

Reliable Decision Loops in Socio-Technical Collective Adaptive Systems

Marino Miculan

ISoLA 2026 · REoCAS track · 25 October 2026

MADS Lab

University of Udine, Italy

`marino.miculan@uniud.it`

Motivation

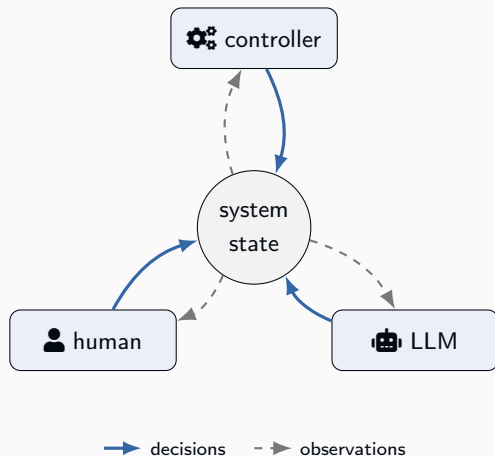
CAS are becoming *socio-technical*

Rigorous engineering of CAS: formal models, quantitative reasoning, guarantees despite decentralisation, openness, adaptation.

Today's collectives increasingly mix:

- **software controllers** executing local control logic
- **human operators** supervising / overriding critical actions
- **generative AI** (LLMs) proposing diagnoses, plans, actions

LLM-based multi-agent systems *are* CAS: many interacting entities, open environments, emergent behaviour, runtime adaptation.



The methodological problem

Formal reasoning presupposes components with **explicit, stable semantics**. But:

Humans

workload, situational awareness, trust
calibration, automation bias, organisational
context

LLMs

probabilistic inference, hallucinations,
context and prompt sensitivity,
distributional shift

Neither is a fully specified controller.

The methodological problem

Formal reasoning presupposes components with **explicit, stable semantics**. But:

Humans

workload, situational awareness, trust
calibration, automation bias, organisational
context

LLMs

probabilistic inference, hallucinations,
context and prompt sensitivity,
distributional shift

Neither is a fully specified controller.

Question

How can we reason rigorously about **reliability** in CAS when some decision components are uncertain, adaptive, or only partially specified?

Shift the object of analysis: from *components* to the **decision loops** they participate in.

1. A formal model of **decision components** in CAS: controllers, humans, LLMs as stochastic decision operators
2. A reliability analysis based on **correlated failures**: joint failure q , not individual accuracy
3. A unified view of **Human-** and **LLM-in-the-Loop**
4. A **guarded architecture** decoupling generation from verification, with safety, monotonicity, cumulative guarantees
5. An illustrative scenario + a **pilot study** on five LLMs measuring marginal and joint failure

Formal results deliberately simple: the point is the unifying framework and the architectural recommendations it yields.

Decision components and decision loops

A formal model of CAS with decision components

$$C = (A, E, S, D, \alpha, \delta)$$

- A : **entities** (services, resources, units): the governed loci
- E : environmental inputs; S : global state space
- $D = \{d_1, \dots, d_k\}$: **decision components** (the *agents*)
- $\alpha : A \rightarrow \mathcal{P}(D)$: governance assignment
- $\delta : S \times U_1 \times \dots \times U_k \times E \rightarrow S$

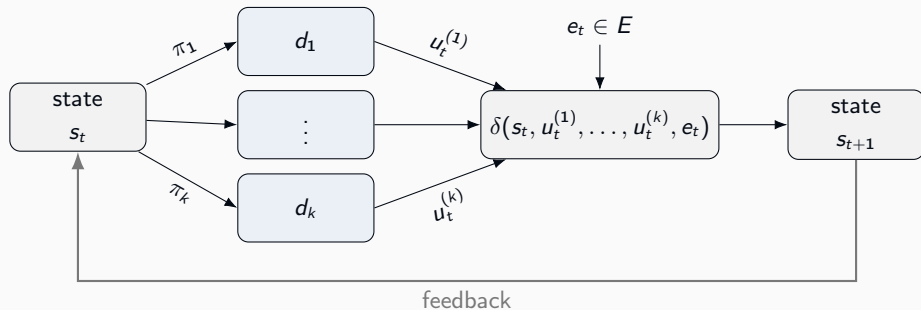
Each component observes and decides:

$$\pi_i : S \rightarrow O_i \quad d_i : O_i \rightarrow \mathcal{D}(U_i)$$

One abstraction for all: what changes is the operator's *semantics*, not the existence of the loop

- deterministic controller: point-mass distribution
- human, LLM: genuinely stochastic, context-dependent

Decision loops



- k loops run concurrently: $u_t^{(i)} \sim d_i(\pi_i(s_t))$
- the successor state feeds back into every loop

Reliability and correlated failures

State-dependent reliability

For each s , let $Adm(s) \subseteq U_1 \times \cdots \times U_k$ be the *admissible joint decisions*.

Marginal $Adm_i(s)$ = projection on the i -th coordinate.

Reliability of each decision component and of the system:

$$R_i(s) = P(u_i \in Adm_i(s)), \quad u_i \sim d_i(\pi_i(s)) \quad R_{\text{sys}}(s) = P\left(\bigcap_{i=1}^k \text{correct}_i(s)\right)$$

- reliability is **state dependent**: ambiguity, partial observability, workload, distribution shift, prompt sensitivity
- independence would give $R_{\text{sys}} = \prod_i R_i$, but in socio-technical systems **errors are correlated**

Feedback amplification

A wrong decision at t degrades s_{t+1} , making later observations less informative; an incorrect AI diagnosis can bias the human operator $\Rightarrow$ reinforcing errors.

Two protective components

d_1, d_2 jointly protect a transition (redundancy, supervision, cross-check).

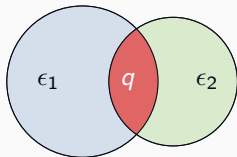
$\epsilon_i(s)$: d_i **fails** to reject/avoid an unsafe decision.

$$p_i = P(\epsilon_i(s)) \quad q = P(\epsilon_1(s) \cap \epsilon_2(s))$$

The system is unsafe only when both fail.

Proposition 1

1. $R_{\text{pair}}(s) = 1 - q$
2. $R_{\text{pair}}(s) \geq 1 - \min(p_1, p_2)$
3. independent failures: $R_{\text{pair}}(s) = 1 - p_1 p_2$
4. for fixed p_1, p_2 , R_{pair} is maximal when q is minimal

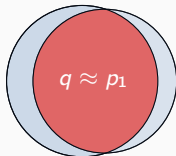


Redundancy helps **only if it reduces joint failure**.

Homogeneous vs. diverse redundancy

Homogeneous pair

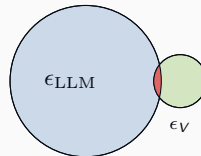
(e.g. LLM + LLM, same-model self-check)



failures largely overlap
 $\Rightarrow$ marginal redundancy gain

Diverse pair

(e.g. LLM generator + formal policy verifier)



structurally different failure mechanisms
 $\Rightarrow q \ll p_1, R_{\text{pair}} \gg 1 - p_1$

Adding a human supervisor or an LLM checker improves reliability only if its failures are **not aligned** with the primary's (automation bias, interface design ...)

LLM-in-the-loop and guarded architectures

$$d_{\text{LLM}}(o) \sim P_{\theta}(u \mid o, c)$$

o : observation, c : prompt / context

θ : parameters, inference config

Sources of uncertainty

- *epistemic*: training
- *contextual*: prompt, retrieval
- *generative*: decoding

Hence R_{LLM} is

- *state-sensitive* (distribution shift)
- *prompt-dependent* ($c \equiv c'$, different outputs)
- *high-variance* under resampling

LLM failure mechanisms differ structurally from those of conventional controllers
 $\Rightarrow$ expect joint failure probability $q \ll \min(p_{\text{LLM}}, p_{\text{ctrl}})$.

The same treatment applies **verbatim to humans** (d_H): different semantics and accountability, same role of *uncertain generator*.

*Decision generation and decision verification
should be decoupled.*

Generator d_{gen}

proposes candidates;
controller, human, or LLM;
possibly adaptive

Verifier V

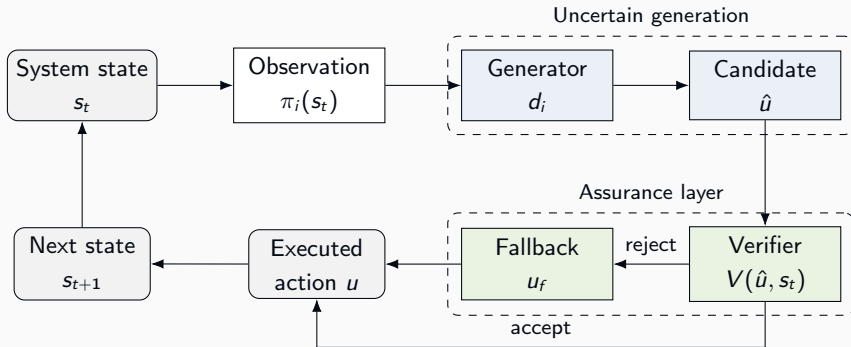
enforces admissibility:
contracts, temporal rules,
policies, monitors

Execution layer

commits only verified
actions; otherwise safe
fallback u_f

In the spirit of **Simplex** (Sha 2001) and runtime assurance for learning-enabled systems, lifted to socio-technical CAS.

The guarded decision loop



$$u = \begin{cases} \hat{u} & \text{if } V(\hat{u}, s_t) = \text{true} \\ u_f \in \text{Adm}(s_t) & \text{otherwise} \end{cases} \quad s_{t+1} = \delta(s_t, u, e_t)$$

Notice that V itself can be *unreliable*.

Safety and monotonicity

A verifier V is *sound* if $\forall u, s : V(u, s) = \text{true} \Rightarrow u \in \text{Adm}(s)$.

Proposition 2

Let $u_f \in \text{Adm}(s)$ for all s .

1. **Safety:** if V is sound then the executed action is in $\text{Adm}(s)$ *whatever the generator proposes*.
2. **Monotonicity:** for an *arbitrary* (even unsound) V , $P_{\text{safe}}^{\text{ver}} \geq P_{\text{safe}}^{\text{gen}} = P(\hat{u} \in \text{Adm}(s))$.

Proof of (2):
$$\begin{aligned} P_{\text{safe}}^{\text{ver}} &= P(\hat{u} \in \text{Adm} \wedge V \text{ accepts}) + P(V \text{ rejects}) \\ &\geq P(\hat{u} \in \text{Adm} \wedge V \text{ accepts}) + P(\hat{u} \in \text{Adm} \wedge V \text{ rejects}) = P_{\text{safe}}^{\text{gen}} \quad \square \end{aligned}$$

$\Rightarrow$ verification **never decreases safety**, however unreliable the verifier.

With *sound* verifier V : gain $\Delta P_{\text{safe}} = p_{\text{err}}$, the generator's error rate.

- (A1) **Verifier soundness** (for Prop.2(1)): a sound verifier never certifies an inadmissible action
completeness not required: rejecting admissible actions costs decision quality, not safety
 - (A2) **Fallback admissibility**: $u_f \in \text{Adm}(s)$ for every s
 - (A3) **Fallback availability**: u_f computable in the current state
- residual risk comes from soundness failing *in practice* (specification gaps) $\Rightarrow$ quantified by Prop. 1, with V as second protective component
 - (A2)–(A3) are **load-bearing**: without a fallback, verification can only *reject* (block execution), not substitute

From steps to trajectories

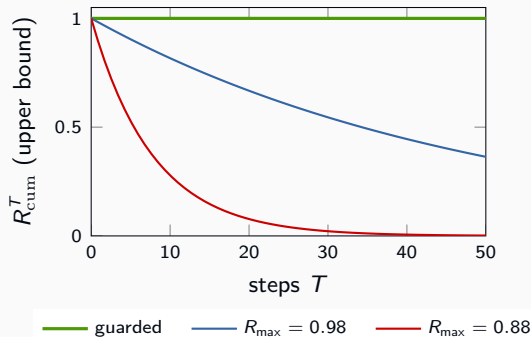
Cumulative reliability over T steps:

$$R_{\text{cum}}^T(s_0) \triangleq P\left(\bigcap_{t < T} \{u_t \in \text{Adm}(s_t)\}\right)$$

Proposition 3

Let $R_{\text{max}} \triangleq \sup_s R_{\text{sys}}(s)$.

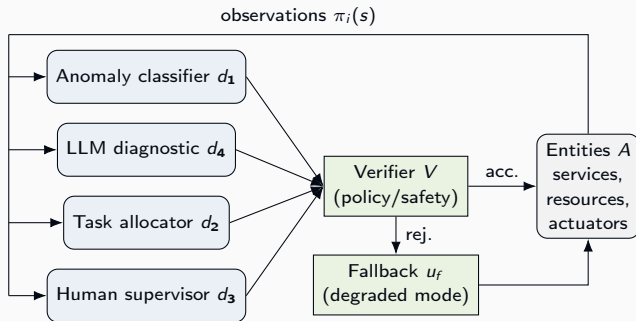
1. unguarded: $R_{\text{cum}}^T(s_0) \leq (R_{\text{max}})^T$
 $\Rightarrow$ exponential decay if $R_{\text{max}} < 1$
2. guarded (sound V , safe u_f):
 $R_{\text{cum}}^T(s_0) = 1$ for all T



Small per-step failure probabilities compound; feedback makes it worse. The guarded loop **eliminates compounding**.

Illustration and pilot study

Illustration: incident response on an infrastructure platform



- LLM proposes mitigations from logs and metrics
- verifier checks policy, safety, resources, consistency
- **human actions go through the same verifier** (shutdowns, reconfigurations, transfers of control)

Illustration: what the numbers say

Inputs p_1, p_2, q *assumed* within literature ranges (LLM planning 10–40% failure; formal checkers $\leq 10^{-2}$); $R_{\text{pair}} = 1 - q$ *computed*. Illustrative data from literature, not measured.

Configuration	p_1	p_2	q	R_{pair}
Unguarded LLM (routine)	0.12	—	—	0.88
Unguarded LLM (distributional shift)	0.30	—	—	0.70
LLM + LLM self-check (routine)	0.12	0.12	0.10	0.90
LLM + formal verifier	0.12	0.02	0.002	0.998
LLM + LLM (distributional shift)	0.30	0.25	0.22	0.78

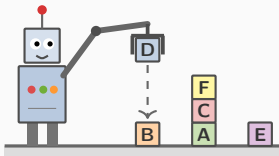
Reliable integration needs verifiers that are **mechanistically diverse** from the generator.

Can we *measure* this?

Pilot study: Blocksworld planning

Classical planning benchmark; standard testbed for LLM planning.

- **30 tasks:** 10 easy (3 blocks), 10 medium (4–5), 10 hard (6–8 blocks, plans up to 24 steps)
- **5 LLMs:** Claude Haiku 4.5, Sonnet 4.6, Opus 4.7, GPT-4o-mini, GPT-4o
- identical prompts, one sample, temp. 0, no chain-of-thought / few-shot / tools
- failure = invalid or non-goal-reaching plan



actions: pickup, putdown, stack, unstack

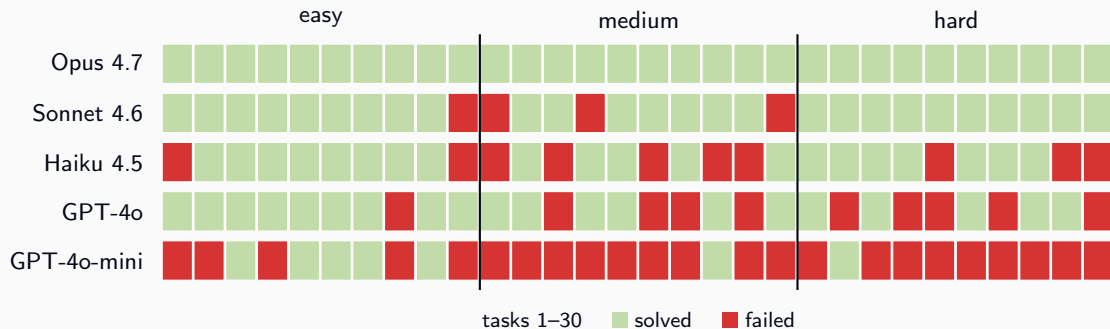
Validator

deterministic plan simulator:
well-formedness, preconditions at
every step, goal.

⇒ a **sound and complete** verifier in
the sense of Prop. 2

 [github.com/miculan/
casllm-pilot-study](https://github.com/miculan/casllm-pilot-study)
tasks, responses, validator, analysis

Who fails where



Clear capability ordering within each vendor. But redundancy depends on *which columns overlap*: compare Sonnet with GPT-4o vs. with GPT-4o-mini.

Measured joint failure

Config.	Pair	p_a	p_b	q	q_{indep}	$q/\min p$	R_{pair}
<i>Same vendor (homogeneous LLM pair)</i>							
Anthropic	Haiku + Sonnet	0.333	0.133	0.067	0.044	0.50	0.933
OpenAI	GPT-4o-mini + GPT-4o	0.767	0.333	0.300	0.256	0.90	0.700
<i>Cross vendor (heterogeneous LLM pair)</i>							
A + O (best)	Sonnet + GPT-4o	0.133	0.333	0.000	0.044	0.00	1.000
A + O (worst)	Sonnet + GPT-4o-mini	0.133	0.767	0.133	0.102	1.00	0.867
<i>Generator + sound formal verifier (diverse)</i>							
LLM + checker	Haiku + plan checker	0.333	0.000	0.000	0.000	—	1.000

$q_{\text{indep}} = p_a p_b$: joint failure under independence; $q/\min p \in [0, 1]$: share of the stronger component's failures also made by the weaker

What the pilot does (and does not) show

1. $q > q_{\text{indep}}$ in **5 of the 6** LLM pairs without Opus; the exception is Sonnet + GPT-4o ($q = 0$): LLM failures are correlated
2. **vendor diversity** $\neq$ **failure diversity**: across vendors, $q/\min p$ ranges from 0 to 1
3. among pairs sharing failures, $q/\min p \in [0.50, 1.00]$ brackets the illustrative 0.83: the gain is pair-specific

Limitations

single domain, $N = 30$, one sample per task, in-distribution;

Sonnet+GPT-4o's perfect complementarity may be a benchmark artefact

Takeaways

- the framework's quantities p_i , q , R_{pair} are **operationally measurable**
- measurements are consistent with the central role of q
- diversity must be **measured**, not assumed from labels

Conclusions

- **Rigorous engineering of CAS** (SCEL 2014, AbC 2015, CARMA 2015, AbU 2023, ...): our operators sit one level above; a policy in SCEL or a process in AbC can instantiate d_i
- **Supervisory control** (Sheridan 1992; Parasuraman & Riley 1997): humans are biased, context-sensitive, not infallible oracles
- **Simplex and runtime assurance** (Sha 2001; Bak et al. 2014; Cofer et al. 2020): we lift the pattern to socio-technical CAS, treat humans and LLMs uniformly, and make q the object of analysis
- **Verified LLM agents** (RV guardrails, Ferrando 2025; LLM plans $\rightarrow$ LTL, Hao et al. 2025): agent-level; we reason at the system level across all components

Conclusions

1. **Architectural diversity** matters at least as much as component accuracy: pair generators with verifiers that fail differently
2. Focus on **robust architectures**, not raw model performance: even an unreliable generator is safe under sound verification
3. **One pattern** for human and AI components: unified socio-technical engineering
4. From component-centric to **loop-centric assurance**

Next steps

- coalgebraic formulation of decision loops
- larger, richer, out-of-distribution evaluation; online estimation of $R_i(s)$
- stochastic dynamics (δ as a Markov kernel); adversarial behaviour
- the role of the fallback: what does the generator buy? what if there is none?
- multi-level loops (humans/LLMs *in, on, out of* the loop); deliberative governance

Thank you!

Questions?

 `marino.miculan@uniud.it`

 `github.com/miculan/casllm-pilot-study`