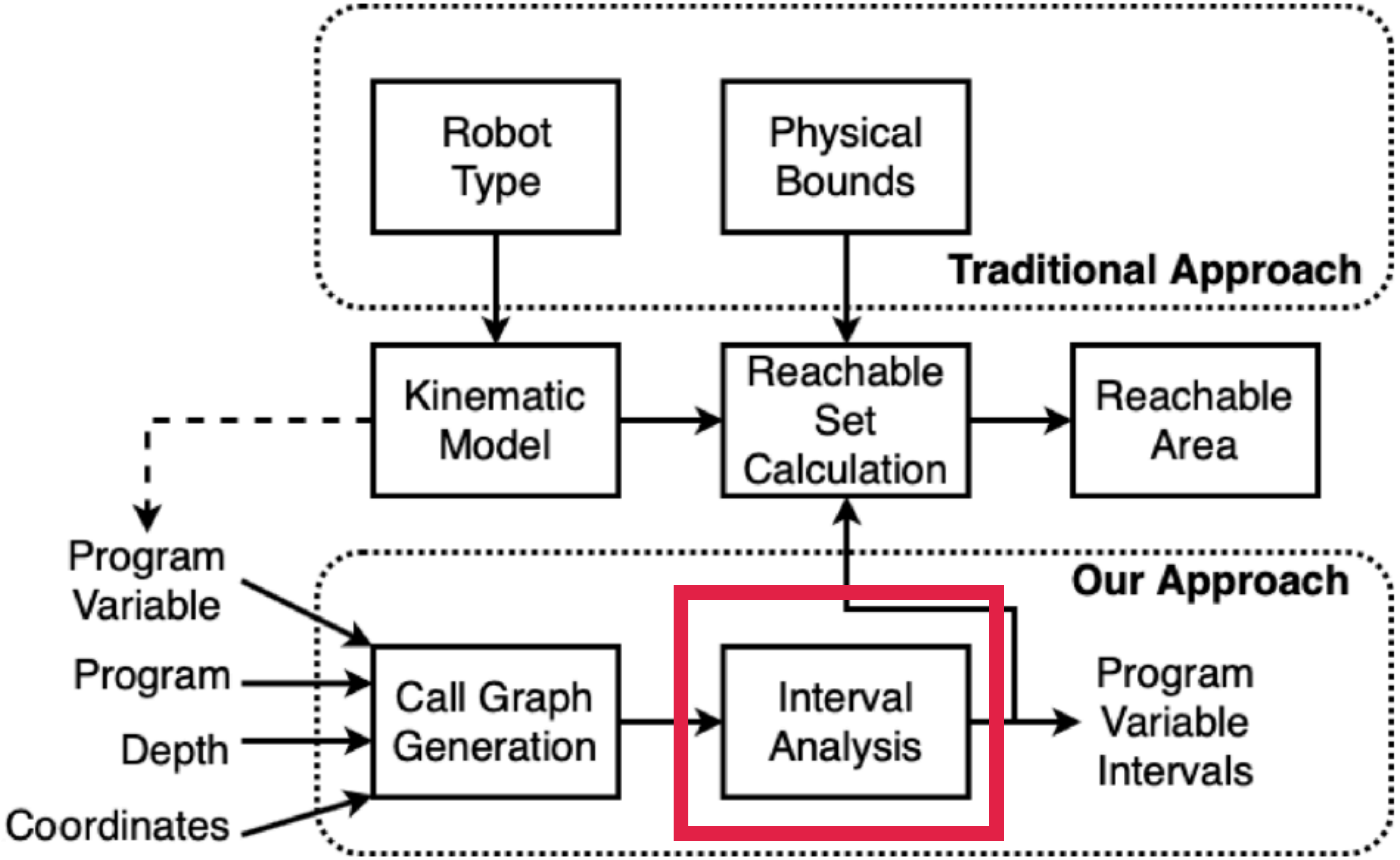


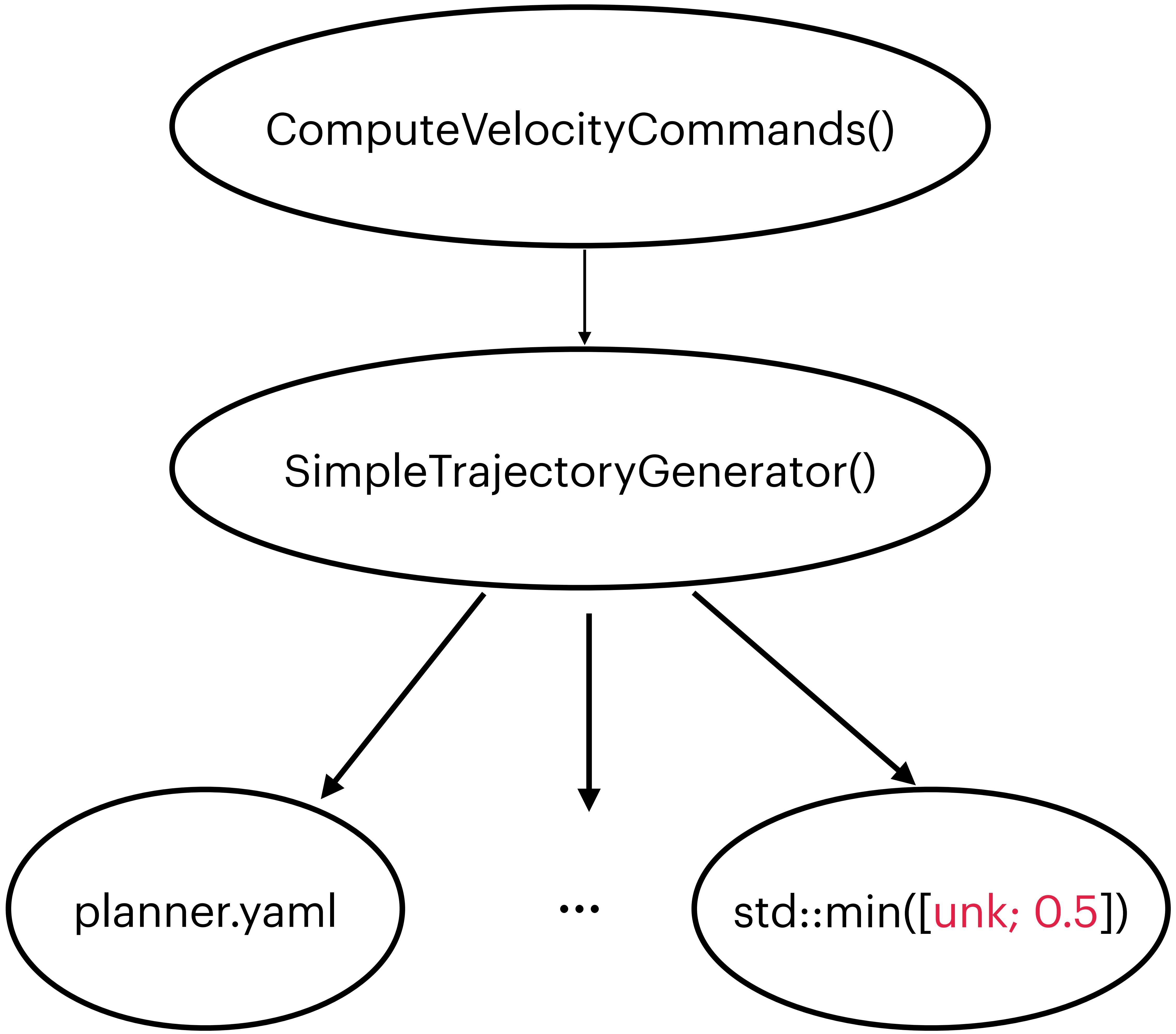
Robot Type
Physical Bounds
Traditional Approach
Kinematic Model
Reachable Set Calculation
Reachable Area
Program Variable
Program
Depth
Coordinates
Our Approach
Call Graph Generation
Interval Analysis
Program Variable Intervals



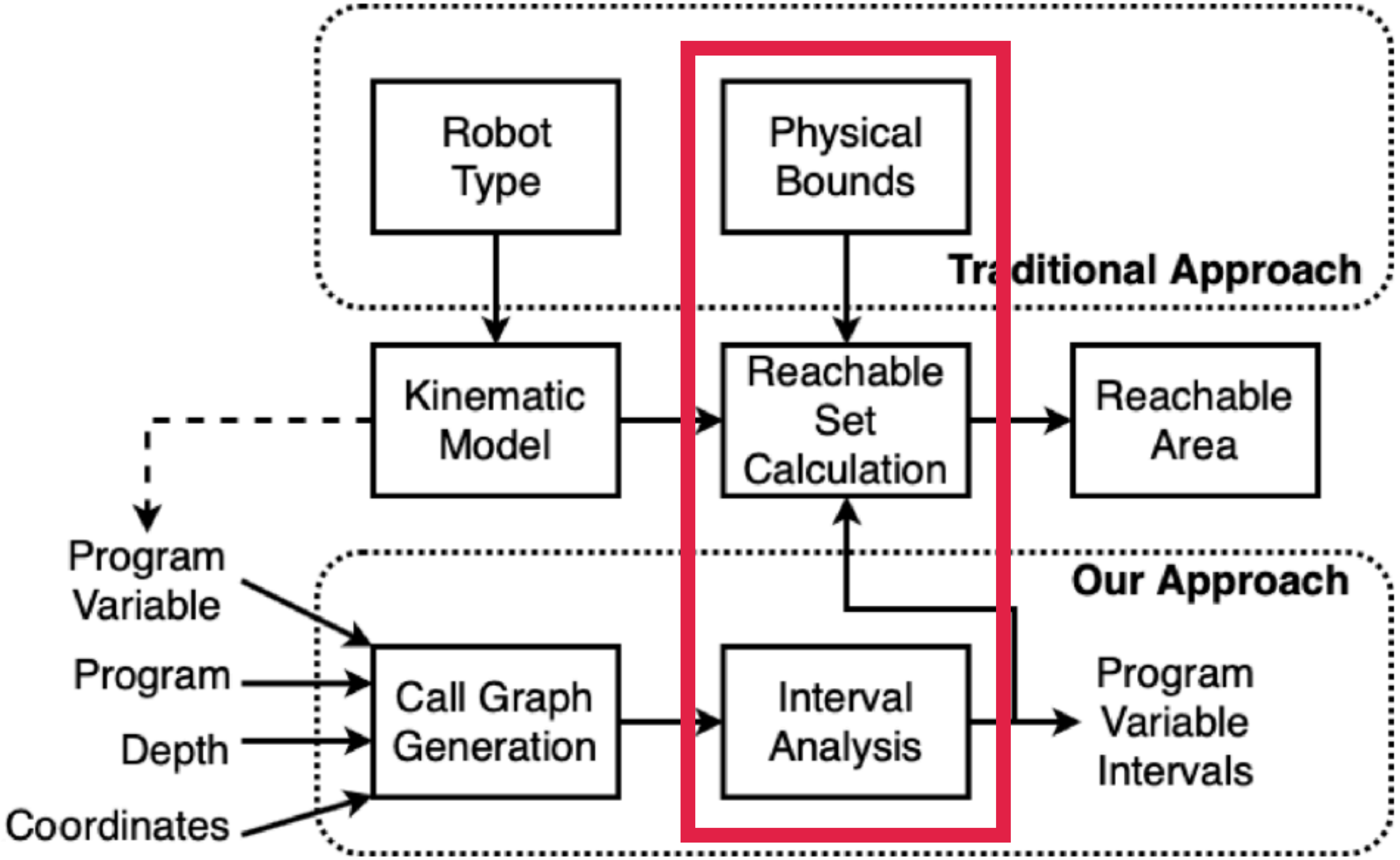
ComputeVelocityCommands()
SimpleTrajectoryGenerator()
planner.yaml
...
std::min(...)

Solution


Robot Type
Physical Bounds
Traditional Approach
Kinematic Model
Reachable Set Calculation
Reachable Area
Program Variable
Program
Depth
Coordinates
Our Approach
Call Graph Generation
Interval Analysis
Program Variable Intervals


ComputeVelocityCommands()
SimpleTrajectoryGenerator()
planner.yaml
...
std::min([unk; 0.5])

Solution



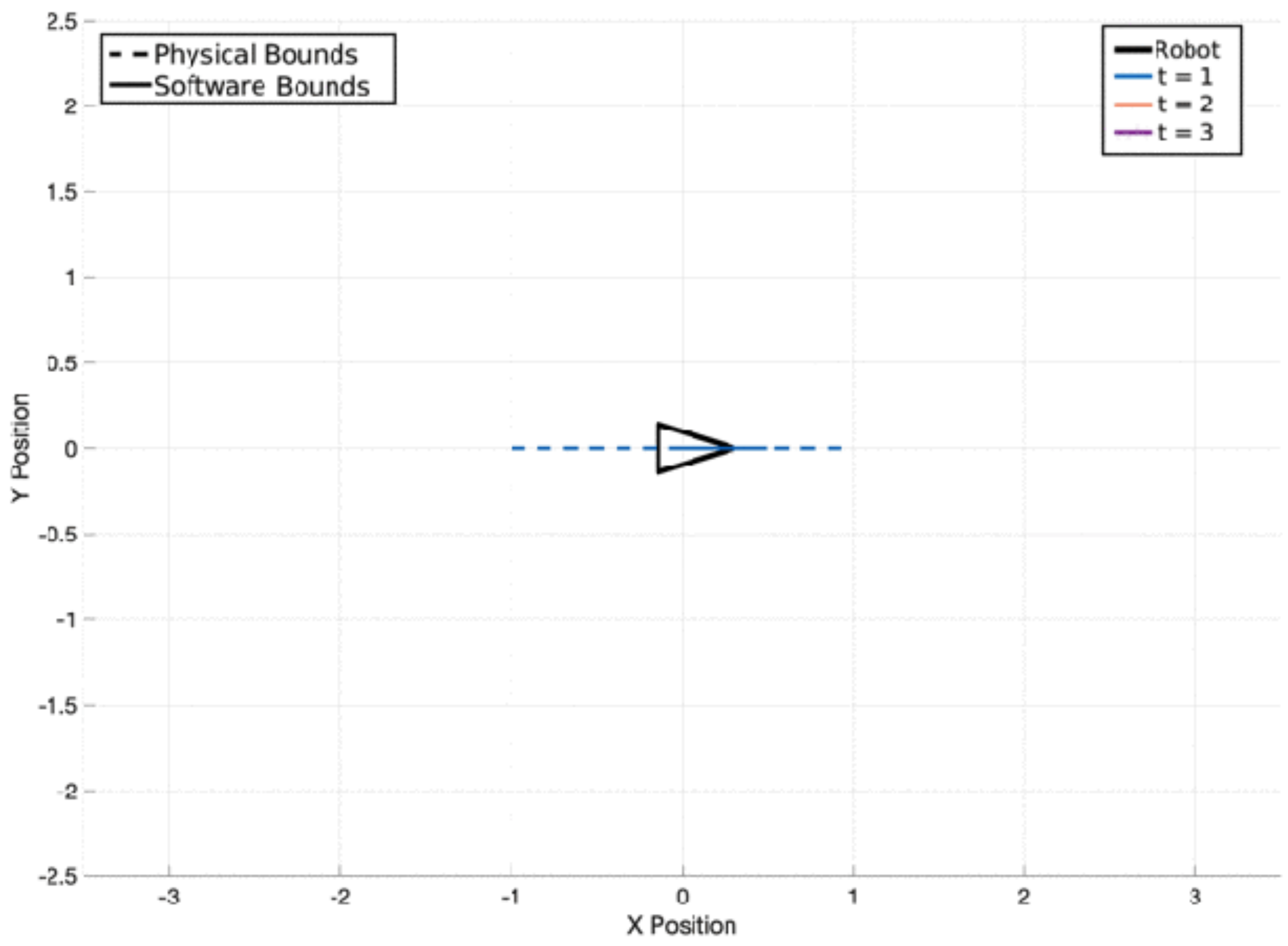
Robot Type	Physical Bounds	Software Bounds
	Max Velocity: 1 m/s Min Velocity: $\boxed{-1\ m/s}$ Turn Rate: 2 rad/s	Max Velocity : 0.5 m/s Min Velocity: $\boxed{0\ m/s}$ Turn Rate: 0.63 rad/s
	Thrust: 45 N Max Pitch: $\boxed{45\ degrees}$ Max Roll: $\boxed{45\ degrees}$	Thrust: ? N Max Pitch: $\boxed{19\ degrees}$ Max Roll: $\boxed{19\ degrees}$

Preliminary Results

Robot type	Physically Bound Reachability	Software Bound Reachability	Reduction
	$20.24m^2$	Max Velocity: $17.10m^2$ Min Velocity: $15.10m^2$ Velocity: $3.77m^2$ Max Turn Rate: $17.06m^2$ All Constraints: $1.85m^2$	16% 25% 81% 16% 91%
	$716930m^3$	Max Pitch: $343428m^3$ Max Roll: $343428m^3$ All Constraints: $163563m^3$	52% 52% 77%

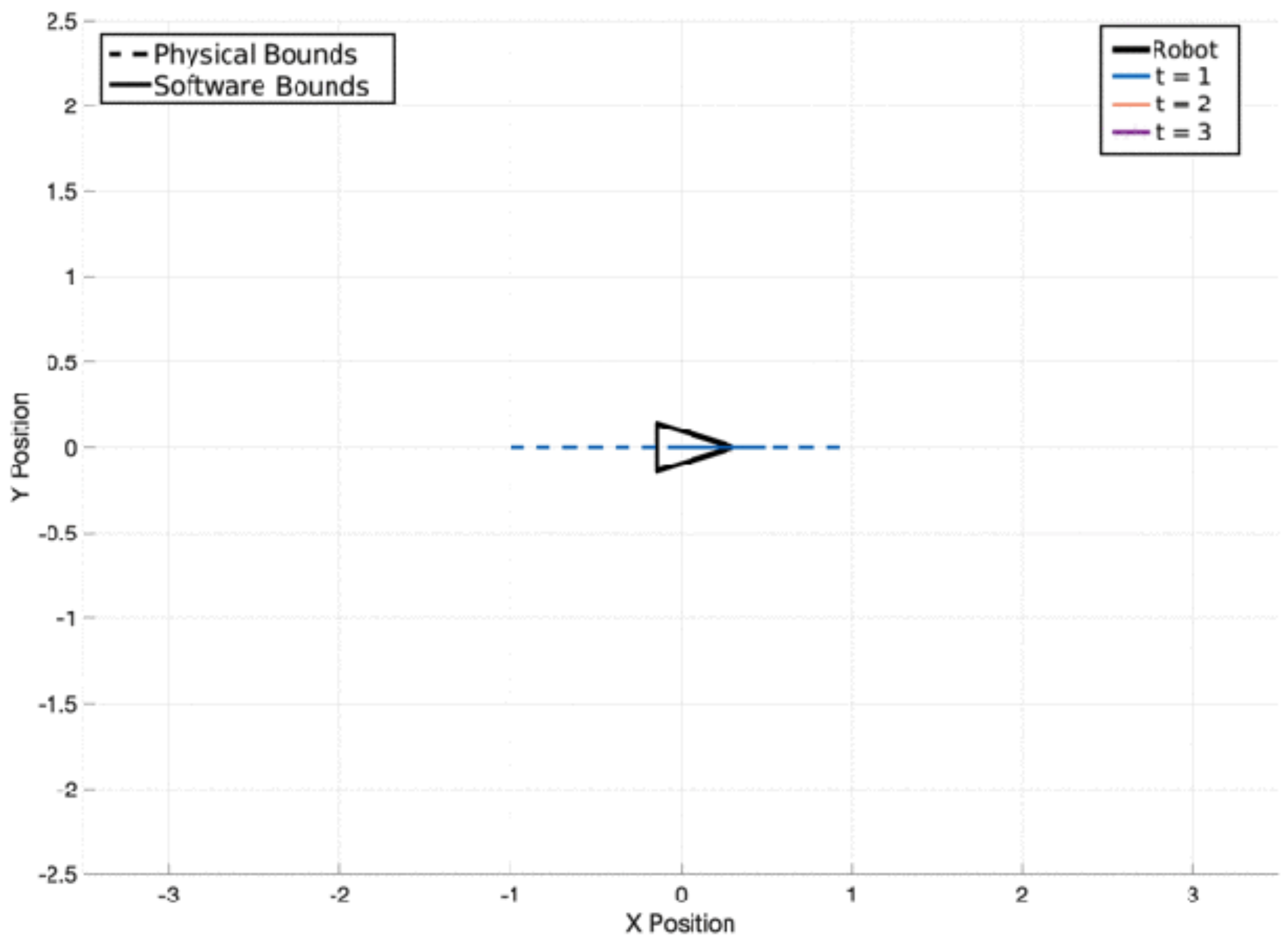
Preliminary Results

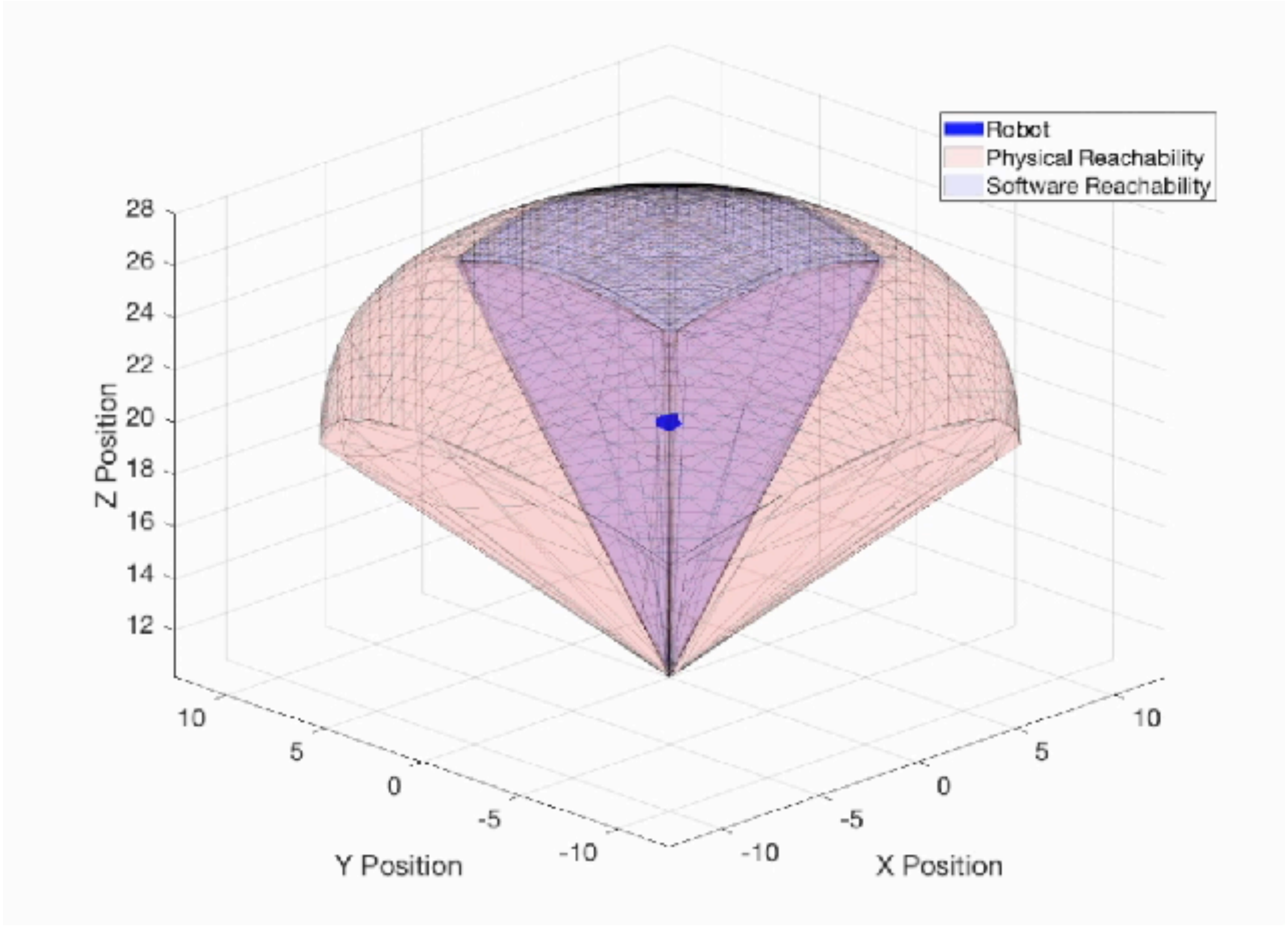
Robot type	Physically Bound Reachability	Software Bound Reachability	Reduction
	$20.24m^2$	Max Velocity: $17.10m^2$ Min Velocity: $15.10m^2$ Velocity: $3.77m^2$ Max Turn Rate: $17.06m^2$ All Constraints: $1.85m^2$	16% 25% 81% 16% 91%
	$716930m^3$	Max Pitch: $343428m^3$ Max Roll: $343428m^3$ All Constraints: $163563m^3$	52% 52% 77%



Preliminary Results

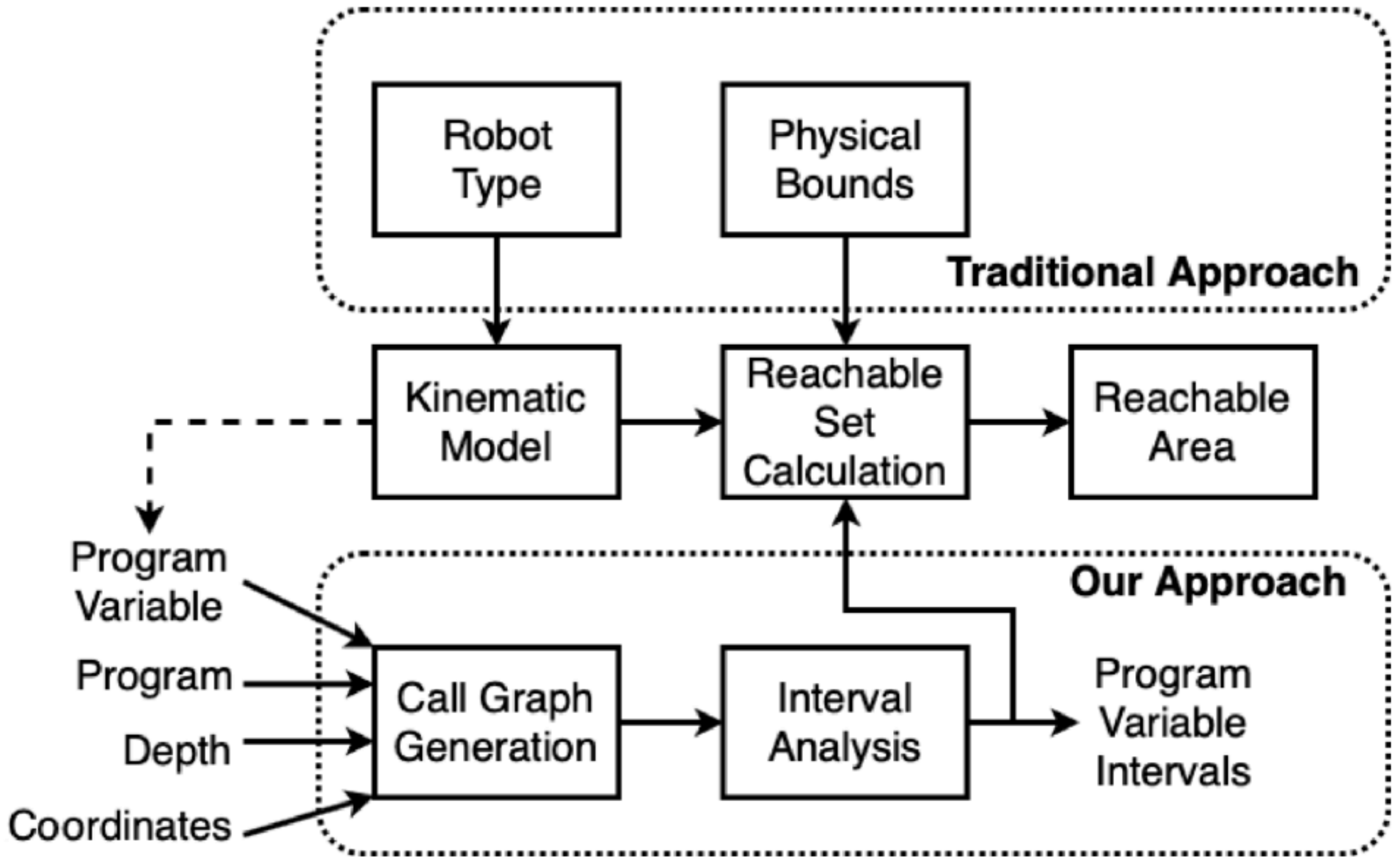
Robot type	Physically Bound Reachability	Software Bound Reachability	Reduction
	$20.24m^2$	Max Velocity: $17.10m^2$ Min Velocity: $15.10m^2$ Velocity: $3.77m^2$ Max Turn Rate: $17.06m^2$ All Constraints: $1.85m^2$	16% 25% 81% 16% 91%
	$716930m^3$	Max Pitch: $343428m^3$ Max Roll: $343428m^3$ All Constraints: $163563m^3$	52% 52% 77%

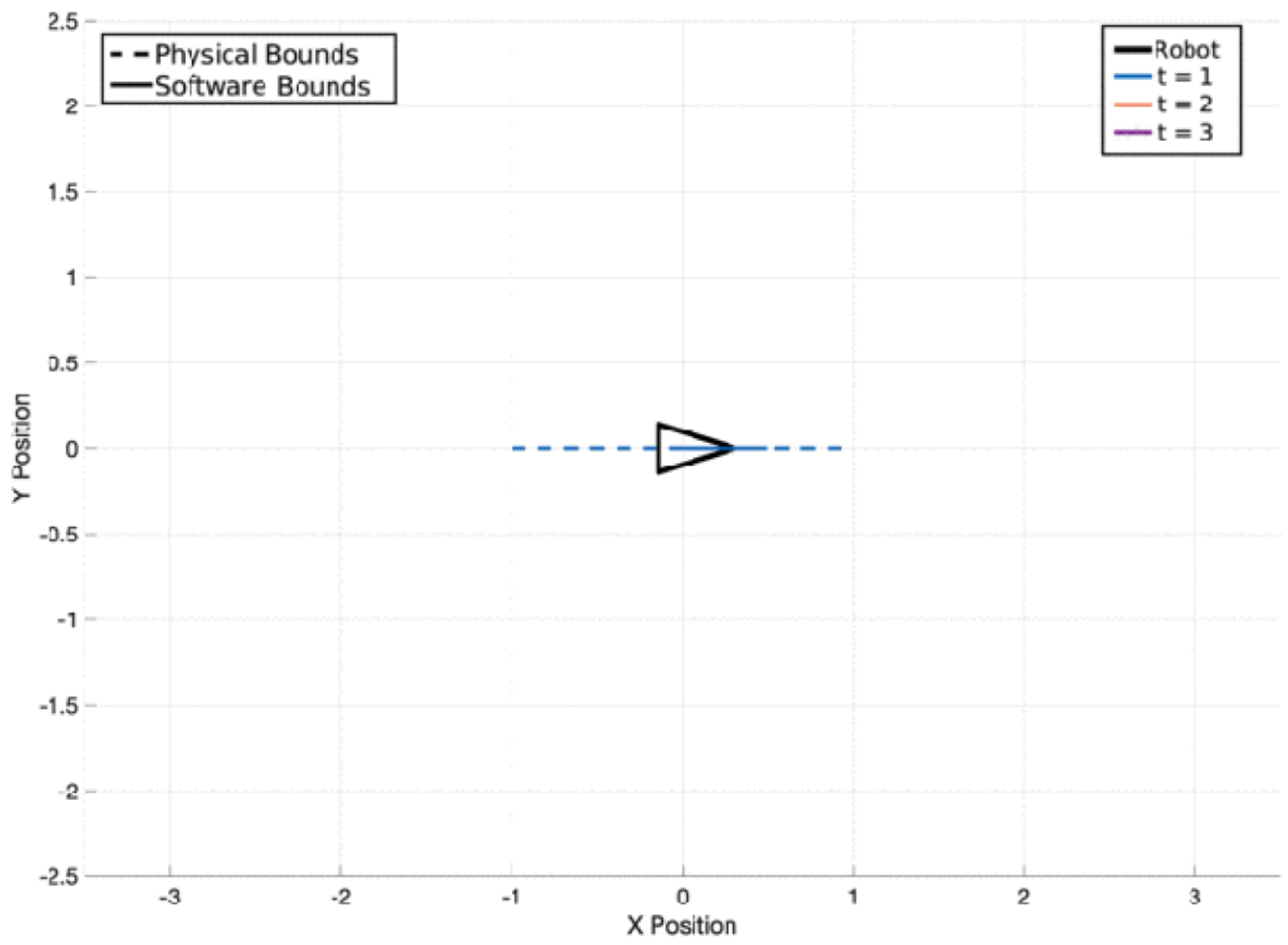


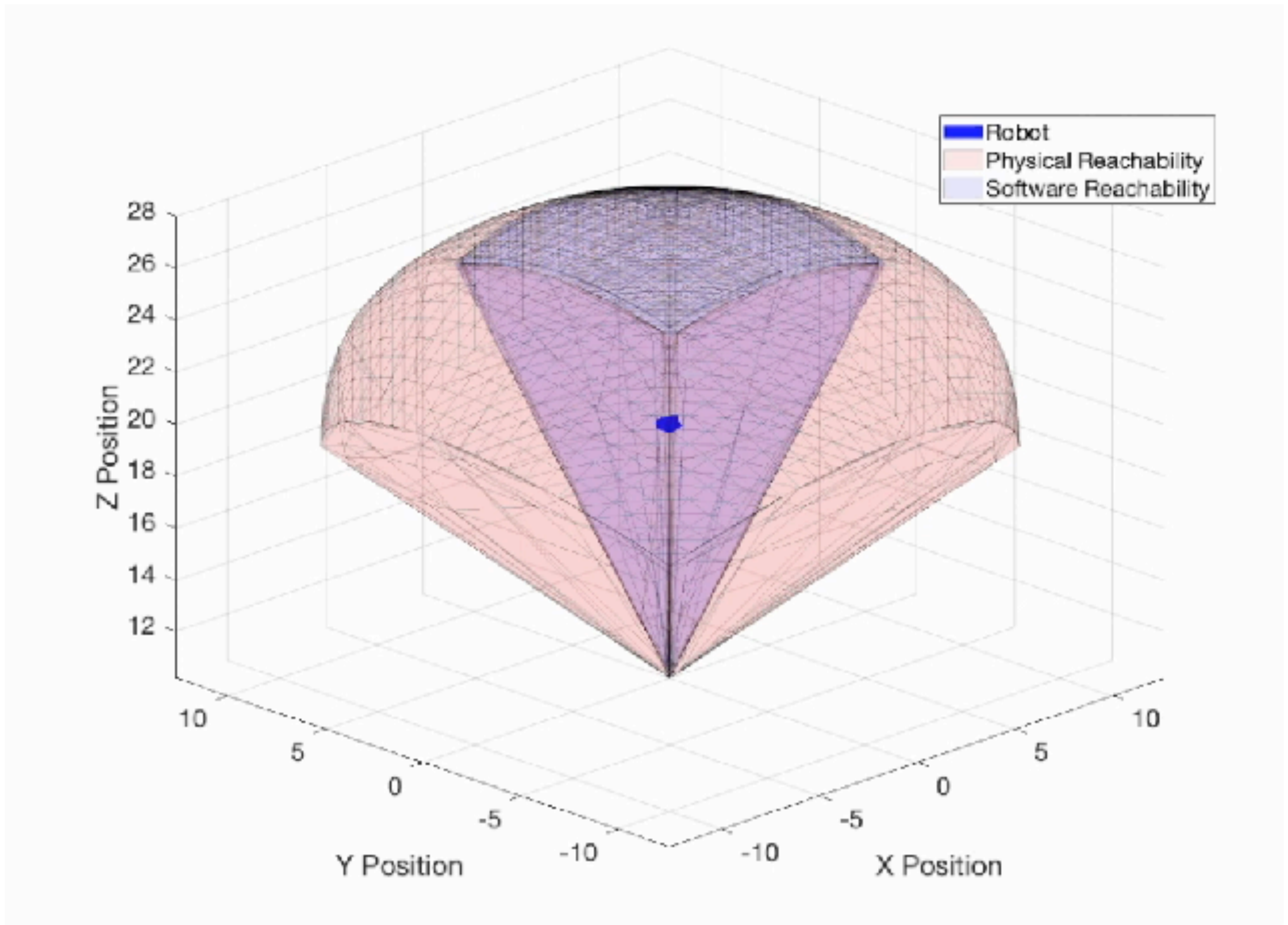


Conclusion

Takeaway: The **precision of reachable sets** used by autonomous vehicles **could be dramatically higher by considering the constraints imposed by software**. We now have an approach to uncovering and applying those constraints.







Contact: hildebrandt.carl@virginia.edu

This work was funded in part by the NSF