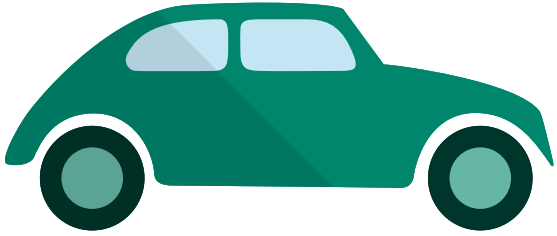
Planning Abstract Consistency Repairs Between Heterogeneous Models

H. Färber, R. Pascual*, S. Singh, T. Stübinger, M. Ulbrich | 25 October 2026

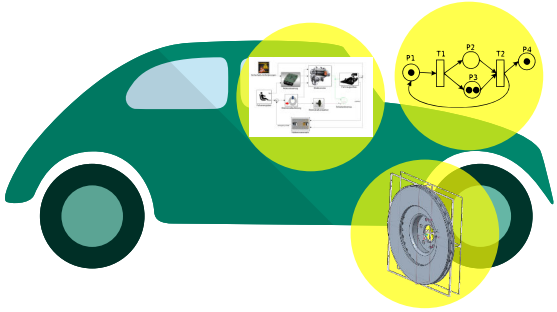
Karlsruhe Institute of Technology *MICS, CentraleSupélec, Université Paris-Saclay



Starting Point: Model-Driven Engineering



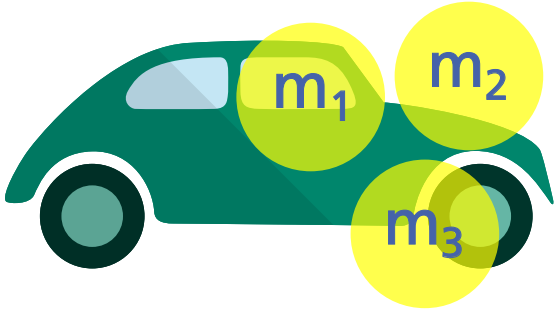
Starting Point: Model-Driven Engineering



Starting Point: Model-Driven Engineering

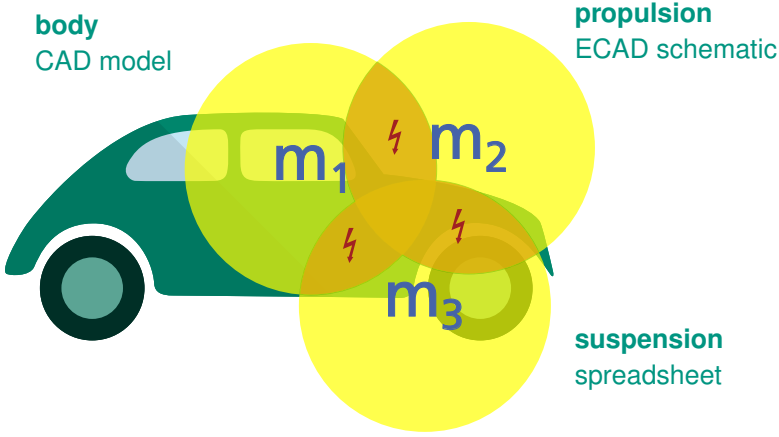
body
CAD model

propulsion
ECAD schematic



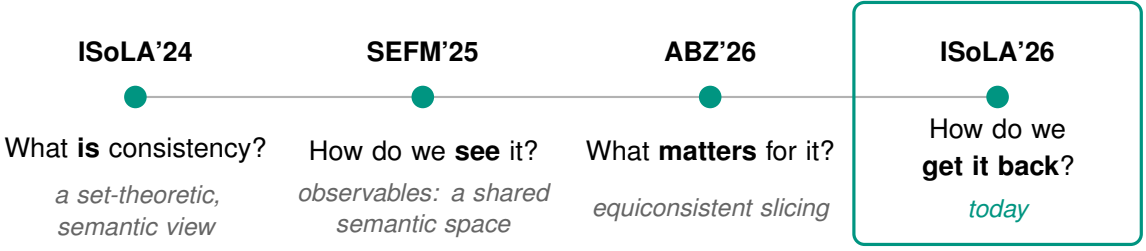
suspension
spreadsheet

Starting Point: Model-Driven Engineering



$\Longrightarrow$ They overlap. Sooner or later, they disagree.

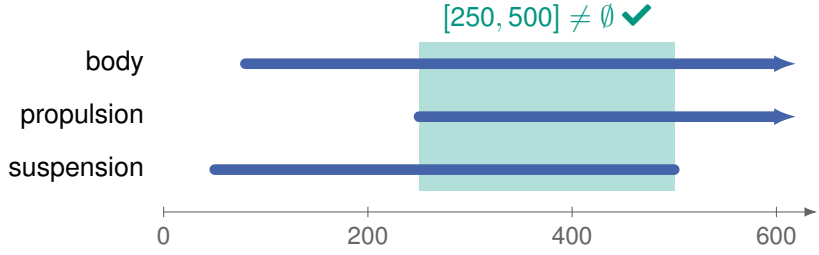
Previously, on this German car . . .

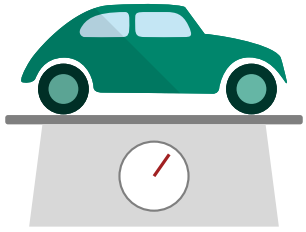


Recap: Observables (SEFM'25)

Each model m constrains the **set of admissible values** in V of a system property: $o(m) \subseteq V$

Observable: total weight w_{tot} (kg)





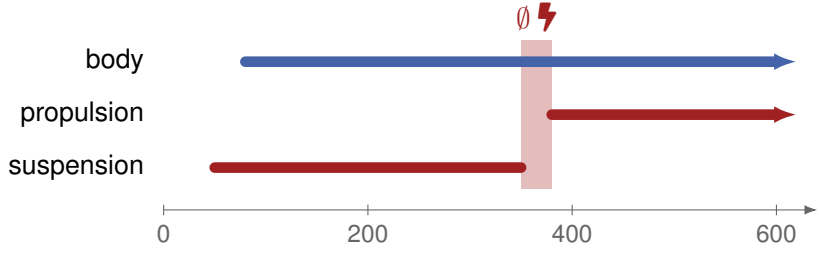
Observable-induced consistency

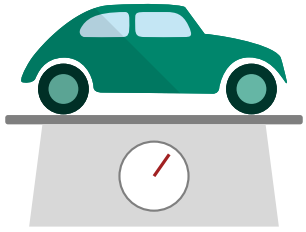
Models are **consistent** iff their admissible values share at least one value: $\bigcap_i o_i(m_i) \neq \emptyset$

Recap: Observables (SEFM'25)

Each model m constrains the **set of admissible values** in V of a system property: $o(m) \subseteq V$

Observable: total weight w_{tot} (kg)





Observable-induced consistency

Models are **consistent** iff their admissible values share at least one value: $\bigcap_i o_i(m_i) \neq \emptyset$

Heterogeneous Modelling Landscape

Consistency is stated like this:

“The total weight must not exceed the load the suspension supports.”

One sentence. Spanning Three models.
No modelling language.

Repair is spread over models:

- **CAD:** resize the shell, change the material
- **ECAD:** swap the cell, rewire the pack
- **Spreadsheet:** pick another suspension

Three languages. Three tools. Three teams.

In which language do you write a repair that spans all three?

Heterogeneous Modelling Landscape

Consistency is stated like this:

“The total weight must not exceed the load the suspension supports.”

One sentence. Spanning Three models.
No modelling language.

Repair is spread over models:

- **CAD:** resize the shell, change the material
- **ECAD:** swap the cell, rewire the pack
- **Spreadsheet:** pick another suspension

Three languages. Three tools. Three teams.

In which language do you write a repair that spans all three?

Heterogeneous Modelling Landscape

Consistency is stated like this:

“The total weight must not exceed the load the suspension supports.”

One sentence. Spanning Three models.
No modelling language.

Repair is spread over models:

- **CAD:** resize the shell, change the material
- **ECAD:** swap the cell, rewire the pack
- **Spreadsheet:** pick another suspension

Three languages. Three tools. Three teams.

In which language do you write a repair that spans all three? **None of them.**

Heterogeneous Modelling Landscape

Consistency is stated like this:

“The total weight must not exceed the load the suspension supports.”

One sentence. Spanning Three models.
No modelling language.

Repair is spread over models:

- **CAD:** resize the shell, change the material
- **ECAD:** swap the cell, rewire the pack
- **Spreadsheet:** pick another suspension

Three languages. Three tools. Three teams.

In which language do you write a repair that spans all three? **None of them.**

. . . We need a formalism spanning heterogeneous models

The Shift

If consistency is semantic,

The Shift

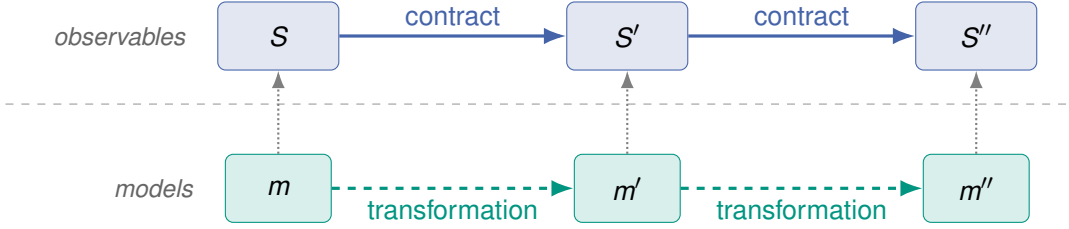
If consistency is semantic,
repair better be planned semantically,

The Shift

If consistency is semantic,
repair better be planned semantically,
but implemented syntactically.

The Shift

If consistency is semantic,
repair better be planned semantically,
but implemented syntactically.



Plan upstairs. Execute downstairs.

What is the output of a repair?

Nobody only wants the repaired consistent model.

What is the output of a repair?

Nobody only wants the repaired consistent model.

They want to know **how to obtain it**.

- The product of repair is a **plan**: a traceable, auditable sequence of admissible changes.
- Temporary inconsistency along the way is **fine**.

Upstairs: Observable States

- An **observable state**: $S = (S_i)_{i \in I}$ with $S_i \subseteq \prod_K V_k$
  joint per model: voltage depends on cell type **and** series count
- S is **consistent** iff $\bigcap_i S_i \neq \emptyset$
- Each collection of het. models m induces an observable state $\nu(m)$

Proposition

$m \in CR_O$ iff $\nu(m)$ is admissible and consistent.

Admissible: claimed observable values can actually be observed

$\Longrightarrow$ Finding a consistent model becomes finding a consistent **state**.

Contracts: What, Not How

Contract

A relation on observable states: $c \subseteq \prod_I \mathcal{S} \times \prod_I \mathcal{S}$, with $\mathcal{S} = \mathscr{P}\left(\prod_K V_k\right)$

Contracts: What, Not How

Contract

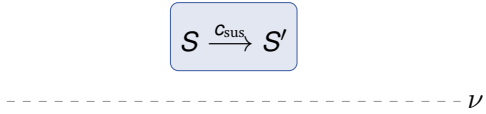
A relation on observable states: $c \subseteq \prod_I \mathcal{S} \times \prod_I \mathcal{S}$, with $\mathcal{S} = \mathscr{P}\left(\prod_K V_k\right)$

Note: contracts on **evolution of observables**,
not on behaviour of component (unlike assume/guarantee).

Contracts: What, Not How

Contract

A relation on observable states: $c \subseteq \prod_I \mathcal{S} \times \prod_I \mathcal{S}$, with $\mathcal{S} = \mathscr{P}\left(\prod_K V_k\right)$



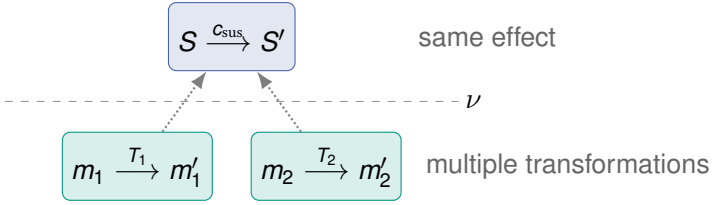
$\Longrightarrow$ One contract = a **family** of transformations with the same observable effect.

Note: contracts on **evolution of observables**,
not on behaviour of component (unlike assume/guarantee).

Contracts: What, Not How

Contract

A relation on observable states: $c \subseteq \prod_I \mathcal{S} \times \prod_I \mathcal{S}$, with $\mathcal{S} = \mathscr{P}(\prod_K V_k)$



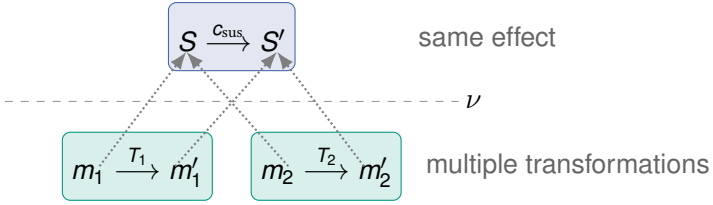
$\Longrightarrow$ One contract = a **family** of transformations with the same observable effect.

Note: contracts on **evolution of observables**,
not on behaviour of component (unlike assume/guarantee).

Contracts: What, Not How

Contract

A relation on observable states: $c \subseteq \prod_I \mathcal{S} \times \prod_I \mathcal{S}$, with $\mathcal{S} = \mathscr{P}(\prod_K V_k)$



$\Longrightarrow$ One contract = a **family** of transformations with the same observable effect.

Note: contracts on **evolution of observables**,
not on behaviour of component (unlike assume/guarantee).

Example Contracts

Let $I = \mathbf{3}$ (3 models) and $K = \mathbf{1}$ (one observable) and $V = \mathbb{R}$ (real-valued)

A state (X, Y, Z) is a triple of admissible value sets,
a contract is a relation between states

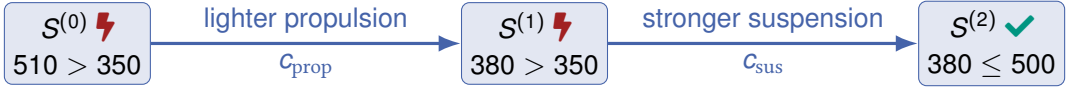
(1) $c_1 = \{((a, b], Y, Z), ([a + s, b + s], Y, Z)) \mid a, b, s \in \mathbb{R}_+\}$ shift a range

(2) $c_2 = \{((X, [0, L], Z), (X, [0, L'], Z)) \mid L' > L\}$ relax a bound

(3) $c_3 = \{((X, Y, Z), (X, Y, \{5\})) \mid 5 \in Z\}$ contraction to a fixed value

Repair Plans

A **repair plan** takes an inconsistent state to states that are **all** consistent.
Here: total weight vs. maximal load.

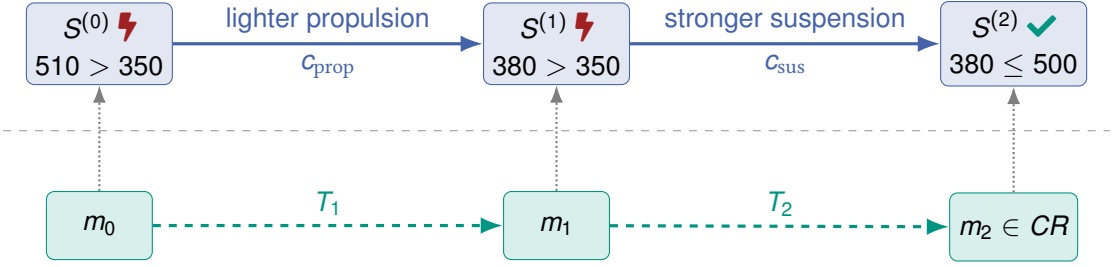


$\Longrightarrow$ $S^{(1)}$ is **still inconsistent**, and that is fine.

Repair Plans

A **repair plan** takes an inconsistent state to states that are **all** consistent.

Here: total weight vs. maximal load.



Theorem

If every contract is **sound**/realisable, the plan lifts to models $m_0 \to \cdots \to m_n$ with $m_n \in CR$.

Solvers Don’t Speak Powerset

States are **sets of sets of values**. Model checkers want variables.



Sets ✗
faithful, but needs
set reasoning



Values ✗
easy to solve, but
uncertainty is lost

Solvers Don't Speak Powerset

States are **sets of sets of values**. Model checkers want variables.



Sets ✗
faithful, but needs
set reasoning



Values ✗
easy to solve, but
uncertainty is lost

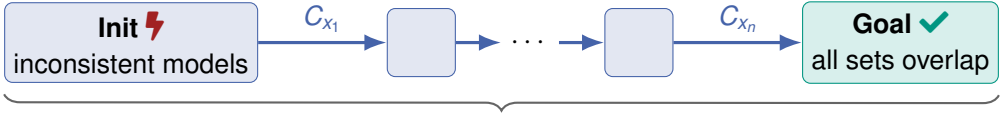


Abstract domains ✓
a set = a few numbers,
e.g. $[l, r]$

$\Longrightarrow$ Parameters $d \in D$ describe the set $\mu_D(d) = \{a \mid a \, \varepsilon_D \, d\}$: the theory is kept, the solver is happy.

Repair = Reachability

State: one variable per model, observable and abstract-domain parameter.

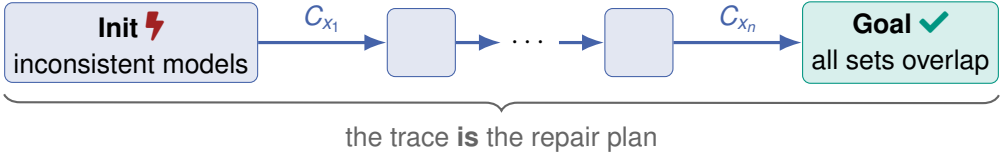


Transitions: one disjunct per contract, plus a frame condition

$$\mathit{Trans} := \bigvee_{x} \left(C_x^{\mathit{pre}} \wedge C_x^{\mathit{post}} \wedge C_x^{\mathit{frame}} \wedge t' = x \right)$$

Repair = Reachability

State: one variable per model, observable and abstract-domain parameter.



Transitions: one disjunct per contract, plus a frame condition

$$\mathit{Trans} := \bigvee_{x} \left(C_x^{\mathit{pre}} \wedge C_x^{\mathit{post}} \wedge C_x^{\mathit{frame}} \wedge t' = x \right)$$

We did not write a planner. Model checkers already are one.

Trying To Prove It Can't Be Fixed

We ask nuXmv to prove the invariant

$$\neg \left(\mathit{step} \leq 10 \wedge \mathit{cons} \wedge \neg \mathit{overflow} \right)$$

"*Within 10 steps, this car can never be made consistent.*"

Trying To Prove It Can't Be Fixed

We ask nuXmv to prove the invariant

$$\neg\left(\mathit{step} \leq 10 \wedge \mathit{cons} \wedge \neg \mathit{overflow}\right)$$

"*Within 10 steps, this car can never be made consistent.*"

nuXmv **fails**.



The counterexample **is** the repair plan.

Case Study: An Electric Vehicle That Is Too Heavy and Underpowered

Observables

- cell $\in$ {LFP, NMC, HiVolt}
- body $\in$ {steel, alu, composite}
- intervals: series s, parallel p, body volume v_b

Derived

- $v_{\text{bat}} = s \cdot v_{\text{cell}}$
- $w_{\text{tot}} = s \cdot p \cdot w_{\text{cell}} + w_{\text{eng}} + v_b \cdot \rho$

Consistency

- `cons_volt`: $v_{\text{bat}} \subseteq [300, 450]$
- `cons_weight`: $\max w_{\text{tot}} \leq 1500$

Contracts

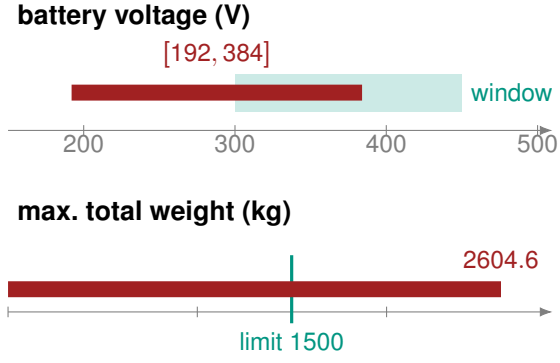
- C1, C2: switch cell, switch body material
- C3–C8: narrow an interval bound by one step

Initial state

LFP, steel, $s \in [60, 120]$, $p \in [4, 12]$, $v_b \in [0.20, 0.30]$ $v_{\text{bat}} = [192, 384]$ ✗ $\max w_{\text{tot}} \approx 2605$ ✗

Watch the Plan (nuXmv, incremental COI)

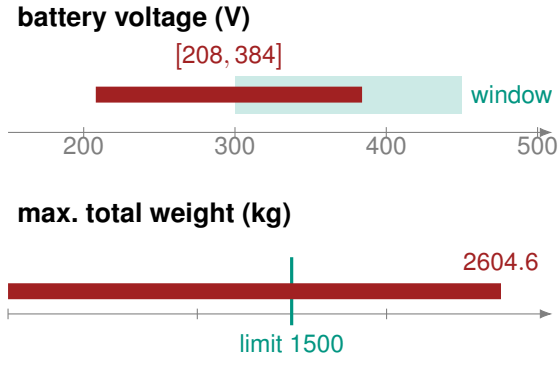
Step	Change



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

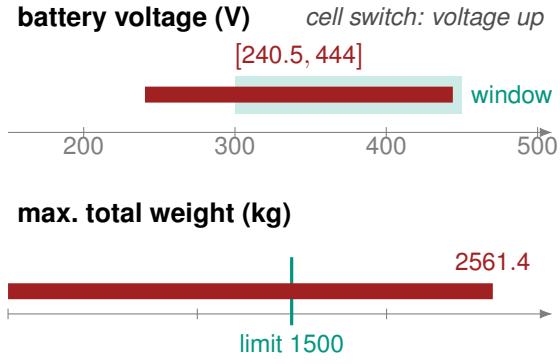
Step		Change
1	C3	**raise** lo_s: 60 → 65



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

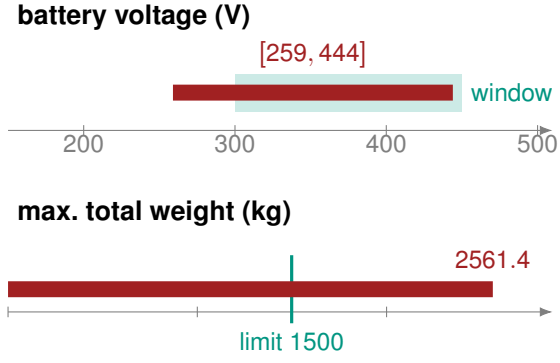
Step		Change
1	C3	raise lo_s: 60 $\rightarrow$ 65
2	**C1**	**cell: LFP $\rightarrow$ HiVolt**



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

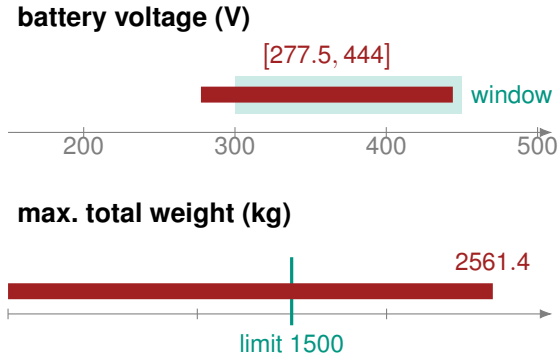
Step		Change
1	C3	raise lo_s: 60 $\rightarrow$ 65
2	C1	cell: LFP $\rightarrow$ HiVolt
3	**C3**	**raise lo_s: 65 $\rightarrow$ 70**



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

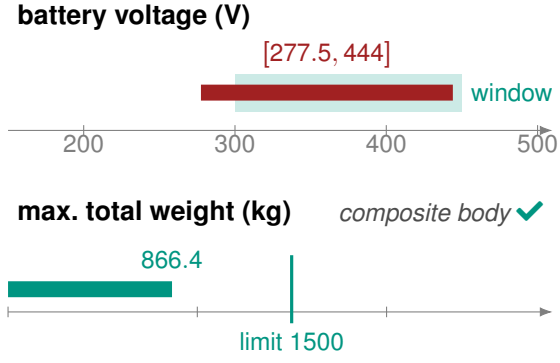
Step		Change
1	C3	raise lo_s: 60 $\rightarrow$ 65
2	C1	cell: LFP $\rightarrow$ HiVolt
3	C3	raise lo_s: 65 $\rightarrow$ 70
4	**C3**	**raise lo_s: 70 $\rightarrow$ 75**



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

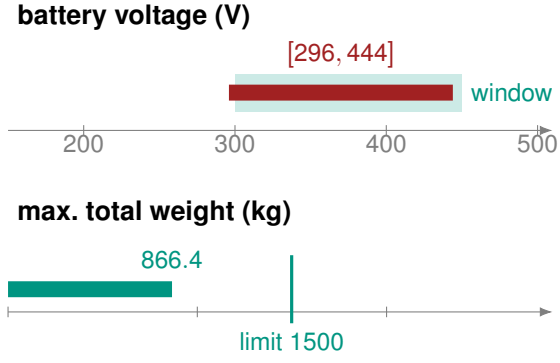
Step		Change
1	C3	raise lo_s: 60 → 65
2	C1	cell: LFP → HiVolt
3	C3	raise lo_s: 65 → 70
4	C3	raise lo_s: 70 → 75
5	**C2**	**body: steel → composite**



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

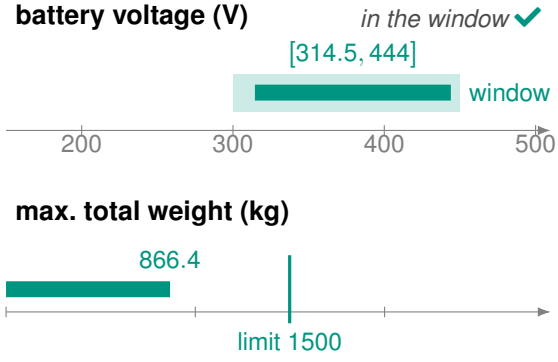
Step		Change
1	C3	raise lo_s: 60 → 65
2	C1	cell: LFP → HiVolt
3	C3	raise lo_s: 65 → 70
4	C3	raise lo_s: 70 → 75
5	C2	body: steel → composite
6	**C3**	**raise lo_s: 75 → 80**



IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

Watch the Plan (nuXmv, incremental COI)

Step		Change
1	C3	raise lo_s: 60 → 65
2	C1	cell: LFP → HiVolt
3	C3	raise lo_s: 65 → 70
4	C3	raise lo_s: 70 → 75
5	C2	body: steel → composite
6	C3	raise lo_s: 75 → 80
7	**C3**	**raise lo_s: 80 → 85**



⟹ Consistent after 7 steps, inconsistent all the way there.

IC3, BMC, COI: same 7 moves, 13–19 s. **Feasibility, not a benchmark.**

“Upstairs only:” Where Are the Models?

You may be thinking:
take models and transformations out of this talk,
and nothing changes.

“Upstairs only:” Where Are the Models?

You may be thinking:
take models and transformations out of this talk,
and nothing changes.

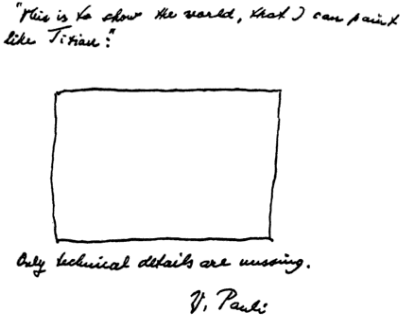
Exactly.

Models appear exactly twice:

- $\uparrow \nu$: extract observables from models (e.g. Liu et al. 2026)
- $\downarrow$ **soundness**: every contract step has a concrete transformation

$\Longrightarrow$ Tools share the **effects** of their edits, not their editing procedures.
$\Longrightarrow$ Technique is modelling language agnostic.

The Fine Print

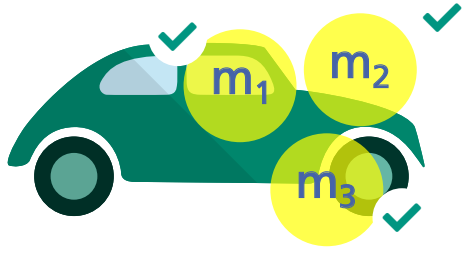


"This is to show the world that I can paint like Titian. [. . .] Only technical details are missing."
— W. Pauli, 1958

- **Realisation: Automatic or manual?**
  Both! It might be a trivial edit or a intectually challenging task
- **Who writes the contracts?**
  Domain experts. Good: Can share without knowledge about other domains
- **Is it a good plan?**
  Consistent, not cheapest or least disruptive. (Costs functions possible.)
- **Does it scale?**
  One car, eight contracts, 13 seconds.

Conclusion

- Consistency is semantic: **plan** repairs semantically.
- **Contracts** abstract transformations by their observable effect.
- Sound contracts: plans lift back to the models.
- Planning is **reachability**: nuXmv finds the plans.
- Works on an electric-vehicle case study.



Plan upstairs. Execute downstairs.

Backup: Is It Fast?

	IC3	BMC	COI
plan length	7	7	7
wall time	16.6 s	18.6 s	**12.7 s**
peak mem	369 M	149 M	197 M

- All three algorithms return the **same 7 moves**, in different orders.
- **Why?** Contracts local to disjoint components commute.
  (Proposition in the paper)

Feasibility study, not a benchmark

13 seconds for 7 moves: your engineers are still faster. **For now.**

Backup: Full Evaluation Setup

Data tables

cell	v_{cell}	w_{cell}	body	ρ
LFP	3.2	0.090	steel	7850
NMC	3.5	0.075	alu	4500
HiVolt	3.7	0.060	comp.	2200

Consistency relations

`cons_volt`	$300 \leq lo\, v_{\text{bat}},\ hi\, v_{\text{bat}} \leq 450$
`cons_weight`	$hi\, w_{\text{tot}} \leq 1500$
`cons_s_rng`	$60 \leq lo_s,\ hi_s \leq 120$
`cons_p_rng`	$2 \leq lo_p,\ hi_p \leq 12$
`cons_v_rng`	$0.10 \leq lo_{v_b},\ hi_{v_b} \leq 0.30$

Contracts

C1 switch cell	—	$cell' \neq cell$
C2 switch body	—	$body' \neq body$
C3 raise lo_s	$lo_s + 5 \leq hi_s$	$lo'_s = lo_s + 5$
C4 lower hi_s	$hi_s - 5 \geq lo_s$	$hi'_s = hi_s - 5$
C5 raise lo_p	$lo_p + 1 \leq hi_p$	$lo'_p = lo_p + 1$
C6 lower hi_p	$hi_p - 1 \geq lo_p$	$hi'_p = hi_p - 1$
C7 raise lo_{v_b}	$lo_{v_b} + 0.025 \leq hi_{v_b}$	$lo'_{v_b} = lo_{v_b} + 0.025$
C8 lower hi_{v_b}	$hi_{v_b} - 0.025 \geq lo_{v_b}$	$hi'_{v_b} = hi_{v_b} - 0.025$

AMD Ryzen AI 7 PRO 360, 30 GiB RAM, Ubuntu 24.04, nuXmv 2.1.0

Backup: nuXmv Encoding (Excerpts)

Observables

```
VAR
 cell_type  : 1..3;
 lo_series  : 1..200;
 lo_vol_bod : real;  ...
```

Initial state

```
INIT
 cell_type = 1 & body_mat = 1 &
 lo_series = 60 & hi_series = 120 &
 lo_vol_bod = 0.20 & ...
```

Contract C3

```
pre_3  := lo_series + 5 <= hi_series;
post_3 := next(lo_series)
                          = lo_series + 5;
frame_lo_series := {3};
```

Transition relation

```
TRANS
 ((pre_1 & post_1 & next(t)=1) |
    (pre_2 & post_2 & next(t)=2) | ...) &
 (!(next(t) in frame_cell_type) ->
        next(cell_type) = cell_type) & ...
```

What Makes a Good Contract?

Property	Meaning	Closed under $\circ$
admissible	compound observables stay coherent	✓
local to i	only S_i changes	✗ (✓ same i)
well-formed	both ends are realizable by models	✓
sound w.r.t. T	each step has a concrete transformation	✓

$$\text{sound} \implies \text{well-formed} \implies \text{admissible}$$

What Makes a Good Contract?

Property	Meaning	Closed under $\circ$
admissible	compound observables stay coherent	✓
local to i	only S_i changes	✗ (✓ same i)
well-formed	both ends are realizable by models	✓
sound w.r.t. T	each step has a concrete transformation	✓

$$\text{sound} \implies \text{well-formed} \implies \text{admissible}$$

Counterexample

"Raise the supported load to **any** value" is not well-formed:
no such suspension exists in the catalogue.

Backup: Composition

Contract composition

$c_2 \circ c_1 = \{(S, S'') \mid \exists S'.\ (S, S') \in c_1 \wedge (S', S'') \in c_2\}$

Proposition

Admissibility, well-formedness and soundness are closed under composition. Locality is not, unless both contracts are local to the same component.

Proposition (commutativity)

Contracts local to disjoint sets of components commute: $c_2 \circ c_1 = c_1 \circ c_2$.

$\Longrightarrow$ A plan can be reorganised: group contracts per component, permute independent ones.

Backup: Related Work

- **Controlled updates / bidirectional transformations** (Ehrig et al., Xiong et al., Macedo & Cunha): consistency by construction, no temporary inconsistency
- **Repair hints** (Easterbrook & Nuseibeh, Kolovos et al., Reder & Egyed): metamodel-independent, but one hint per constraint, written by hand
- **Graph transformation** (Kosiol et al., Fritsche et al.): strong guarantees, but rules live at the graph level
- **Search-based repair** (Badger, Pinna Puissant et al.): regression planning, intra-model only
- **Planning as SAT/SMT** (Kautz & Selman, Rintanen, Bofill et al.): the engines we reuse
- **Contract-based design** (Benveniste et al., Pacti): contracts on **components**; ours are on **changes**

$\Longrightarrow$ We plan on observable semantics: no constraint-specific repair hints, heterogeneous by design.

Backup: Future Work

- **Category Theoretic View on Concistency Repair**
  - CT theory of lenses
  - Independence of consistency relations

- **Engineering view**
  Identifying concrete repair actions on concrete examples.