Closing the Control Abstraction Gap:
From Linear Control Models to Fine-Grained Verification of Control Software

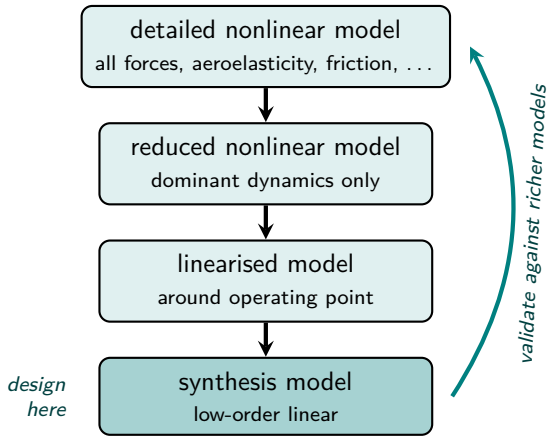
Martina Maggio

Saarland University, Germany
Lund University, Sweden

RTSOPS 2026

Control design lives on a ladder of abstractions

On the physics side, multi-granularity modelling is standard practice

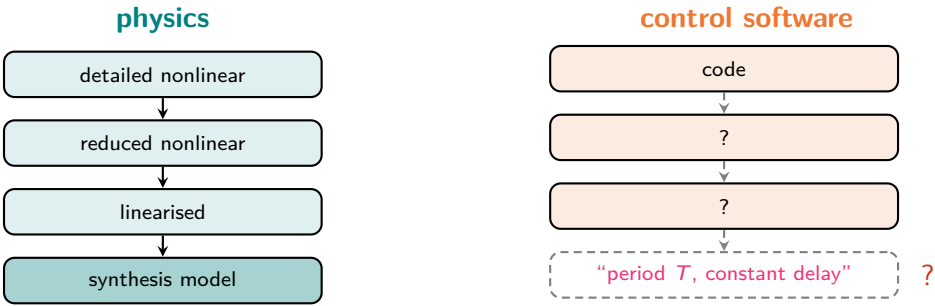


- every level **drops detail deliberately** — and we know what was dropped
- controller synthesis on the simplest model
- confidence built by walking back up the ladder: simulation, HIL, flight tests

This works because the levels are **related**: each abstraction is accountable to the one above it.

The digital side does not follow the same pattern

The implementation is real — its model is a cartoon



- the physics gains and loses detail across levels; the software jumps from **full implementation** straight to **one coarse timing assumption**
- intermediate levels — delays, missed updates, unfinished computations, synchronisation errors — are not modelled at all

➔ **this asymmetry is the *control abstraction gap***

The gap is not hypothetical

Mars helicopter Ingenuity, Flight 6 [NASA, 2021]

"This glitch caused a single image to be lost, but more importantly, it resulted in all later navigation images being delivered with **inaccurate timestamps**. [...] estimates being constantly 'corrected' to account for phantom errors. **Large oscillations ensued**."

- the controller was robust by design: "designed to tolerate significant errors without becoming unstable, including errors in timing"
- yet the failure emerged at the **interaction of communication and computation** — a level that no design model captured
- patched after Flight 6; a different synchronisation failure appeared on Flight 53
- Ingenuity survived thanks to an $80M budget — most systems cannot afford that

➔ **timing failures live exactly in the unmodelled middle of the software ladder**

Zooming in on one timing failure: deadline misses

And industry has accepted this as a fact of life

Industrial survey [Akesson et al., 2020]

"45% of respondents indicated that the most time critical functions in the system **could miss some deadlines**, and 20% indicated that deadline misses more often than 1 in 100 could be tolerated"

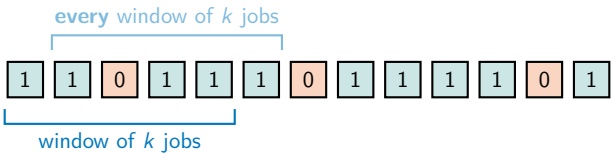
- sensing → computing → actuating, periodically, under timing constraints
- complexity (e.g., vision pipelines) makes timing unpredictable
- sporadic **deadline misses** are a reality
- even infrequent misses can affect safety and stability of the controlled system

➔ **let's model deadline misses properly**

The weakly-hard model

The de-facto industrial standard for deterministic timing failures

A task execution is an infinite word $(a_j)_{j \in \mathbb{N}^+}$, $a_j \in \{0, 1\}$: 1 = deadline hit, 0 = deadline miss



$$\tau \vdash \texttt{AnyMiss}(m, k) := \forall i \in \mathbb{N}^+, \sum_{j=i}^{i+k-1} (1 - a_j) \leq m$$

Why this model is pretty cool

- **expressive**: captures realistic "at most m misses per k jobs" guarantees
- **deterministic**: certification-friendly, no probabilistic assumptions
- **formal**: every constraint is a regular language — automata, tools, theory for free

Verifying stability under deadline misses

Two ingredients combined in a cross-product, model-checking style

1. Closed-loop dynamics

- switched linear system

$$x_{i+1} = \Phi_{\sigma(i)}\, x_i,\ \ \sigma(i) \in \{0, 1\}$$

- Φ_1: nominal step (deadline hit)
- Φ_0: faulty step (deadline miss)

2. Fault pattern

- the weakly-hard constraint is a regular language
- acceptor automaton $\mathcal{A}$ with adjacency matrices Π_1 (hit), Π_0 (miss)

Verifying stability under deadline misses

Two ingredients combined in a cross-product, model-checking style

1. Closed-loop dynamics

- switched linear system

$$x_{i+1} = \Phi_{\sigma(i)}\, x_i,\ \ \sigma(i) \in \{0, 1\}$$

- Φ_1: nominal step (deadline hit)
- Φ_0: faulty step (deadline miss)

2. Fault pattern

- the weakly-hard constraint is a regular language
- acceptor automaton $\mathcal{A}$ with adjacency matrices Π_1 (hit), Π_0 (miss)

Stability criterion (joint spectral radius) [Vreman et al., LCSS 2022]

the system is stable under all admissible miss patterns iff

$$\mathrm{jsr}\left(\Pi_0 \otimes \Phi_0,\ \Pi_1 \otimes \Phi_1\right) < 1$$

➔ **clean, sound, automatic — so what is the catch?**

The catch: combinatorial automaton growth

Minimal acceptor size is binomial in the window length

Minimal automaton size [Vreman et al., 2022]

$\mathcal{A}$ for **AnyMiss** (m, k) has $\binom{k}{m}$ states
(all ways of placing m zeros in a window of k)

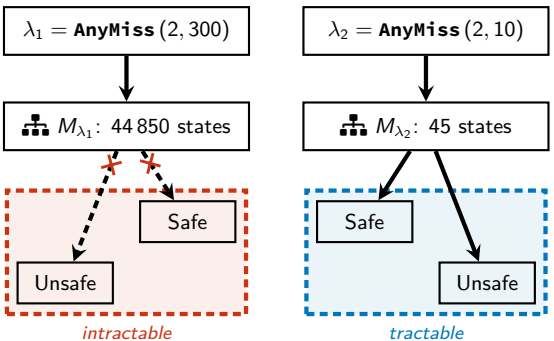
- industrial windows are large ($k \approx 300$) with small miss budgets m
- the matrices to analyse are $\Pi_\sigma \otimes \Phi_\sigma$: **automaton size** $\times$ **plant order**
- plant Φ: $\approx 10 \times 10 \ldots$ the automaton dominates by three orders of magnitude

constraint	states
AnyMiss $(3, 5)$	10
AnyMiss $(2, 50)$	1 225
AnyMiss $(2, 300)$	44 850
AnyMiss $(3, 100)$	161 700
AnyMiss $(3, 300)$	4 455 100

➔ **the fault model (not the plant!) sets the limit of what can be verified**

Why is this so hard? Rare failures need many states

Describing a world where little happens takes a lot of memory



the classical workaround:
shrink the window

- every word in λ_1 is a word of λ_2:
  safe under $\lambda_2 \Rightarrow$ safe under λ_1 👍
- but $\mathbf{AnyMiss}\,(2, 10)$ admits up to
  60 misses per 300 jobs —
  30× the original fault budget
- extra fault patterns push
  jsr above 1: verdict unsafe
- **unsafe does not carry back**:
  the result is inconclusive for λ_1

➲ **faithful models are intractable; tractable models are too pessimistic to be useful**

Open problem: fault models that are meaningful and scalable

An automaton approximation problem, with structure to exploit

Given the minimal acceptor M_λ (n states) of a realistic constraint λ, find $\underline{M}$ such that:

(i)	**soundness**	$\mathcal{L}(M_\lambda) \subseteq \mathcal{L}(\underline{M})$	safety under $\underline{M}$ implies safety under λ
(ii)	**compression**	$\lvert\underline{M}\rvert = k \ll n$	small enough for the verification engine
(iii)	**fidelity**	$\underline{M} = \arg\min \lvert\mathcal{L}_m(\underline{M}) \setminus \mathcal{L}_m(M_\lambda)\rvert$	as few spurious fault patterns as possible

- target scale: $n \approx 10^6$, $k \approx 10^3$, word length $m \approx 300$
- generic DFA reduction won't do it — but our automata are **highly structured**: binary alphabet, transition function constrained by the failure pattern
- and "fidelity" should ultimately be measured on the closed loop, not just the language

A first data point [de Maeyer, Hermanns, M., CAV 2026]

Exploit the structure: merge states that only differ in recovery bookkeeping

- rename states to tuples of **distances to recovery**: how many hits until the next miss is tolerable
- **critical** states all just wait for hits — merge them in groups of c (a tunable compression factor)
- keep the **most permissive** transitions: language only grows

automaton	states
AnyMiss $(2, 300)$	44 850
Trim $(2, 300, 260)$	635

- $70\times$ fewer states, near-identical language growth
- first stability certificates for realistic $k \approx 300$ constraints
- verification memory down 1–2 orders of magnitude

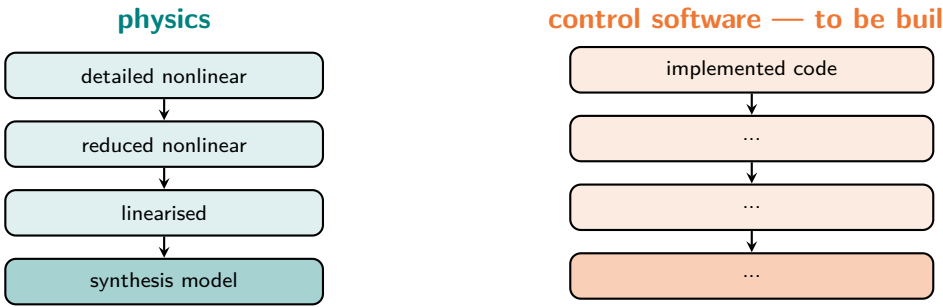
Soundness for every c

Trim (m, k, c) simulates the original automaton $\Rightarrow$ language inclusion $\Rightarrow$ sound verdicts

➔ **one point in the design space — the general problem is wide open**

From one rung to a ladder

Deadline misses are only the best-understood timing failure



- each software level needs what the physics levels have: a **quantified relation** to the level above (what was dropped, and what verdicts carry across)
- the scalability lesson generalises: every added level of fidelity must stay **analyzable**
- ultimately: match plant-model granularity with software-model granularity

Takeaways

1. **Asymmetry:** physics is modelled at many granularities, control software at (essentially) one — this is the control abstraction gap
2. **Probe:** deadline misses show what a good software fault model looks like: the weakly-hard model is expressive, deterministic, and formally grounded ...
3. **Barrier:** ... but its automata grow as $\binom{k}{m}$ — realistic constraints are intractable, and pessimistic simplifications give inconclusive verdicts
4. **Open problem:** find fault models that are meaningful and scalable — sound, compressed, faithful approximations; $\texttt{Trim}(m, k, c)$ is a first data point
5. **Vision:** a full abstraction ladder for control software — delays, missed updates, unfinished computations, synchronisation errors — with quantified relations between levels, matching the one we already trust for physics

maggio@cs.uni-saarland.de — https://www.martinamaggio.com