製造業の現場と自動運転の未来を支えるソフトウェア技術

研究成果及び技術の概要紹介

蓮尾 一郎

国立情報学研究所 アーキテクチャ科学研究系 教授
同　数理的高信頼ソフトウェアシステム研究センター長

総合研究大学院大学 情報学専攻

JST ERATO 蓮尾メタ数理システムデザインプロジェクト 研究総括

自己紹介

- 数学 ＞ 数学基礎論・応用数学 ＞ 数理論理学
- 数学 ＞ 代数学 ＞ 圏論
- 情報学 ＞ ソフトウェア ＞ ソフトウェア科学・形式検証・数理論理学

- 東大 数学（学士）
  ＞ 東工大 数理・計算 （修士）
  ＞ Radboud U. Nijmegen 計算機科学 （PhD）
  ＞ 京大数理研 （助教）
  ＞ 東大 コンピュータ科学専攻 （講師・准教授）
  ＞ 国立情報研 （准教授・教授）

- さきがけ 西浦領域（2007-2011）

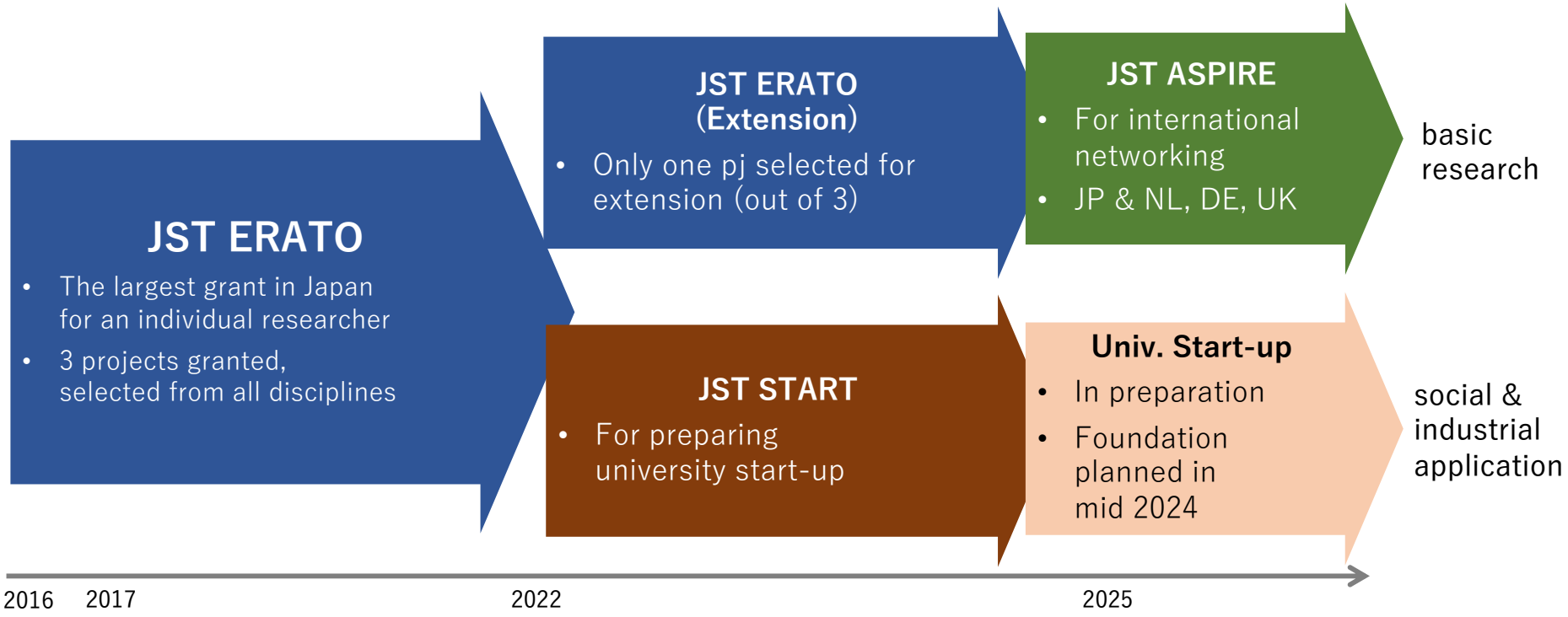
共著書： 圏論の歩き方
（日本評論社，2015）

研究チーム紹介

- **国立情報学研究所 数理的高信頼ソフトウェアシステム研究センター：**
  ソフトウェア分野の包括的研究グループ
  - 研究員 7
  - 博士課程学生 6
  - プログラマー 2

- JST funding
  - 基礎研究
  - 国際展開
  - 産業化支援

- 数学のチカラを社会に示す
  - 数学基礎論
  - 代数学



NII MathTrust

自動運転・生成AIなどの新情報技術は「侵襲的」
社会受容を支える技術で，人間中心の情報化社会へ

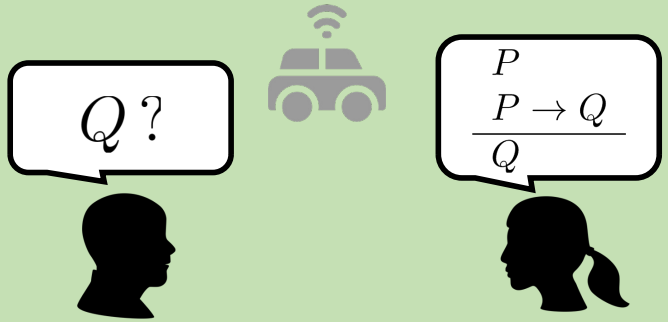
ブラックボックス「安全性」

- 中身の見えない「安全性主張」
- 精査・批判的検討・改良・責任所在の議論が困難

⇑ 現状の多く

VS

説明可能安全性



- 説明可能でトレーサブルな論理的安全性主張
- 実世界の安全性保証は終わりのない営為．対話による社会全体の取り組みをサポート
  → 情報技術への社会的信頼，社会受容
  → 本格展開，真の情報化社会の実現

⇑ **ソフトウェア科学・論理学を使い我々の追求する未来像**

NII MathTrust

— 新情報技術の社会受容を可能にする数学 —

- 2024年6月19日（水）11:30～20:30
  （受付開始 11:00）
- 参加登録締め切り： 今日
- 「蓮尾 erato シンポジウム」で検索

Software Science in Manufacturing Industry: Historical Perspective & Overview

<u>Formal methods for (conventional) software</u>
(e.g. model checking)

Goal:
to ***prove*** $\mathcal{M} \vDash \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Software Science in Manufacturing Industry: Historical Perspective & Overview

Formal methods for (conventional) software
(e.g. model checking)

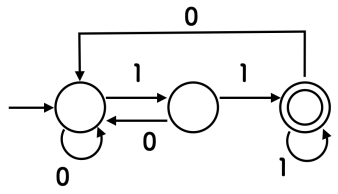
Goal:
to ***prove*** $\mathcal{M} \vDash \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Heavy-weight formal methods for manufacturing

Goal:
to ***prove*** $\mathcal{M} \vDash \varphi$

- $\mathcal{M}$ is a car
  φ is safety/quality/…
- But we need a mathematical/logical model $\mathcal{M}$ of a car!



Modeling is costly

- MBD usually doesn't help…
  Simulink is for numeric simulation,
  not principally for logical verification

Software Science in Manufacturing Industry: Historical Perspective & Overview

Formal methods for (conventional) software
(e.g. model checking)

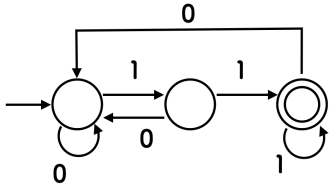
Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Heavy-weight formal methods for manufacturing

Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- $\mathcal{M}$ is a car
  φ is safety/quality/…
- But we need a mathematical/logical model $\mathcal{M}$ of a car!



Modeling is costly
- MBD usually doesn't help
- Simulink is for numeric simulation, not principally for logical verification

Light-weight formal methods
→ *Manufacturing DX*

- Formal/math./software description of φ
- Feasible yet powerful

Proof-based *mathematical traffic laws* for automated driving

- For int'l safety standards, a sound ecosystem, & social acceptance

Software Science in Manufacturing Industry: Historical Perspective & Overview

Formal methods for (conventional) software
(e.g. model checking)

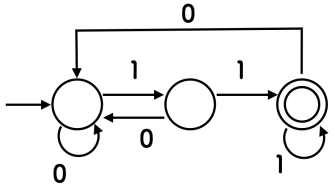
Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Heavy-weight formal methods for manufacturing

Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- $\mathcal{M}$ is a car
  φ is safety/quality/…
- But we need a mathematical/logical model $\mathcal{M}$ of a car!



Modeling is costly
- MBD usually doesn't help
- Simulink is for numeric simulation, not principally for logical verification

Part 1

Light-weight formal methods
→ *Manufacturing DX*

- Formal/math./software description of φ
- Feasible yet powerful

Part 2

Proof-based *mathematical traffic laws* for automated driving

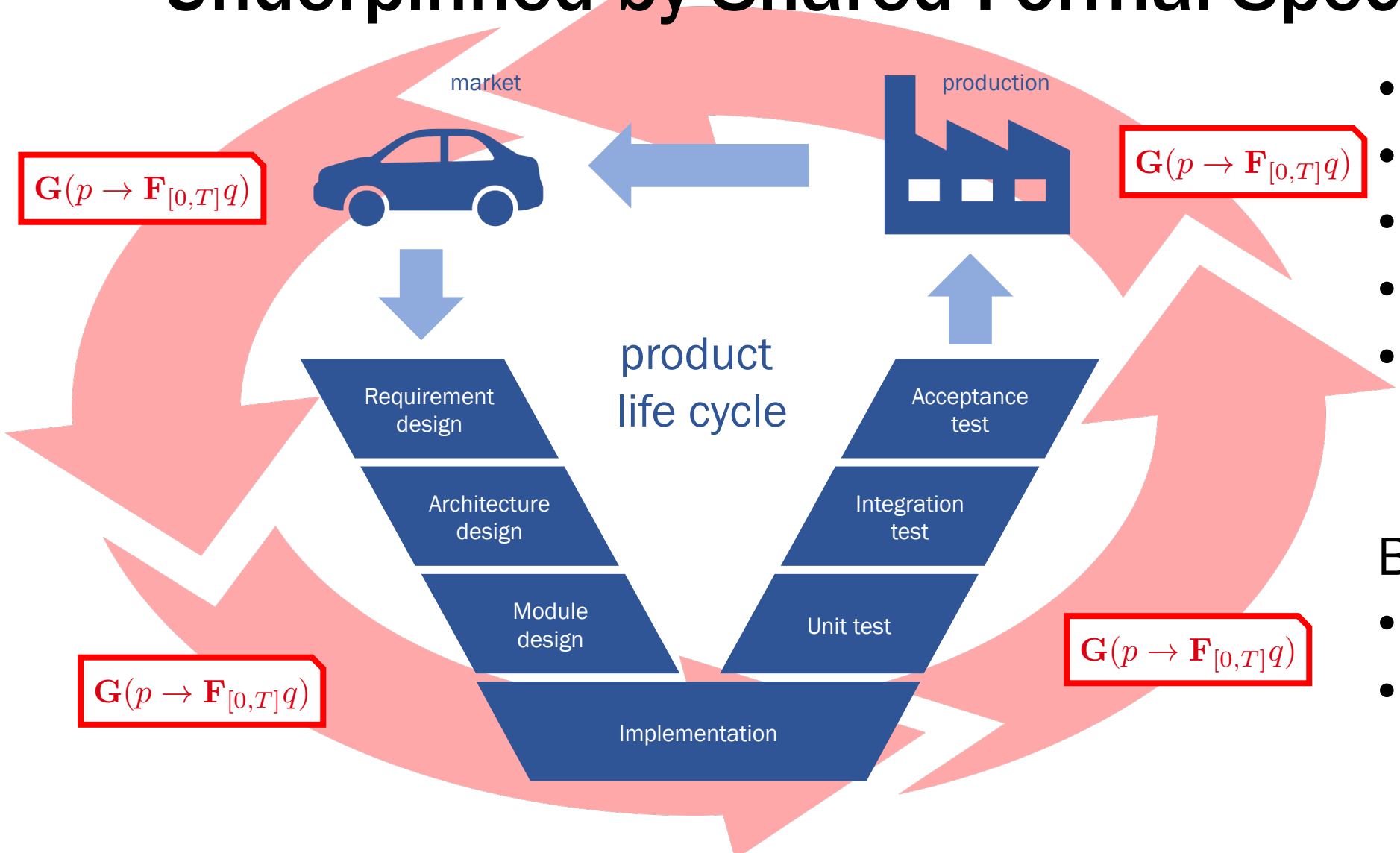
- For int'l safety standards, a sound ecosystem, & social acceptance

Part 1

Specification Driven Engineering: Its Concept and Tool Support

Collaboration w/ Mitsubishi Electric, Mitsubishi Heavy Industry, TOYOTA, …

Design, V&V, Production, and Deployment—Underpinned by Shared Formal Specifications



- automation
- inter-operability
- communicability
- traceability
- responsibility

Benefits…

- within each phase
- in connecting different phases

Rigorous and Unambiguous Definition of "What Is Needed"

Formal specification

$$\mathbf{G}(p \rightarrow \mathbf{F}_{[0,T]} q)$$

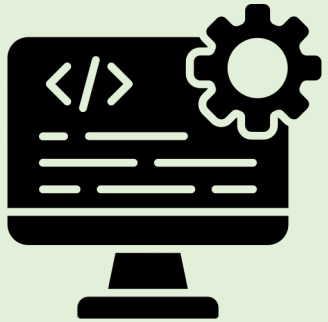
written in a formal language such as STL

$$\varphi :\equiv \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{F}_I \varphi \mid \varphi_1 \mathbf{U}_I \varphi_2$$

1. Its meaning is mathematically defined.
   No ambiguity, no dispute

$$\begin{aligned}
&\sigma \models \alpha \iff f(\sigma(0)) \geq 0 \qquad (\sigma \models \bot \text{ never holds}) \\
&\sigma \models \neg\varphi \iff \sigma \not\models \varphi \\
&\sigma \models \varphi_1 \wedge \varphi_2 \iff \sigma \models \varphi_1 \text{ and } \sigma \models \varphi_2 \\
&\sigma \models \varphi_1 \mathcal{U}_I \varphi_2 \iff \exists t \in I.\, (\sigma^t \models \varphi_2 \wedge \forall t' \in [0,t).\, \sigma^{t'} \models \varphi_1) \\
&\sigma \models \varphi_1 \mathcal{R}_I \varphi_2 \iff \forall t \in I.\, (\sigma^t \not\models \varphi_2 \Rightarrow \exists t' \in [0,t).\, \sigma^{t'} \models \varphi_1)
\end{aligned}$$

2. Machine processable.
   Write, *record*, *inspect*, and *use* it in software



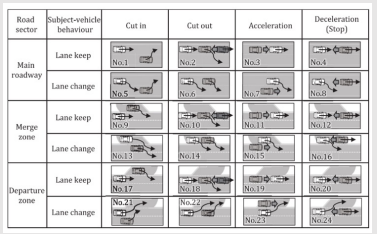
Spec: "What is Needed"
Model: "How Spec is Realized"

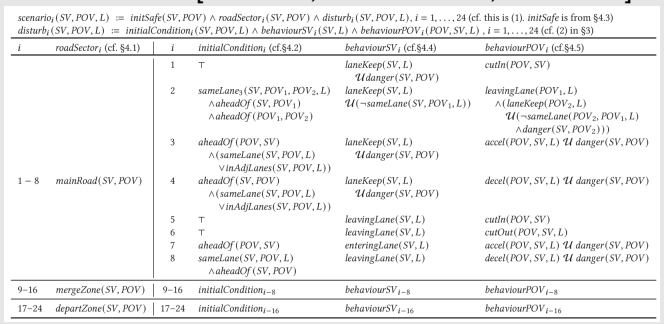
Specs:

$$G(gear = 1 \wedge rpm > 3500 \Rightarrow F_{[0,1]}\, gear = 2)$$

"When gear is 1st and rpm is > 3500, then gear will be 2nd within 1 sec."

Disturbance scenarios of ISO 34502 [Reimann, Mansion et al., SAC 2024]



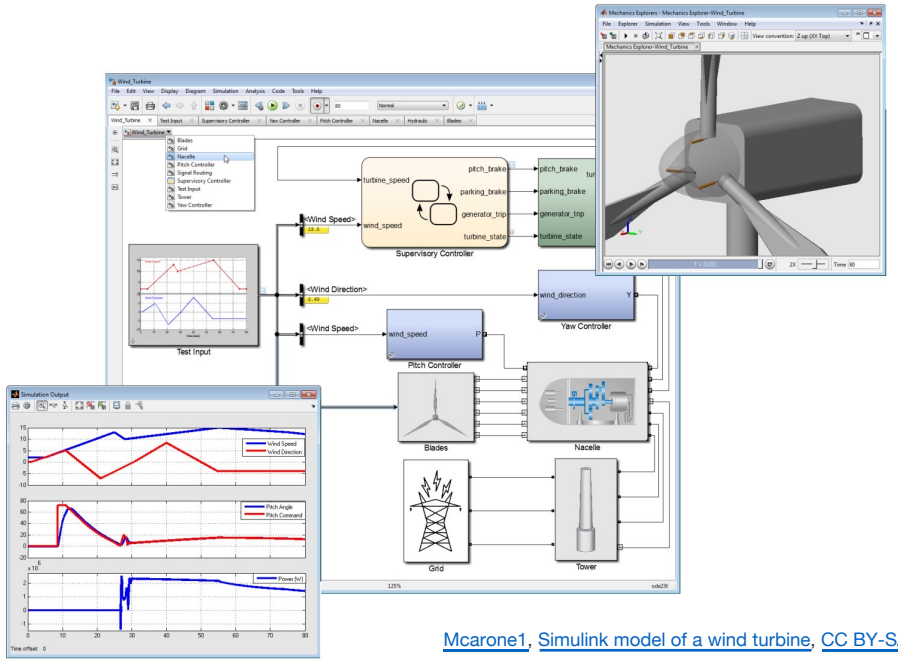


$scenario_i(SV, POV, L) := initSafe(SV, POV) \wedge roadSector_i(SV, POV) \wedge disturb_i(SV, POV, L),\ i = 1, \ldots, 24$ (cf. this is (1). *initSafe* is from §4.3)
$disturb_i(SV, POV, L) := initialCondition_i(SV, POV, L) \wedge behaviourSV_i(SV, L) \wedge behaviourPOV_i(POV, SV, L)\ ,\ i = 1, \ldots, 24$ (cf. (2) in §3)

i	$roadSector_i$ (cf. §4.1)	i	$initialCondition_i$ (cf.§4.2)	$behaviourSV_i$ (cf.§4.4)	$behaviourPOV_i$ (cf.§4.5)
1 – 8	$mainRoad(SV, POV)$	1	$\top$	$laneKeep(SV, L)$ $\mathcal{U} danger(SV, POV)$	$cutIn(POV, SV)$
		2	$sameLane_3(SV, POV_1, POV_2, L)$ $\wedge aheadOf(SV, POV_1)$ $\wedge aheadOf(POV_1, POV_2)$	$laneKeep(SV, L)$ $\mathcal{U}(\neg sameLane(SV, POV_1, L))$	$leavingLane(POV_1, L)$ $\wedge (laneKeep(POV_2, L)$ $\mathcal{U}(\neg sameLane(POV_2, POV_1, L)$ $\wedge danger(SV, POV_2)))$
		3	$aheadOf(POV, SV)$ $\wedge (sameLane(SV, POV, L)$ $\vee inAdjLanes(SV, POV, L))$	$laneKeep(SV, L)$ $\mathcal{U} danger(SV, POV)$	$accel(POV, SV, L)\ \mathcal{U}\ danger(SV, POV)$
		4	$aheadOf(SV, POV)$ $\wedge (sameLane(SV, POV, L)$ $\vee inAdjLanes(SV, POV, L))$	$laneKeep(SV, L)$ $\mathcal{U} danger(SV, POV)$	$decel(POV, SV, L)\ \mathcal{U}\ danger(SV, POV)$
		5	$\top$	$leavingLane(SV, L)$	$cutIn(POV, SV)$
		6	$\top$	$leavingLane(SV, L)$	$cutOut(POV, SV, L)$
		7	$aheadOf(POV, SV)$	$enteringLane(SV, L)$	$accel(POV, SV, L)\ \mathcal{U}\ danger(SV, POV)$
		8	$sameLane(SV, POV, L)$ $\wedge aheadOf(SV, POV)$	$leavingLane(SV, L)$	$decel(POV, SV, L)\ \mathcal{U}\ danger(SV, POV)$
9–16	$mergeZone(SV, POV)$	9–16	$initialCondition_{i-8}$	$behaviourSV_{i-8}$	$behaviourPOV_{i-8}$
17–24	$departZone(SV, POV)$	17–24	$initialCondition_{i-16}$	$behaviourSV_{i-16}$	$behaviourPOV_{i-16}$

- In temporal logic, state machines, etc.
- One-line (top) to ~50 lines (bottom)

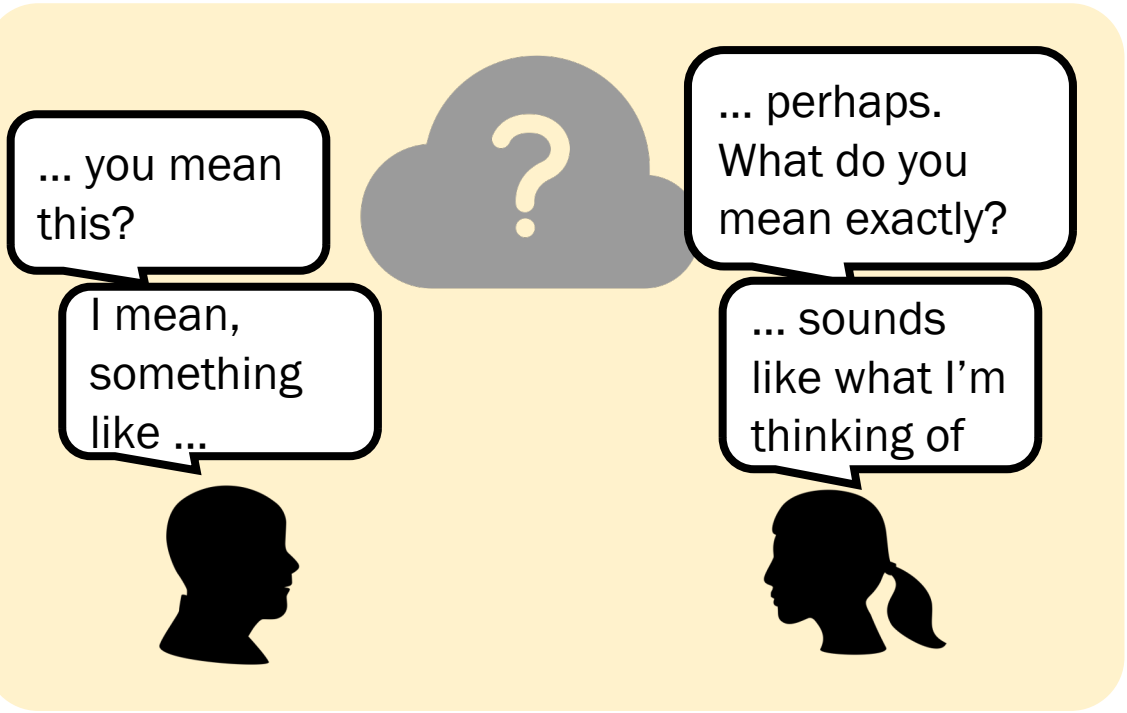
Models:

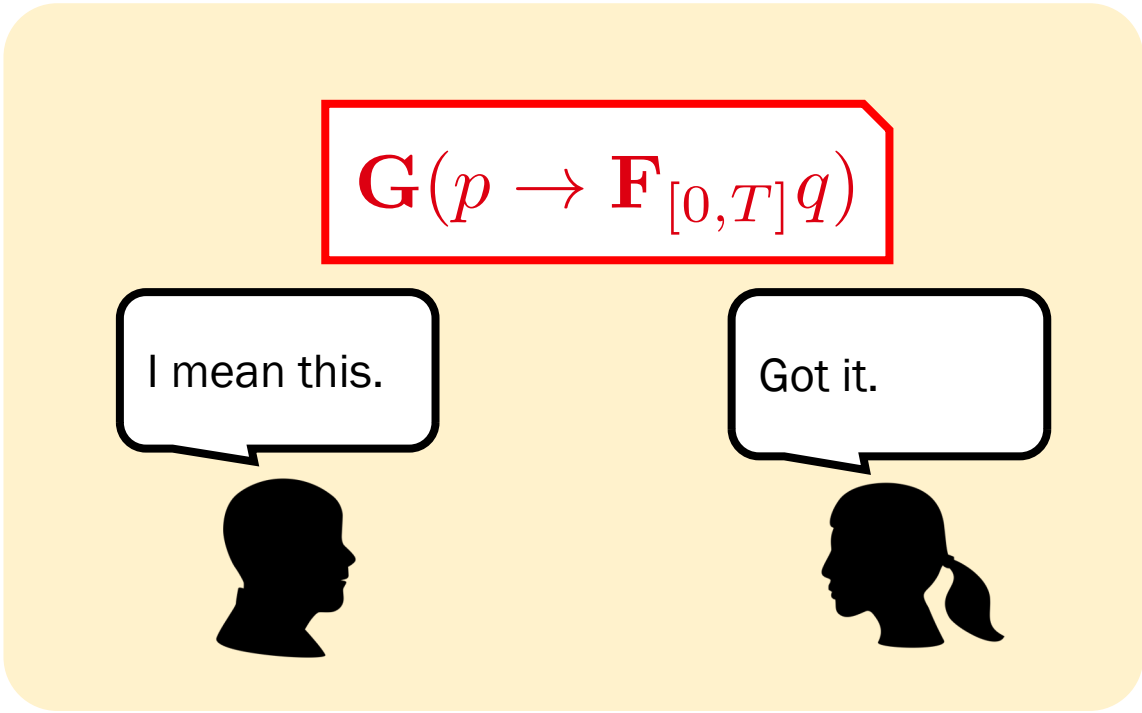


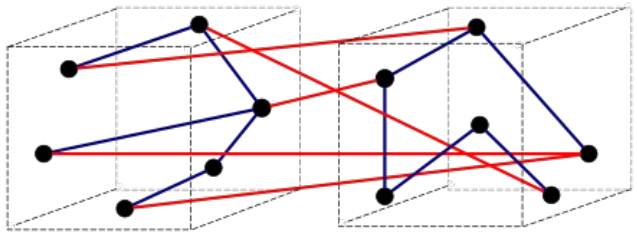
Mcarone1, Simulink model of a wind turbine, CC BY-SA 3.0

- In Simulink, Modelica, MapleSim, …
- Takes time and cost to build (months)

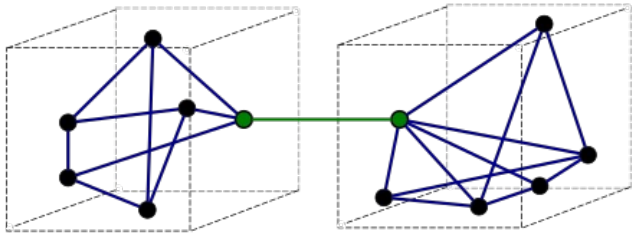
Communication on Solid and Unambiguous Bases







b) Bad (high coupling, low cohesion)

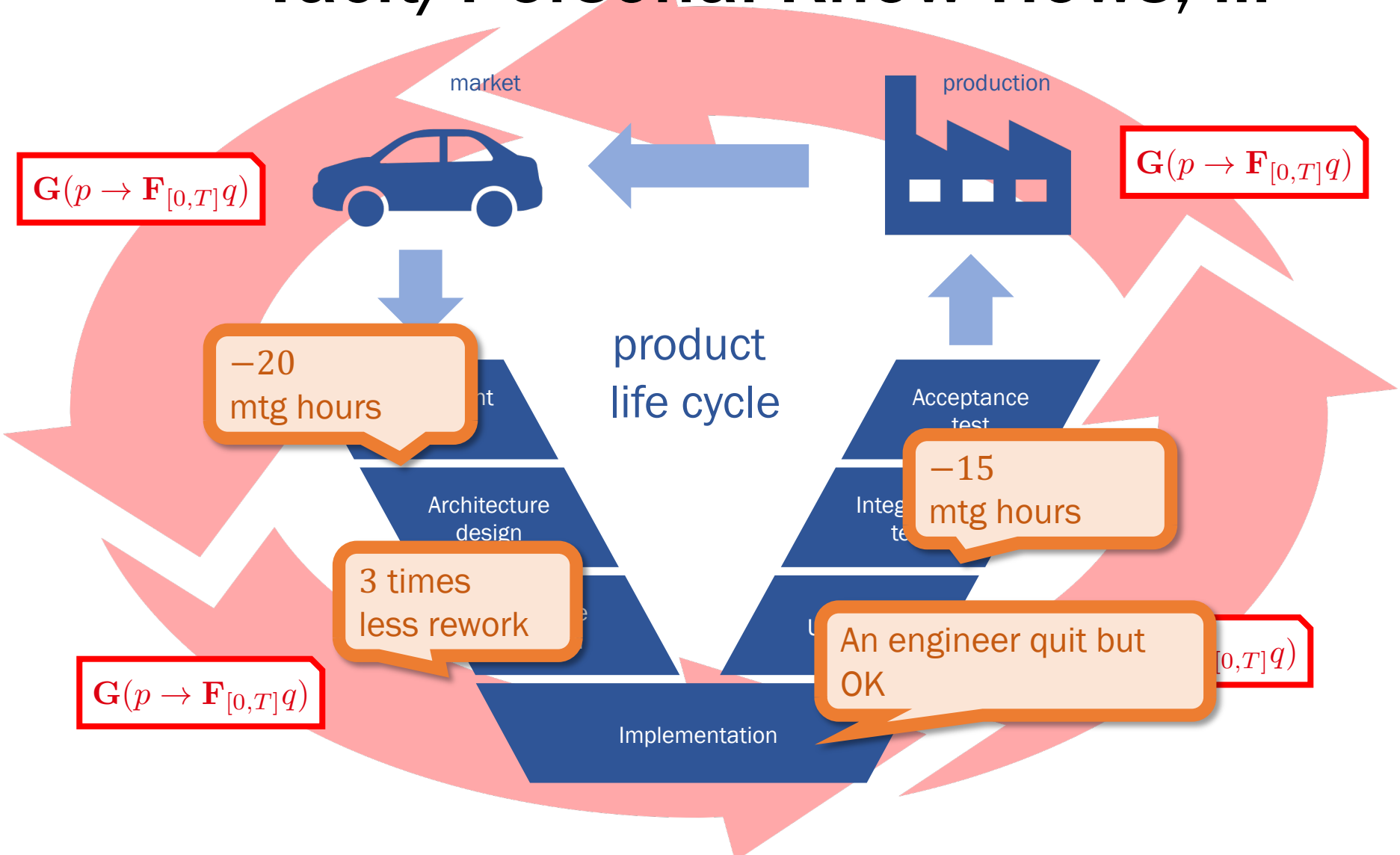


a) Good (loose coupling, high cohesion)

"Low Coupling, High Cohesion"

https://commons.wikimedia.org/wiki/File:CouplingVsCohesion.svg

Reducing Meetings, Reworks, Tacit/Personal Know-Hows, …

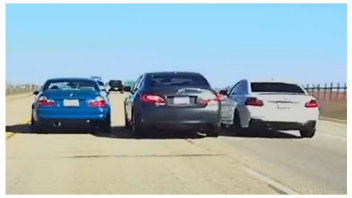


Automating Daily Analysis Tasks, Reducing Person-Hours

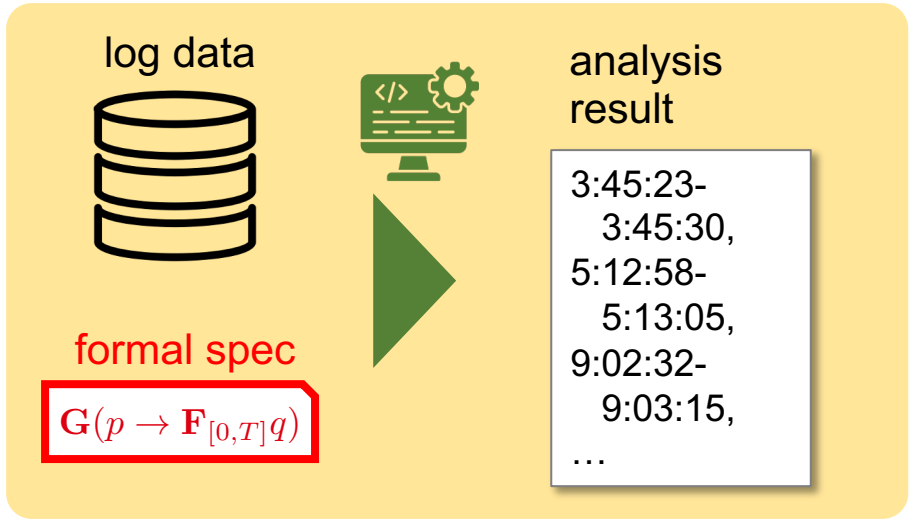
Task example: **log extraction**

Collab. w/ MITSUBISHI ELECTRIC

[Reimann, Mansion, et al., SAC'24]



Where in the log does a dangerous passing occur?



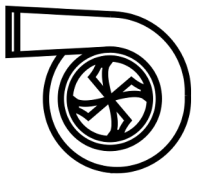
BEFORE:
- Tens of hours of manual inspection or ad-hoc C-coding

AFTER: w/ our algorithm
- Automated & fast analysis (100K events/sec.)
- User effort is mininal (writing a logical formula, < 10 lines)

Task example: **parameter optimization**

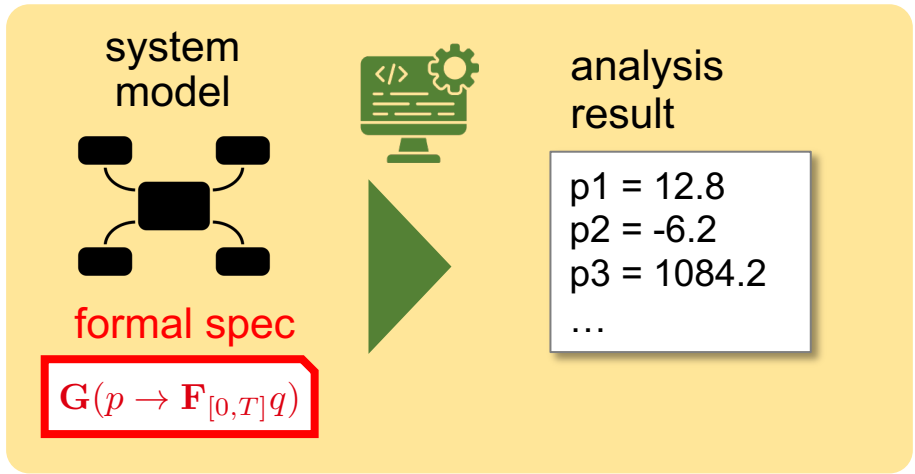
Collab. w/ MITSUBISHI HEAVY INDUSTRIES

[Sato+, FM'21]



gas turbine for power plants

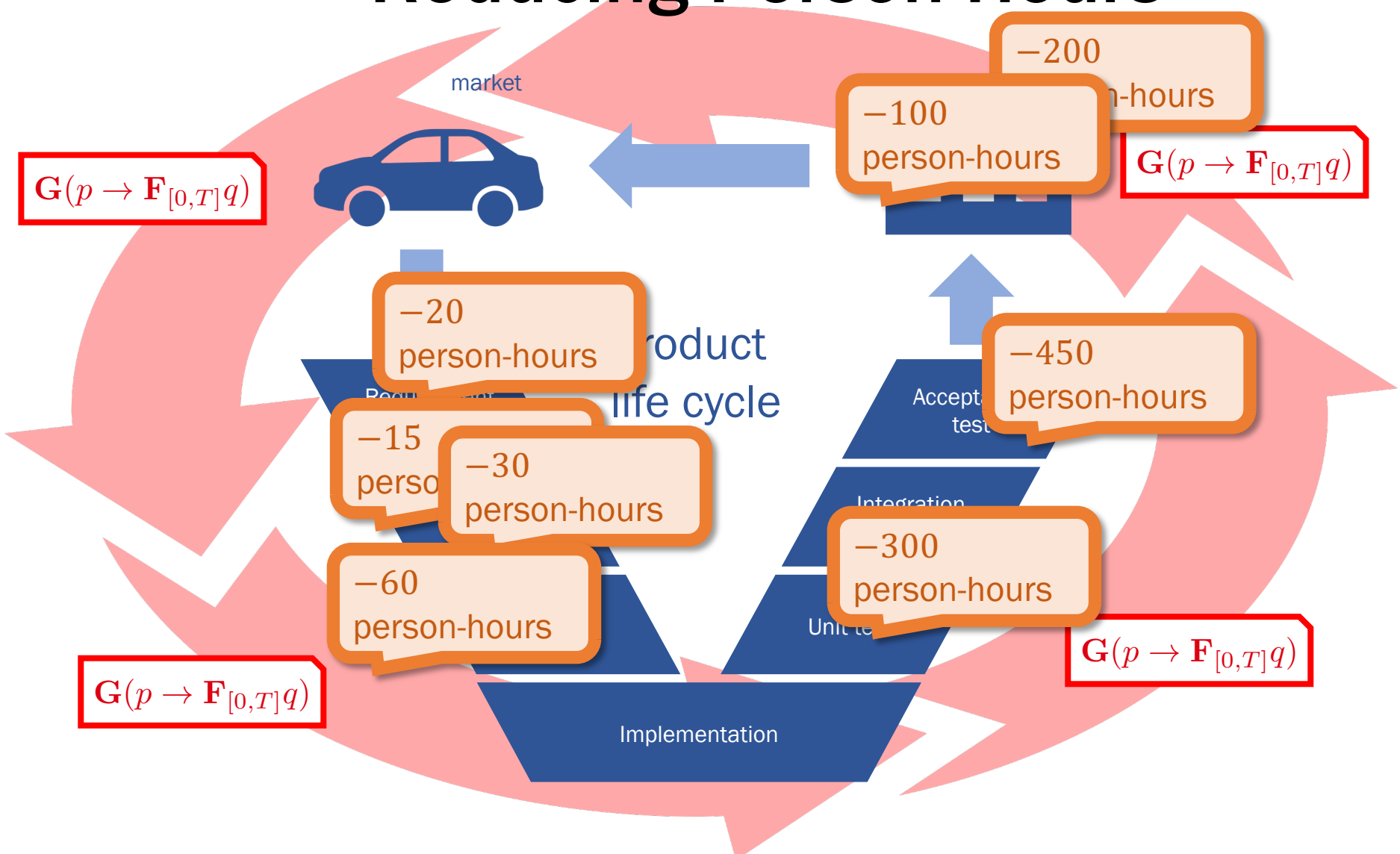
Need to tune parameter values for safety and efficiency



BEFORE:
- Manual efforts by experts (7 man-days in [Sato+, FM'21])

AFTER: w/ our algorithm
- Automated & fast analysis (3 hrs in [Sato+, FM'21])
- Better answer

Automating Daily Analysis Tasks, Reducing Person-Hours



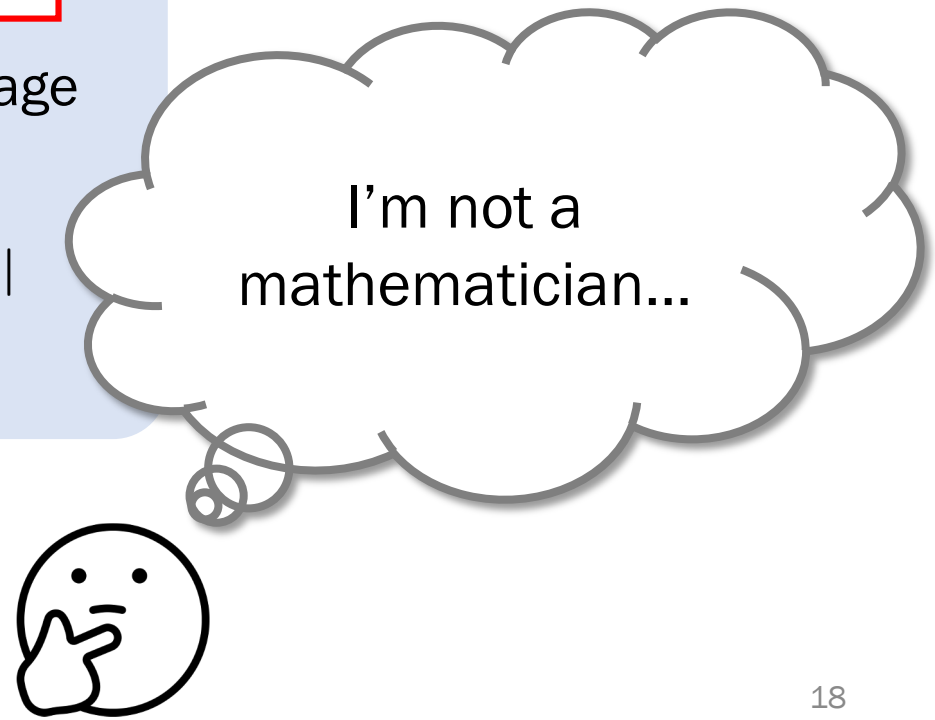
Writing and Managing Formal Specs Is Not Easy

Formal specification

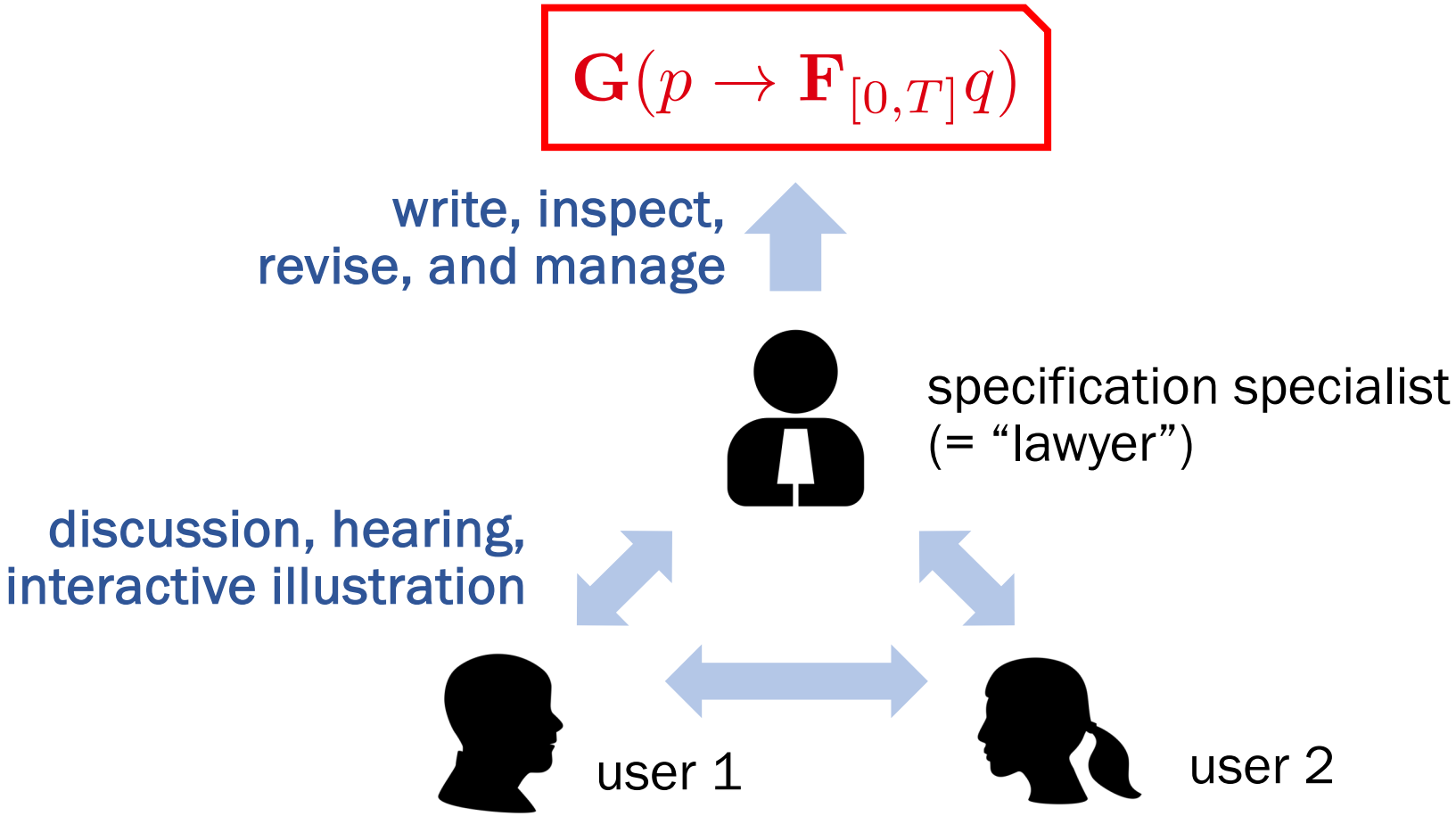
$$\mathbf{G}(p \rightarrow \mathbf{F}_{[0,T]} q)$$

written in a formal language such as STL

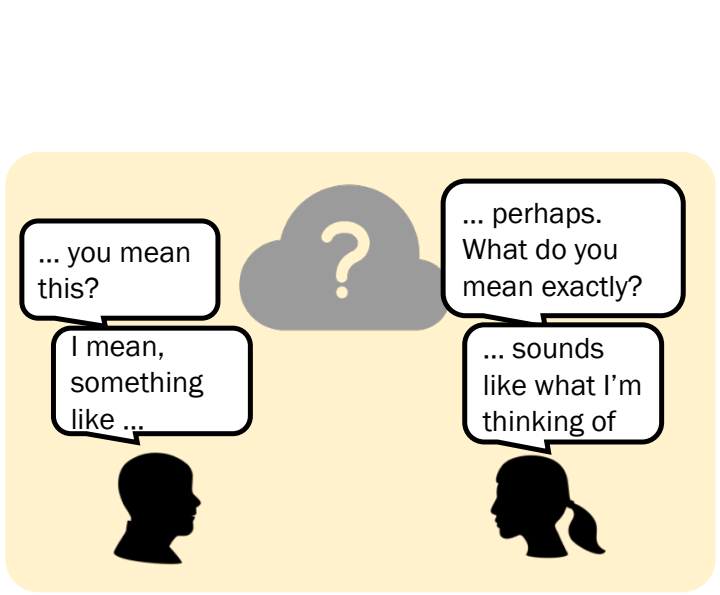
$$\varphi :\equiv \alpha \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \mathbf{F}_I \varphi \mid \varphi_1 \mathbf{U}_I \varphi_2$$



The *Lawyer Model* for Formal Specifications

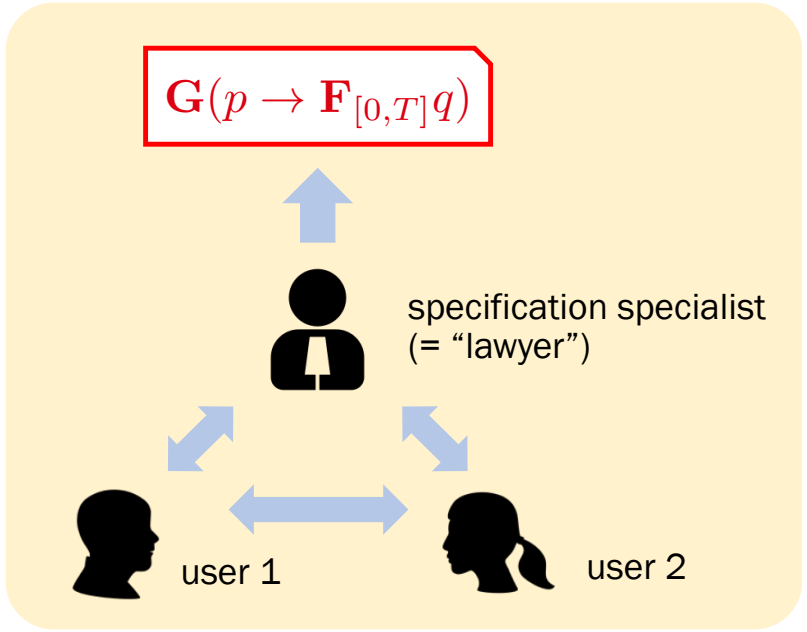


The *Lawyer Model* for Formal Specifications



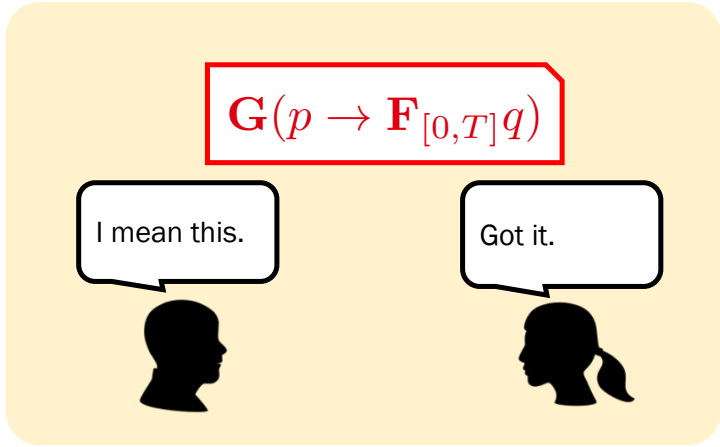
Current practice

expensive



The lawyer model
(our immediate goal)

admissible comm. cost
and feasible now



Communication
in formal specs
(our ultimate goal)

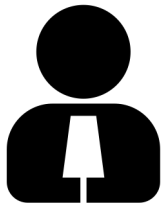
cheap comm. cost but
learning cost is needed

The *Lawyer Model* for Formal Specifications and Its Software Support

It does cost, but history points to this

$$\mathbf{G}(p \rightarrow \mathbf{F}_{[0,T]}q)$$

write, inspect, revise, and manage

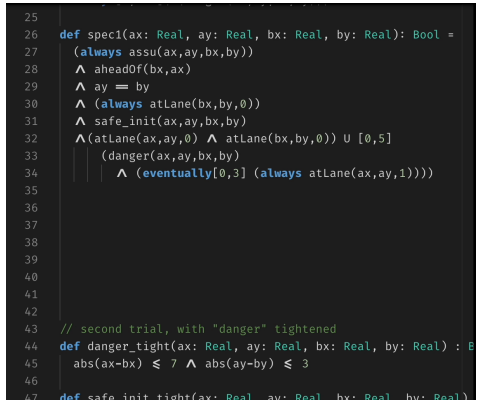


specification specialist (= “lawyer”)

discussion, hearing, interactive illustration

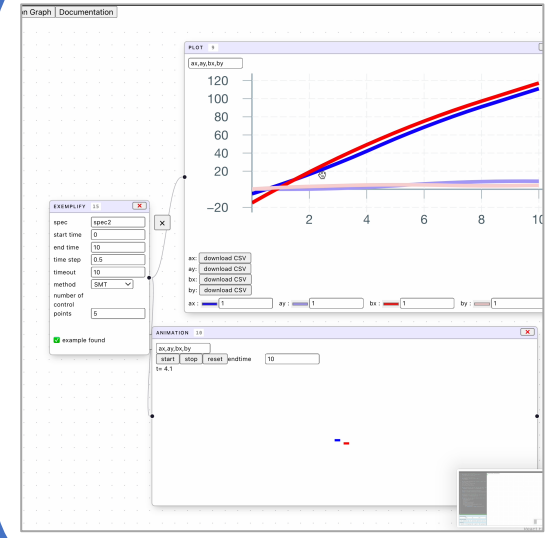
user 1

user 2



spec editor

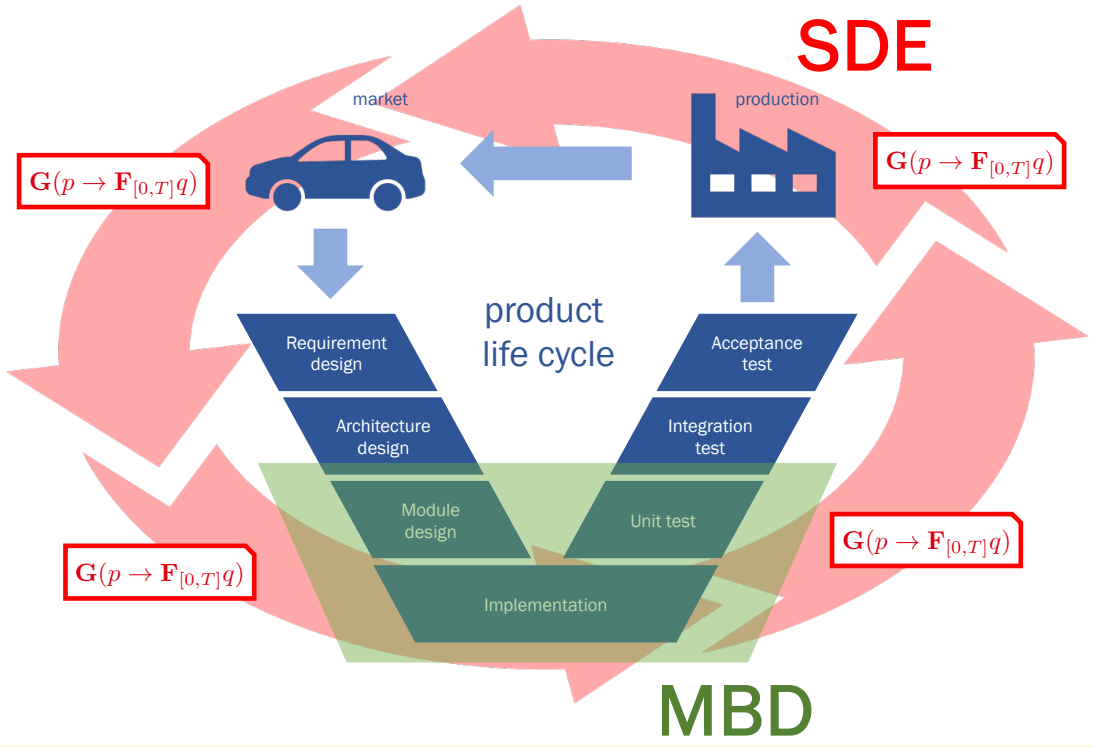
(IDE for formal specs)



interactive spec illustrator

(visual illustration of formal specs)

Mutually Complementary Paradigms towards Responsible & Efficient Manufacturing



V&V asks

mechanized in MBD ⇓

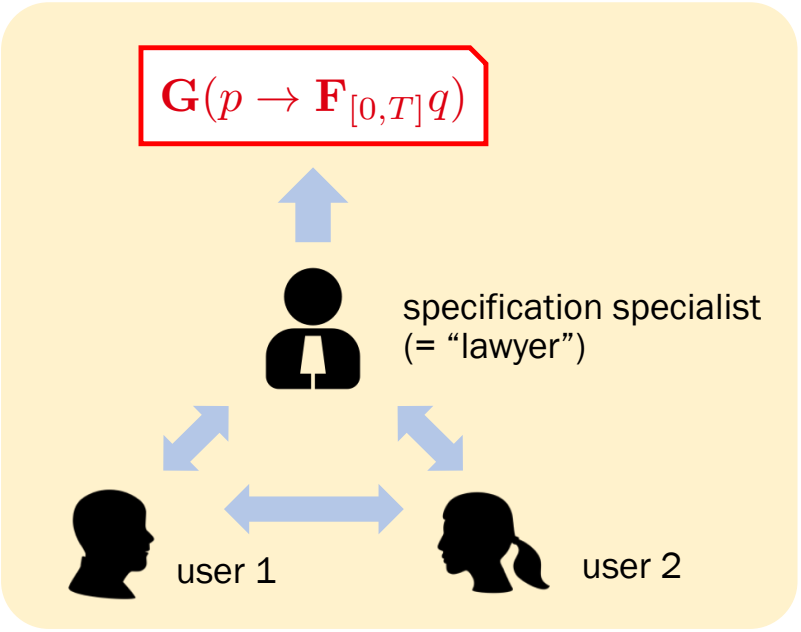
mechanized in SDE ⇓

if a system (model) M satisfies a spec φ

	MBD	SDE
target phases	design, impl., V&V	the whole product life cycle
requirement	system model (costly)	formal spec (cheaper)
advantage	model ➔ V&V ➔ impl. (eliminating rework)	• V&V automation • communicability • traceability

Part 1 (Specification-Driven Engineering) Summary

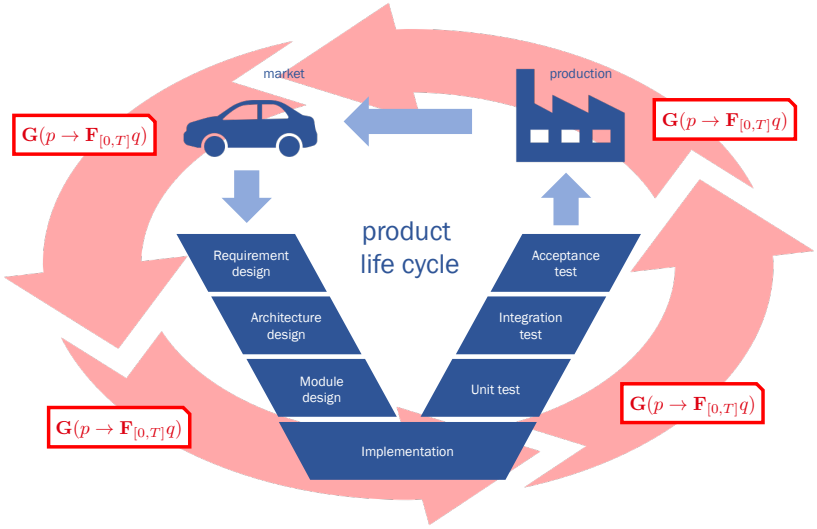
The SDE workflow—
supported by tools from software science!

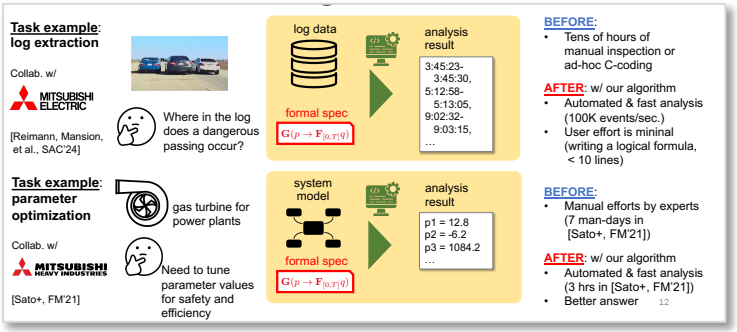


Write, examine, correct, and agree on **formal specs**,
with the help of spec specialist

$$\mathbf{G}(p \rightarrow \mathbf{F}_{[0,T]}q)$$

A <u>formal spec</u>:
an unambiguous and machine-processable description of requirements and contracts





- automation
- inter-operability
- communicability
- traceability
- responsibility

Software Science in Manufacturing Industry: Historical Perspective & Overview

Formal methods for (conventional) software
(e.g. model checking)

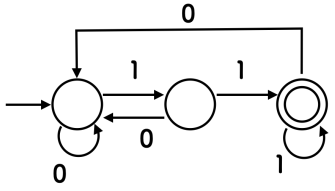
Goal:
to ***prove*** $\mathcal{M} \vDash \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Heavy-weight formal methods for manufacturing

Goal:
to ***prove*** $\mathcal{M} \vDash \varphi$

- $\mathcal{M}$ is a car
  φ is safety/quality/…
- But we need a mathematical/logical model $\mathcal{M}$ of a car!



Modeling is costly
- MBD usually doesn't help
- Simulink is for numeric simulation, not for logical verification

Part 1

Light-weight formal methods
→ *Manufacturing DX*

- Formal/math./software description of φ
- Feasible yet powerful

Part 2

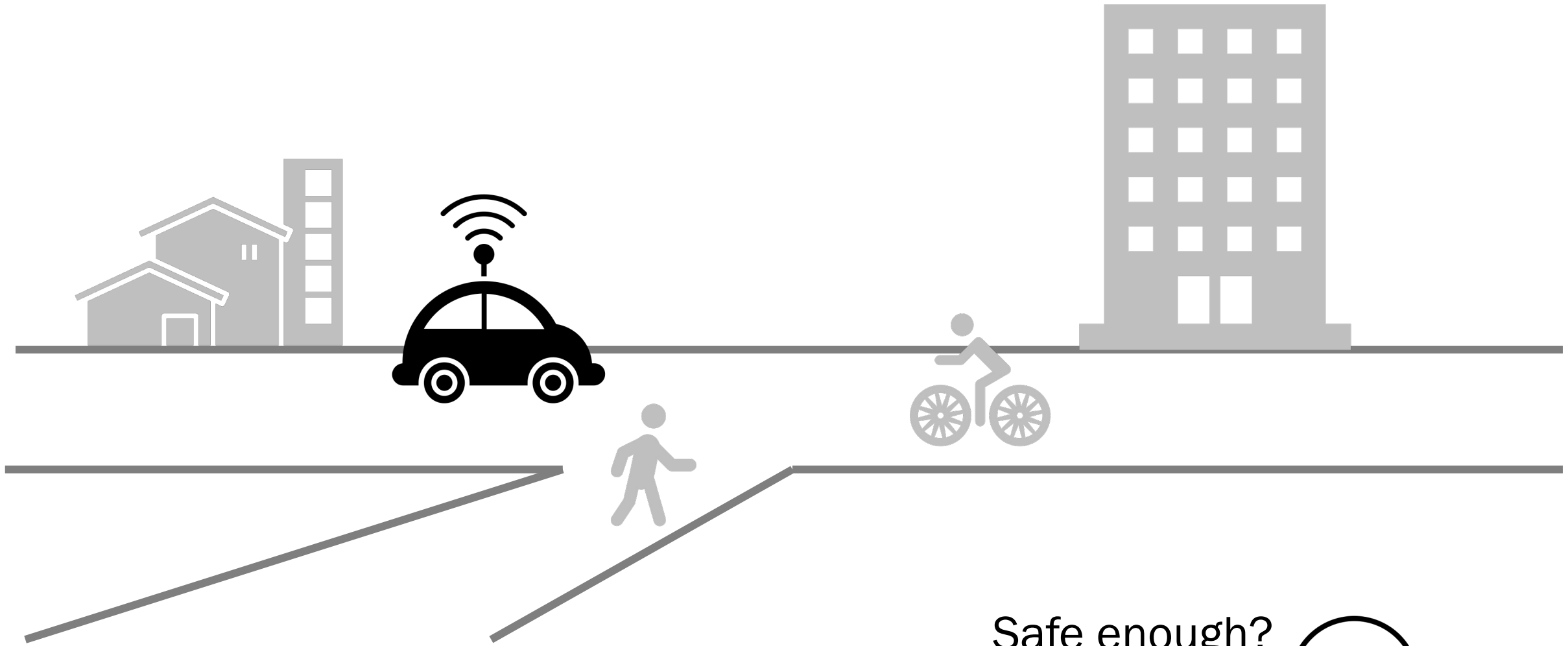
Proof-based *mathematical traffic laws* for automated driving

- For int'l safety standards, a sound ecosystem, & social acceptance

Part 2

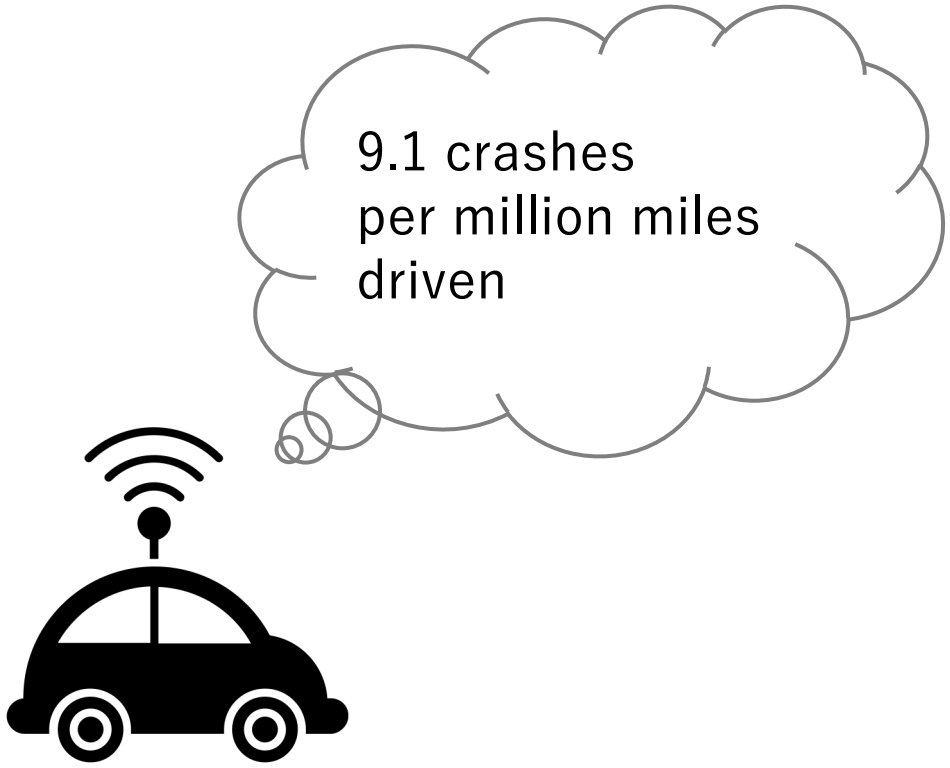
Proof-based *Mathematical Traffic Laws* for Automated Driving

Collaboration w/ Mazda, …
Standardization efforts at IEEE SA, …



Safe enough?

Guarantee
by statistical data



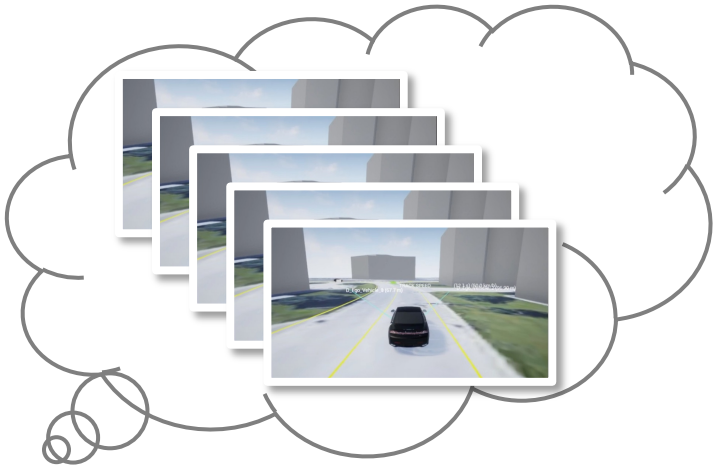
Guarantee
by testing and simulation

Guarantee strong enough?

Guarantee by statistical data

9.1 crashes
per million miles
driven

Guarantee by testing and simulation



Explainability?

Proof.

We prove the first statement. The rest is shown symm

Let $S \subseteq L$ be an arbitrary subset. We let $S^{\downarrow}$ be th
that is,

$$S^{\downarrow} \coloneqq \{y \in L \mid y \sqsubseteq s \text{ for each } s \in$$

Since $S^{\downarrow} \subseteq L$ is a subset of L, it has its supremum $\bigsqcup S^{\downarrow}$ in the
semilattice $(L, \sqsubseteq)$. We claim that $\bigsqcup S^{\downarrow}$ is the infimum

To prove the claim, it suffices to show the two-way
acterization in (2.1), that is, we need to show

$$\frac{y \sqsubseteq s \quad \text{for each } s \in S}{y \sqsubseteq \bigsqcup S^{\downarrow}}.$$

For the downward implication in **??**,

$$\begin{aligned} & y \sqsubseteq s \quad \text{for each } s \in S \\ \Longrightarrow \quad & y \in S^{\downarrow} && \text{by def. of } S^{\downarrow} \\ \Longrightarrow \quad & y \sqsubseteq \bigsqcup S^{\downarrow} && \text{since } \bigsqcup S^{\downarrow} \text{ is an u} \end{aligned}$$

For the upward implication in **??**, we first observe

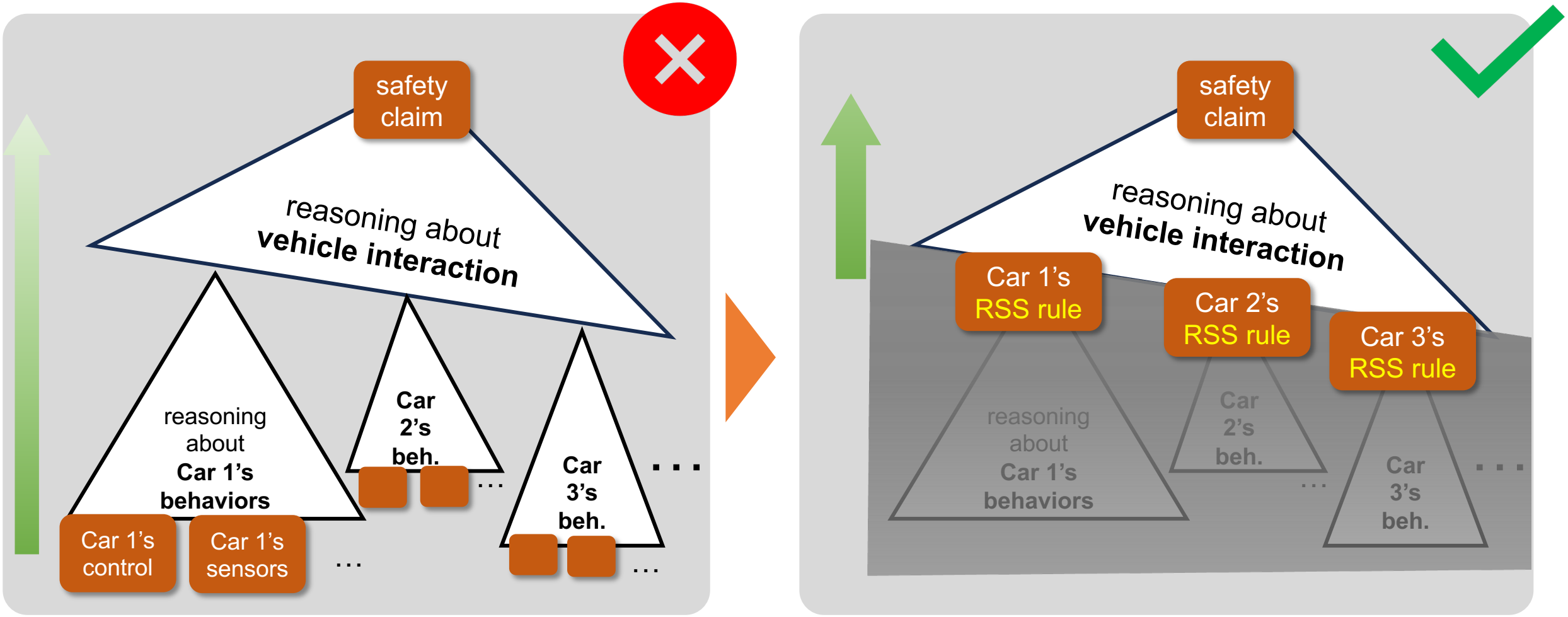
$$\bigsqcup S^{\downarrow} \sqsubseteq s \quad \text{for each } s \in S.$$

Mathematical safety proofs would certainly be great...

But are they ever feasible?

Responsibility-Sensitive Safety (RSS)

[Shalev-Shwartz et al., arXiv preprint, 2017]



- **Full safety proofs are infeasible**
  - Lack of white-box models
  - Ultimate safety claim is too far

- Ignore the internal working of individual vehicles
- Instead, impose "behavioral constracts" on them
  - Called **RSS rules**. "Mathematical traffic laws"
- Mathematical proofs assume rule compliance → feasible

RSS Rule, an Example

[Shalev-Shwartz et al., arXiv preprint, 2017]

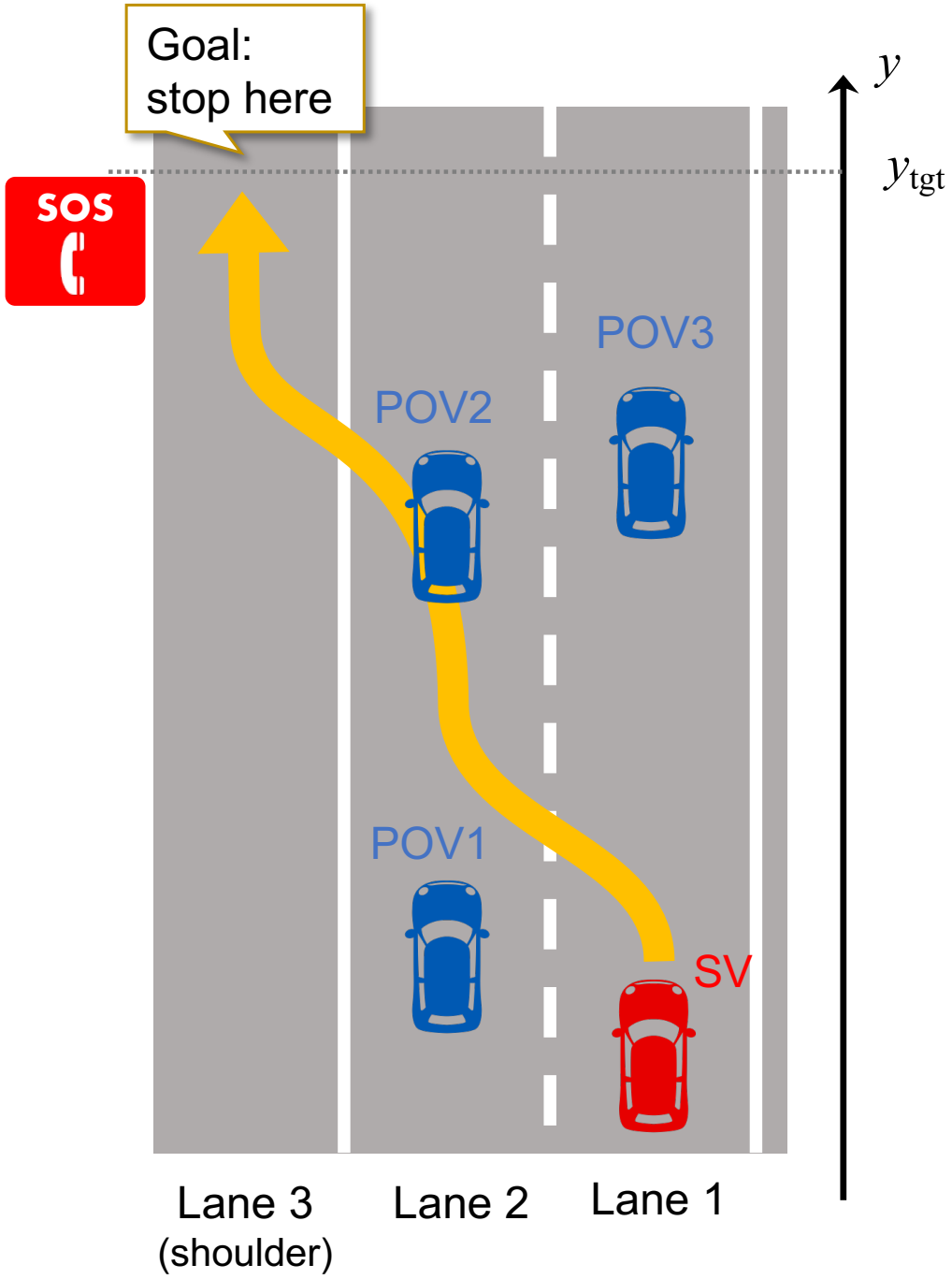
- An RSS rule is a pair (A, α) of an *RSS condition* A and a *proper response* α

RSS condition A: ("You can still escape if A is true")
Maintain an inter-vehicle distance at least

$$d_{\min} = \left[v_r\,\rho + \frac{1}{2} a_{\max,\text{accel}}\,\rho^2 + \frac{(v_r + \rho\, a_{\max,\text{accel}})^2}{2a_{\min,\text{brake}}} - \frac{v_f^2}{2a_{\max,\text{brake}}} \right]_+$$

Proper response α: ("When you escape, use the control strategy α")
Brake at rate $a_{\text{min, brake}}$ within ρ seconds

Conditional safety lemma:
Any execution of α, from a state that satisfies A, is collision-free.



- Now what about this pull over scenario?
- Essential for eyes-off ADVs to hand the control over to human drivers
- Requires complex decision making
  - Merge before POV1, or after?
  - Accelerate to pass POV1…
    - ➔ Risk of overrun?

Our Work: Logical Formalization of RSS ➔ More Scenarios (Collab. w/ Mazda)

RSS

Responsibility-Sensitive Safety,
Shalev-Shwartz et al., 2017

- Basic methodology of logical safety rules
- Standardization (IEEE 2846)
- Lack of formal implemantion
  ➔ **appl. to complex scenarios is hard**
- Guarantees only collision-freedom so far

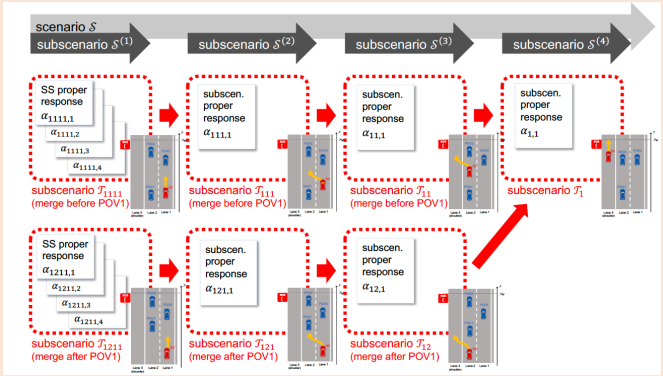
↓ Software science research

differential program logic dFHL (our contribution)

$$\frac{\begin{array}{ll}\text{inv}: & A \Rightarrow e_{\text{inv}} \sim 0 \quad e_{\text{var}} \geq 0 \wedge e_{\text{inv}} \sim 0 \Rightarrow \mathcal{L}_{\dot{\mathbf{x}}=\mathbf{f}}\, e_{\text{inv}} \simeq 0 \\ \text{var}: & A \Rightarrow e_{\text{var}} \geq 0 \quad e_{\text{var}} \geq 0 \wedge e_{\text{inv}} \sim 0 \Rightarrow \mathcal{L}_{\dot{\mathbf{x}}=\mathbf{f}}\, e_{\text{var}} \leq e_{\text{ter}} \\ \text{ter}: & A \Rightarrow e_{\text{ter}} < 0 \quad e_{\text{var}} \geq 0 \wedge e_{\text{inv}} \sim 0 \Rightarrow \mathcal{L}_{\dot{\mathbf{x}}=\mathbf{f}}\, e_{\text{ter}} \leq 0\end{array}}{\{A\}\ \text{dwhile}\,(e_{\text{var}} > 0)\,\dot{\mathbf{x}} = \mathbf{f}\ \ \{e_{\text{var}} = 0 \wedge e_{\text{inv}} \sim 0\} : e_{\text{inv}} \sim 0 \wedge e_{\text{var}} \geq 0}\ (\text{DW}_{\text{H}})$$

- A logical system for deriving and proving safety rules

Compositional rule derivation workflow by dFHL (our contribution)

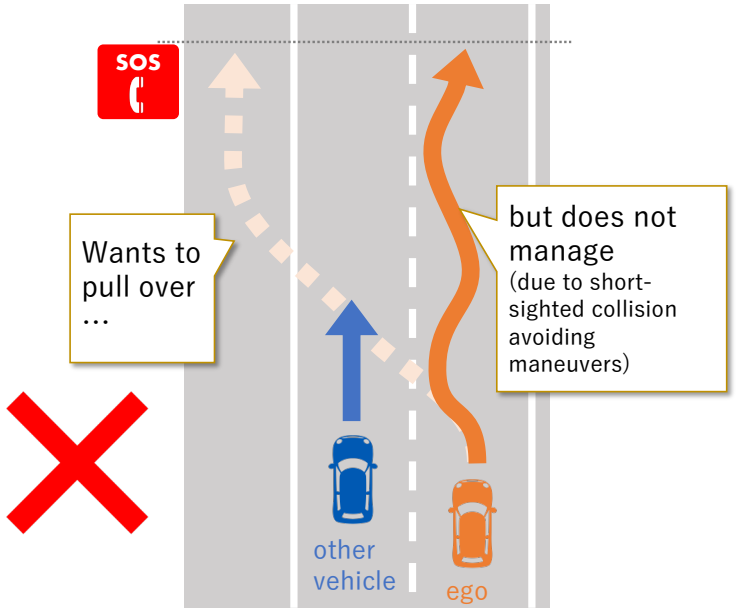


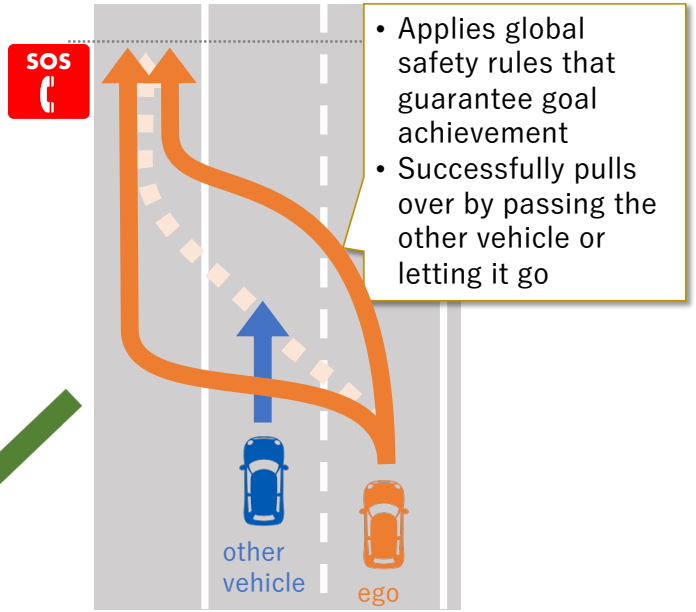
- ”Divide and Conquer” complex driving scenarios
- Tool support by autom. reasoning

GA-RSS (our contribution)

Goal-Aware
Responsibility-Sensitive Safety
[Hasuo+, IEEE T-IV, 2023]

- Guarantees **goal achievement** (e.g. successful pull over) and collision-freedom
- Global safety rules that combine mult. maneuvers
- Necessary for real-world complex driving scenarios





What is Formalization?

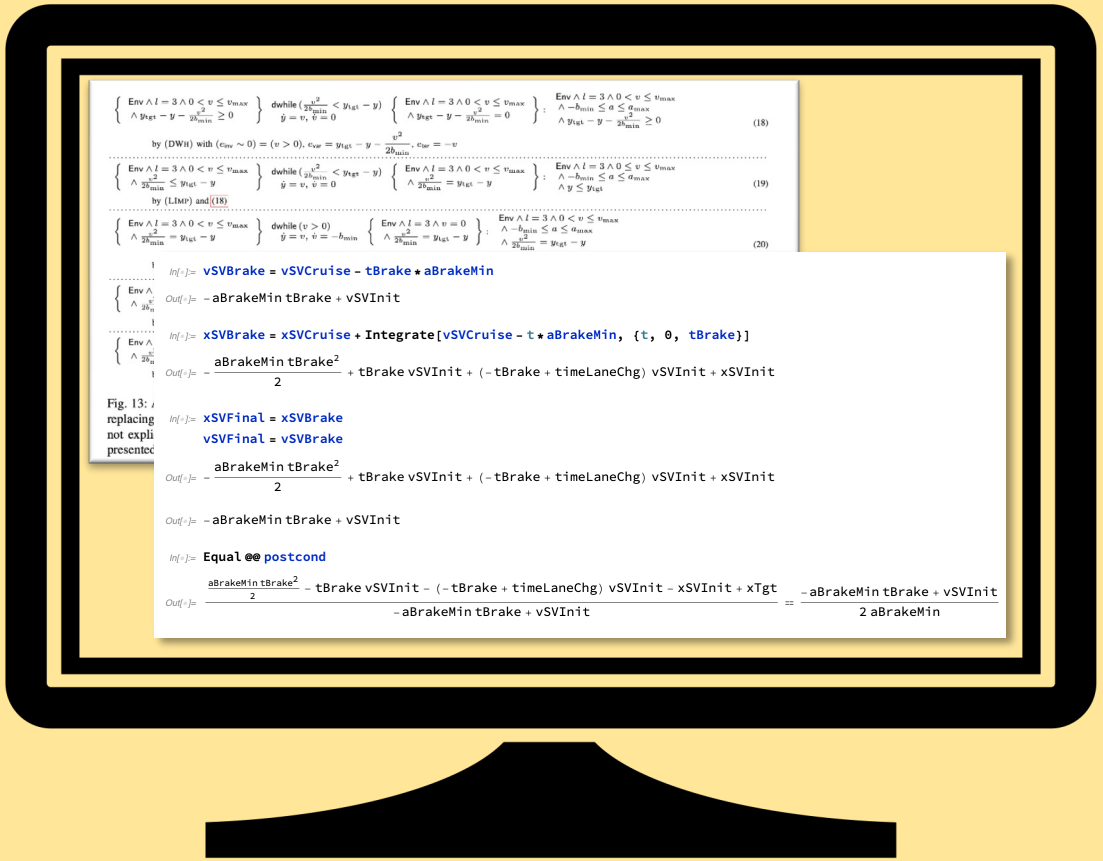
Informal

pen-and-paper proofs

- Error-prone
- Poor traceability

Formal

software-assisted proofs



- Symbolic proofs in our formal logical system dFHL
- Software tool checking the validity of each logical step of reasoning

RSS Rules as *Mathematical Traffic Laws*:
Proof-Based Ecosystem for Safe Automated Driving

- Decompose **safety** (a complex goal) into **logical safety rules** (explicit, easy to check and enforce)
- "Ultimate assurance" in the form of **mathematical proofs**. Logical explanation by following their reasoning steps
- Safety rules are generic and reusable
  → regulation, standard → social acceptance
- Attribution of liabilities
  (collision → someone must have broken the rules)

Regulation/Standardization Body

R_1
R_2
R_3
⋱

Safety Rule R_1

In the *same-lane same-direction* driving scenario,

- Maintain the safety distance

$$d_{\min} = \left[v_r\,\rho + \frac{1}{2} a_{\max,\text{accel}}\,\rho^2 + \frac{(v_r + \rho\, a_{\max,\text{accel}})^2}{2a_{\min,\text{brake}}} - \frac{v_f^2}{2a_{\max,\text{brake}}} \right]_+$$

  from the preceding car
- When that's hard, brake at acceleration $a_{\max,\text{brake}}$

Theorem (Safety)

There is no collision attributed to the ego vehicle as long as the safety rule R_1 is respected

Proof (of the safety thm.)

⋮

The only non-obvious point is that $e_{\text{inv},2}$ is preserved by the dynamics. We first observe

$$\mathcal{L}_{\delta_f,\delta_r^1} e_{\text{inv},2} = \begin{cases} 0 & \text{if } \mathsf{dRSS}_\pm(v_f, v_r, \rho - t) \geq 0 \\ v_f - v_r & \text{otherwise,} \end{cases}$$

where $\mathsf{dRSS}_\pm(v_f, v_r, \rho)$ is given by

$$\mathsf{dRSS}_\pm(v_f, v_r, \rho) = v_r\rho + \frac{a_{\max}\rho^2}{2} + \frac{(v_r + a_{\max}\rho)^2}{2b_{\min}} - \frac{v_f^2}{2b_{\max}}.$$

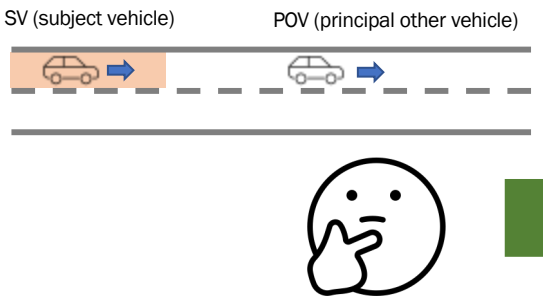
Therefore, we can infer as follows.

$$\mathsf{dRSS}_\pm(v_f, v_r, \rho - t) < 0$$
$$\iff v_r(\rho - t) + \frac{a_{\max}(\rho - t)^2}{2} + \frac{(v_r + a_{\max}(\rho - t))^2}{2b_{\min}} - \frac{v_f^2}{2b_{\max}} < 0$$

⋮

⋱

RSS Rules as *Mathematical Traffic Laws*: Proof-Based Ecosystem for Safe Automated Driving



Want an RSS rule C …
s.t. compliance with C guarantees collision-freedom

A **proof engineer** works on each driving scenario and derives an RSS rule C

- Correctness proof for C, at the same time

RSS rule C

+

Safety proof for C

Checkable by the public
…
proof-search is hard, proof-check is easy

- Attribution of liabilities
  (collision ➔ ∃ a party who did not comply)
- Safety standards
  (Show compliance before selling cars)
  e.g. IEEE P2846

→ standardization bodies, gov't, …

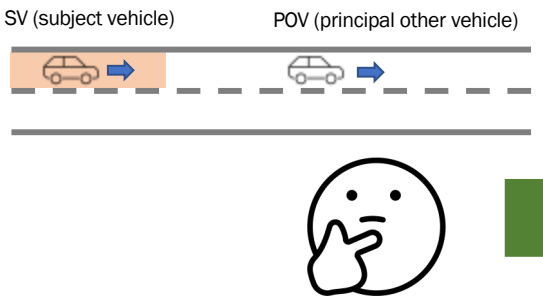
- Safety verification
  (the car complies with these rules. Thus safe)
- Runtime monitoring
  (more freedom in ADS design)

→ Car makers, ADS vendoes

- Risk analysis in insurance
  (Safety rule compliant ➔ lower risk ➔ cheaper insurance)

→ insurance companies

RSS Rules as *Mathematical Traffic Laws*: Proof-Based Ecosystem for Safe Automated Driving



Want an RSS rule C …
s.t. compliance with C guarantees collision-freedom

A **proof engineer** works on each driving scenario and derives an RSS rule C

- Correctness proof for C, at the same time

RSS rule C

+

Safety proof for C

Checkable by the public
…
proof-search is hard, proof-check is easy

- Attribution of liabilities (collision ➔ ∃ a party who did not comply)
- Safety standards (Show compliance before selling cars) e.g. IEEE P2846

→ standardization bodies, gov't, …

- Safety verification (the car complies with these rules. Thus safe)
- Runtime monitoring (more freedom in ADS design)

→ Car makers, ADS vendoes

- Risk analysis in insurance (Safety rule compliant ➔ lower risk ➔ cheaper insurance)

→ insurance companies

Safety Envelope by RSS Rules

Can Be Retrofit to Any ADV Controller
Monitor & Intervene ➜ Runtime Safety Guarantee

RSS Rule, an Example
[Shalev-Shwartz et al., arXiv preprint, 2017]

- An RSS rule is a pair (A, α) of an *RSS condition* A and a *proper response* α

RSS condition A:
Maintain an inter-vehicle distance at least

$$d_{\min} = \left[v_r \rho + \frac{1}{2} a_{\max,\text{accel}} \rho^2 + \frac{(v_r + \rho a_{\max,\text{accel}})^2}{2a_{\min,\text{brake}}} - \frac{v_f^2}{2a_{\max,\text{brake}}} \right]_+$$

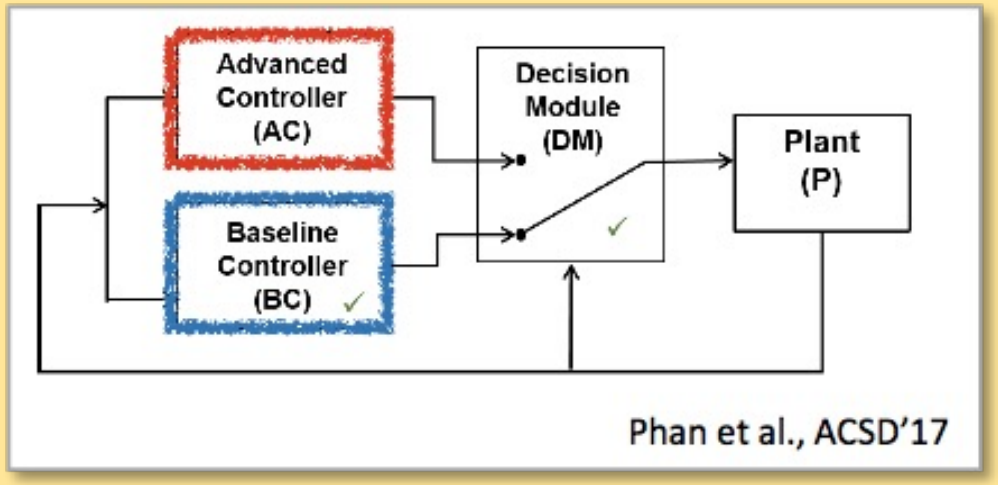
Proper response α:
If A is about to be violated, brake at rate $a_{\text{min, brake}}$ within ρ seconds

Conditional safety lemma:
Any execution of α, from a state that satisfies A, is collision-free.

escape = MRM (minimum risk maneuver)

Structure of an RSS rule

- RSS Condition A:
  "You can still *escape* if A is true"
- Proper response α:
  "control strategy to *escape*"



Simplex architecture

- AC pursues performance and safety
- BC pursues safety (only)
- DM (decision module) switches between them—
  "use BC to escape"

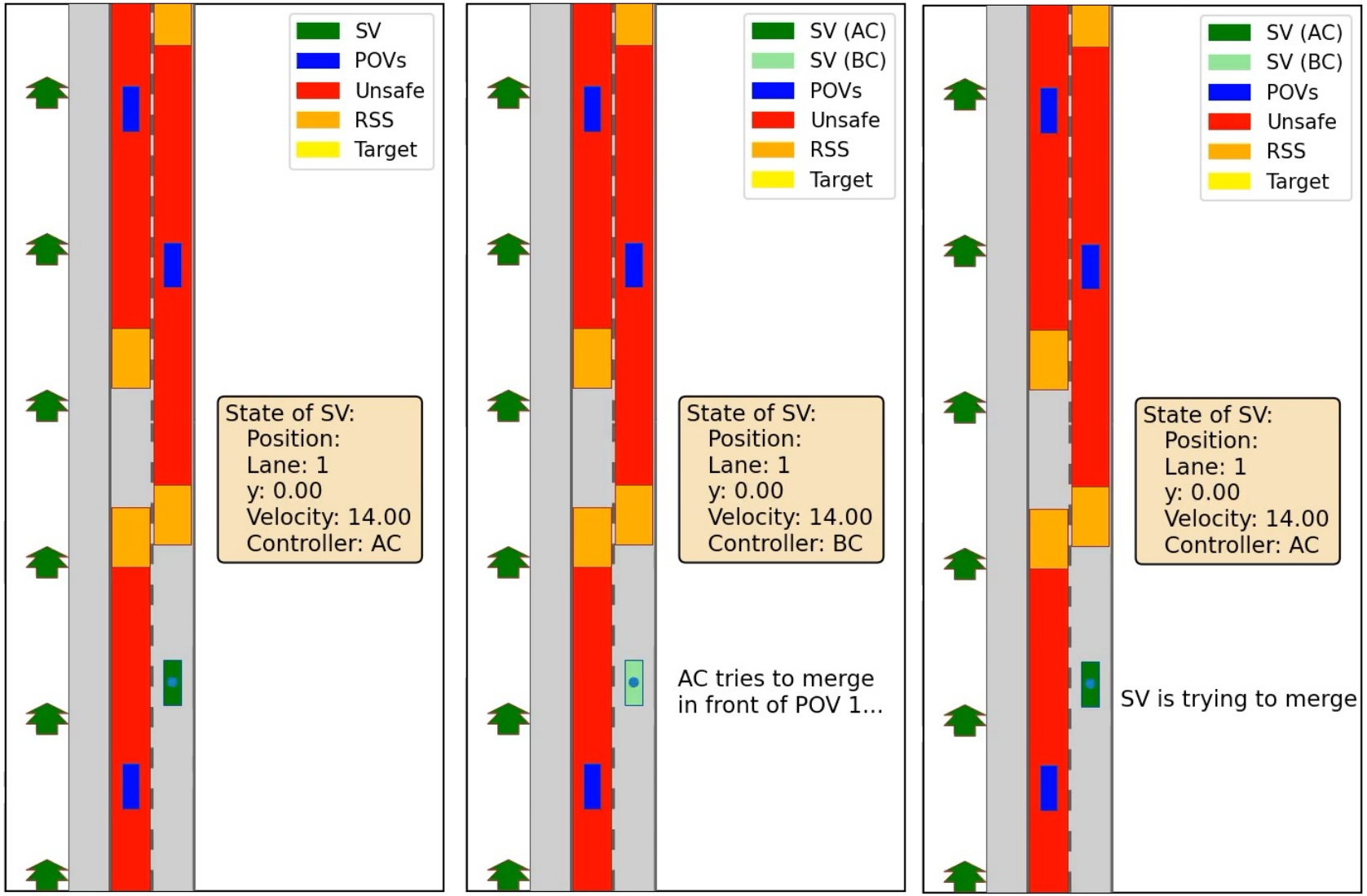
➜ RSS rules fit perfectly!

- AC: existing controller (optimization-based, ML, …)
- BC: executes a proper response
- DM: monitors an RSS condition.
  Violation foreseen ➜ switch to BC

RSS Safety Envelopes in Action, Scenario I

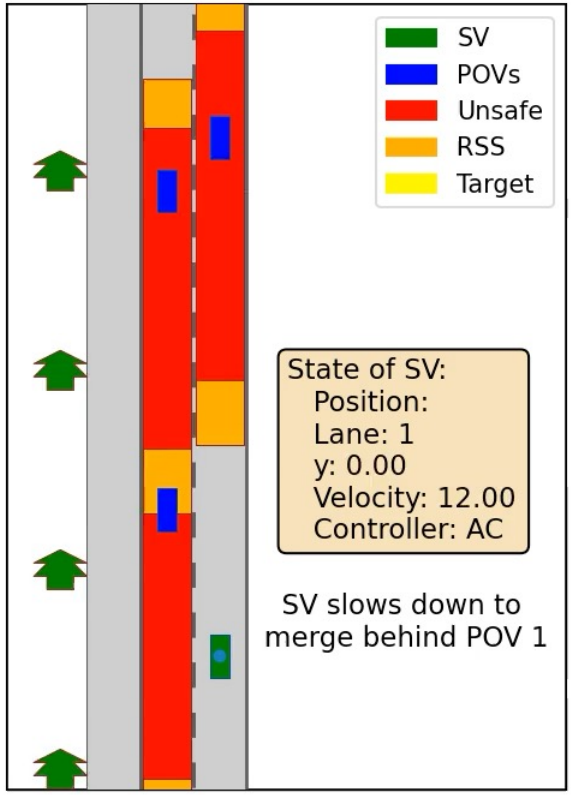
- AC: no safety envelope
- AC+RSS:
  Original RSS rule [Shalev-Shwartz et al., arXiv, 2017]
  as a safety envelope
  ("short-sighted" collision avoidance)
- AC+RSS^{GA} :
  Our RSS rule [Hasuo+, IEEE T-IV]
  as a safety envelope
  (goal achievement too with longer-term planning)

- AC is not safe (hazadous cut-in)
- AC+RSS does not reach the shoulder
- AC+RSS^{GA} successfully deployed the long term strategy of (brake → merge behind).
  Achieved both safety and the goal

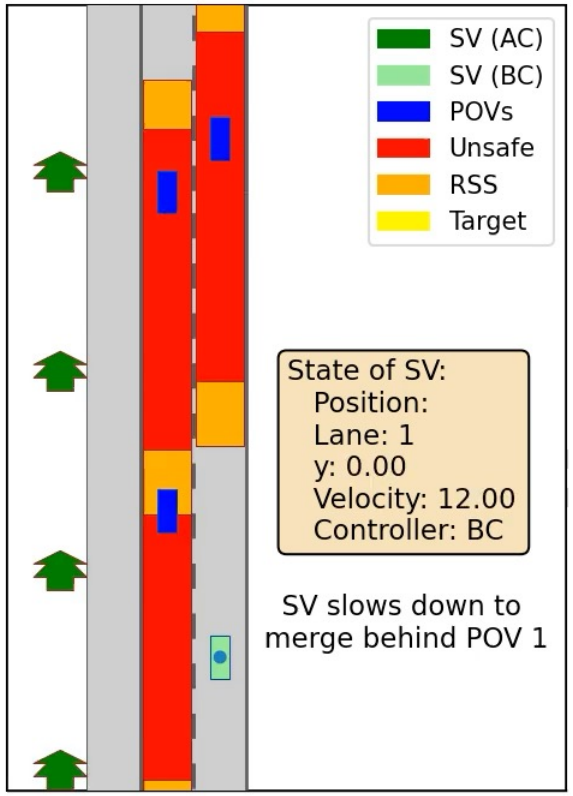


RSS Safety Envelopes in Action, Scenario II

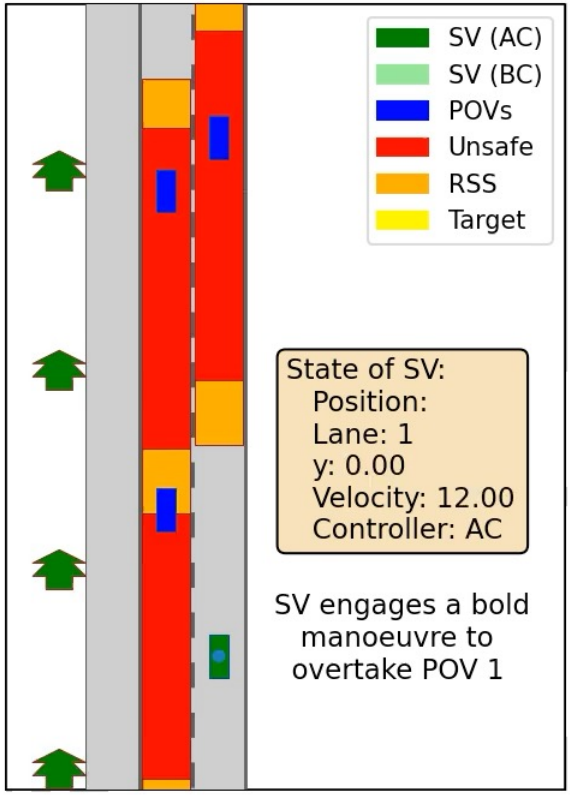
- AC: no safety envelope
- AC+RSS:
  Original RSS rule
  [Shalev-Shwartz et al., arXiv, 2017]
  as a safety envelope
  ("short-sighted" collision avoidance)
- AC+RSS^{GA} :
  Our RSS rule [Hasuo+, IEEE T-IV]
  as a safety envelope
  (goal achievement too
  with longer-term planning)



AC



AC+*RSS*

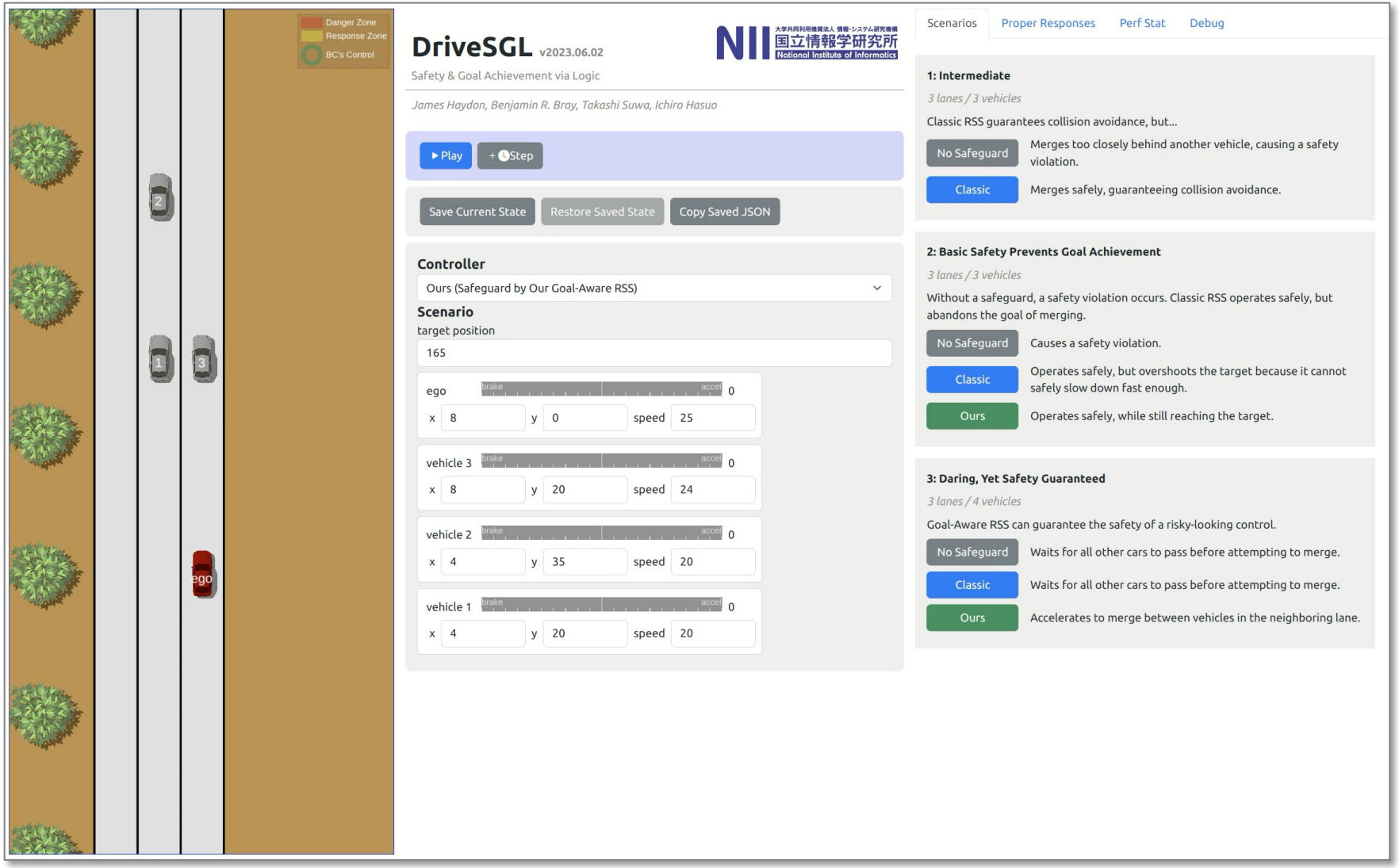


AC+*RSS*^{GA}

- AC & AC+RSS safety achieve the goal, but are **slow**
- AC+RSS^{GA},
  under mathematical safety guarantee,
  boldly accelerates and merge in front
  - … who says safe ADVs are conservative and boring? ☺

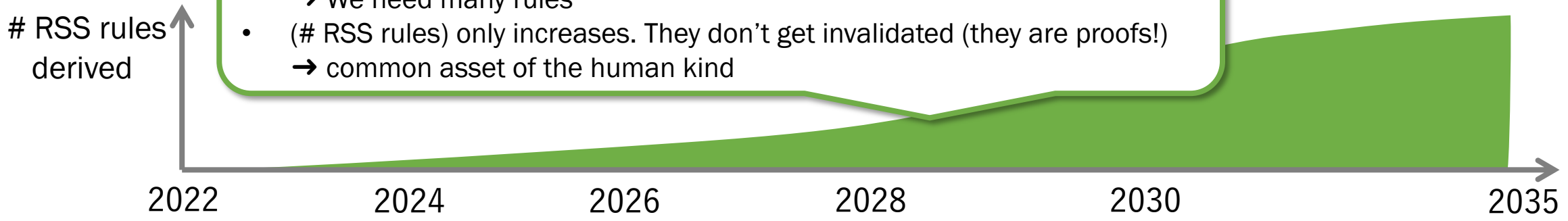
DriveSGL – Our Live Demo (Under Devel.)

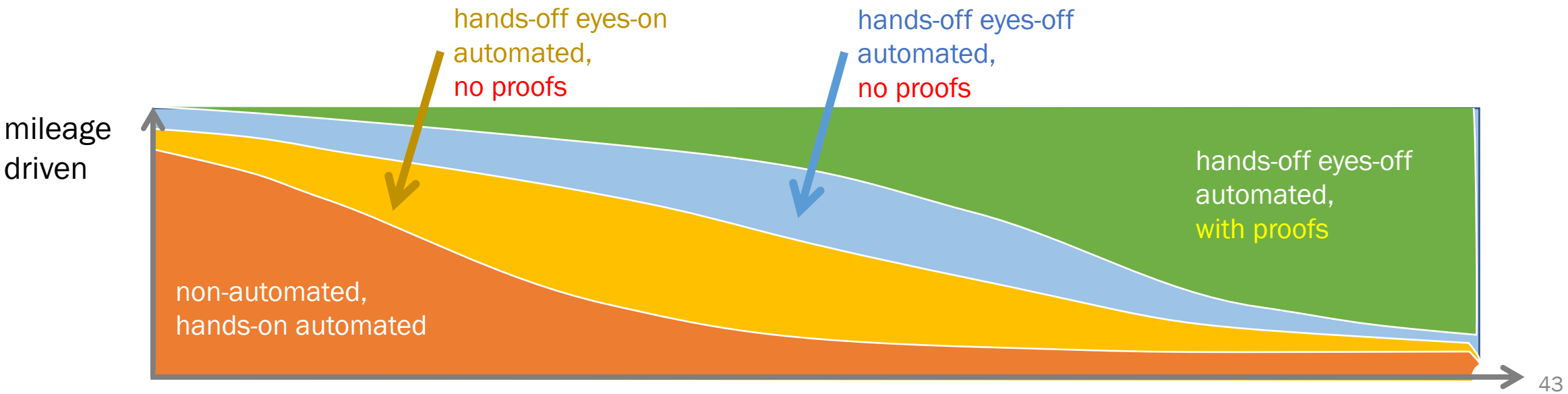
Tease an ADV... Can you cause an accident?



Incremental Accumulation of RSS Rules, Incremental ODD Expansion of “ADVs with Proofs”

- ~ 1 person.month for each scenario
- Different rules for different traffic laws & customs
  → We need many rules
- (# RSS rules) only increases. They don’t get invalidated (they are proofs!)
  → common asset of the human kind





Two Possible Shapes of ADV Safety. Which is Better?

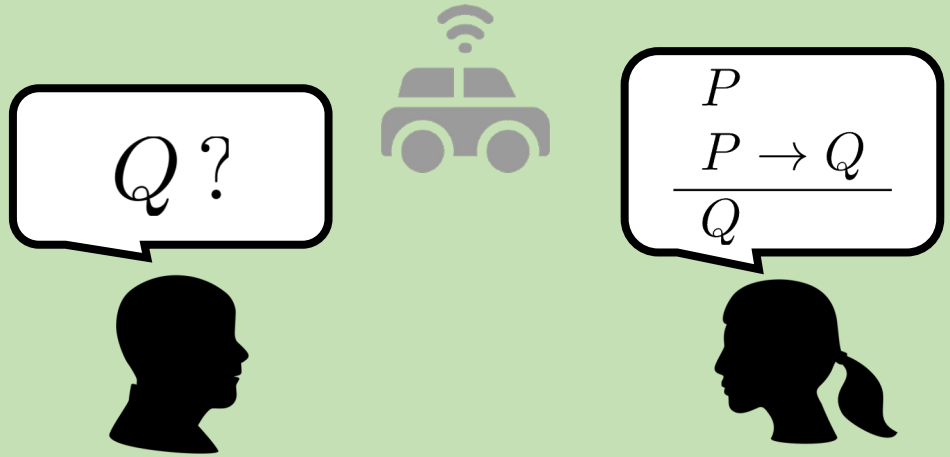
NII MathTrust

Blackbox Safety

- Monolithic “safety claims”
- Hard to examine, criticize, or improve

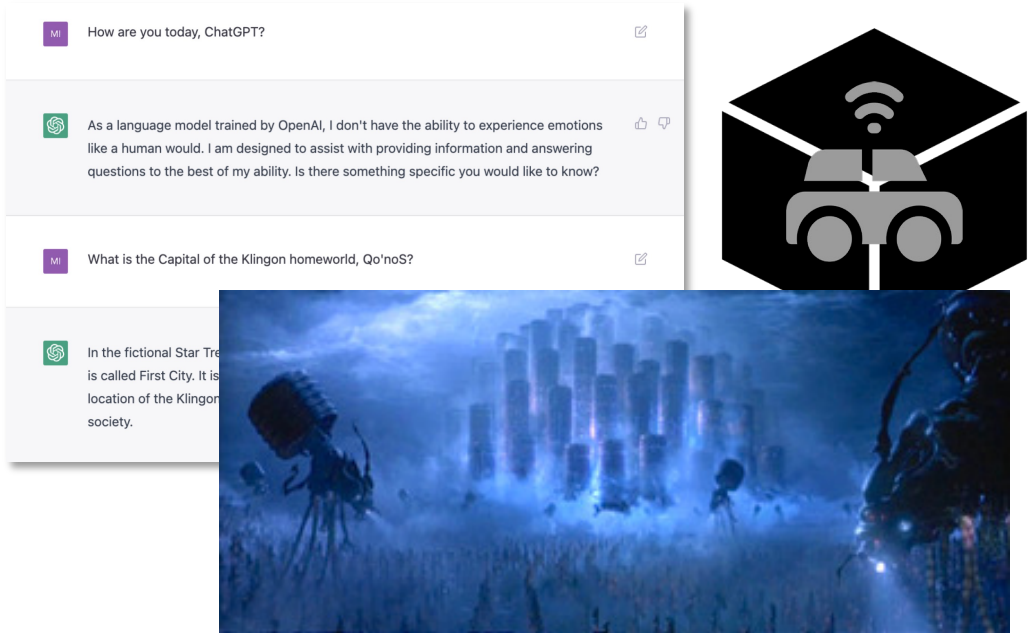
vs

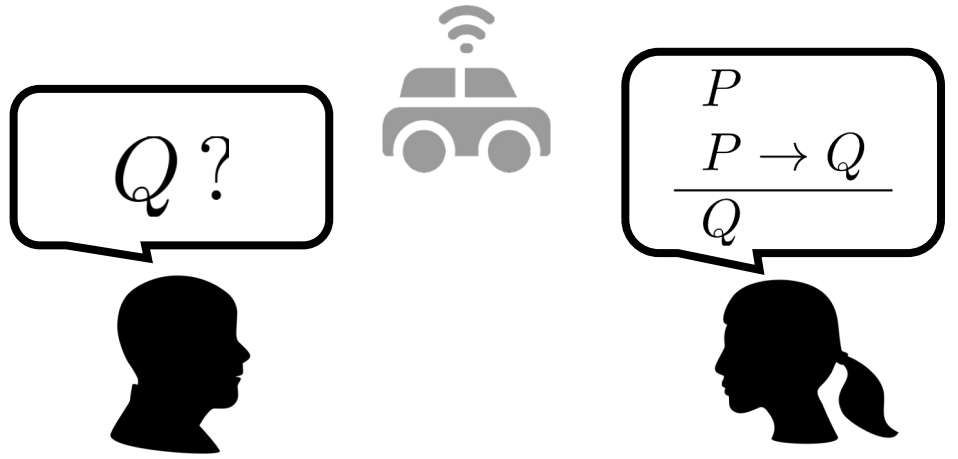
Accountable Safety



- Explainable and traceable safety cases structured by **logic**
- Supporting society’s collective and endless efforts towards ADV safety
- **The shape that we pursue**

Safety-Critical Systems Should Never be Blackbox
Proofs Explicate Assumptions, Contracts, ODDs, and Responsibilities





- Many emerging technologies are statistical and blackbox
- We shouldn't let them operate in safety-critical domains
- (… fight against the "lawyer up" approach towards safety!)

- Conventionally:
  Proofs are for establishing absolute truths
- New: proofs are communication media for
  - explicating assumptions and contracts,
  - showing who's responsible for what, and
  - writing and assessing safety cases
- Logiic as a social infrastructure for trust in ICT

Summary: Logical Manifestation of Specifications, Requirements and Responsibilities

Formal methods for (conventional) software
(e.g. model checking)

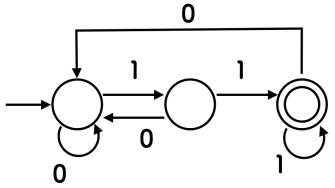
Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- "A system $\mathcal{M}$ satisfies a req./spec./property φ"
- Conventionally:
  $\mathcal{M}$ is a program
  φ is "bug-free"
  → *bug-free proofs*

Heavy-weight formal methods for manufacturing

Goal:
to ***prove*** $\mathcal{M} \models \varphi$

- $\mathcal{M}$ is a car
  φ is safety/quality/…
- But we need a mathematical/logical model $\mathcal{M}$ of a car!



Modeling is costly
- MBD usually doesn't help
- Simulink is for numeric simulation, not principally for logical verification

Part 1

Light-weight formal methods
→ *Manufacturing DX*

- Formal/math./software description of φ
- Feasible yet powerful

Part 2

Proof-based *mathematical traffic laws* for automated driving

- For int'l safety standards, a sound ecosystem, & social acceptance