CESAR: Control Envelope Synthesis via Angelic Refinements

Aditi Kabra[1] Jonathan Laurent[1,2] Stefan Mitsch[1,3] André Platzer[1,2]

[1]Carnegie Mellon University
[2]Karlsruhe Institute of Technology
[3]DePaul University

TACAS 2024

Supported by:

FRA contract number 693JJ620C000025

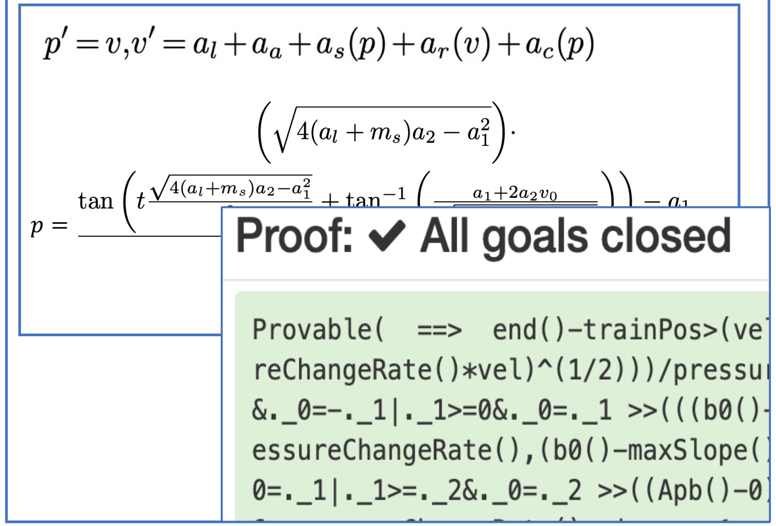
Swartz Center Innovation Commercialization Fellowship

an Alexander von Humboldt Professorship

Control Envelope Synthesis

Cyber-physical systems

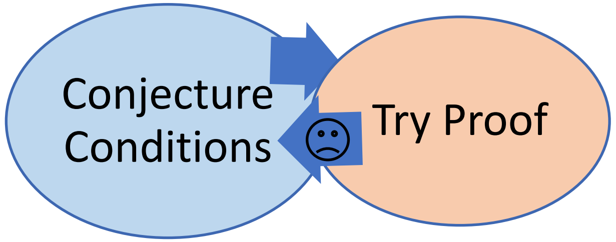
Verification



Requires

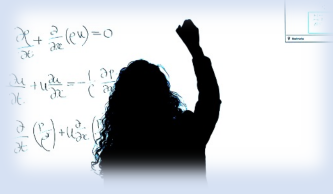
Designing Control Conditions

- Difficult: the main point control theory
- Needs creativity, careful reasoning about dynamics



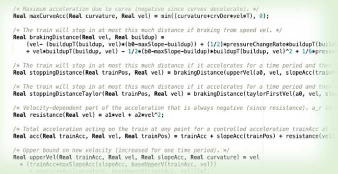
Solution: Synthesis justified by verification

Human provides:
Shape of model



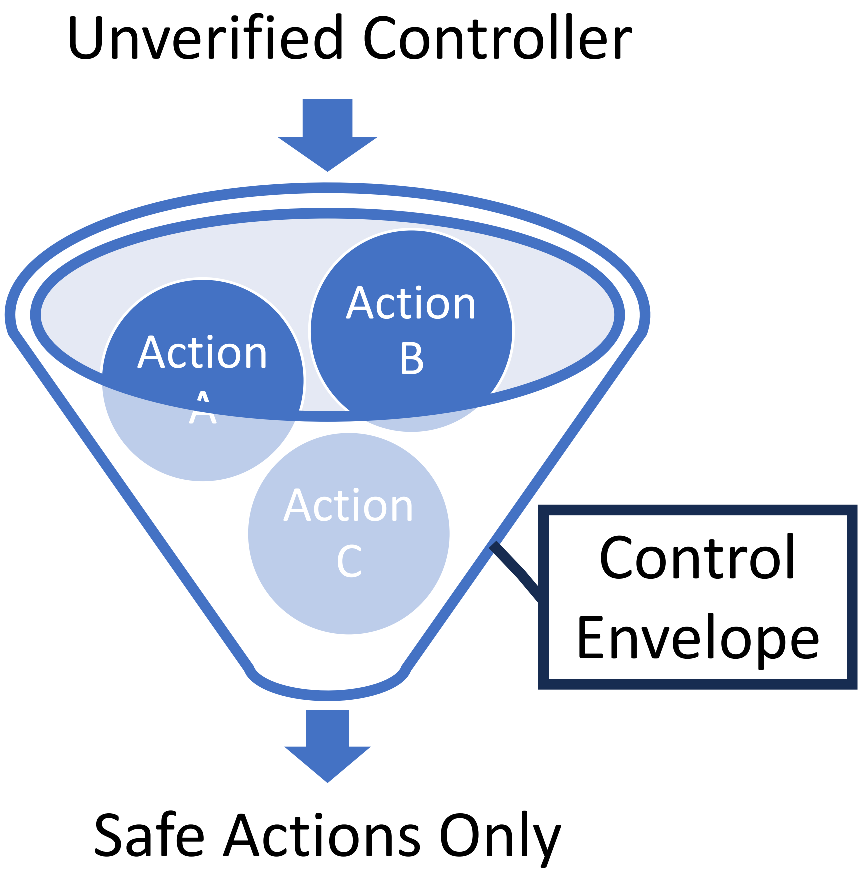
Synthesis procedure:
fills control conditions

Correct by Construction Control Envelope!



Control **Envelopes**

- Non-deterministic: allow ***all*** safe actions
- Define ***families*** of safe controllers
- Full system monitored for adherence at runtime
- Higher-order constraint compared to controllers: solutions permit as many safe control solutions as possible

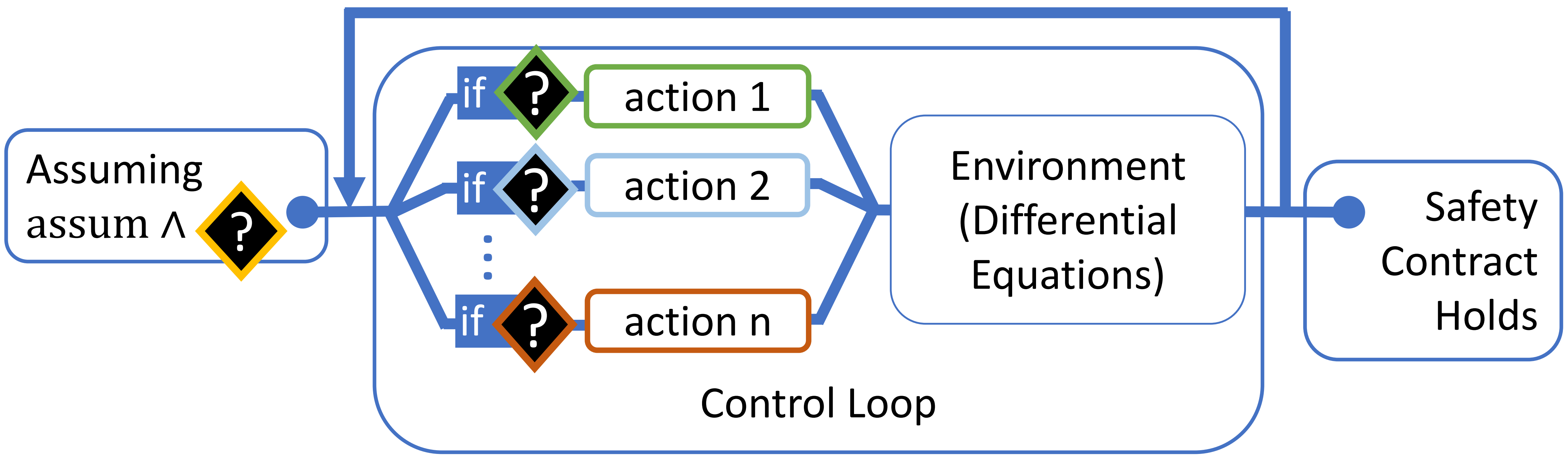


Overview

- Introduction
- **Problem Statement**
- Game Logic and Solution
- Refinement
- Evaluation

Problem

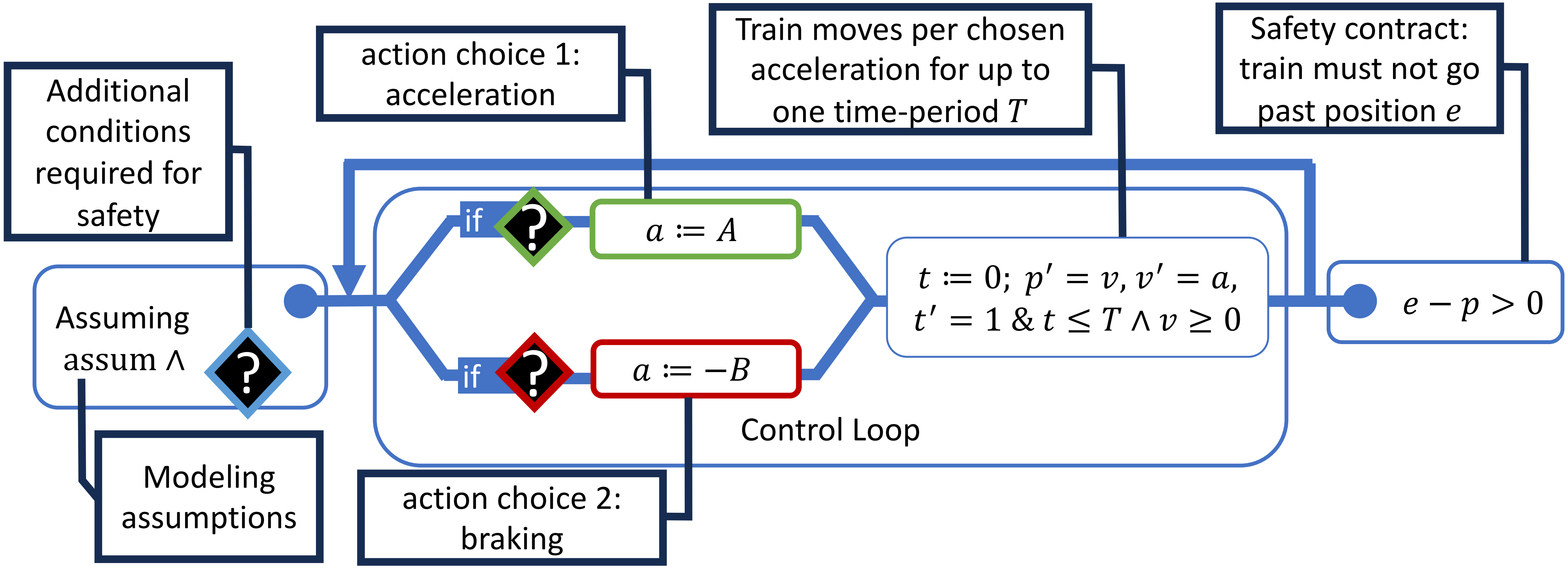
Synthesis procedure fills holes (?). Which action is safe when?



$$\text{prob} \equiv \overbrace{\text{assum} \land \text{\textvisiblespace}}^{\text{Assumptions}} \to [\overbrace{((\cup_i (?\text{\textvisiblespace}_i ; \text{act}_i)) ; \text{plant})^*}^{\text{Control Loop}}]\,\overbrace{\text{safe}.}^{\text{Contract}}$$

Example: Train

Synthesis procedure fills holes (?). Which action is safe when?

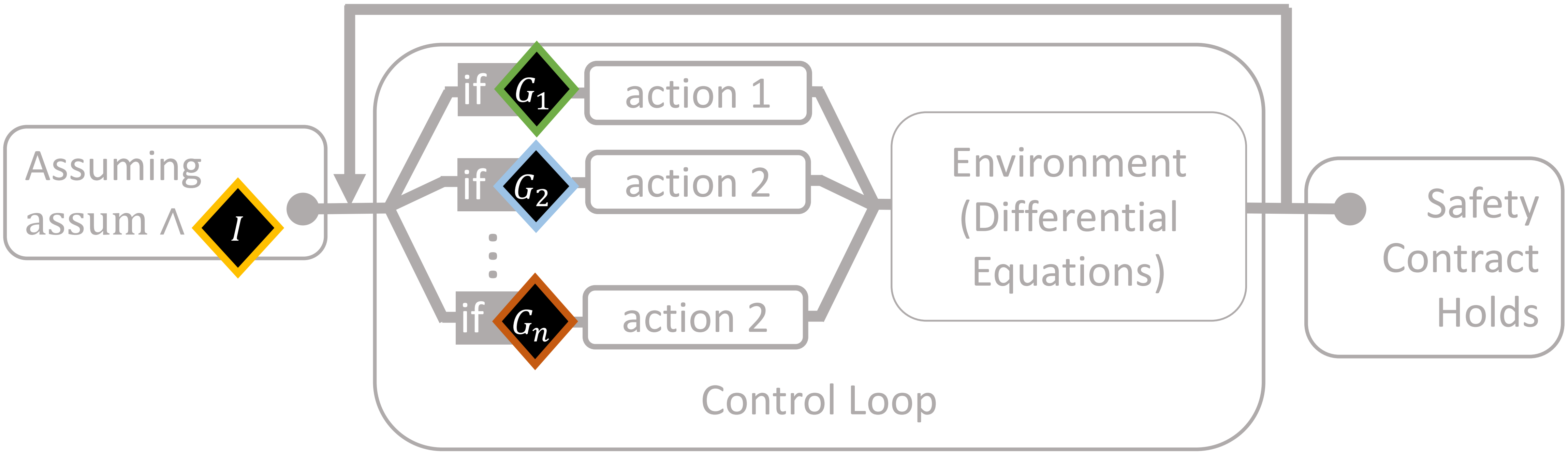


Where assum=$A > 0 \wedge B > 0 \wedge T > 0 \wedge v \geq 0$

[1] Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings. pp. 246–265 (2009). doi: 10.1007/ 978-3-642-10373-5_13

Solution

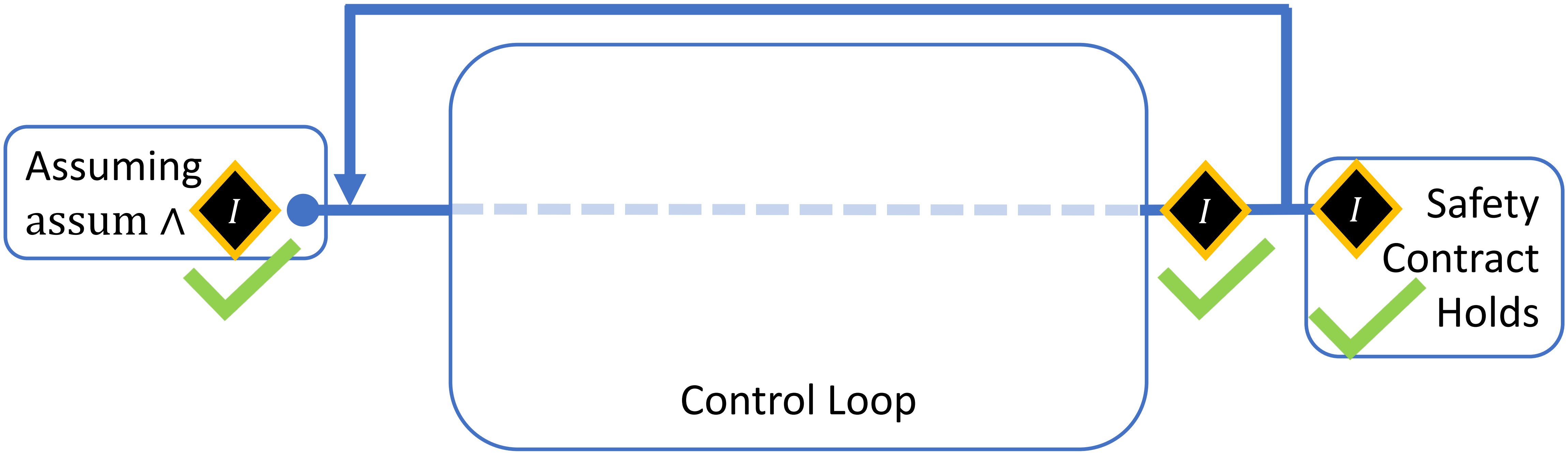
Synthesis procedure fills holes (?). Which action is safe when?



Solution (I, G_i)

Solution

Synthesis procedure fills holes (?). Which action is safe when?

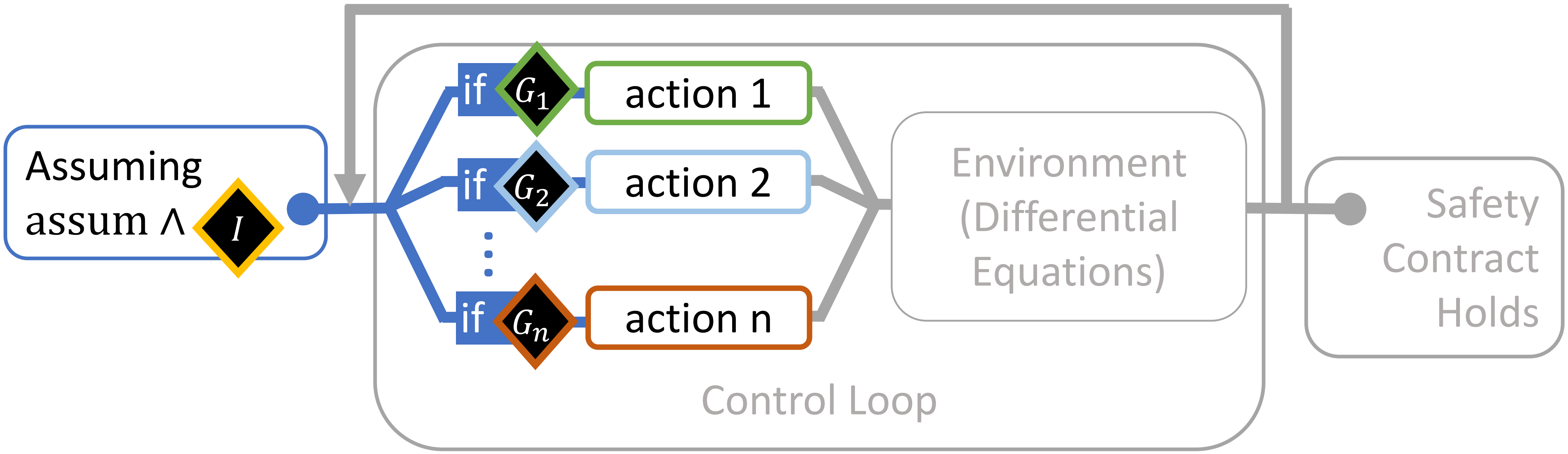


Solution (I, G_i) ensures:

1. Safety (valid formula, as proved by loop invariant assum $\wedge$ I)

Solution

Synthesis procedure fills holes (?). Which action is safe when?

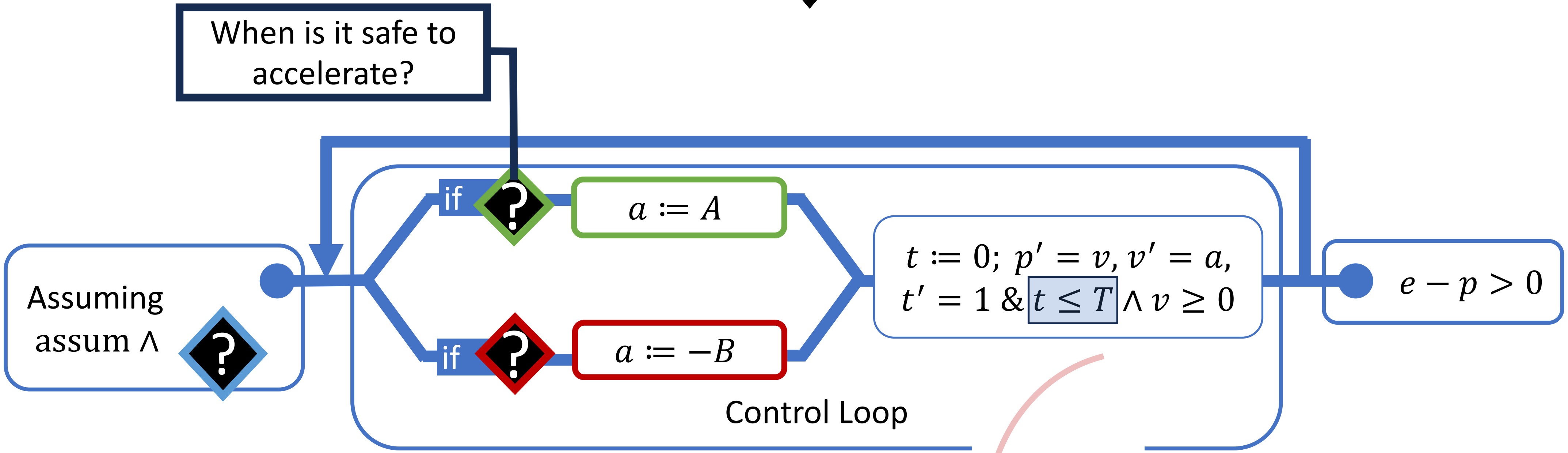


Solution (I, G_i) ensures

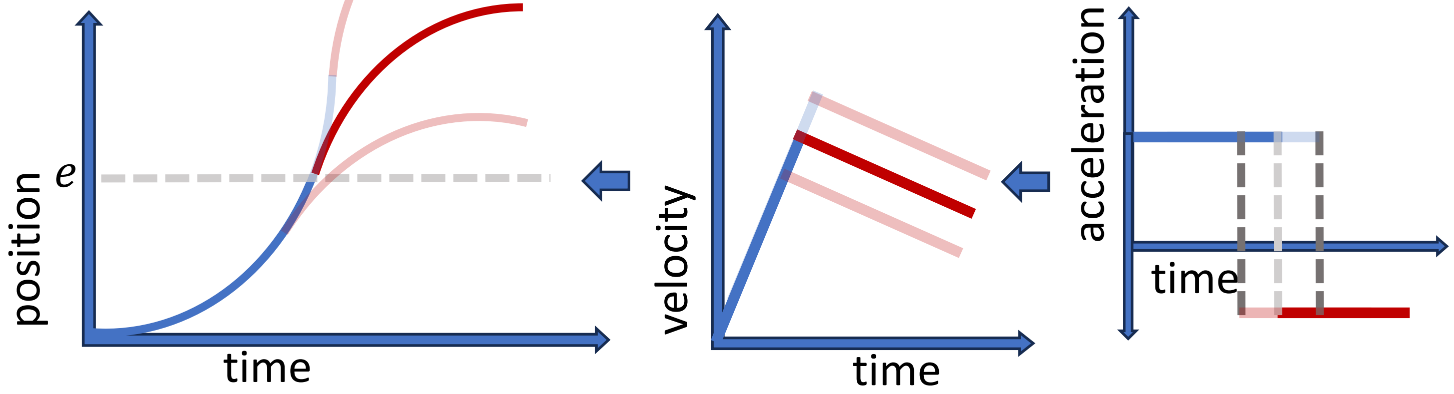
1. Safety (valid formula)
2. Controllability (always some control option: $(\text{assum} \wedge I) \rightarrow \vee_i \; G_i$)

Example: Train

Synthesis procedure fills holes (◆?). Which action is safe when?



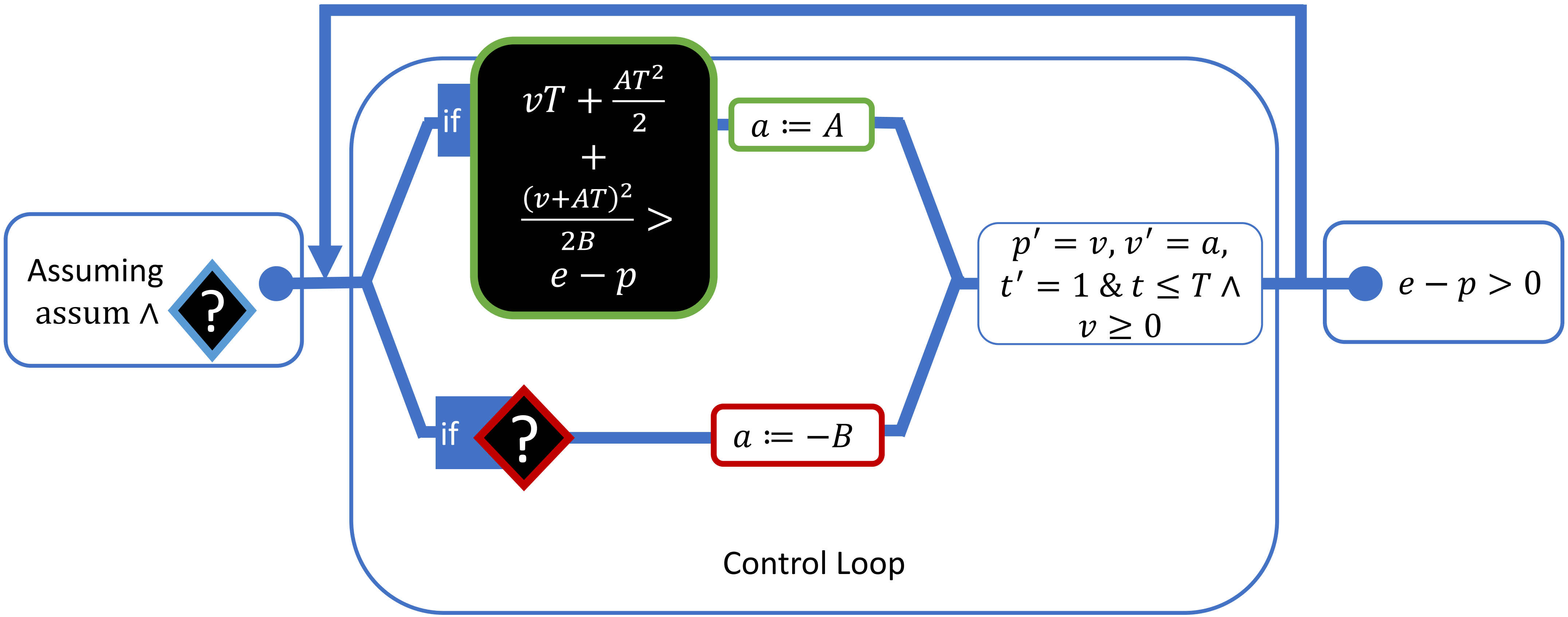
Back compute to ensure the safety contract isn't breached



[1] Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings. pp. 246–265 (2009). doi: 10.1007/ 978-3-642-10373-5_13

Example: Train

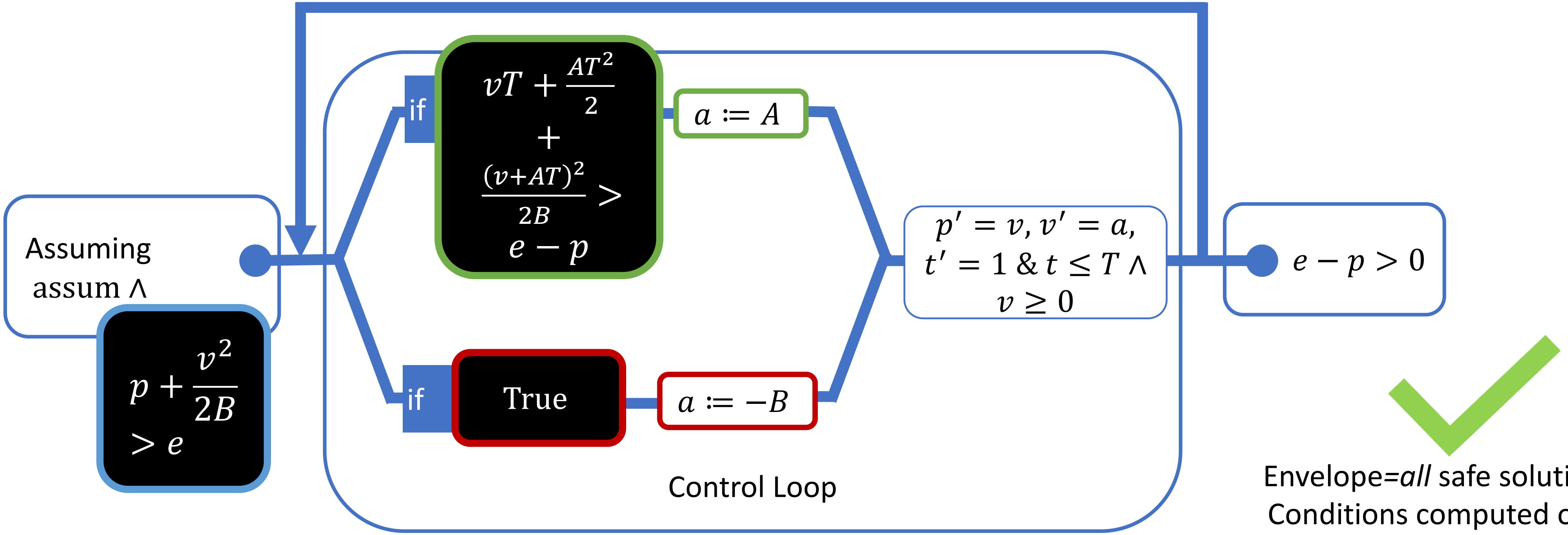
Synthesis procedure fills holes (?). Which action is safe when?



[1] Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings. pp. 246–265 (2009). doi: 10.1007/ 978-3-642-10373-5_13

Example: Train

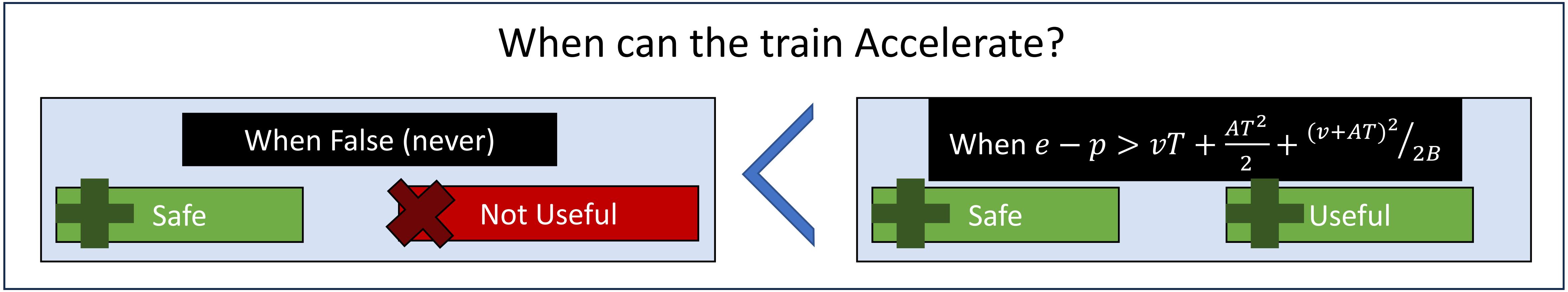
Synthesis procedure fills holes (?). Which action is safe when?



Envelope=*all* safe solutions.
Conditions computed once and for all

[1] Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings. pp. 246–265 (2009). doi: 10.1007/ 978-3-642-10373-5_13

Quality of Solution



- Good solution: more permissive
- $S' \geq S$ when either I' is strictly more permissive than I,

  $\vDash \text{assum} \rightarrow (I \rightarrow I')$ and $\vDash \text{assum} \rightarrow \neg(I' \rightarrow I)$

  or they are equally permissive and each ${G_i}'$ is more permissive than G_i

  $\vDash \text{assum} \rightarrow (I \rightarrow I')$ and $\vDash (\text{assum} \wedge I) \rightarrow \wedge_i (G_i \rightarrow G_i')$
- Optimum exists, expressible in dGL!

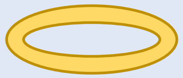
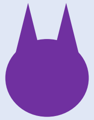
Overview

- Introduction
- Problem Statement
- **Solution**
- Evaluation

Background: Differential Game Logic (dGL)

Systems have nondeterminism
$(x := A \cup x := B)$

Players resolve nondeterminism


vs


Operators

$(x := A \cup x := B)$

$(x := A \cap x := B)$

$\alpha \cap \beta, \alpha^*, ?\phi, \{x\hat{}' = f(x) \& Q\}$

$\alpha \cap \beta, \alpha^\times, ?\phi^d, \{x' = f(x) \& Q\}^d$

Demonic Win Condition

$[(x := A \cap x := B)]x = A$
Demon wins if in the end, $x = A$

Winning Strategy

$[(\boxed{x := A} \cap x := B)]x = A$
Demon strategy: choose left

Formulas

$[(x := A \cap x := B)]x = A$

Formula true in states where demon has a **winning strategy**

dGL Axioms

Provide a way to get a *propositional arithmetic* formula saying "when can demon win this game"?

$[(v := 1 \cap v := -1); \{x' = v\}]x \neq 0$

$\Updownarrow$

$[(v := 1)][\{x' = v\}]x \neq 0 \lor$
$[(v := -1)][\{x' = v\}]x \neq 0$

$\Updownarrow$

$[\{x' = 1\}]x \neq 0 \lor [\{x' = -1\}]x \neq 0$

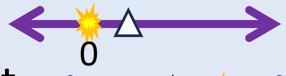
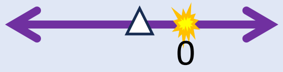
$\Updownarrow$

$\forall t \geq 0 x + t \neq 0 \lor \forall t \geq 0 x - t \neq 0$

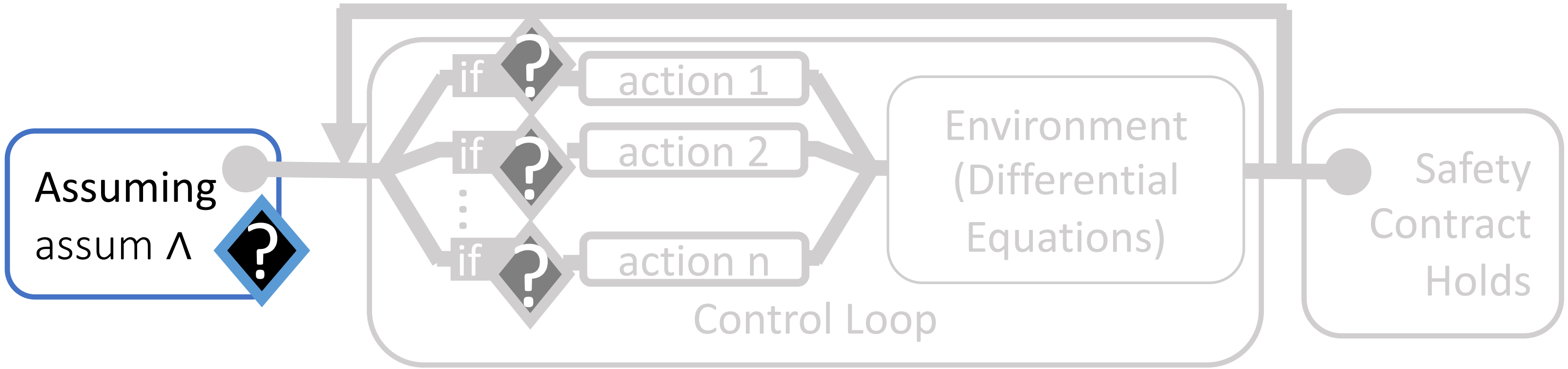
$\Updownarrow$

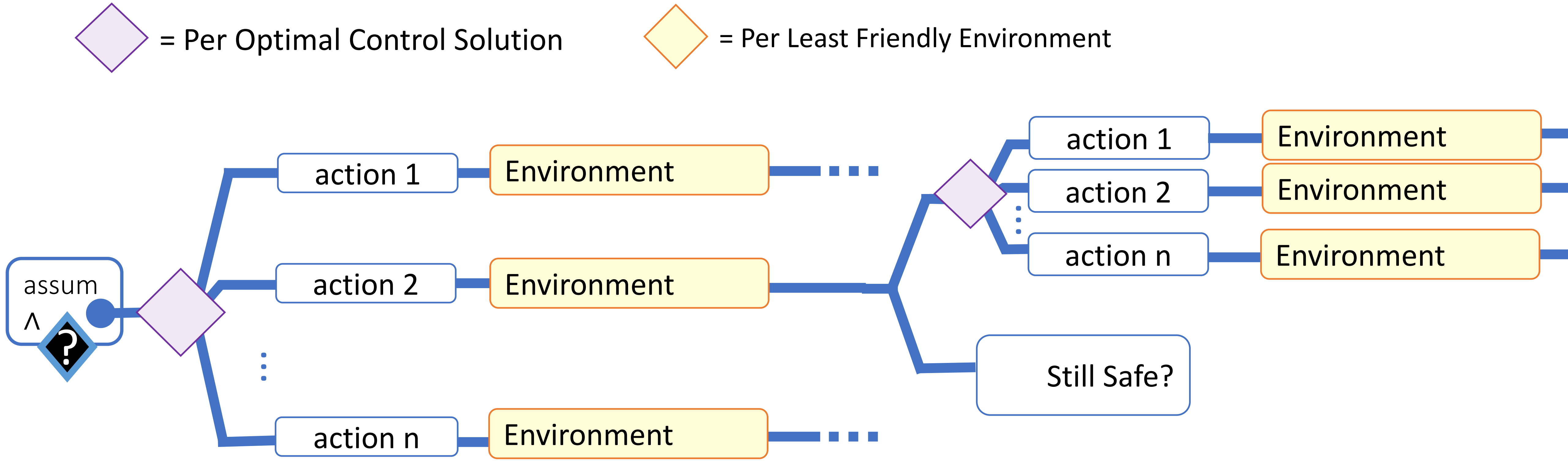
$x > 0 \lor x < 0$

Demon has a winning strategy if:

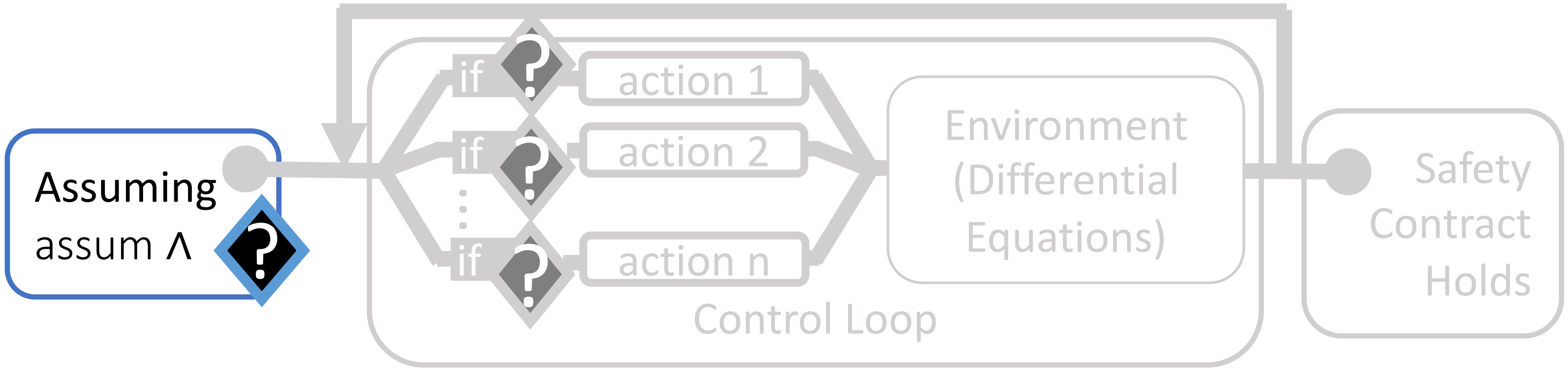
- $x > 0$: choose left 
- $x < 0$: choose right 

Optimal Solution



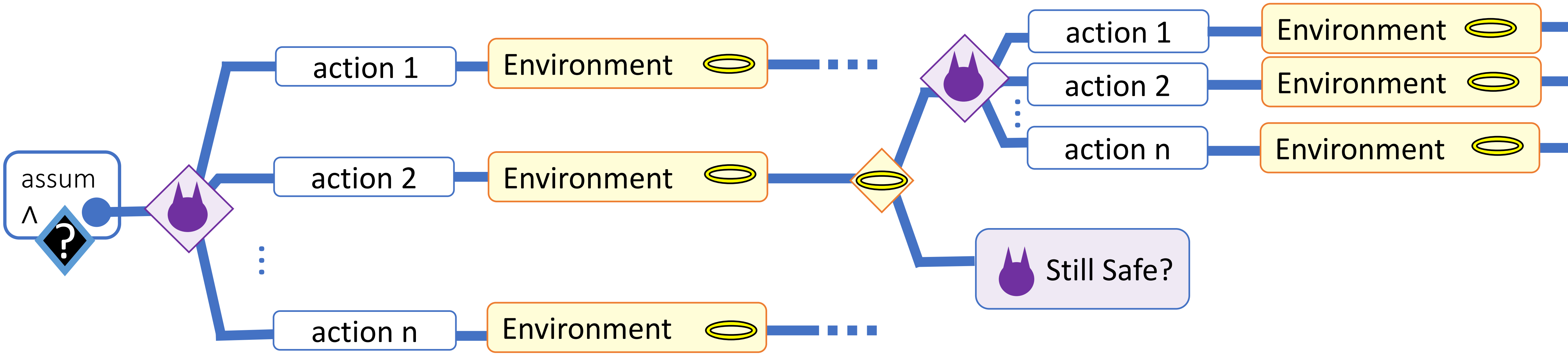


Optimal Solution

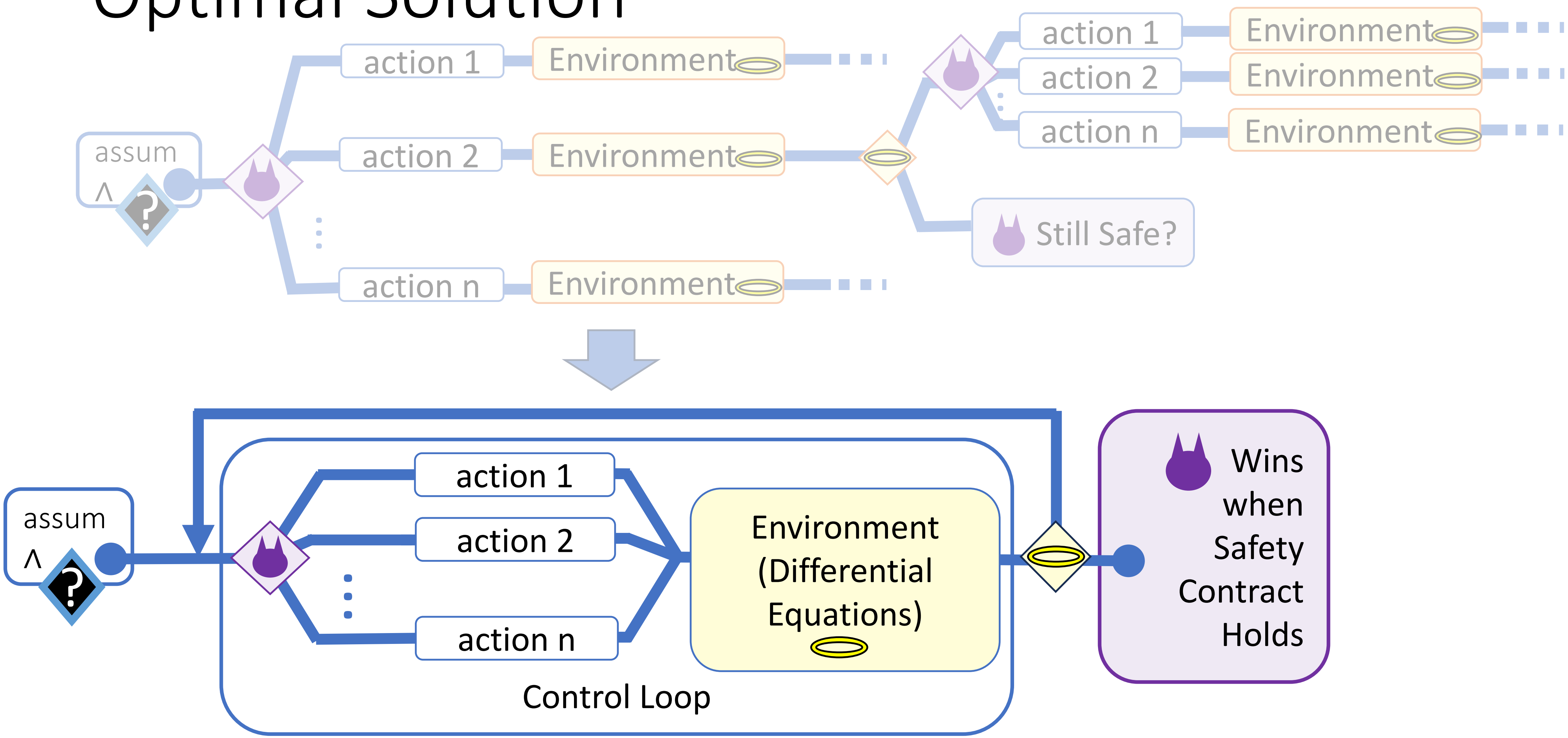


= Per Optimal Controller

= Per Least Friendly Environment



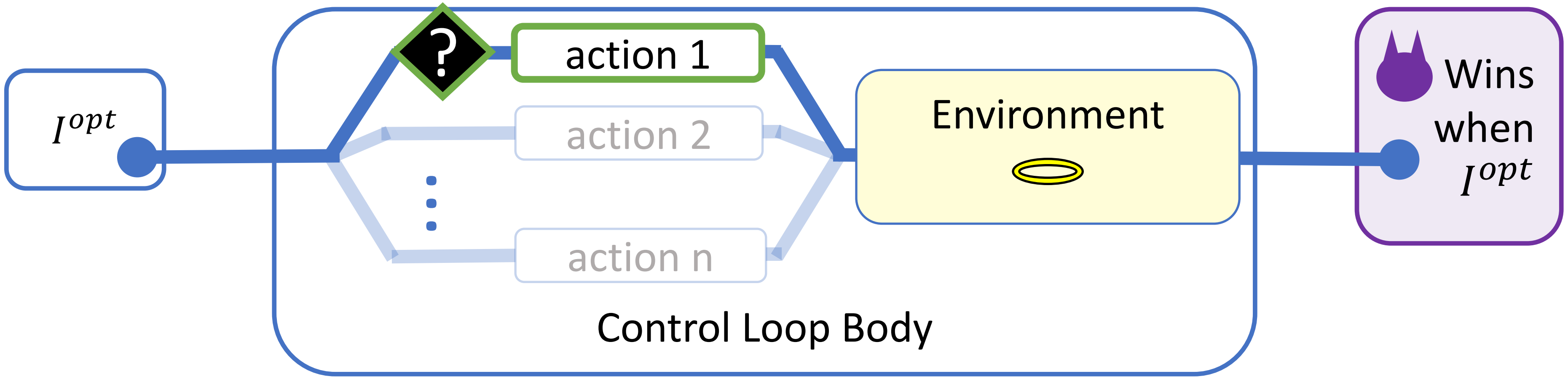
Optimal Solution



$$I^{\text{opt}} \equiv [((\cap_i \text{act}_i)\ ;\ \text{plant})^*]\,\text{safe}$$

Optimal Solution: Guards

Computed I^{opt} is loop invariant. Guards ensure inductive step.



Allow a control action when it is guaranteed to keep the system within I^{opt}

$$G_i^{\mathrm{opt}} \equiv [\mathsf{act}_i \,;\, \mathsf{plant}]\, I^{\mathrm{opt}}.$$

Next: Extracting Explicit Solutions

- Propositional arithmetic: easily checked at runtime
- Use the axioms of dGL (which are in terms of FOL^*)
- *But two dGL constructions need more than FOL.

  - Loops: Defined in terms of fixed point → Approximate with "Refinement"
  - Differential equations: Presupposes an ODE solution → Approximate using continuous invariants

Action Choice Refinement

The game obtained by restricting the controller to one action

$$\left[\left(\begin{matrix} \boldsymbol{a := -B}; t := 0; \\ \{p' = v, v' = a, t' = 1 \;\; t \le T \wedge v \ge 0\} \end{matrix}\right)^*\right] e - p > 0$$

2

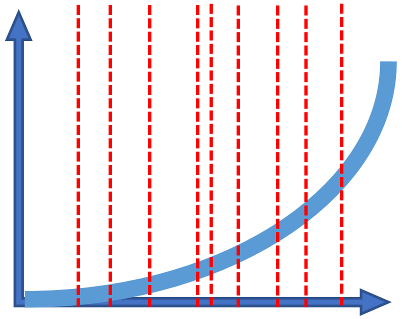
implies

Is harder than the game where the controller chooses between multiple actions

$$\left[\left(\begin{matrix} (\boldsymbol{a := -B \cap a := A}); t := 0; \\ \{p' = v, v' = a, t' = 1 \;\; t \le T \wedge v \ge 0\} \end{matrix}\right)^*\right] e - p > 0$$

1

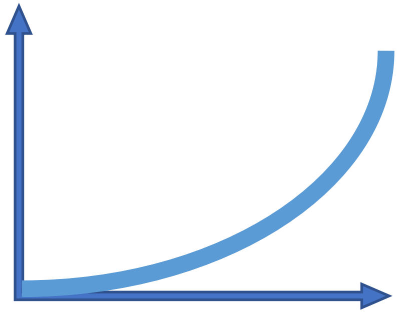
One Shot Unrolling



If you repeat a time bounded ODE

$$\left[\left(\begin{matrix} a := -B; \boldsymbol{t := 0}; \\ \{p' = v, v' = a, t' = 1 \;\; \boldsymbol{t \le T} \wedge v \ge 0\} \end{matrix}\right)^*\right] e - p > 0$$

3



That's like executing the ODE for arbitrarily long

$$[a := -B;\ \{p' = v, v' = a, t' = 1 \,\&\, v \ge 0\}] e - p > 0$$

4

After One Shot Unrolling

$$[a := -B;\ \{p' = v, v' = a, t' = 1\ \&\ v \geq 0\}]e - p > 0$$

Axioms of dGL

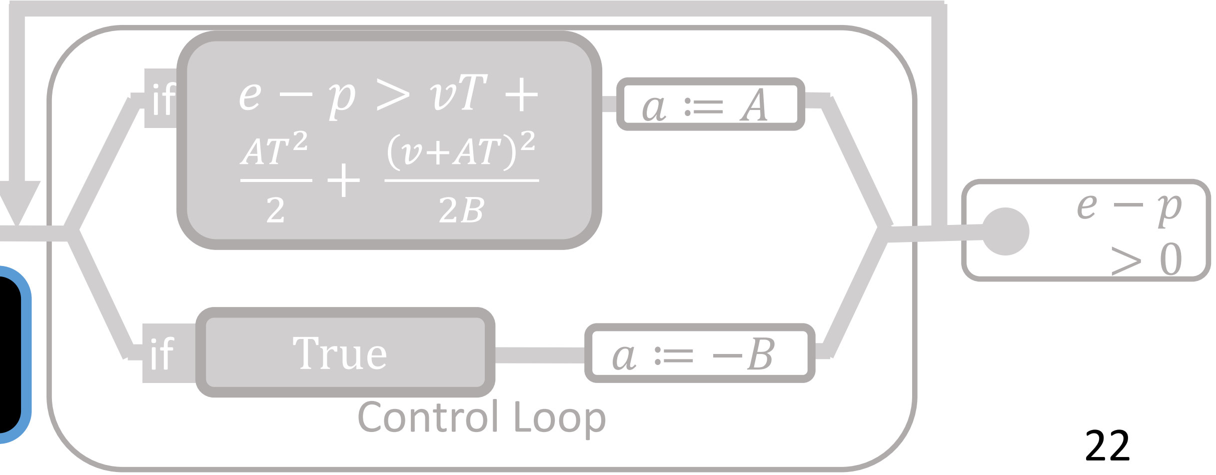
$$\forall t \left(v - Bt \geq 0\ \rightarrow p + vt - \frac{Bt^2}{2} > e\right)$$

Quantifier Elimination

$$I = \boxed{p + \frac{v^2}{2B} > e}$$

Assuming
assum $\wedge$

$$p + \frac{v^2}{2B} > e$$



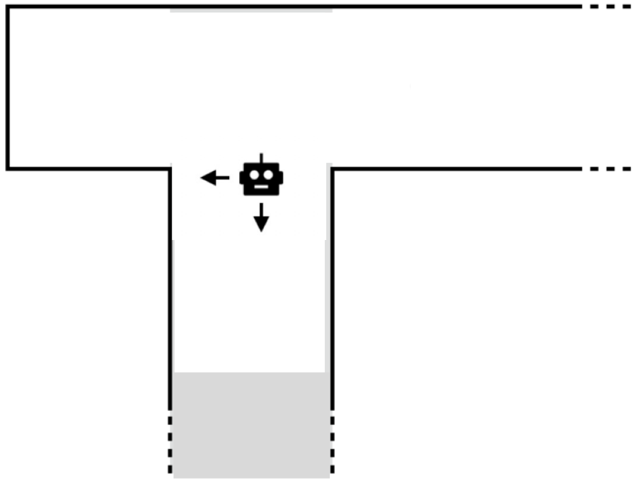
1. Define optimal solution game using hybrid system game theory

2. Systematic *refinements* make games easier to reason about

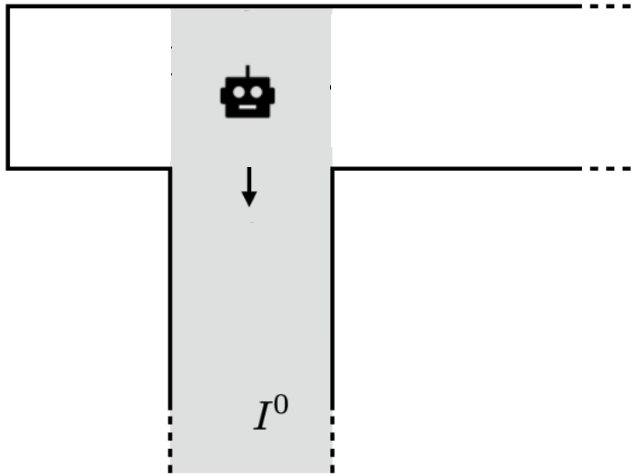
3. Symbolic execution using game axioms produces solution formulas

More Unrolling

- 1-shot unrolling lets the controller choose one action and run it forever.



1 iteration

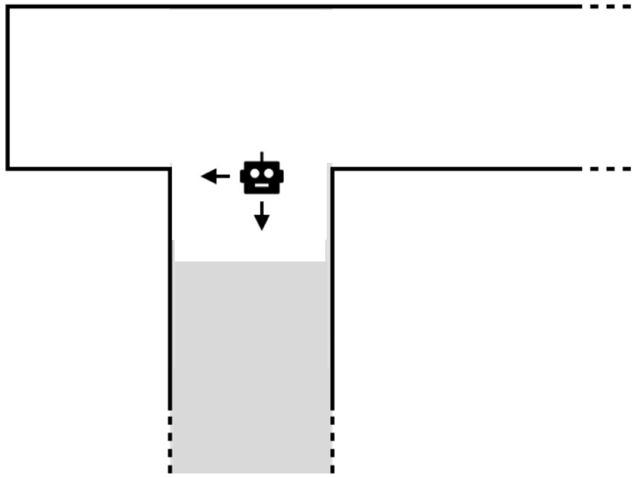


1-shot unroll

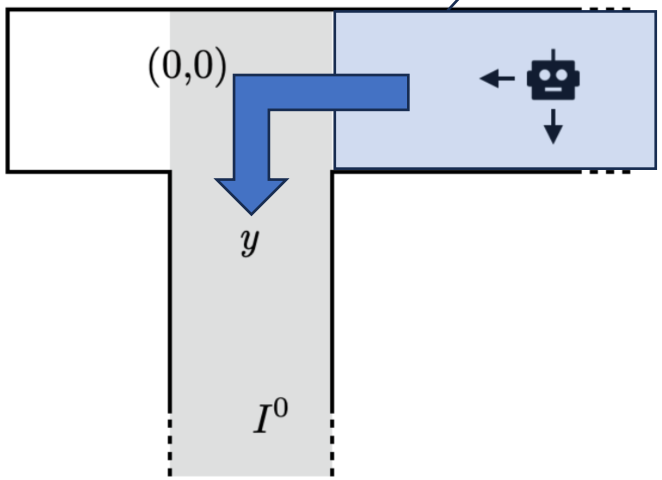
Multi-shot Bounded Unrolling

- 1-shot unrolling lets the controller choose one action and run it forever.
- Bounded unrolling allows a “switch” in action choice
- Recursive game formulation for each switch

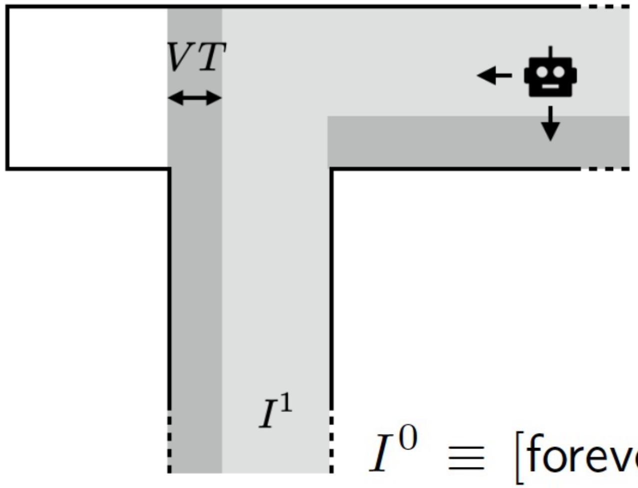
This is safe too, but requires robot to switch choice of action



2 iterations



1-shot unroll



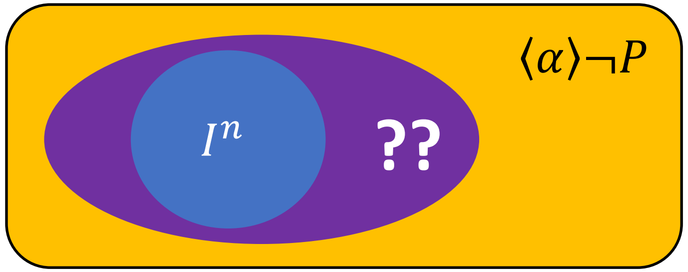
2-shot unroll

$$I^0 \equiv [\text{forever}]\,\text{safe}$$

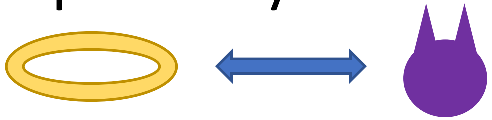
$$I^{n+1} \equiv I^n \vee [\text{step}]\, I^n$$

Other Ideas that Make CESAR Work

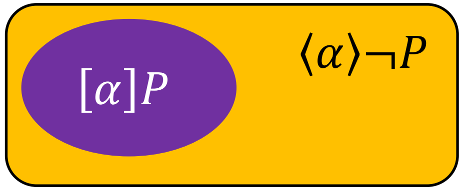
Problem: Is the synthesized envelope still optimal after all those refinements?



Solution: Optimality Checking by Duality

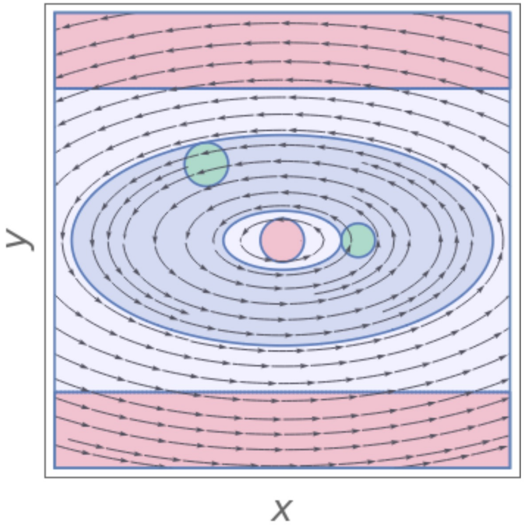


$\neg\langle\alpha\rangle\neg P \quad \leftrightarrow \quad [\alpha]P$



Problem: Symbolic reasoning about unsolvable ODEs

Solution: Approximate with Pegasus invariant generator



Problem: Complicated arithmetic expressions resulting in slow quantifier elimination

Solution: Proposition Arithmetic simplification using heuristics

Overview

Part 2: Synthesis

- Introduction
- Problem Statement
- Solution
- **Evaluation**

Evaluation

Summary of CESAR experimental results

Benchmark	Synthesis Time (s)	Checking Time (s)	Optimal	Needs Unrolling	Non Solvable Dynamics
ETCS Train [3]	14	9	✓		
Sled	20	8	✓		
Intersection	49	44	✓		
Parachute [4]	46	8			✓
Curvebot	26	9			✓
Coolant	49	20	✓	✓	
Corridor	20	8	✓	✓	
Power Station	26	17	✓	✓	

[3],[4]: Solved Manually in the Literature

State Dependent Fallback

Non Solvable Dynamics

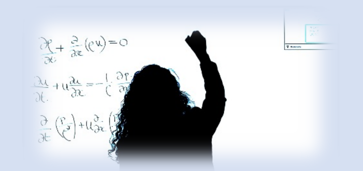
Optimal control requires a careful sequence of actions

[3] Platzer, A., Quesel, J.: European train control system: A case study in formal verification. In: Formal Methods and Software Engineering, 11th International Conference on Formal Engineering Methods, ICFEM 2009

[4] Fulton, N., Mitsch, S., Bohrer, R., Platzer, A.: Bellerophon: Tactical theorem proving for hybrid systems. In: Ayala-Rincon, M., Munoz, C.A. (eds.) ITP. LNCS, vol. 10499, pp. 207–224. Springer (2017).

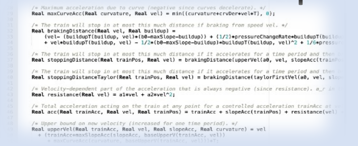
Summary

Model Shape



Synthesis procedure fills control conditions

Correct by Construction Control Envelope



Automate

Evaluation: Benchmark Suite with Diverse Control Challenges

Table 2: Summary of CESAR experimental results

Benchmark	Synthesis Time (s)	Checking Time (s)	Optimal	Needs Unrolling	Non Solvable Dynamics
ETCS Train	14	9	✓		
Sled	20	8	✓		
Intersection	49	44	✓		
Parachute	46	8			✓
Curvebot	26	9			✓
Coolant	49	20	✓	✓	
Corridor	20	8	✓	✓	
Power Station	26	17	✓	✓	

Solution Ordering

When False (never)

Safe

Not Useful

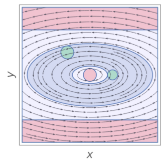
$<$

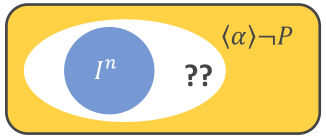
When $e - p > {(v+AT)^2}/{2B}$

Safe

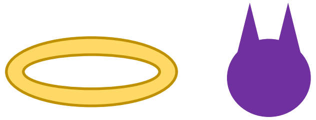
Useful

Other Techniques



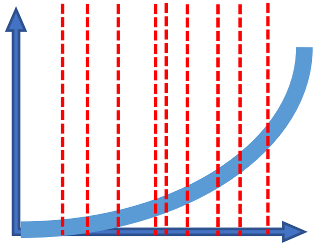


Characterize optimal solution using games



Compute explicit solution with symbolic execution of **refined** games

Refinement: One-Shot Unrolling



Refinement: Multi-shot Unrolling