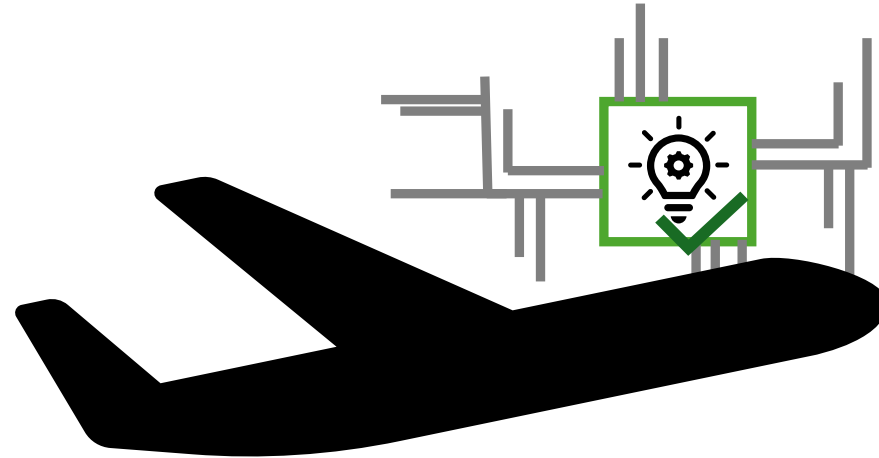


Hybrid Game Control Envelope Synthesis

OOPSLA 2026



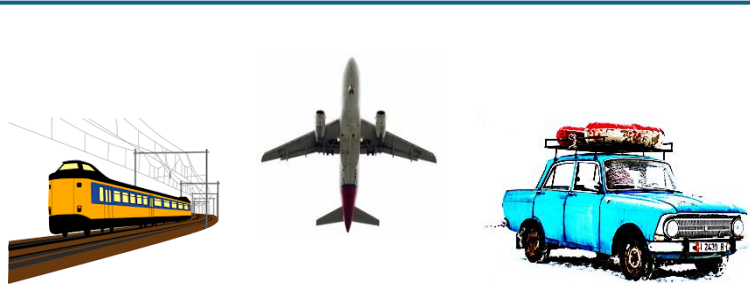
Aditi Kabra *

Jonathan Laurent °

Stefan Mitsch †

André Platzer °

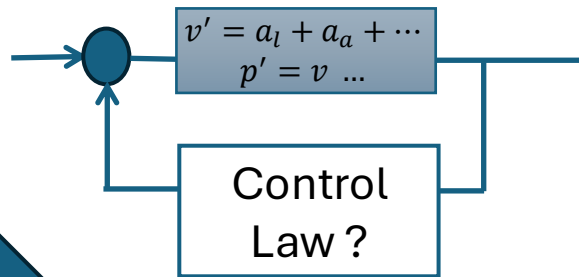
Control Envelope Synthesis



Hybrid systems
Verification

Requires

Designing Control
Conditions



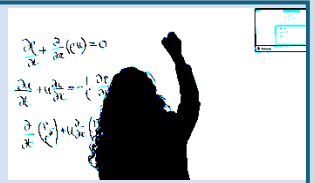
- Difficult: the main point control theory
- Needs creativity, careful reasoning about dynamics

Conjecture
Conditions

Try Proof

This Paper: Synthesize Correct
Control Conditions

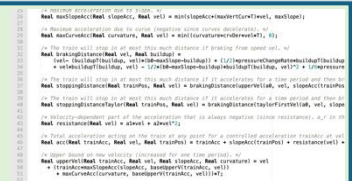
Human provides spec:
Control options,
Environment behavior



Synthesis procedure:
Which control decision
is safe when



Verified Control
Envelope!

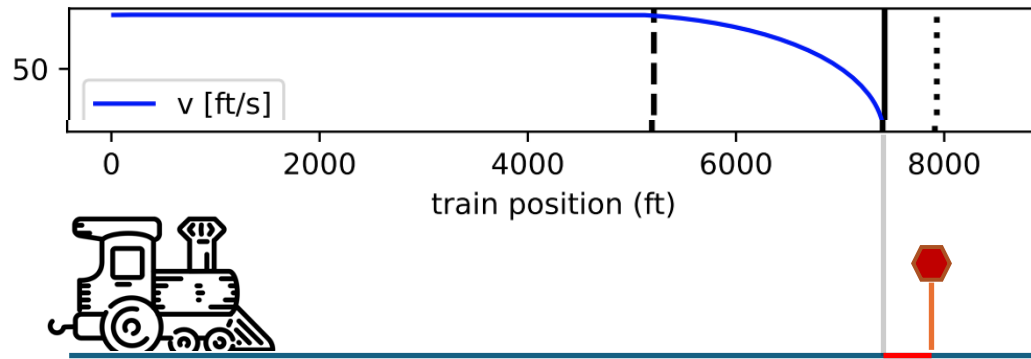


Proof: ✓ All goals closed

```
Provable( ==> end()-trainPos>(ve
reChangeRate()*vel)^(1/2))/pressu
&._0=-._1|._1>=0&._0=-._1 >>(((b0()-
essureChangeRate(),(b0()-maxSlope(
0=-._1|._1>=._2&._0=-._2 >>((Apb()-0
```

What do Control Envelopes do?

Verified controller 1:
follow signals correctly



Start braking



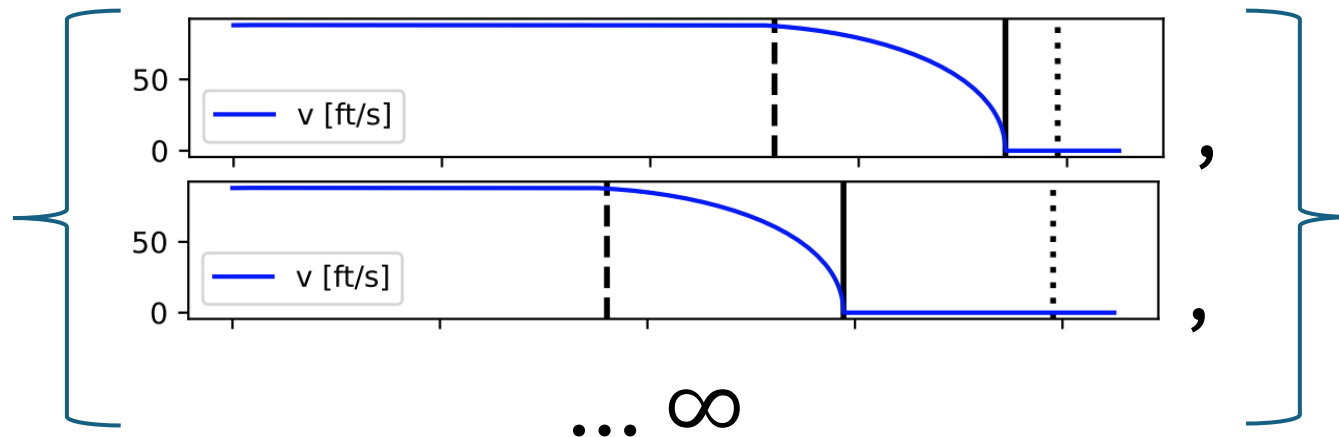
Train stops



End of movement authority

Want: follow signals + **energy efficient given current weather conditions**

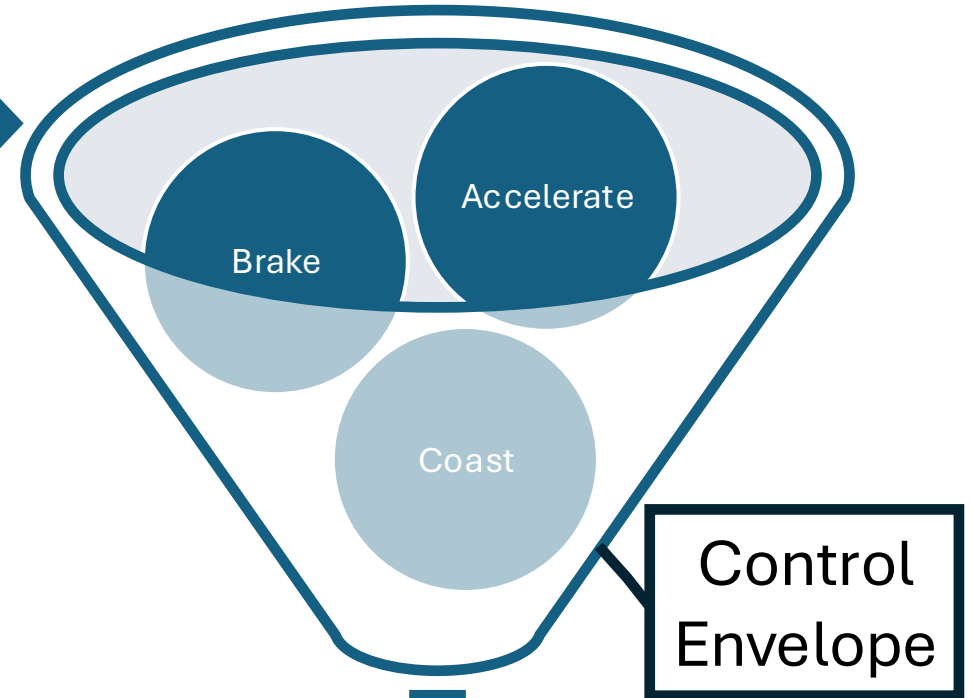
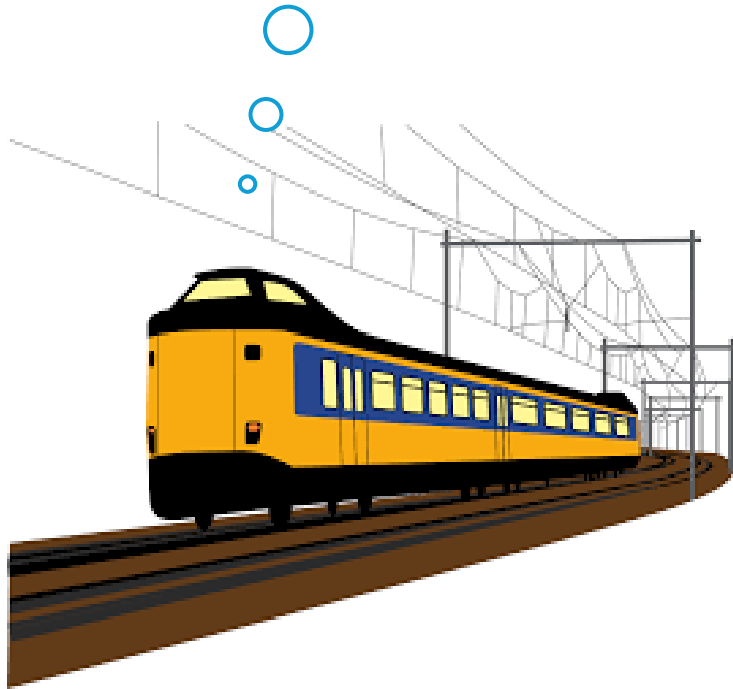
Control Envelope: **family** of controllers that follow signals correctly



Optimize within this space
for energy efficiency.

What do Control Envelopes do?

$\sigma \equiv \{p = 3 \text{ miles from Oakland}, v = 80\text{mph}\}.$
Can I accelerate?
Can I brake?



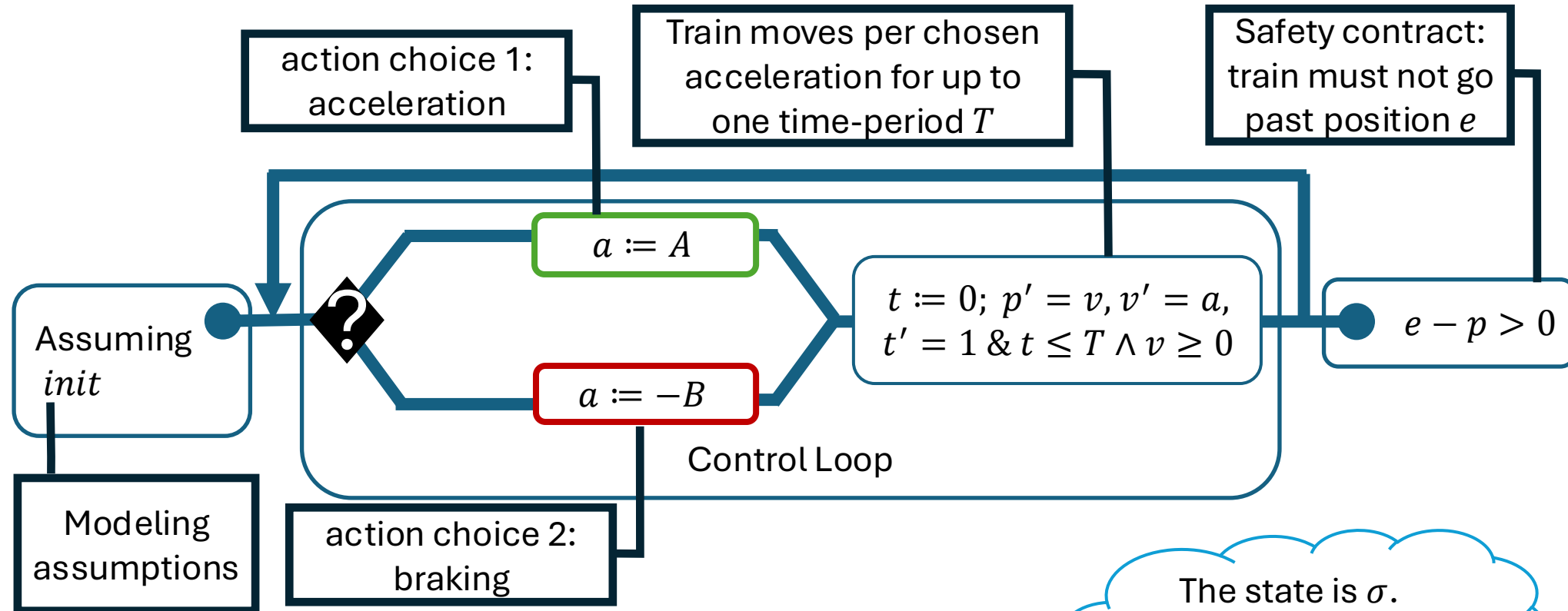
✗ Accelerate

✓ Brake

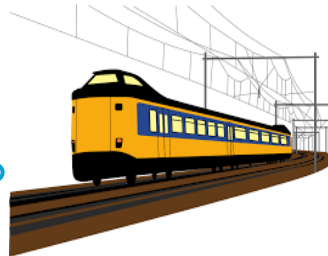
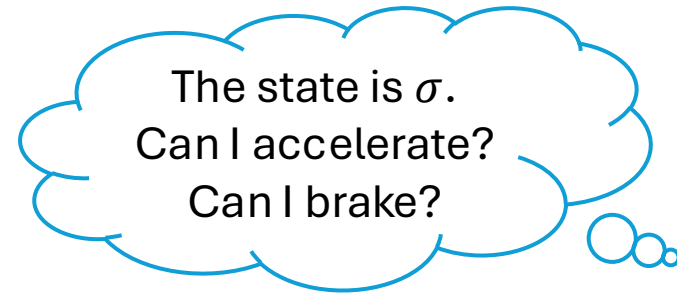
✓ Coast

Example Control Envelope Spec: Train

Envelope fills (❓). Which action is safe when?

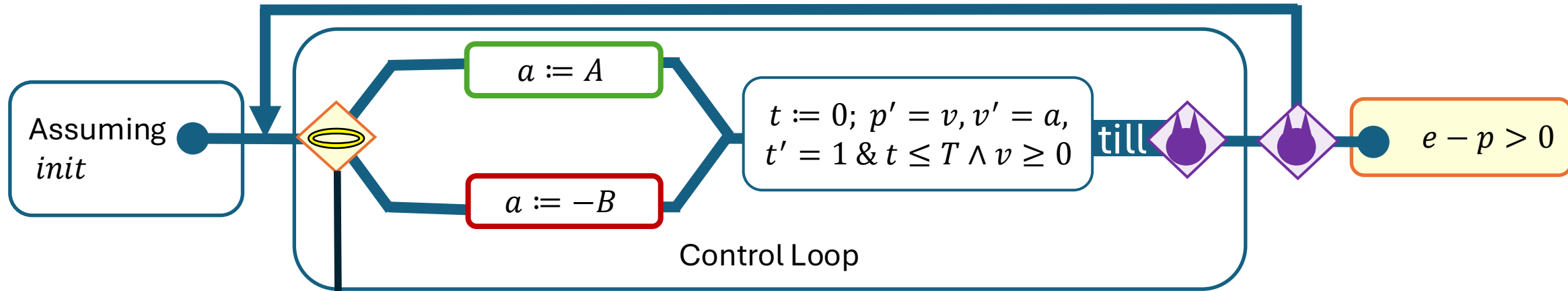
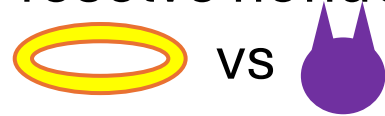


Where $init = A > 0 \wedge B > 0 \wedge T > 0 \wedge v \geq 0$



Control Envelope Synthesis **Spec**: Hybrid Game

Players resolve nondeterminism



when can Angel accelerate = when can train accelerate
when can Angel brake = when can train brake

dGL: a logic for hybrid games

$\alpha, \beta ::= x := \theta \mid x' = \theta \ \& \ \psi \mid ?\psi \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \alpha^d$

Background: Differential Game Logic (dGL)

Systems have nondeterminism

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

Players resolve nondeterminism



vs



Operators

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

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

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

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



Demonic Win Condition

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

Demon wins if in the end, $x = A$

Winning Strategy

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

Demon strategy: choose left

Formulas

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

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

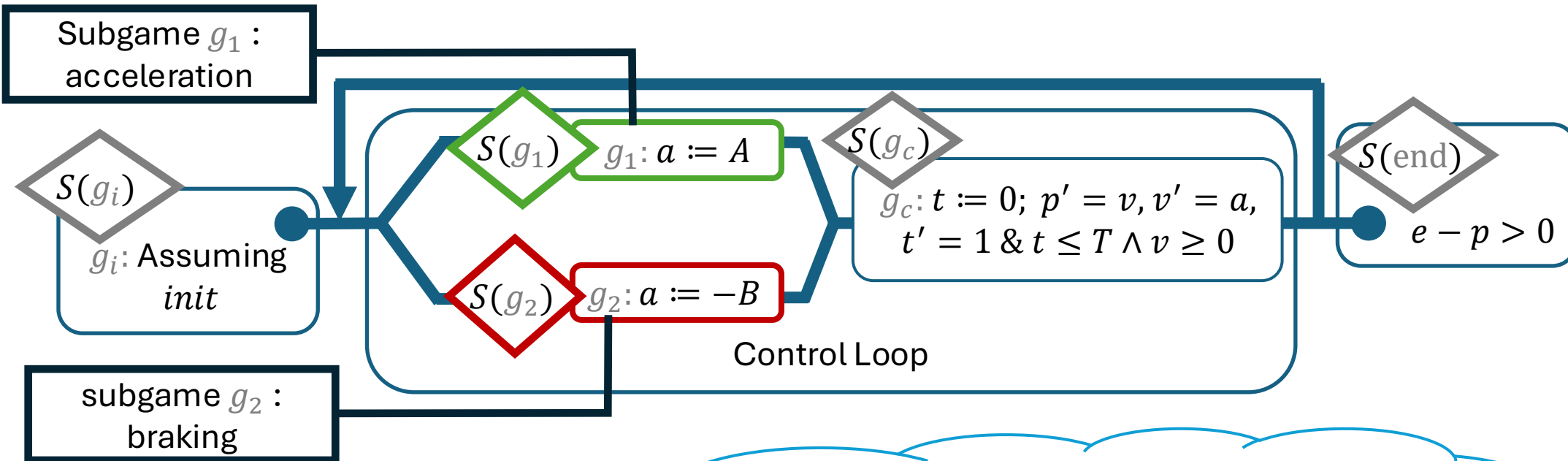
The Question

*How can we represent control envelopes for **all** dGL specifications in a way that is:*

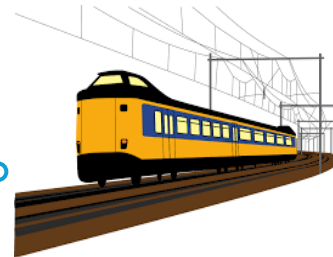
- *Compositional*
- *Symbolic (in terms of parameters like A, T)*

Control Envelope Representation: Train

Inductive Subvalue Map S maps subgames to formulas (“subvalues”)

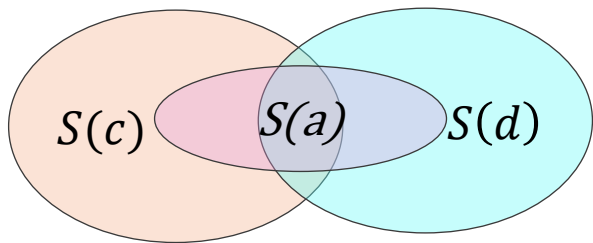
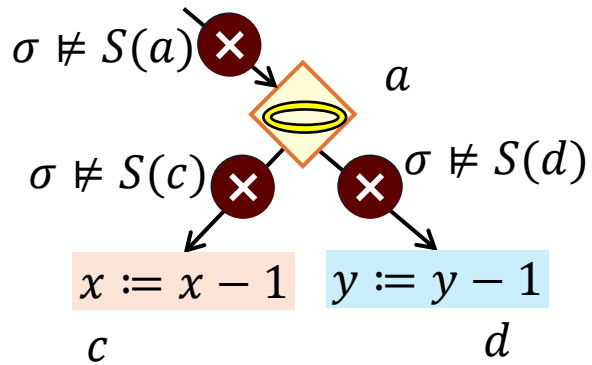


The state is σ .
I can accelerate if $\sigma \models S(g_1)$
I can brake if $\sigma \models S(g_2)$



Control Envelope Representation: Inductive Subvalue Maps

- Want maps that represent strategies that:
 - Don't lose
 - Don't get stuck



$$S(a) \rightarrow S(c) \vee S(d)$$

- Recursive dGL formulation

atomic, i.e., $\alpha \in \{x := e, x := *, x := \otimes, ?Q, !Q, \{x' = f(x) \ \& \ Q\},$

$\{x' = f(x) \ \& \ Q\}^d\}$ then $\models S(a) \rightarrow \langle \alpha \rangle S(\text{end})$.

$a:(g:\gamma \cup d:\delta)$ then $\models S(a) \rightarrow S(g) \vee S(d)$ and $g:\gamma \models S$ and $d:\delta \models S$.

$a:(g:\gamma \cap d:\delta)$ then $\models S(a) \rightarrow S(g) \wedge S(d)$ and $g:\gamma \models S$ and $d:\delta \models S$.

$a:(g:\gamma ; d:\delta)$ then $\models S(a) \rightarrow S(g)$ and $g:\gamma \models S(\text{end} \mapsto S(d))$ and $d:\delta \models S$.

$a:(g:\gamma)^*$ then $\models S(a) \rightarrow \langle a:\alpha \exists S \rangle S(\text{end})$ and $g:\gamma \models S(\text{end} \mapsto S(a))$.

$a:(g:\gamma)^\times$ then $\models S(a) \rightarrow S(g) \wedge S(\text{end})$ and $g:\gamma \models S(\text{end} \mapsto S(a))$.

$a:(g:\gamma)^\times$ then $\models S(a) \rightarrow S(g) \wedge S(\text{end})$ and $g:\gamma \models S(\text{end} \mapsto S(a))$.

Invariant

not
stuck

Implies
post-
condition

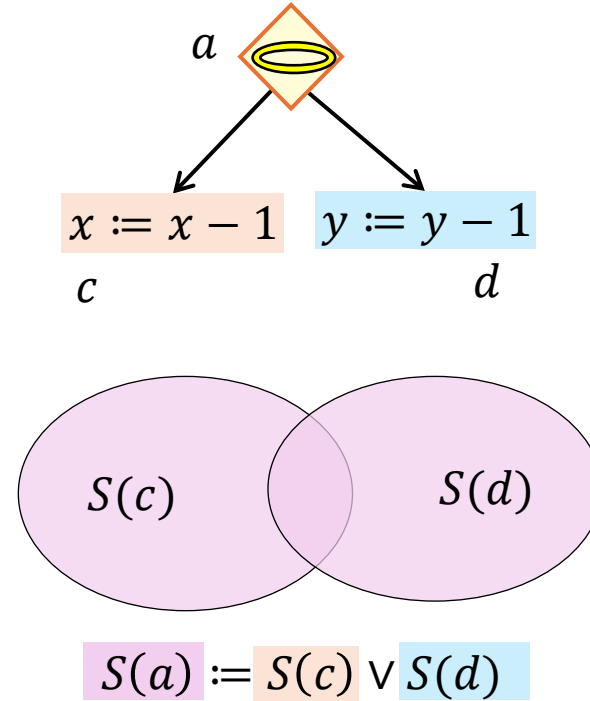
is inductive

- Can be checked in theorem prover



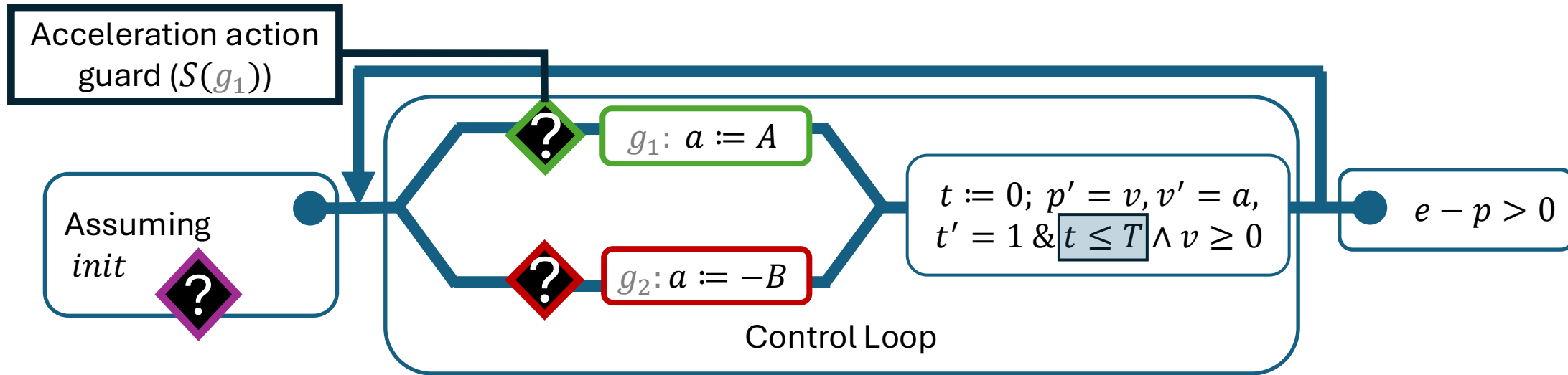
Overview

- Control Envelopes
- Problem Class Generality
- Solution Representation
- **Synthesis**
- Evaluation

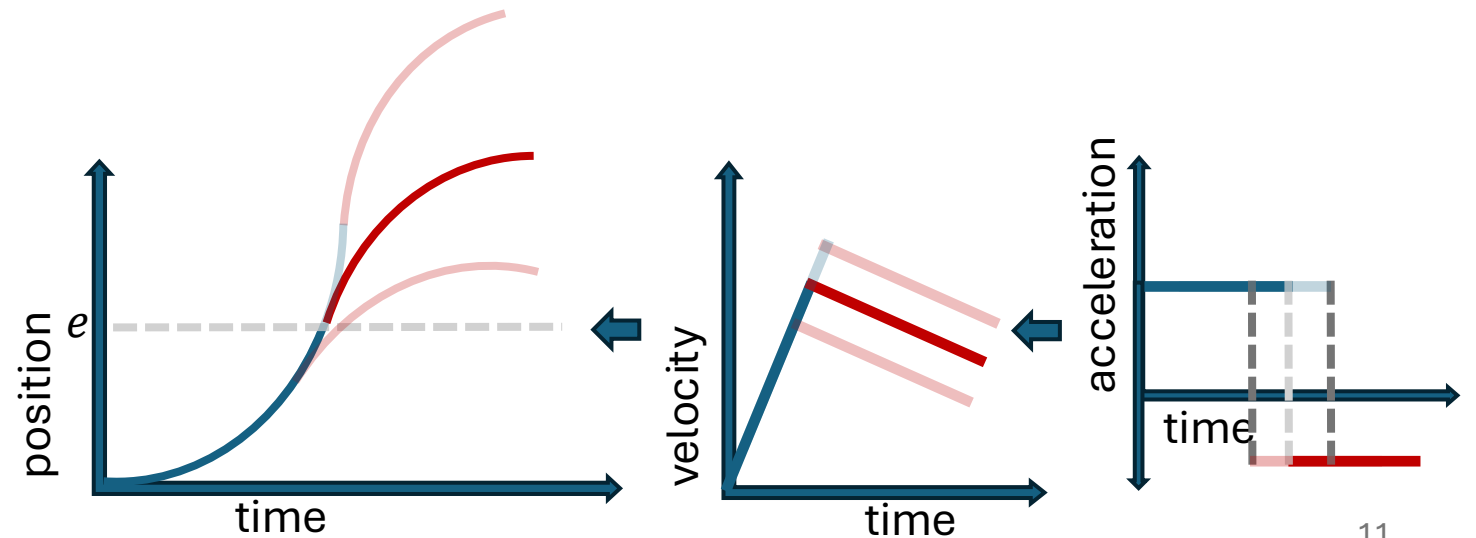


Example Control **Envelope**: Train

Synthesis fills (❖). Which action is safe when?

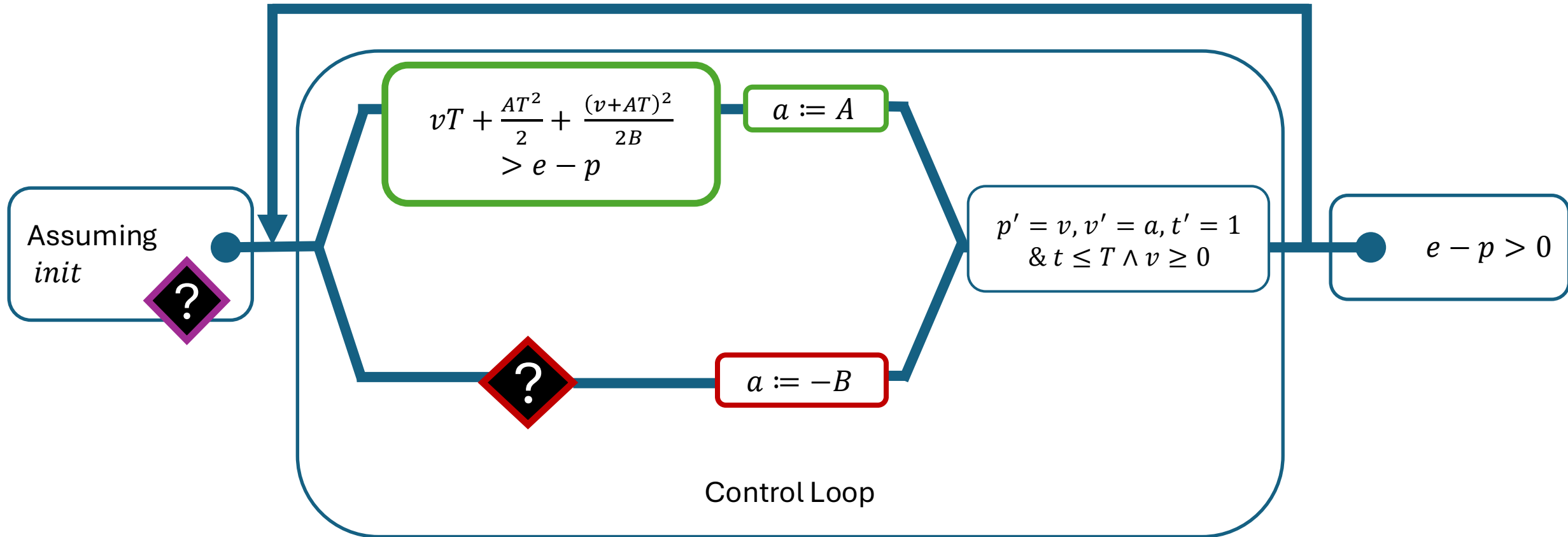


Back compute to ensure the safety contract isn't breached



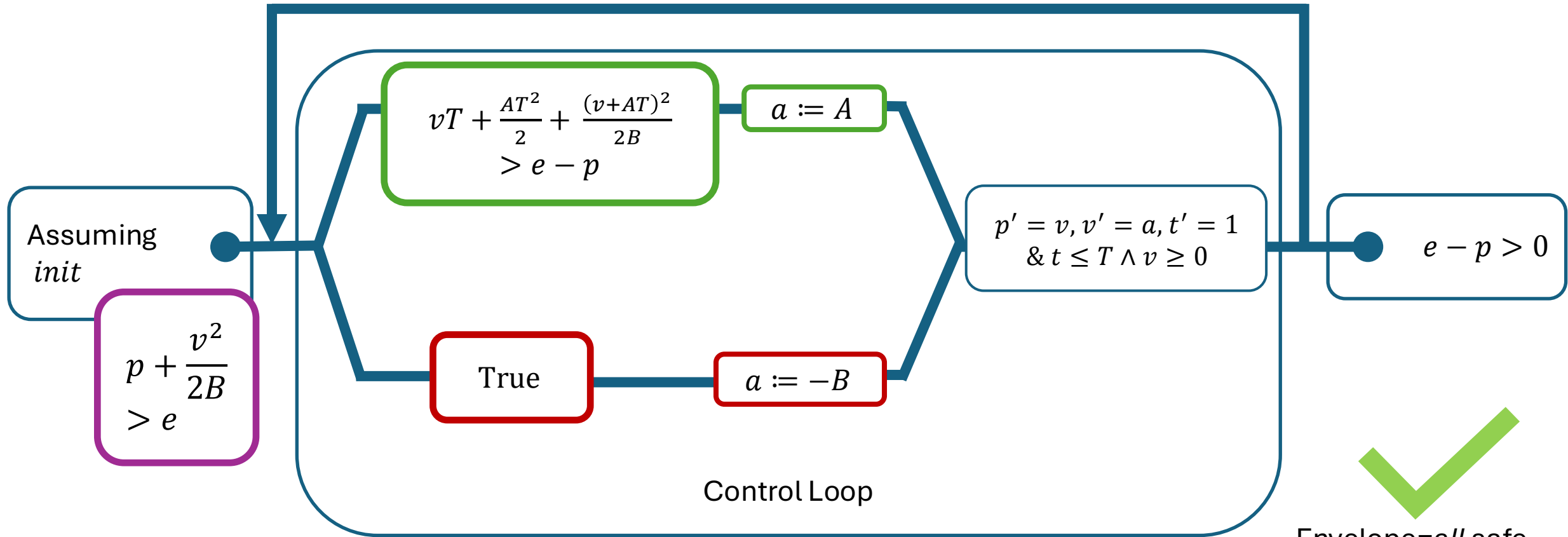
Example: Train

Synthesis fills holes (❖).



Example: Train

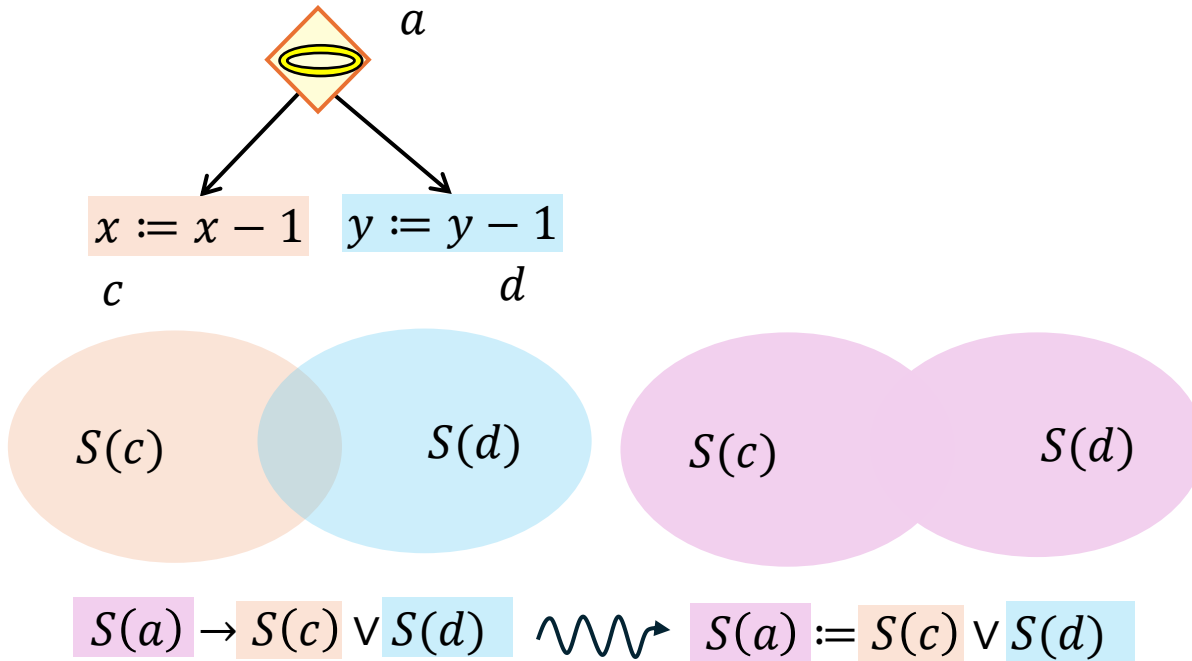
Synthesis fills holes (❓).



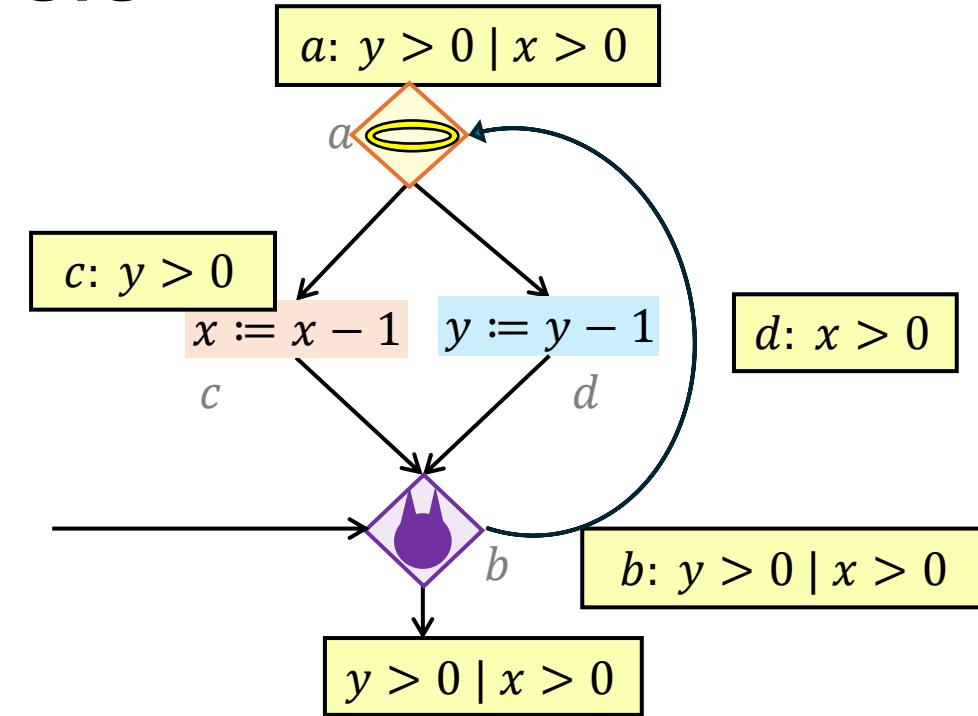
Envelope=*all* safe solutions. Conditions computed once and for all

Inductive Subvalue Map Synthesis

- Recursive dGL formulation allows backwards symbolic reasoning



- Unlike program synthesis, no search explosion
- Still depends on loop invariant synthesis



Algorithm 1 Inductive subvalue map synthesis framework

```

1: function  $\langle \text{map}(a, \alpha, \phi) \rangle$  ▷ Parametric Input:  $\text{genInv}$ 
2:   if  $\alpha \in \{x := e, x := *, x := \otimes, ?Q, !Q, \{x' = f(x) \& Q\}, \{x' = f(x) \& Q\}^d\}$  then
3:     return  $\{a \mapsto \text{simpl}(\langle \alpha \rangle \phi)\}$ 
4:   else if  $\alpha = a.(g; \gamma \cup d; \delta)$  then
5:      $S_1 \leftarrow \langle \text{map}(g; \gamma, \phi) \rangle$ ;  $S_2 \leftarrow \langle \text{map}(d; \delta, \phi) \rangle$ ; return  $S_1 \sqcup S_2 \sqcup \{a \mapsto S_1(g) \vee S_2(d)\}$ 
6:   else if  $\alpha = a.(g; \gamma \cap d; \delta)$  then
7:      $S_1 \leftarrow \langle \text{map}(g; \gamma, \phi) \rangle$ ;  $S_2 \leftarrow \langle \text{map}(d; \delta, \phi) \rangle$ ; return  $S_1 \sqcup S_2 \sqcup \{a \mapsto S_1(g) \wedge S_2(d)\}$ 
8:   else if  $\alpha = a.(g; \gamma; d; \delta)$  then
9:      $S_1 \leftarrow \langle \text{map}(d; \delta, \phi) \rangle$ ;  $S_2 \leftarrow \langle \text{map}(g; \gamma, S_1(d)) \rangle$ ; return  $S_1 \sqcup S_2 \sqcup \{a \mapsto S_2(g)\}$ 
10:  else if  $\alpha = a.(g; \gamma)^*$  then
11:    while true do
12:       $\text{Inv} \leftarrow \text{genInv}(\langle (g; \gamma)^* \rangle \phi)$ ;  $S \leftarrow \langle \text{map}(g; \gamma, \text{Inv} \vee \phi) \rangle$ 
13:      if  $\models \text{Inv} \rightarrow \langle a \alpha (S^0, \phi) \rangle \phi$  then return  $S \sqcup \{a \mapsto \text{Inv}\}$ 
14:  else  $\alpha = a.(g; \gamma)^\times$ 
15:    while true do
16:       $\text{Inv} \leftarrow \text{genInv}(\langle (g; \gamma)^\times \rangle \phi)$ ;  $S \leftarrow \langle \text{map}(g; \gamma, \text{Inv}) \rangle$ 
17:      if  $\models \text{Inv} \rightarrow S(g) \wedge \phi$  then return  $S \sqcup \{a \mapsto \text{Inv}\}$ 

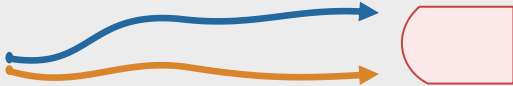
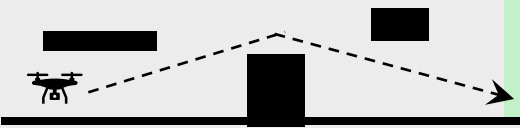
```

Overview

- Control Envelopes
- Problem Class Generality
- Solution Representation
- Synthesis
- **Evaluation**

Benchmark	Ours	CESAR	NYCS	Problem Type
Event ETCS	23s	∞	∞	Event-triggered control
Infinite Track	233s	∞	∞	Infinite horizon switching
Surgical Robot [33]	168s	∞	∞	Nonlinear control
Highway Driving [35]	91s	∞	∞	Time-triggered control with adversarial agent
Reach-avoid Robot	21s	∞	∞	Reach-avoid problem

Representatives of Different Problem Classes

Problem Type	
Event-triggered control	
Infinite horizon switching	
Nonlinear control	
Time-triggered control in presence of adversarial agent	
Reach-avoid problem	

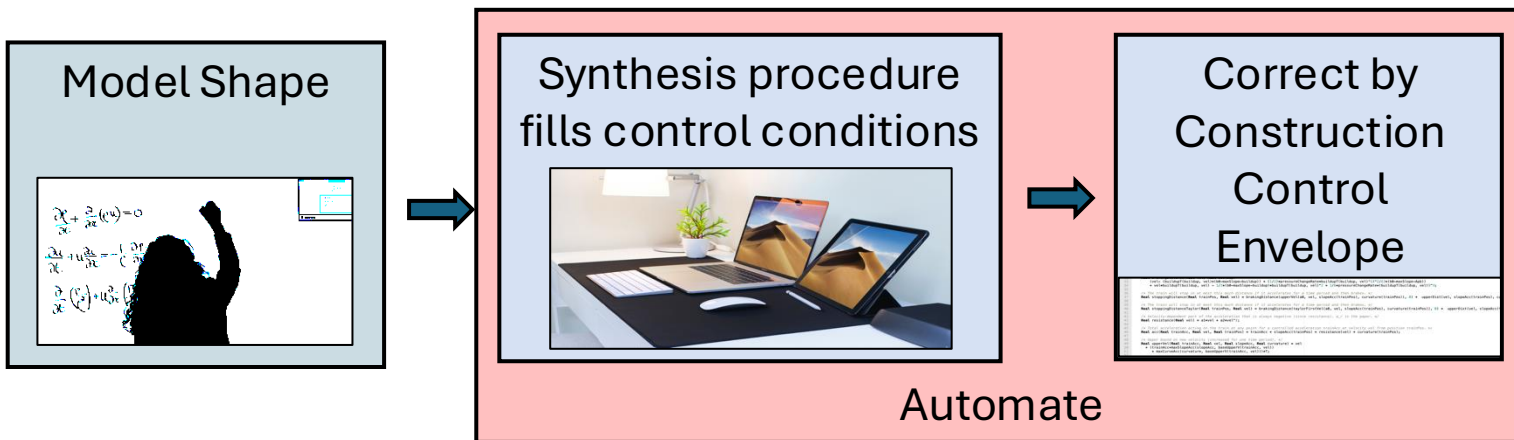
Representatives of Different Problem Classes

Benchmark	Problem Type	Ours	CESAR ^[1]	NYCS ^[2]
Event ETCS	Event-triggered control	23s	∞	∞
Infinite Track	Infinite horizon switching	233s	∞	∞
Surgical Robot	Nonlinear control	168s	∞	∞
Highway Driving	Time-triggered control in presence of adversarial agent	91s	∞	∞
Reach-avoid Robot	Reach-avoid problem	21s	∞	∞

[1] CESAR: Control Envelope Synthesis via Angelic Refinements. Aditi Kabra, Jonathan Laurent, Stefan Mitsch, André Platzer. TACAS 2024.

[2] Automatic Synthesis of Switching Controllers for Linear Hybrid Systems: Reachability Control. Massimo Benerecetti and Marco Faella. ACM Trans. Embed. Comput. Syst. 16, 4, Article 104. 2017.

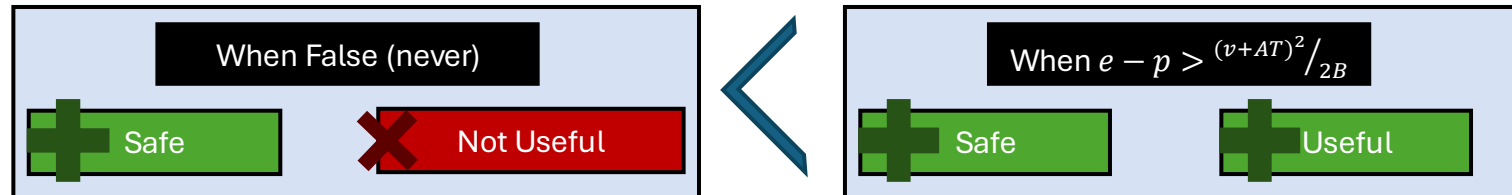
Summary



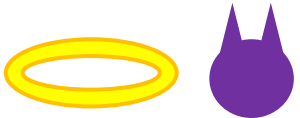
Solves a Variety of Control Challenges

Benchmark	Problem Type	Ours	CESAR ^[1]	NYCS ^[2]
Event ETCS	Event-triggered control	23s	∞	∞
Infinite Track	Infinite horizon switching	233s	∞	∞
Surgical Robot	Nonlinear control	168s	∞	∞
Highway Driving	Time-triggered control in presence of adversarial agent	91s	∞	∞
Reachavoid Robot	Reachavoid problem	21s	∞	∞

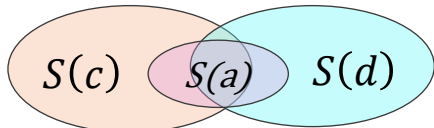
Solution Ordered by Permissiveness



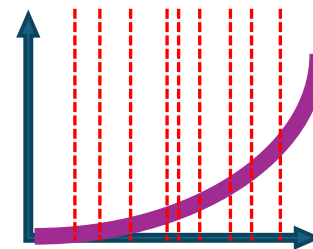
Games Specify Envelopes



Subvalue Maps represent envelopes compositionally



Synthesis heuristics close the gap



Synthesis by Precondition Calculation

