PERSLIS RESEARCH

Fail-First Models

Failure becomes structure. Structure changes the next attempt.

How a model can learn only from failures it has actually had, keep each lesson as a rule that cites its evidence, and never widen what it is allowed to do

Perslis Research | September 2026

Research prototype · simulation and games · not a certified safety system

Abstract. A model that learns from experience must fail before it learns: a rule is justified only by evidence of the failure it prevents. The design question is where a failure may lead. We define a *fail-first model*: it learns only from failures the environment, not the model, has verified; keeps each as an explicit constraint that cites the failures behind it; and updates no weights. Fail-first and fail-safe are two names for one model: fail-first is how it learns, fail-safe is what that learning may never change. When authority is computed before the learner, from sourced licences and human orders, and the learner may only remove and reorder options, every decision lies in the admissible set or a safe hold for every learner state (Theorem 1), and an order outranks any amount of experience (Theorem 2). The evidence: on real Atari 2600 ROMs, from raw pixels, one failure memory gains 32% on Space Invaders and loses 12% on Freeway, because avoiding failure breaks when risk and objective share an action; 54 individually justified rules score below none, and retiring rules by coverage takes dead ends from 6 to 0; on DOOM the memory is parity ($t = 0.78$); in Fallout, after a credit-assignment fix, a pilot that died 49 times at one guard in a live run dies once in a scripted scene, then chooses the peaceful line; a simulated drone relearns a wiped route in 7 rounds. Admitting a model's 85.8%-accurate guesses as facts cut correct identification from 1.000 to 0.753.

1 Introduction: every model fails

Every model fails. A controller drifts into an obstacle, a game agent walks back into the fight that killed it, a language model states a fact that is not one. Most engineering effort goes into failing less often. This paper is about a different question, the one engineers ask of brakes, lifts and reactors: *when this fails, where does the failure lead?*

A model that learns from experience cannot avoid the question, because it cannot learn a rule from a failure it has not had. Before an action has been tried there is no evidence that it fails, only a guess. Two kinds of system sidestep this. Systems that never learn carry only rules written in advance; they need not fail, but they never become better than their authors. Systems trained on a large corpus inherit other people's failures second-hand, as statistical patterns they cannot cite. A third kind learns from its own failures, in its own environment, and keeps each lesson as a readable rule. We call it a **fail-first model**.

A fail-first model has to be allowed to fail, so the environment it fails in must be arranged so that failing cannot widen what it does. That arrangement is what makes it **fail-safe**. The two words describe one model from two sides: *failing first is how it learns; being fail-safe is why failing first is acceptable.* The short form, used throughout, is:

The learning loop can change behaviour. It cannot change the safety floor.

The *floor* is everything that decides what is permitted; the *learner* is everything that decides, among permitted things, what to try. Figure 1 shows the loop: STATE → ATTEMPT → VERIFY → (success: *preserve*)

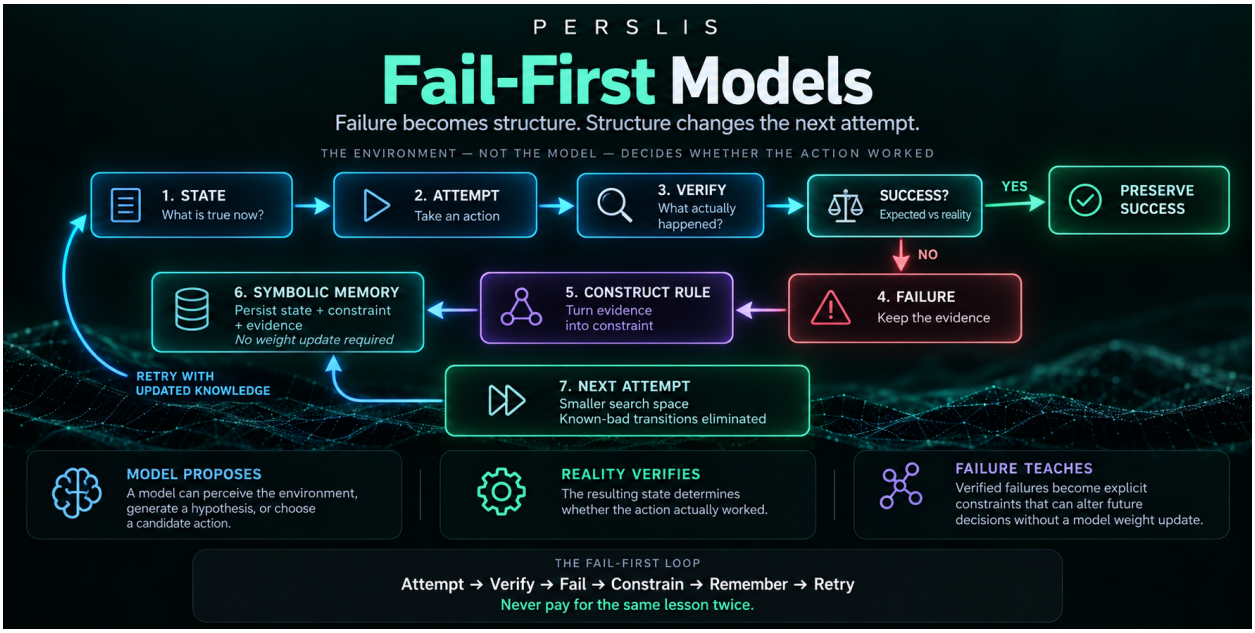


Figure 1. The fail-first loop at a glance. The environment, not the model, decides whether an action worked. A verified failure is kept with its evidence, turned into an explicit constraint and stored in symbolic memory, so the next attempt starts from a smaller search space. No weight is updated. Sections 3–7 make each box precise and measure it.

or FAILURE (*keep the evidence*) → CONSTRUCT RULE → SYMBOLIC MEMORY → NEXT ATTEMPT, with known-bad transitions eliminated. Written as a slogan: *Attempt → Verify → Fail → Constrain → Remember → Retry. Never pay for the same lesson twice.*

1.1 Contributions

C1. A definition, and one model under two names (§2–§3). We define a fail-first model precisely and show that the architectural placement of its learner, innermost, inside an authority computed before it runs, makes it fail-safe. We state and prove bounded learning (Theorem 1), that orders outrank experience (Theorem 2), that orders only narrow (Proposition 3), a precise form of "never pay twice" with its two exceptions (Proposition 4), that pure accumulation is monotone toward paralysis (Proposition 5), and that a safety stage applied after a learned plan can only slow it (Proposition 6).

C2. A measured case study on two real Atari games (§4). The same failure memory, reading raw 210 × 160 pixels with zero emulator RAM and no weights, gains 32% on Space Invaders and loses 12% on Freeway. The pair is the finding: failure avoidance breaks when risk and objective share an action. We report the five defects that each produced a plausible wrong answer, the saturation point (54 rules score below no learning), and the follow-up arms that fixed one regression and lost the win.

C3. The same loop carried into a runtime (§5–§7): coverage-aware rule retirement (dead ends 6 → 0), evidence tiles that trace one refusal back to the individual deaths behind it, a commanded game runtime (VDSG) on DOOM, Wolfenstein 3D and Fallout in which tests pin the authority boundary, and a drone course in simulation where the loop relearns a route from a wiped memory and a stop-distance floor acts after the learned plan.

C4. The same discipline on the knowledge side (§8). A fail-first model admits only verified outcomes as evidence; a fail-safe model admits only verified facts. We report what happened when that was broken on purpose: a frontier model's guesses, 85.8% accurate, cut measurement cost by 38.4% and

cut correct identification from 1.000 to 0.753.

1.2 What we claim, and what we do not

None of the ingredients is new, and §9 names where each comes from. Removing unsafe actions at run time is shielding [1]; learning rules from failure goes back to PRODIGY, CHEF and Ripple-Down Rules [16, 9, 6]; remembering a failure so the same mistake is not repeated is the second principle of Haralick and Elliott's fail-first search [10]; and learning a shield from catastrophic failures has been done by Shperberg, Liu and Stone [19, 20], the closest prior work. What we claim is exactly this:

> To our knowledge, the first model to combine (1) rules built from observed failures, each citing the failures that earned it, (2) an authority computed before the learner from sourced cards and human orders, which the learner provably cannot widen, and (3) no neural network in the loop that decides.

If an earlier system does all three, we will cite it. We do not claim that the model plays games well, that the results generalise beyond the environments measured, or that anything here is qualified for a safety-critical path. Every figure is ours, and none has been replicated by a third party. The fail-safe guarantee is claimed for the VDSG runtime, where tests pin it; in the Atari prototype the parameter-search layer is not yet separated from the floor, and the guarantee is not claimed there (§10).

2 Fail-first and fail-safe: one model, two names

Fail-safe is an engineering term older than computing. George Westinghouse's automatic air brake (1872) holds the brakes off with air pressure; if a hose bursts, pressure is lost and the brakes apply. Elisha Otis demonstrated his safety catch in 1854 by having the hoist rope cut while he stood on the platform. A fail-safe design does not prevent failure. It decides, in advance, where failure is allowed to lead.

A modern learning model, left to itself, fails the other way. On knowledge it fails open: asked what it does not know, a generative model still produces the most plausible continuation [11]. On constraints it learns around them: a system optimised against a measure exploits the gaps in the measure, which the literature calls reward hacking or specification gaming [3, 14]. If the learner can touch the rules, the cheapest improvement is often to loosen them. And its usual gate is a single number, a confidence threshold, which (§3.8) is a single price on every kind of failure.

A fail-safe model is built the other way round. It holds three properties together:

(i) **It fails closed.** When evidence for an answer or action is missing it returns *unknown*, refuses, or holds still.

(ii) **Its learning is bounded.** The learner may only remove or reorder options inside an authority computed before it runs. No amount of experience adds a permission.

(iii) **Its refusals are accountable.** Every refusal names the rule that produced it, and every learned rule cites the failures that earned it. A human can read, challenge and delete any of them.

A fail-first model is the same model described by how it learns. Table 1 sets the two halves side by side. A system that failed first without being fail-safe would be learning by breaking things. A system that was fail-safe without failing first would be a fixed rulebook that never improves. It is useful only when both are true.

Neighbouring uses of the words. "Fail-first" and its relatives are used elsewhere with related but different meanings (Table 2). The closest cousin is in constraint satisfaction. Haralick and Elliott showed that backtracking search improves when it follows two principles: try first where failure is most likely, so dead ends are found early, and remember what has been done so the same mistake is not repeated [10]. The

Table 1. One model, two names. Neither half works alone.

	fail-first	fail-safe
describes	how it learns	what learning is allowed to change
loop steps	fail, observe, explain, build rule	verify, retry inside the floor
guarantee	every rule cites the failures that earned it	no rule can widen what the model may do
without the other	learning by breaking things	safe, but never improves

first principle is what the phrase "fail-first" has meant in search ever since. A fail-first model applies the *second* principle to a whole model instead of a search tree: every failure is remembered as a rule, so the same mistake is not made twice.

Table 2. Neighbouring terms. Only the first row is the subject of this paper.

term	field	meaning
fail-first model	AI models (this paper)	Learns from its own verified failures, as explicit rules, inside an authority it cannot widen.
fail-safe	engineering	A failure drives the system to a safe state (air brakes, safety lifts).
fail-fast	software engineering	Stop at the first error instead of continuing in a bad state.
fault-tolerant	engineering	Keep operating through a fault. A fail-safe system instead goes to the state that cannot do harm.
fail-first principle	constraint satisfaction	Search heuristic: branch first where failure is most likely [10].

When failing first is not acceptable. Some failures cannot be allowed even once: a collision with a person, a wrong dose, an irreversible transfer. For those the fail-first half does not apply. The floor has to be written in advance, not learned, and the learner operates only above it. This is where the boundary between the two halves sits: learned rules handle the failures one can afford to have; written rules handle the ones one cannot.

3 The fail-first loop

3.1 The loop, step by step

Figure 2 draws the loop and the floor beneath it; Table 3 says what each step is in the systems measured later. The step that defines the class is VERIFY: *the environment, not the model, decides whether the action worked.* A ROM loses a life, a game engine reports a death, a simulator reports a contact or a lap time. The model's own opinion of how it did never enters memory.

3.2 Authority is computed first

A fail-safe model does not first ask "what should I do?". It asks "what am I allowed to do here?" and computes the answer from three sources, none of which the learner can edit: what the situation offers (rules over facts read from the system's own state); what the sources license (typed, sourced cards; in the game runtime, cards compiled from a printed manual, each licence carrying its page receipt, in a store opened read-only); and what a human has ordered (standing orders in plain language, such as "don't fire" or "hold position", which can only narrow). Figure 3 draws the nesting.

3.3 Setup

Definition 1 (situation, goals, authority)**.** Let s be the situation at a decision and G a finite vocabulary of goals. Let $\mathrm{App}(s) \subseteq G$ be the goals the rules find applicable, $\mathrm{Lic}(s) \subseteq G$ the goals licensed by the

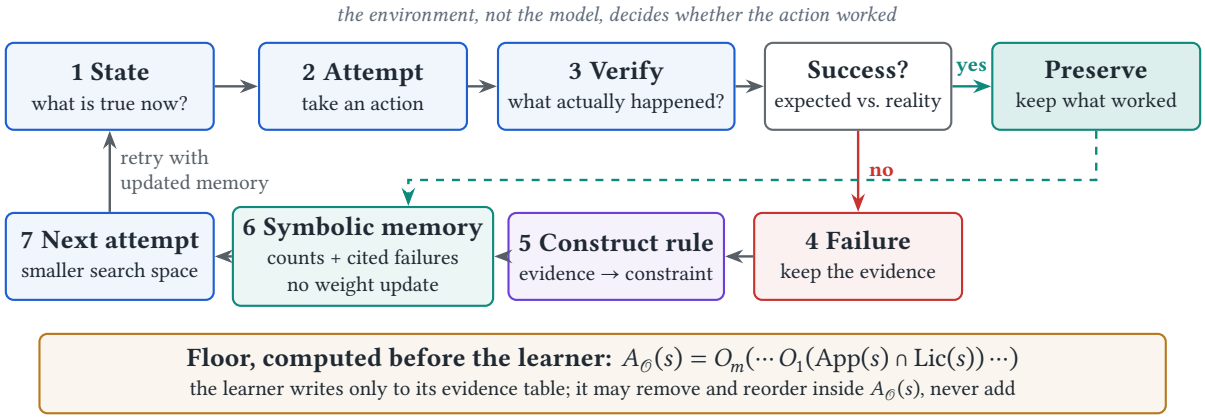


Figure 2. The fail-first loop (top two rows) and the floor it runs on (bottom). A success is preserved; a verified failure is kept with its evidence, turned into a rule only when the evidence is statistically clear (§3.5), and stored as readable counts. Every attempt, including the next one, is drawn from the admissible set computed before the learner runs (Definition 1), which is what makes the loop fail-safe (Theorem 1).

sourced cards in force, and $O_1, \ldots, O_m$ the standing orders, each a map on sets of goals with $O_i(X) \subseteq X$. The *admissible set* is

$$A_{\mathcal{O}}(s) \;=\; O_m\big(\cdots O_1\big(\mathrm{App}(s) \cap \mathrm{Lic}(s)\big)\cdots\big) \;\subseteq\; \mathrm{App}(s) \cap \mathrm{Lic}(s).$$

$A_{\mathcal{O}}(s)$ is computed from s, the card store and the orders; it does not read the learner's state.

Definition 2 (the learner). The learner's state M is a finite table of counts: for each situation pattern and goal, how often it was tried and how often it ended in a verified failure, together with the identifiers of those failures. From M it derives two operations on any non-empty set of goals X: a *veto* V_M with $\varnothing \neq V_M(X) \subseteq X$, and a *ranking* π_M, a permutation of its argument. Concretely, if $C_M(s)$ is the set of goals condemned in s (§3.5), then $V_M(X) = X \setminus C_M(s)$ when that is non-empty and otherwise the single least-condemned goal of X. A fixed rule policy c picks one element of an ordered non-empty set. The decision is

$$\delta(s, M) = \begin{cases} c\big(\pi_M(V_M(A_{\mathcal{O}}(s)))\big) & \text{if } A_{\mathcal{O}}(s) \neq \varnothing, \\ \mathrm{w} & \text{otherwise (hold still: the safe state).} \end{cases}$$

Definition 3 (fail-first model). A decision model is *fail-first* if (a) its learned state changes only by recording outcomes that the environment has verified, not outcomes the model has estimated; (b) that state is a finite, readable table (Definition 2) and no continuous parameter is fitted; (c) every constraint it holds is justified by a statistical bar on verified failures (§3.5) and cites the failures that earned it; and (d) every constraint can only remove or reorder options.

Definition 4 (fail-safe model). A decision model is *fail-safe* if its admissible set is computed without reading the learner's state, and for every situation s and every learner state M, $\delta(s, M) \in A_{\mathcal{O}}(s) \cup \{\mathrm{w}\}$.

3.4 The guarantees

Theorem 1 (bounded learning). *For every situation s and every learner state M, however much it has learned, $\delta(s, M) \in A_{\mathcal{O}}(s) \cup \{\mathrm{w}\}$.*

Proof. If $A_{\mathcal{O}}(s) = \varnothing$ the decision is w. Otherwise $V_M(A_{\mathcal{O}}(s)) \subseteq A_{\mathcal{O}}(s)$ by Definition 2, a permutation adds no elements, and c returns an element of its argument. So the chosen goal lies in $A_{\mathcal{O}}(s)$. ∎

The proof is one line on purpose. The guarantee does not depend on the learner being clever, correct or well-trained; it depends only on where the learner sits. A bad learner makes worse choices among

Table 3. Each step of the loop in the systems measured in this paper. Middle column: the Atari floor of §4; right column: the same step in the VDSG game runtime (§6) and the drone (§7).

step	Atari floor	VDSG and the drone
1 State	A situation bucketed from raw 210×160 pixels, zero emulator RAM.	Facts read from engine or simulator state: health, who is present, free space by sector.
2 Attempt	The pilot proposes from the admissible set.	The rules choose a goal inside the admissible set; a model may propose, never admit.
3 Verify	The ROM decides: a life lost, read at the true impact frame.	The game or simulator decides: a death, a contact, a lap time.
4 Failure	A card: the situation and the action taken.	Charged to the decision that opened the episode: the line that started the fight, not the heal inside it.
5 Construct rule	A pattern becomes a rule only when its Wilson lower bound clears the base rate.	The same bar; one unlucky death does not become a rule.
6 Symbolic memory	Rules are counts that cite experience identifiers, traced into 203 evidence tiles.	An evidence table the learner writes; the licensing cards are read-only to it.
7 Next attempt	The admissible set shrinks, bounded by rule retirement.	The learner can only narrow and reorder; human orders outrank it.
Success	Later arms keep both sides of every outcome, so good actions are remembered too.	The drone keeps a change only if the round's cost falls.

permitted options; it cannot make an unpermitted one. Definition 2 therefore makes any fail-first learner fail-safe in the sense of Definition 4: that is the formal content of "one model, two names".

Corollary 1 (no learned permission). *Let $R(s) = \{\delta(s, M) : M \text{ any learner state}\}$. Then $R(s) \subseteq A_{\mathcal{O}}(s) \cup \{\mathrm{w}\}$. Experience changes which permitted goal is chosen, never whether it is permitted.*

Theorem 2 (orders outrank experience). *If the orders narrow the admissible set to a single goal, $A_{\mathcal{O}}(s) = \{g\}$, then $\delta(s, M) = g$ for every M, even if M records that g has failed every time.*

Proof. $V_M(\{g\})$ is a non-empty subset of $\{g\}$, so it equals $\{g\}$; the permutation and c return g. ∎

Theorem 2 encodes a deliberate choice: a human order outranks the model's experience. The learner can report that an order is costly; it cannot countermand it. The veto is never empty for a related reason: standing still and failing is not an adaptation, so when experience condemns everything the least-condemned permitted option comes back.

Proposition 3 (orders only narrow). *Adding an order never adds a goal: for any additional order O', $A_{\mathcal{O},O'}(s) \subseteq A_{\mathcal{O}}(s)$.*

Proof. Immediate from $O'(X) \subseteq X$. ∎

These three statements are not only on paper. The VDSG paper states order narrowing and the bounded learner as propositions [24], and the Fallout test suite pins them in code (§6.3):

- `test_the_learner_can_never_add_a_goal`
- `test_the_learner_cannot_overrule_a_standing_order`
- `test_ranking_is_a_permutation_and_nothing_more`

We re-ran them while writing this paper; they pass.

The slogan "never pay for the same lesson twice" is true only with its exceptions stated.

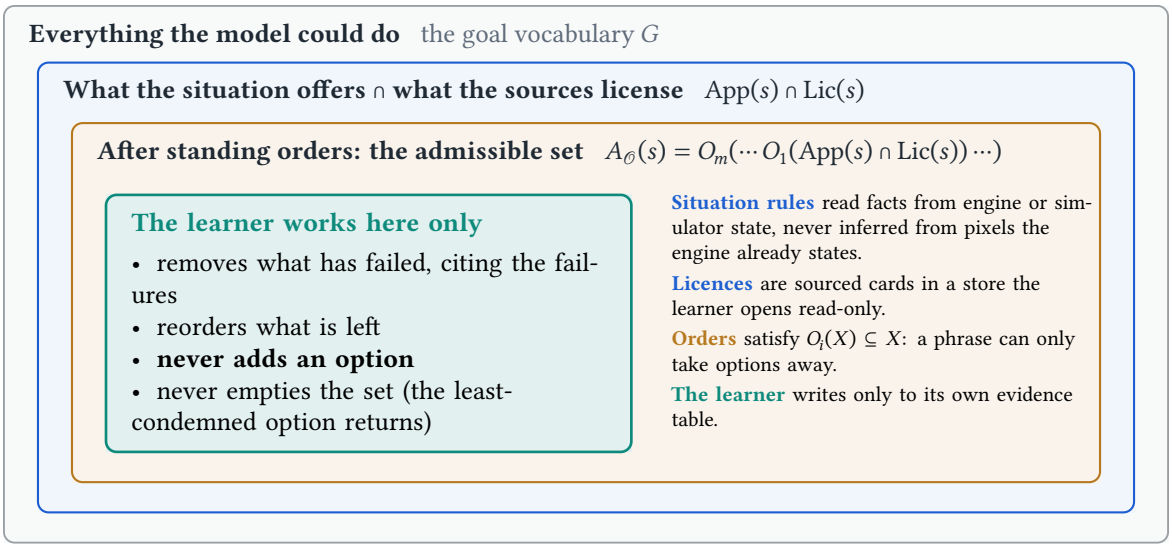


Figure 3. The order of authority. Each box can only shrink the one outside it; the learner sits innermost and cannot reach outward. Two stores do not mix: the cards say what is permitted and hold the source for it; the evidence table says what has gone wrong and holds the experiences for it. Nothing the model lives through can change what it is licensed to do.

Proposition 4 (never pay twice, with two exceptions). *Suppose goal g is condemned in situation s by M. Then $\delta(s, M) \neq g$ unless (a) every goal of $A_{\mathcal{O}}(s)$ is condemned and g is the least condemned, or (b) $A_{\mathcal{O}}(s) = \{g\}$.*

Proof. If some goal of $A_{\mathcal{O}}(s)$ is not condemned, $V_M(A_{\mathcal{O}}(s)) = A_{\mathcal{O}}(s) \setminus C_M(s)$ excludes g, and neither π_M nor c can reintroduce it. Otherwise V_M returns the least-condemned goal, which is case (a); case (b) is Theorem 2. ∎

A third exception is operational rather than logical: a condemned goal returns if its rule is *retired* (§5.1). Retirement is a change to M, so Theorem 1 still holds after it. The reason retirement is needed is the next proposition. Fix a finite set $\mathcal{S}$ of encountered situations and, for each $S \in \mathcal{S}$, the set $\mathrm{Seen}(S)$ of goals observed there. For a rule set $\mathscr{R}$ let $\mathrm{Blk}_{\mathscr{R}}(S)$ be the goals of $\mathrm{Seen}(S)$ some rule blocks, $\mathrm{left}_{\mathscr{R}}(S) = \mathrm{Seen}(S) \setminus \mathrm{Blk}_{\mathscr{R}}(S)$, and $D(\mathscr{R}) = \{S : \mathrm{left}_{\mathscr{R}}(S) = \emptyset\}$ the *dead ends*.

Proposition 5 (accumulation only moves toward paralysis). *If $\mathscr{R} \subseteq \mathscr{R}'$ then $\mathrm{left}_{\mathscr{R}'}(S) \subseteq \mathrm{left}_{\mathscr{R}}(S)$ for every S, and $D(\mathscr{R}) \subseteq D(\mathscr{R}')$.*

Proof. A goal blocked by some rule of $\mathscr{R}$ is blocked by the same rule in $\mathscr{R}'$, so $\mathrm{Blk}_{\mathscr{R}}(S) \subseteq \mathrm{Blk}_{\mathscr{R}'}(S)$; take complements in $\mathrm{Seen}(S)$. ∎

In a dead end the model does not stall (the veto returns the least-condemned goal), but the choice is no longer made by evidence. A learner that only accumulates rules can therefore only lose the ability to choose. Every rule may be individually correct and the set still paralysing; §5.1 measures exactly that.

3.5 When does a failure become a rule?

A rule should form when a pattern is clearly more dangerous than normal, not when it was unlucky once. For a pattern tried n times with d failures, the point estimate $\hat{p} = d/n$ is misleading at small n: one failure

in one try reads as 100%. The decision therefore uses the lower edge of the Wilson score interval [22],

$$L(d,n) \;=\; \frac{\hat{p} + \dfrac{z^2}{2n} - z\sqrt{\dfrac{\hat{p}(1-\hat{p})}{n} + \dfrac{z^2}{4n^2}}}{1 + \dfrac{z^2}{n}},$$

and a pattern is condemned when it has enough tries and its lower bound clears the base rate b (the overall failure rate) by a margin μ:

$$\text{condemn}(d,n) \iff n \geq n_{\min} \;\wedge\; L(d,n) > b + \mu.$$

We use $z = 1.645$, the one-sided 95% bound (equivalently the lower edge of a two-sided 90% interval); the Fallout lane uses $z = 1.64$. Table 4 and Figure 4 give computed values.

Table 4. Wilson lower bounds at $z = 1.645$, computed from the formula above. The point estimate overstates the evidence at small n.

failures / tries	point estimate	lower bound L	reading
1 / 1	100%	0.270	One failure is weak evidence, whatever the point estimate says.
4 / 4	100%	0.596	Four in a row is strong evidence.
13 / 19	68%	0.496	Against a base rate near 0.19, clearly condemned.
3 / 4	75%	0.356	Suspicious, not yet proven.

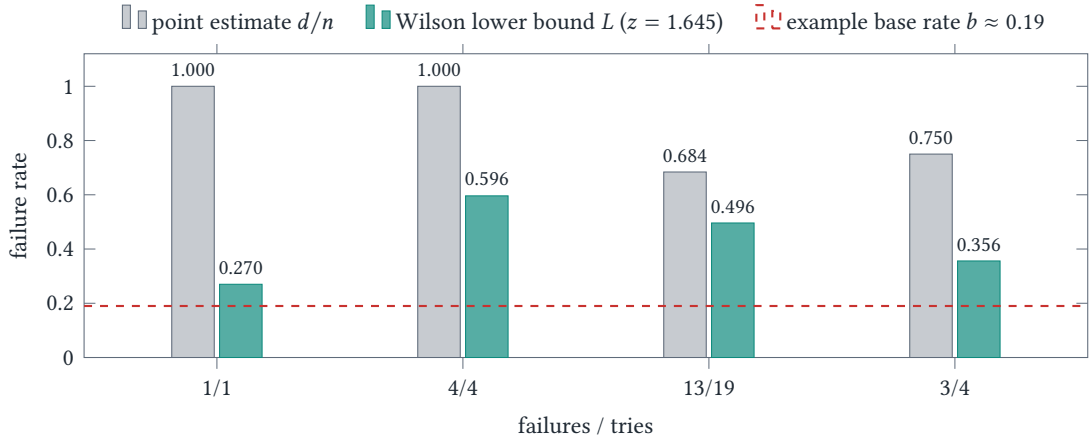


Figure 4. Why a rule needs more than one failure. Grey: the naive rate. Teal: how low the rate could plausibly be. One death in one try clears a base rate of 0.19 on the point estimate but not by a margin on the lower bound; four in four does.

Why the bar is relative. In an environment where the model survives 99.5% of decisions, a situation that kills it 5% of the time is ten times more lethal than normal and must be refused, yet it never approaches an absolute threshold such as 60%. On Space Invaders the base death rate was 2.8%, and an absolute 60% gate produced **0 rules from 268 real failures** (§4.5). That is a measured design decision, not a preference. A relative bar has the opposite hazard: a model that dies at everything has a base rate near 1, and nothing is "worse than average". The Fallout lane therefore adds two clauses: a pattern fatal on at least 2 tries

at $\geq 90\%$ is condemned outright, and the relative bar is capped at an absolute lethality of 0.6, so that the more a model dies the more, not the less, it can learn. Where enough alternatives have been tried in the same situation, the base rate is replaced by the failure rate of the *other* goals in that situation, so that a dangerous situation is not blamed on the goal that happened to be chosen in it.

3.6 Credit assignment: blaming the right decision

When a failure happens at time t_f, some set of earlier decisions $B(t_f)$ receives the blame. The naive choice, the last w decisions, is usually wrong: the decisions just before a failure are often the *response* to the danger, not its cause. The rule has to charge the decision that opened the dangerous episode.

We have published two cases in which we got this wrong, and both produced believable learning curves. In Atari, the emulator reported a lost life at the end of a 127-frame death animation, so every failure was recorded after the ship was already destroyed: 0 of 374 blamed frames showed the hazard; at the true impact frame, 17 of 17 did (§4.5). In Fallout, blame covered only the last six decisions inside a fight: 26 deaths were charged to HEAL, and the 446 conversational replies that started those fights were charged nothing (§6.3). Credit assignment is where a learning system fails silently. That is why a fail-safe model must never let a mis-blamed rule widen its authority: by Theorem 1, a wrong rule can only make it more cautious. It can still make it useless, which is a performance failure, not a safety one.

3.7 Retry: improvement that cannot regress

When the model improves a plan, for example a route, it changes one parameter θ at a time and keeps the change only if a cost that prices failure explicitly falls. For the drone (§7) the cost of a round is

$$C(\theta) = T(\theta) + \kappa\, k(\theta) + P(\theta),$$

the lap time, plus $\kappa = 15$ s per contact k, plus a penalty P for not finishing (60 s plus 2 s per metre short), averaged over a fixed set of starts. A change is kept only if C falls below the best so far by at least 0.05 s; a rejected move halves its step. The accepted sequence therefore satisfies $C_{j+1} \leq C_j - 0.05$ by construction: the best-so-far can only improve, and because the simulator is deterministic it can be replayed exactly. The floor still applies to every trial, because it acts *after* the learned plan.

Proposition 6 (a floor applied after the plan can only slow it)**.** *Let the learned plan propose a forward speed u, and let the command be $v = g_k(\cdots g_1(u)\cdots)$ where each stage satisfies $g_i(x) \leq \min(x, \beta_i(s))$ for a bound $\beta_i(s)$ computed from the current state without reading the learner's memory. Then $v \leq \min(u, \beta_1(s), \ldots, \beta_k(s))$, whatever u the learner proposes.*

Proof. By induction: $g_1(u) \leq \min(u, \beta_1)$, and if the output of stage $i-1$ is at most $\min(u, \beta_1, \ldots, \beta_{i-1})$ then stage i returns at most that value and at most β_i. ∎

In the drone's time-trial guidance the stages are, in order: the learned cap for the current 6 m block (never below 0.8 m/s); a stop-distance stage that, when $\sqrt{2a(d-m)}$ with $a = 4\,\mathrm{m/s^2}$ and margin $m = 1.5$ m is below the current command, replaces it by $\max(0.3, \sqrt{2a(d-m)})$; and the upstream reflex, which below its trigger distance returns at most 0.25 m/s or a reverse command. Each satisfies the hypothesis, with the stop-distance bound carrying a 0.3 m/s creep floor. The proposition covers forward speed only; lateral and vertical manoeuvres are the upstream project's reactive layer and the floor's climb rule.

3.8 When is refusing the right call?

Take an action with failure probability p, failure cost c, and value v if it succeeds. It beats doing nothing when

$$(1-p)\,v - p\,c > 0 \quad \Longleftrightarrow \quad p < \frac{v}{v+c}.$$

A terminal failure forfeits everything that could follow, so c is very large and the threshold on p collapses toward zero: refusing anything clearly dangerous is right. A recoverable failure costs only lost ground, so c is small; if the dangerous action is also the only one with value, refusing it is wrong. A confidence threshold fixes one cut-off on p for every action, which amounts to assuming one c/v everywhere (Figure 5). No single value is right in both regimes. That is the formal reason a fail-safe model must know what kind of failure it is avoiding, and it is the open problem of §10: Freeway (§4) is the measured instance.

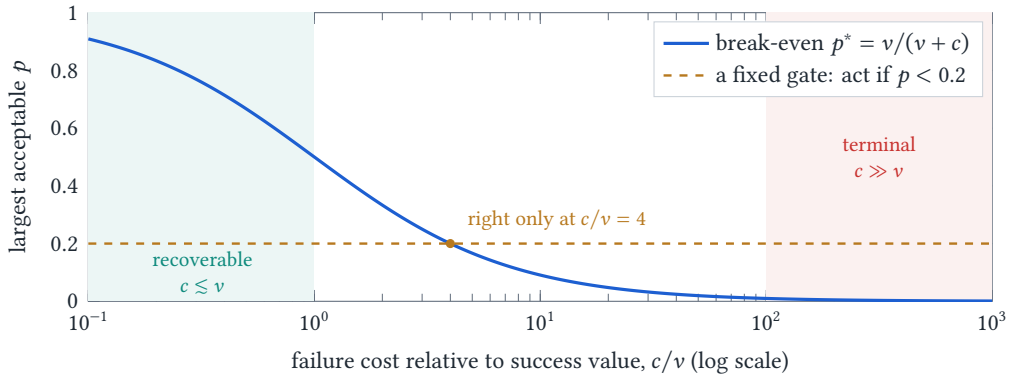


Figure 5. The break-even failure probability as a function of how much a failure costs. A fixed confidence gate is a horizontal line: too permissive where failure is terminal, too strict where it is recoverable. Arithmetic, not a measurement.

4 Case study: two Atari games

4.1 Setup

The first fail-first floor was pointed at two real Atari 2600 ROMs, Space Invaders and Freeway, through the Arcade Learning Environment (ALE 0.11.2, Stella emulator) [4]. Three commitments make the result meaningful:

- **No emulator RAM.** Everything is read from the raw 210 × 160 RGB frame, found by colour. Reading entity positions out of the emulator's memory makes the task nearly trivial.
- **No weights, no gradients.** A failure writes a *card*: a coarse symbolic signature of the situation plus the action taken. When a signature has enough evidence (§3.5) it becomes a *rule* that removes that action from the admissible set at the moment of decision. The memory never suggests an action; it only takes them away. If every candidate is condemned, the least-condemned one is taken.
- **Every rule cites its evidence.** A rule reports its trials, failures, observed rate, lift over the base rate, and the experience identifiers that built it. Rules must also name a hazard: the only two that formed early without one were "do not move when the screen is empty", left and right, which is the shape of a superstition.

The pipeline is identical in both games; only the vision layer and the hazard vocabulary change. On Freeway the code imports the Space Invaders failure memory unchanged, with no per-game strategy. Evaluation uses held-out seeds (Space Invaders 9000–9015, Freeway 900–907), frameskip 1, and a memory that is frozen during every evaluation. The result was frozen on 25 September 2026 on two experiment cards, which are not edited to make later metrics look better; a later version gets its own card and cites them.

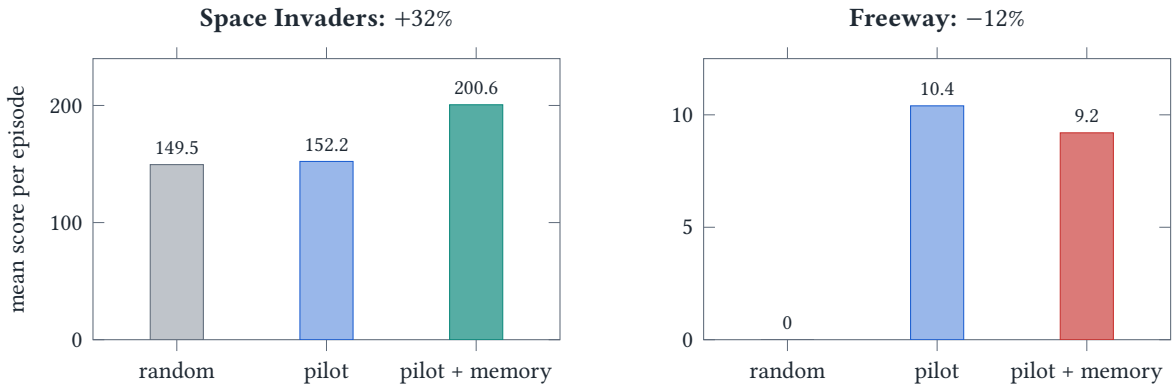


Figure 6. The same failure memory on two ROMs. Left: held-out seeds 9000–9015, memory frozen. Right: seeds 900–907. The architecture transferred between games; the learning did not.

4.2 Results

On Space Invaders the memory is worth +32%: random play 149.5, the rule-based pilot 152.2, pilot plus memory 200.6 (deaths 46 against 43). At a more conservative operating point with 21 rules the same comparison gave 184.4 against 152.2 (+21%) with deaths falling from 46 to 23. On Freeway the identical mechanism *loses* 12%: random 0.0, pilot 10.4, pilot plus memory 9.2. The Freeway memory wrote 17,611 cards over 169 signatures and formed three rules, **every one blocking up**: `car dx+2 dy+1|up` (2 of 4 died), `car dx+4 dy+1|up` (2 of 3), `car dx-4 dy+1|up` (2 of 3).

4.3 The finding: what a failure destroys

The pair is the finding, and the reason is not that one game is harder. In Space Invaders the ship fires on every frame wherever it dodges, so safety is bought for free: risk and objective are separable. In Freeway the hazard lies across the objective: `up` is both the dangerous action and the only scoring action. A learner whose only operation is "I died doing X, therefore inhibit X" correctly concludes that `up` is dangerous, and correctly stops playing. The rule is true and useless, because *risk is not the same thing as bad.* The learner knows what an action costs, never what it is worth.

Put in the terms of §3.8: a death in Space Invaders is *terminal.* It removes every remaining reward in the episode, so its true cost is the expected remaining return, and an absolute veto prices it correctly, by accident. A collision in Freeway is *recoverable*: the chicken is knocked back and play continues, so the cost is only the lost ground. One failure price cannot be right in both regimes, and a confidence threshold is a single failure price.

Read the pair together. Anyone citing the +32% without the −12% is misreading the work. The mechanism transferred between games; the learning did not.

4.4 Saturation: 54 correct rules are worse than none

The learning curve (Figure 7) rises from 162.5 with no training to a peak of 276.2 after 50 episodes, then declines to 246.7 at 100 episodes, with the working set bounded at 30 rules. Remove the bound and 100 episodes collapse to **167.1 with 54 rules**. Every one of those 54 rules was individually justified by real deaths and at least 50% lethal; together they paralyse the pilot. This is Proposition 5 measured: the admissible set only ever shrinks, so a floor that never retires a rule eventually refuses everything. Bounding the working set is load-bearing, not a tuning detail, and it is the argument against our own approach, so we state the number.

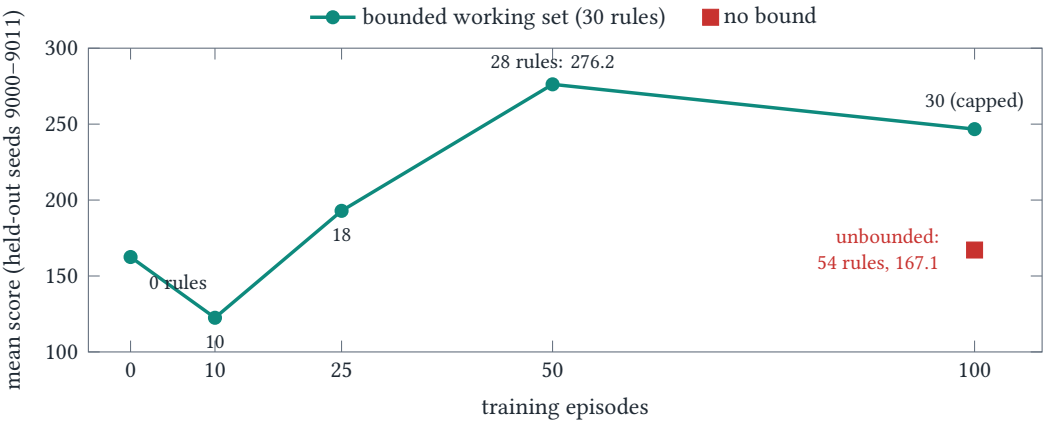


Figure 7. Space Invaders learning curve from the frozen V1 card (memory frozen during each evaluation; rule counts annotated). The dip at 10 episodes is in the record and is not smoothed away. The red square is the same 100 episodes without the bound on the working set.

4.5 Five defects, each of which produced a plausible wrong answer

The experiment card records every defect found, because each one silently returned a believable result before it was caught (Table 5). The largest is a credit-assignment error: the floor was learning from the wrong moment entirely and still produced a curve that looked like learning. We report these because a result that only ever went up would be less trustworthy, not more.

Table 5. The five defects on the V1 experiment card. Each produced a plausible wrong answer.

#	defect	evidence
1	Credit anchored to the lives counter	ALE drops `lives` at the *end* of a 127-frame death animation, so cards were written after the ship was destroyed: **0 of 374** blamed frames had a bomb visible; at the true impact frame, **17 of 17** did.
2	One card per frame	About 79,000 cards whose failure rates were all ≈ 0, so no signature could clear a threshold and no rules formed. Fixed by one card per *encounter*; rules then formed within 12 episodes.
3	Absolute 60% gate	Base death rate 2.8%: a situation killing 5% of the time is catastrophic yet never nears 60%. **268 real failures produced 0 rules.** Fixed: relative lift + Wilson bound + absolute lethality floor.
4	Own laser same grey as bombs	Column matching paired falling bombs with the rising laser: of 374 streaks, 339 were read as "ours". The ship was blind to nearly every bomb aimed at it.
5	`frameskip=4`	Bombs crossed the danger band in 2–3 samples: visible in **6%** of frames at frameskip 4 against **76%** at frameskip 1.

4.6 What the next arms did

A second card (V2) kept both sides of every (situation, action) pair and ranked by a utility, expected return minus a failure cost, so nothing is permanently excluded and later evidence can un-rank a bad option. A variant (V2.1) added a hard floor for catastrophic outcomes on top. Table 6 gives the result exactly as frozen. **V2 fixed the Freeway regression and lost the Space Invaders win. V2.1 lost both.** On Freeway a hard floor vetoes `up` (several `up` signatures sit above 80% observed fatality) and simply reconstructs V1: there is no catastrophic tail to exclude when the fatal action is the only scoring action. A sweep of the fixed failure cost over $\{0.5, 1, 2, 4\}\times$ the observed return scale gave Freeway $\{10.2, 7.5, 10.3, 7.7\}$: parity at best, never above the pilot.

Table 6. Three arms, two games, mean score per episode. **Read within a row, not across cards**: the Space Invaders pilot baselines differ between cards (V1's comparison ran to natural game-over, V2's used a 3,000-frame cap), so comparing 200.6 with 139.4 would be wrong.

arm	Space Invaders	Freeway
pilot only (no learning), V2 card	139.4	10.3
V1: avoidance only	200.6 (+32% vs. its own 152.2)	9.2 (−12%)
V2: utility only	128.8 (−8%)	10.2 (parity)
V2.1: catastrophic floor + utility	121.9 (−13%)	5.0 (−51%)

The same card records a further arm, V2.2, that credits each decision with the discounted return from that moment to the end of the episode (textbook Monte-Carlo return estimation), so a terminal death is charged the whole remainder and a recoverable knock-back only the setback. It scored 157.5 on Space Invaders (+13% over its 139.4 pilot) and 10.0 on Freeway (parity); with a signature that buckets cars by time to arrival instead of raw distance, 11.2 on Freeway (+8%). We report it with its limits: it is a *ranking*, not a floor; asked which action it prefers, it chooses up in 13 of 17 situations, including when a car arrives immediately, so it learned that crossing is worth the risk, not *when* to cross; and the same memory on DOOM is parity (§6.2). A floor, a hard constraint, that prices recoverability remains open.

4.7 What carried over

Four things carried from this prototype into the runtime of §5–§7: *learning only removes* (here the memory could only take actions away; in VDSG this became Theorem 1 with tests); *every refusal cites its evidence* (the counts became traced evidence tiles); *the saturation collapse is addressed* by coverage-aware retirement, so far measured on replay; and *credit assignment* was repeated and caught again in Fallout. One thing did not: risk as the objective. Freeway remains the benchmark for a *floor* that prices recoverable failure; no floor we have built beats its pilot.

5 From rules to a runtime

Table 7 is the line from the frozen Atari result to the runtime that now carries the loop, one step per row, with its number. Negatives are kept.

Table 7. From Atari to the fail-safe model. Every step since the frozen result, with its number.

step	result	what it taught
Arcade floor V1 Space Invaders, Freeway	+32% · −12%	Failure memory helps where failure is terminal and hurts where the risky action is the only useful one (§4).
Rule retirement the 54-rule collapse	30 rules → 6 dead ends · 50 → 12 · retirement → 0	Every rule individually justified, collectively paralysing. Retiring by coverage restores choice (§5.1).
Evidence tiles every veto traced	4,280 cards → 30 rules → 203 tiles	A refusal walks back to the individual deaths that earned it (§5.2).
VDSG · DOOM rule-based, no neural network	random 3.2 · rules 17.9 · + memory 20.3 ($t = 0.78$)	The rules win decisively; the memory on top is parity, not a win (§6.2).
VDSG · orders DOOM, Wolfenstein 3D	orders only narrow · learner bounded	Stated as propositions; no order and no amount of experience can add an action.
VDSG · Fallout (1997) scripted scene, real loop	49 deaths at one guard → dies once	Blame had to reach the line that started the fight, not the heal inside it (§6.3).
Drone course simulation, held-out starts	10/10 whole course, 0 contacts · baseline 1/10	The learned plan sets the speed; the stop-distance limit and the reflex act after it (§7).

5.1 Coverage-aware retirement

The usual mitigation for a growing rule set is to score each rule by its own evidence quality and keep the best N. That is what the V1 floor did, and it is still blind: it scores every rule alone and never asks what the conjunction does. Retirement scores the *set*. For each situation the agent actually encountered,

$$\text{choices\_left}(S) = |\text{actions\_seen}(S)| - |\text{actions\_blocked}(S)|,$$

where 0 is a **dead end** (every option refused; the floor has become a wall) and 1 is **forced** (one option left; the agent decides nothing there). A rule's *protection* is the failures it accounts for; its *cost* is the situations it pushes into a dead end or into forced. Retirement drops the worst cost-to-protection offenders until choice is restored.

We re-ran the measurement while writing this paper, read-only, on the Space Invaders failure memory: 4,320 replayed decisions over 91 encountered situations (Table 8, Figure 8). The bounded set of 30 rules leaves 6 dead ends and 9 forced situations; all 50 qualifying rules leave 12 and 15. Doubling the rules doubles the dead ends, which is the mechanism of the collapse to 167.1 in §4.4 (the 54-rule run and this 50-rule replay are different snapshots of the same memory). Retiring 6 rules takes dead ends from 6 to 0, at the cost of 29 recorded failures handed back, while forced situations rise from 9 to 14: situations move from no choice to one choice. Retirement lets the admissible set grow back; by Proposition 5 pure accumulation never can.

Table 8. Paralysis measured on replay (Space Invaders memory, 91 situations, 4,320 decisions).

rule set	rules	dead ends	forced	failures handed back
bounded working set (V1)	30	6	9	–
unbounded (all qualifying)	50	12	15	–
bounded, then coverage-aware retirement	24	0	14	29

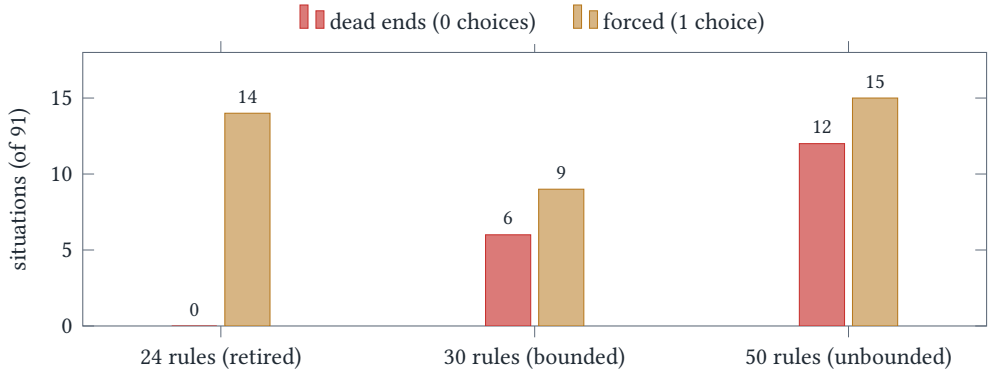


Figure 8. More rules, fewer choices. Retirement is the only operation that moves the red bar down. The retirement row is a replay measurement; its effect on game score has not yet been measured.

5.2 Evidence tiles: a refusal you can walk back

A classifier can report that it is 0.94 sure. It cannot say which experiences made it 0.94, it cannot delete one of them, and after an incident there is nothing to read. In the fail-first floor the learned state is the explanation. The 4,280 cards of the Space Invaders memory, the 30 rules they support, and the chain between them were traced into 203 evidence tiles, linked parent to child, so a single veto walks back to every event that justifies it. A veto reads like this:

What a veto reads like

```
left is removed from the admissible set when bomb dx+0 drop0:
died 4 of 4 times (100.0%), 5.2× the base rate
← experience #0040  ← experience #0042  ← experience #0171  ← experience #0203
```

Counts, a comparison to the base rate, and the individual events by identifier. Delete a row and the behaviour changes; there is nothing else in there. We do not call this "no machine learning". It is learning, since behaviour changes from experience. What it has is no weights, no gradients and no training run. The tile format itself belongs to the Peel evidence floor and is not described here (§10).

6 VDSG: DOOM, Wolfenstein 3D and Fallout

VDSG is the runtime in which the loop now runs with its guarantee pinned: a commanded admission-control runtime that, at every decision, computes the goals an agent may pursue from explicit rules over facts, lets an operator narrow that set in ordinary language, learns from evidence only inside it, and writes down the reason for every choice [24]. It is the deployment of the Peel model in games, and it has no neural network in the loop that decides: its knowledge is sourced cards and its learning is a table of counts.

6.1 The contract

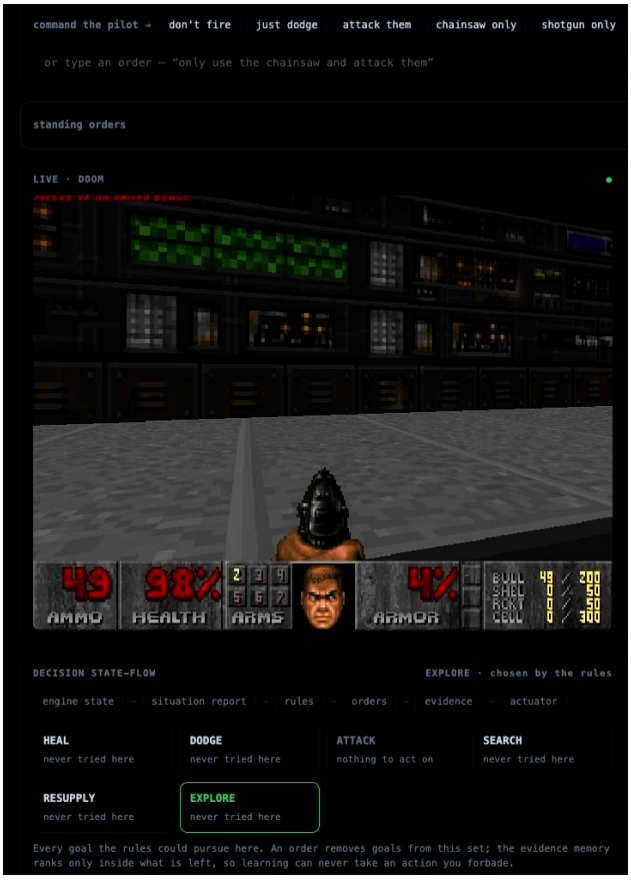
The situation report $F(s)$ is a structured extraction of facts from the engine's own state (enemies with bearing, distance and whether in view; pickups; health, armour, ammunition; position; recent hits). The applicability function returns the goals whose object exists (ATTACK needs an enemy in view, HEAL a health pickup), and a fixed-priority rule policy chooses among them. A standing order is defined by its effect, not its text: the goals it still permits, the buttons it removes, and a weapon it may name. The parser is a fixed phrase vocabulary with explicit negation; text it does not recognise is refused, never guessed. When an order cannot be met ("only use the shotgun" with no shotgun carried) the runtime says so, with a receipt, every decision until it can, and falls back to the applicable set without ATTACK, so an unsatisfiable order never puts the pilot into an engagement it was not already going to enter. The VDSG paper proves that orders only narrow, that the learner is bounded by the narrowed set, and that refusal of an order is sound and complete [24]. Those properties were written after their failure modes were observed during development: a negated weapon obeyed backwards, comma-separated clauses merged into one, and an unsatisfiable order silently widened to ATTACK. Each is now excluded by construction and pinned by per-site tests.

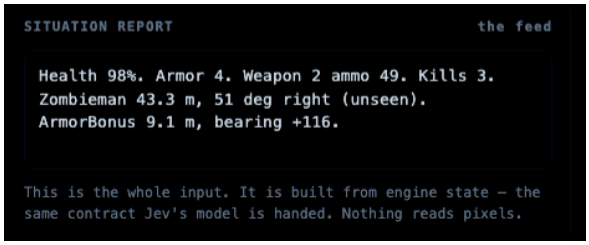
6.2 DOOM

On a Freedoom deathmatch arena under ViZDoom [13], same seeds and cadence for every arm, the rule-based pilot is worth about five times random, and the evidence memory on top does not separate from it (Table 9). The paired per-seed difference, memory minus pilot, has mean +2.4 and standard deviation 17.4 over 32 seeds, $t = 0.78$, 18 wins, 13 losses and 1 tie. That is parity within noise, not an improvement, and we report it as such. The cause is the one named in the V2 card: a return-to-go estimate cannot separate goals within a situation at this sample size (the top goals of the learned table sit within a few points of each other).

Two further measurements temper even the rules' margin. The engine's kill counter is the map's, and counts monsters that kill each other: a pilot ordered never to fire (verified: zero trigger presses over eight episodes) was still credited 55 kills. Decomposed on the same seeds, most of the rules' return over random is *moving at all*; the console's headline counters are therefore damage dealt and shots on target, which are exactly zero when the pilot does not fire. Separately, of six failure rules generalised from the evidence trace, four reproduced on held-out episodes.

On the real 1993 shareware levels the pilot clears E1M1 on the third difficulty (exit switch at 52 s) and





Reading the console. Top: the order buttons and the free-text order line; *standing orders* is empty ("as you were"). Middle: the real DOOM engine on E1M1. Bottom: the decision state-flow, *engine state* → *situation report* → *rules* → *orders* → *evidence* → *actuator*, with every goal the rules could pursue here and its evidence ("never tried here"). The chosen goal, EXPLORE, is outlined. Right, above: the situation report, which is the pilot's whole input, built from engine state; nothing reads pixels to decide.

Still from the recorded full console (t = 8 s), cropped below the header line, whose wording predates the current "no neural network in the loop" phrasing.

Figure 9. VDSG at the wheel of DOOM (1993). An order removes goals from the admissible set; the evidence memory ranks only inside what is left, so learning can never take an action the operator forbade.

does not clear it on Nightmare in 8 attempts (it dies at 23–37 s). On E1M2 it takes the shotgun at 10–20 s and holds the red key in every attempt, and still dies at 112–207 s. The cause of death was measured, not guessed: of 177 points of damage in one full attempt, 144 came from Zombiemen beyond 15 m that were never in view. A hypothesis that followed, retreating to break line of sight when hit by something unseen, measured *worse* (mean 123 s alive over three attempts against 187 s over the four before) and was reverted; the record keeps the number [24].

Wolfenstein 3D. The same runtime on the 1992 data exposed two defects that the loop's own stuck rule made visible. The stuck thresholds had been left in DOOM's map units, so the Wolfenstein lane recorded 150 stuck events in 1,014 decisions, one every eight; in the engine's own tile units, 9. And because the engine's USE toggles a door, a pilot pressing it at a half-open door was closing the doors it was trying to open: 583 of 1,275 decisions spent at doors before the fix, 371 of 1,184 after. The pilot does not yet reach the level's elevator.

6.3 Fallout (1997): the failure we learned the most from was ours

In the Fallout lane the licence comes from a document. The game's printed manual is compiled into **2,250 cards**; the cards in force for the current situation license the goals the pilot may pursue, each licence carrying its page receipt, and the card store is opened read-only. The evidence table is the only store the learner writes. So there are two stores that do not mix, exactly as in Figure 3.

Table 9. DOOM arena, mean return. Rules with memory versus rules alone is a null result.

arm	seeds	return	kills
random buttons	16	3.2	1.4
rules, view-only feed	16	14.1	5.6
rules + radar feed (the pilot)	32	17.9	7.4
pilot + evidence memory (200 training episodes)	32	20.3	8.2
paired difference (memory − pilot): mean +2.4, s.d. 17.4, $t = 0.78$; 18 wins / 13 losses / 1 tie			

The 49-death loop. One live run died **49 times** returning to the same guard, saying the same line. The learner was working; it was learning from the wrong moment. Three defects, all found in the pilot's own records:

(a) **A line's outcome was read one tick too early.** The guard closes the conversation first and draws a moment later, so "Prepare to meet your maker", said four times and followed by four fatal fights, was scored as *fight 0%*.

(b) **Blame covered only the last six decisions inside the fight.** 26 deaths were charged to HEAL, and the 446 replies that started those fights were charged nothing.

(c) **The veto used the widest harmful pattern and a point estimate** against a per-decision base rate near 0.5%, so after a single death the pilot shunned every person in town.

The fixes follow the loop: a line stays open until it is answered by the next line, a fight, a death or a short quiet period; a fight that begins within that grace window of a conversation is charged to the talk that opened it; and a pattern is condemned only by the rule of §3.5, fatal on at least 2 tries at $\geq 90\%$, or at least 4 tries with a Wilson lower bound ($z \approx 1.64$) above the base rate plus a margin.

After the fix. In a scripted Shady Sands scene driven through the real decision loop, the pilot **dies once, then chooses the peaceful line** every time after. With no safe line available, it dies twice, then stops talking to that guard, and still talks to the child: the lesson is about a counterpart, not about talking. These are scripted scenes through the real `Session.tick()`, not a long live campaign.

The tests that pin the boundary. The separation is stated in the suite in one sentence: the learner may rewrite what it *believes works*; it may never rewrite what it is *authorised to do*. Three tests pin it:

- `test_the_learner_can_never_add_a_goal`
  with every goal killed six times and authority {WAIT}, the kept set stays inside authority;
- `test_the_learner_cannot_overrule_a_standing_order`
  an order that narrows the set to FIGHT stays FIGHT after 20 deaths;
- `test_ranking_is_a_permutation_and_nothing_more`
  the learner may reorder the admissible goals, never add one.

As published on 26 September 2026 the suite had 219 passing tests and 1 strict expected failure; it has since grown (488 tests collected at the time of writing). We re-ran the three boundary tests and the situation-bound test beside them for this paper; all four pass.

The expected failure that stays in the suite. Having died only while unarmed, the widest pattern (FIGHT with every other field wildcarded) clears the gate and condemns fighting *armed*, something never tried. The obvious fix, distrusting a wildcard whose field never varied, also breaks a correct case that

generalises across an enemy's name from a single name. The two are logically symmetric: nothing in the evidence says which field is causal, and choosing for the learner would mean hard-coding the lesson. The remedy is exploration, occasionally testing a condemned goal in a context whose specifics were never tried, and it is not built. The test is marked as a strict expected failure until it is.

7 The drone

7.1 Setup

The course and aircraft come from an open-source project, jev-drone [12], pinned at upstream commit `974b473` and not edited: a MuJoCo [21] simulation of a Skydio X2 quadrotor flying a 62 m course with a slalom, low beams, turnstiles, a sliding gate and a pillar cluster. Upstream's flight controller, onboard depth camera, reflex distances and altitudes are used as they are. One seam is swapped: the tactical layer that decides what to do is replaced by the floor, with the same inputs and outputs. The only sensor the rules read is the onboard depth image, summarised as free space by sector, the path ahead and the nearest obstacle. There is no neural network anywhere in the loop.

7.2 Results, with the caveats

On held-out starting positions, run once after the rules were frozen, the floor flew the whole course **10 of 10** times with **0 contacts**. The unmodified baseline finished **1 of 10**. The caveats always go with that number: the course is identical in every run and only the start pose varies, so this is weak evidence of generalisation; it is simulation, not flight; and we did not re-run anyone else's system. At the upstream decision cadence and latency instead of instantaneous decisions, the same floor finished 8 of 10 with 6 contacts: cadence costs.

7.3 Learning the route by failing first

The time trial is where the loop learns. The course is cut into 6 m blocks, each holding a speed cap and an altitude choice. Each round proposes *one* move on *one* block (fly high, faster, or slower), flies it from every start, and keeps it only if the cost of §3.7 falls; otherwise it is reverted and that move's step is halved. The memory stores only per-block speed and altitude, never where a moving gap was. Contacts happen while it learns; each costs 15 s, and a round that adds them is thrown out.

From a cautious initial schedule the learner went from **39.4 s with 4 contacts to 27.8 s with 0 contacts**, converging by round 87; unseen starts then flew 10 of 10 clean. In the recorded flight console, every lap of the learned route finishes in 27.6 s with 0 contacts. A second recording starts from a wiped memory: **7 rounds take the cost from 69.39 to 51.52, with 2 changes kept and 5 reverted** (Figure 10).

7.4 The learners that failed

The failed learners belong in the record as much as the one that worked.

- **Per-segment speed adaptation** (additive increase, multiplicative decrease per 1 m segment) went from 39.1 s to 20.9 s in six laps, then collapsed over laps 7–20. Two stations move: flying faster anywhere upstream changes the phase at which the drone meets the turnstile and the gate, and slowing at the failure point shifts the phase again. A per-segment rule cannot see that coupling; accept/reject on whole rounds can.
- **A slow-down repair** that answered every failure by slowing down repeated the same move, with the same failure, 57 rounds in a row. Slower is not safer here: a crawling aircraft meets the moving stations at a different phase. This is the stuck rule in its general form: when the same movement recurs, it is stuck, and it must break out rather than replan into the loop.

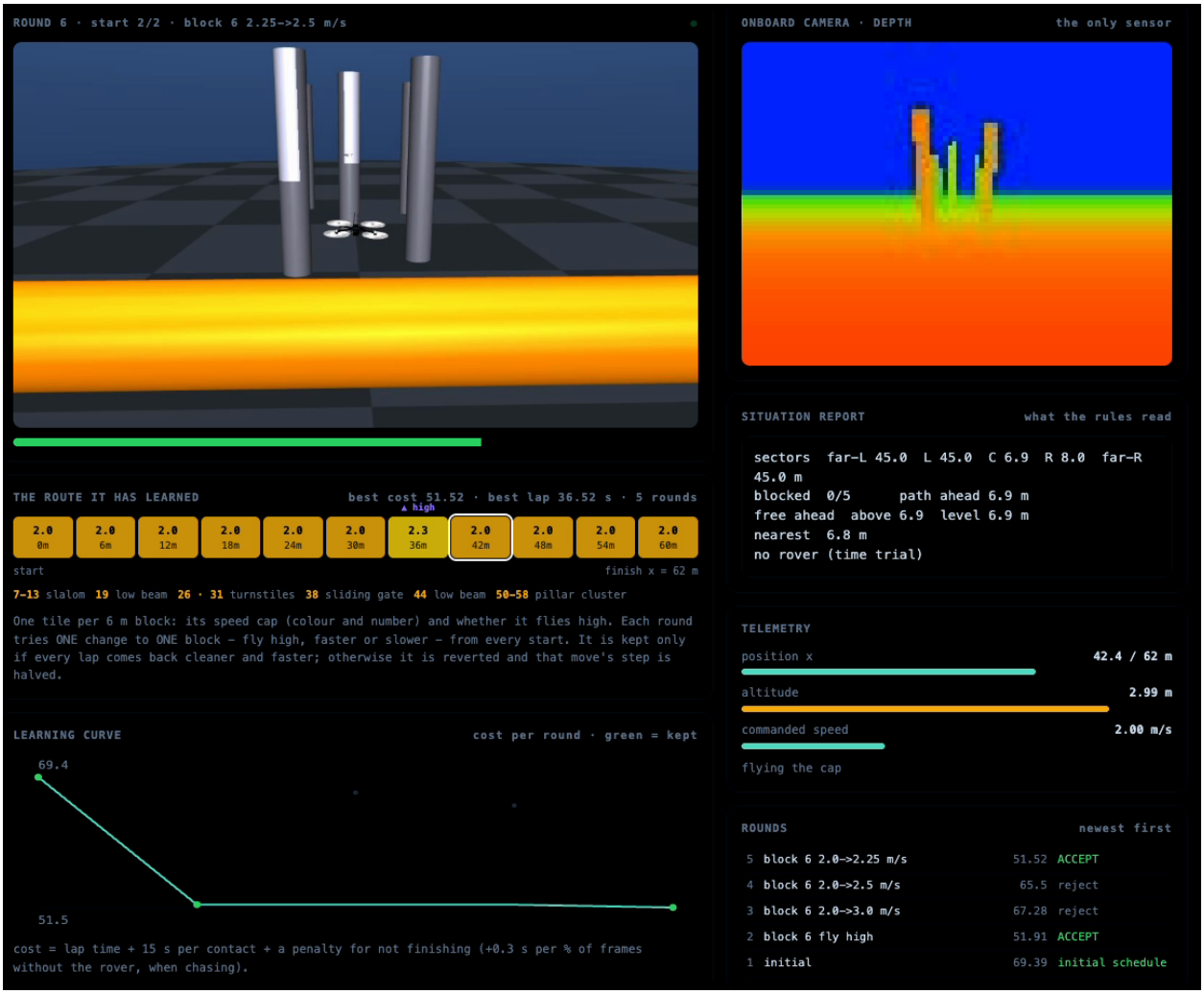


Figure 10. The flight console relearning the course from a wiped memory (still at round 6 of the recording). Top left: the simulator view. Top right: the depth image, the only sensor. Middle: the route it has learned, one tile per 6 m block with its speed cap; block 6 has learned to fly high. Bottom: the learning curve (69.4 → 51.5) and the round log, newest first: two changes ACCEPTed, two rejected and reverted so far. Simulation (MuJoCo), sped up.

7.5 What the learner cannot touch

Forward speed is composed in a fixed order: the learned block cap first, then the stop-distance limit on what is in the path, then the upstream reflex. By Proposition 6 the last two can only reduce what the learned plan proposes. The learner changes what the drone tries; it never changes the physics bound it flies under. This is the fail-safe property in a continuous setting: here the "admissible set" is an interval of speeds, the floor computes its upper end from the current depth image, and the learner only chooses inside it.

8 Why a guess must never become a fact

The loop's defining step, VERIFY, says that the environment and not the model decides whether an action worked. The same discipline governs knowledge. A language model may propose a fact or suggest where to look. A fact is admitted only when an independent, deterministic check confirms it against a source, and a verified fact is complete or absent, never 86% right. When nothing verifies, the answer is *unknown*. This

section explains why the rule is binary, and what happened when we broke it on purpose. We describe the admission rule at the level of its property only; how facts are extracted, checked for contradiction and stored with their provenance is not part of this paper.

The arithmetic. Suppose a model's guesses are each right with probability q and an answer depends on k guessed facts. If the guesses are independent, the answer survives with probability $\Pr[\text{answer correct}] = q^k$; for $q = 0.858$ and $k = 2$, $0.858^2 \approx 0.736$. That is an illustration, not our analysis, but it matches what we measured.

The measurement. The task is diagnostic identification of a hidden target among **160 human kinases** over 2,775 verified features, under a per-feature cost model: each measurement asks whether the target has feature f, and the verified reference eliminates inconsistent candidates [25]. With **40% of the reference masked**, a strengthened cost-aware information-gain planner needs 19.66 cost units at correct identification 1.000. A frontier language model with no truth authority predicted the 2,875 masked entries of the 44 discriminating features with **85.8% accuracy** against sealed truth, a genuinely good prior. Admitted into the elimination step, its guesses cut mean cost from 19.66 to 12.10 (−38.4%, paired 90% CI [6.65, 8.51]), and correct identification fell from **1.000 to 0.753**. All 39 misidentifications were caused by a wrong guess about the true target's own masked entry: one wrong cell eliminates the true answer, and there was no fact to check the guess against.

Table 10. Four lanes fill the same 2,875 masked cells. Only admission of verified facts is both cheaper and correct. Pre-registered bar: $\geq 15\%$ improvement, paired 90% bootstrap CI > 0, no worse correctness; the verified-card lane is reported from the same benchmark's result record [25].

lane	mean cost	correct ID	verdict
baseline planner, no fill	19.66	1.000	reference
model guesses admitted as facts (85.8% accurate)	12.10	0.753	rejected: corrupts truth
model ranks where to look, admission protected	18.86	1.000	rejected: −4.06%, below the bar
verified-card retrieval	11.50	1.000	−41.5% at perfect correctness

The decomposition is the point. Of the 38.4 points of apparent benefit in the admission lane, about 34 were bought by corrupting truth and about 4 were the prior's legitimate contribution to which question to ask next, which is what the protected lane measures (4.06%, real but below the pre-registered bar). Keeping admission to verified facts, and letting the model only rank where to look, held correct identification at 1.000. A guess that is right 86% of the time still corrupts the answer about one time in four once it is allowed to count as a fact.

The connection to failing first. A fail-first model and a fail-safe model apply one rule in two places. On the action side, only a verified outcome becomes evidence: a failure the model merely predicts does not write a card. On the knowledge side, only a verified fact is admitted: a fact the model merely predicts does not enter the answer. In both places the model may propose, and something it cannot write to decides. That is also why the learning half is safe to leave running: its lessons are about which permitted option to try, never about what is true or what is permitted.

9 Related work

Remembering failure in search. Haralick and Elliott [10] gave two principles for efficient backtracking: try first where failure is most likely, and remember what has been done so the same mistake is not repeated. The second is the ancestor of the loop in this paper. Its descendants in constraint processing

record a *nogood* whenever an inconsistency is found, so future partial assignments are pruned without further search [7]; conflict-driven clause learning in SAT solvers adds a learned clause after each conflict [15]. A fail-first model applies the same idea to a whole acting model in an environment, where the "conflict" is a verified failure reported by the world, and the learned constraint is statistical rather than logical.

Learning rules from failure. Explanation-based learning of control rules in PRODIGY [16], failure-driven case memory in CHEF [9], and Ripple-Down Rules [6], where each rule is added in the context of the case that broke the previous one, all turn failures into explicit knowledge. They are the direct ancestors of "every rule cites the failures that earned it".

Language agents that learn from experience. Reflexion [18] keeps verbal reflections on task feedback in an episodic buffer; ExpeL [23] distils insights from successes and failures into natural language; AutoManual [5] builds an instruction manual of rules through interaction. All three learn without weight updates, as a fail-first model does. Their rules are prompt text that a language model reads and may ignore; nothing enforces them, and nothing separates the rules from the skill that applies them.

Safe reinforcement learning and runtime assurance. Safe RL [8] and constrained MDPs [2] bound expected cost during learning. Shielding [1] blocks unsafe actions with a layer synthesised from a written safety specification; the Simplex architecture [17] places a verified controller beside a high-performance one and switches on a safety condition. Both give specification-level guarantees and neither learns from failure; our floor sits *before* the chooser, so the chooser never sees the full action space, and it accepts an operator's constraint as a first-class input.

Learned shields: the closest prior work. Shperberg, Liu and Stone [19] define POMDPs with catastrophic actions and learn a shield by recording catastrophic mistakes in a database and forbidding the agent from repeating them (*never repeat the same mistake*); their rule-based follow-up [20] accumulates safety rules from catastrophic action effects. That is the idea of a learned shield, and we do not claim it. The differences are narrower: their shield sits beside a reinforcement-learning agent (ShieldPPO, a modified PPO, in the first paper), which is the learned policy the shield constrains, while here there is no neural network in the loop that decides; our authority is computed before the learner from sourced cards and human orders, and the learner provably cannot widen it; every refusal cites the individual failures behind it with a statistical bar against the base rate; and we measure the saturation point and the regime (risk coupled to the objective) in which a learned shield degenerates.

Why constraints need a floor. Reward hacking and specification gaming [3, 14] are the reason a learner must not be able to touch its own constraints; hallucination [11] is the reason a model's output must not be admitted as fact without a check.

The claim, stated once more. To our knowledge, this is the first model to combine (1) rules built from observed failures, each citing the failures that earned it, (2) an authority computed before the learner from sourced cards and human orders, which the learner provably cannot widen, and (3) no neural network in the loop that decides. Learned shields [19, 20] are the closest prior work.

Table 11. Typical forms of each approach; individual systems vary.

approach	learns from failure	learner can loosen the rules	refusal cites evidence	abstains when unknown
confidence threshold	no	n/a	a number	no
guardrail filters	usually no	n/a (fixed)	sometimes	sometimes
rules in a prompt (ExpeL, AutoManual)	yes	can be ignored by the model	partly	not enforced
fine-tuning / RLHF	yes, in weights	yes: nothing separates rules from skill	no	not guaranteed
shielding / Simplex	no (hand-specified)	no	cites the specification	varies
learned shields	yes	no	varies	varies
fail-first / fail-safe model	**yes, as readable rules**	**no (Theorem 1)**	**yes, every rule**	**yes**

10 Limits and open problems

Status. Research prototype. Simulation and games only. Nothing here has flown, driven a road vehicle, or been qualified for a safety-critical path; "fail-safe" names an architectural property, not a functional-safety certification. Every figure is ours, and none has been replicated by a third party.

Risk as the objective (open). When the action that accomplishes the task is the action that exposes the model, pure failure avoidance refuses the mission. Freeway shows it: three rules, every one blocking up, and the pilot correctly stopped scoring. A catastrophic floor on top of a utility ranking did not fix it; it reconstructed the original veto and made both games worse. A return-to-go ranking reached parity and, with a time-to-arrival signature, +8%, but without learning timing, and it is a ranking rather than a floor. A floor that prices recoverability instead of fatality (refuse what cannot be undone, permit everything else right up to that edge) has not been built. By §3.8, a correct lesson can still be a costly one.

The model must fail to learn. A rule is earned by observed failures. Where the first failure is unacceptable, the floor must be written, not learned; that is the classical shielding case and a different product posture.

Composition needs bounding. Retirement takes dead ends from 6 to 0 on a replayed memory; its effect on play has not been measured, and it has not been shown at a scale where constraints between many agents dominate.

Over-generalisation (open). Bucketing decides when two situations are "the same"; it is declared in one place and is arguable. Widening a pattern can condemn a context never tried (the strict expected failure of §6.3). The remedy, exploration of condemned goals in untried contexts, is not built.

Credit assignment fails silently. Two of our own defects produced believable curves before they were caught. Theorem 1 keeps a mis-blamed rule from widening authority; it does not keep it from making the model useless.

The guarantee is claimed where it is pinned. Theorem 1 is claimed for VDSG, where the learner's placement is enforced in code and pinned by tests. In the Atari prototype the failure memory only removes actions, but the parameter-search layer above it (an evolutionary search over the pilot's parameters, which promoted one change from 103.0 to 221.8 on sealed seeds) mutates a parameter vector that also holds floor parameters, and one of its mutations relaxes one. The floor and policy parameters are not yet separated there, so the guarantee is not claimed for the Atari prototype.

Small samples and weak generalisation. The Fallout results are scripted scenes through the real decision loop; the drone's held-out starts vary only the start pose on an identical course; the DOOM level results rest on four to eight attempts; the only statistically powered comparison in the game runtime, 32 DOOM seeds, is a null. The kinase task uses a single frozen instance of 160 candidates.

What this paper does not publish. Everything above is either standard mathematics or a property we measure in public. The mechanisms that make the Peel science floor work at scale, including how facts are extracted, how contradictions are vetoed, how provenance is stored, and the admission gate's internal checks, are withheld pending patent review. The properties they are claimed to have are stated; their internals are not.

Threats to validity. All measurements were made by the same group that built the systems. Baselines differ between experiment cards and must be compared within a row. Games are cheap, deterministic and forgiving in a way the physical world is not; the transfer from a game engine to a platform is untested. The rule-based pilots carry most of the performance, and the learning layer's contribution is small where it is measured with power.

11 Conclusion

A model that learns from experience has to fail first. The useful question is where a failure is allowed to lead, and the answer this paper gives is architectural: compute what the model is allowed to do before the learner runs, from sources and orders the learner cannot write to; let verified failures teach it only inside that authority; and keep every lesson as a rule that cites the failures behind it. Then a failure can change what the model tries next and never what it is allowed to do, and the proof of that is one line because it depends on where the learner sits, not on how good it is.

The measurements say what that buys and what it does not. Failure memory helps where failure is terminal (+32% on Space Invaders) and hurts where the risky action is the only useful one (−12% on Freeway). Individually correct rules compose into paralysis unless they are retired. Credit assignment is where learning systems fool themselves, and we fooled ourselves twice. The learning layer on top of good rules can be a null ($t = 0.78$). A route can be relearned from nothing in seven rounds under a stop-distance floor that only slows it, and a guess that is right 86% of the time still breaks one answer in four when it is allowed to count as a fact.

Fail-first is how the model learns. Fail-safe is what that learning may never change. The open problem is to price a recoverable failure without giving the learner the power to widen the floor.

Acknowledgements

The Atari experiments use the Arcade Learning Environment and the Stella emulator; the DOOM experiments use ViZDoom and Freedoom; Wolfenstein 3D runs under ECWolf; Fallout runs under the Fallout Community Edition engine; the drone course and aircraft come from the open-source jev-drone project, which we used unmodified. We thank their authors.

References

[1] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu. Safe reinforcement learning via shielding. In *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[2] E. Altman. *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

[3] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, D. Mané. Concrete problems in AI safety. arXiv:1606.06565, 2016.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling. The Arcade Learning Environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279, 2013.

[5] M. Chen, Y. Li, Y. Yang, S. Yu, B. Lin, X. He. AutoManual: constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.16247.

[6] P. Compton, R. Jansen. A philosophical basis for knowledge acquisition. *Knowledge Acquisition* 2(3):241–257, 1990.

[7] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[8] J. García, F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16:1437–1480, 2015.

[9] K. J. Hammond. CHEF: a model of case-based planning. In *Proc. AAAI-86*, 1986.

[10] R. M. Haralick, G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14(3):263–313, 1980.

[11] Z. Ji et al. Survey of hallucination in natural language generation. *ACM Computing Surveys* 55(12), 2023.

[12] jev-drone. Open-source drone simulation project, `https://github.com/RomanSlack/jev-drone`; used unmodified at commit `974b473`.

[13] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, W. Jaśkowski. ViZDoom: a Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, 2016.

[14] V. Krakovna et al. Specification gaming: the flip side of AI ingenuity. DeepMind blog, 2020.

[15] J. P. Marques-Silva, K. A. Sakallah. GRASP: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers* 48(5):506–521, 1999.

[16] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer, 1988.

[17] L. Sha. Using simplicity to control complexity. *IEEE Software* 18(4):20–28, 2001.

[18] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, S. Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2303.11366.

[19] S. S. Shperberg, B. Liu, P. Stone. Learning a shield from catastrophic action effects: never repeat the same mistake. arXiv:2202.09516, 2022.

[20] S. S. Shperberg, B. Liu, A. Allievi, P. Stone. A rule-based shield: accumulating safety rules from catastrophic action effects. In *Proc. 1st Conference on Lifelong Learning Agents (CoLLAs)*, PMLR 199:231–242, 2022.

[21] E. Todorov, T. Erez, Y. Tassa. MuJoCo: a physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[22] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association* 22(158):209–212, 1927.

[23] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, G. Huang. ExpeL: LLM agents are experiential learners. In *Proc. AAAI Conference on Artificial Intelligence*, 2024. arXiv:2308.10144.

[24] Perslis Research. VDSG: a commanded admission-control runtime for autonomous agents. Systems paper, 2026. `https://research.perslis.com/vdsg.html`

[25] Perslis Research. Inference placement: where learned inference earns authority in a symbolic system. Preprint, 2026. https://research.perslis.com/inference-placement.html

[26] Perslis Research. What is a fail-safe model? Definition, science and math. 2026. https://perslis.com/fail-safe-model

[27] Perslis Research. Fail-first models: failure becomes structure (the arcade floor, frozen result). 2026. https://perslis.com/research/arcade

A Where each number comes from

Every number in this paper is taken from a frozen experiment card, a published Perslis page or paper, or a measurement re-run read-only while writing. Public explanations of the model are at [26]; the frozen Atari result as published is at [27].

result	source
Space Invaders and Freeway (V1), learning curve, 54-rule collapse, five defects, Freeway rules	frozen card ARCADE-FLOOR-V1 (tag `freeze/arcade-floor-2026-09-25`); public arcade page [27]
V2, V2.1, V2.2 arms and the cost sweep	frozen card ARCADE-FLOOR-V2 (cites V1; V1 unedited)
Dead ends and forced situations (30 / 50 / retired), 29 failures handed back	retirement replay, re-run read-only on 2026-09-27 (4,320 decisions, 91 situations)
4,280 cards → 30 rules → 203 tiles; the example veto	tile-trace build of the Space Invaders memory; public evidence chapter
DOOM arena arms and paired statistics	frozen card DOOM-FLOOR-V1
DOOM and Wolfenstein level results, kill-counter defect, trace rules	VDSG paper [24], §§5–9
Fallout: 49 deaths, root causes, scripted scene, test counts	VDSG fail-safe chapter (published 2026-09-26); the four boundary tests re-run on 2026-09-27; collected test count taken the same day
Drone: 10/10 vs. 1/10, route learning, relearn 69.39 → 51.52, failed learners	flight-demo page; drone lane evidence and route-memory source; the recorded console
Kinase task	Inference Placement preprint [25]
Wilson bounds, q^k, $p < \nu/(\nu + c)$	computed from the formulas in §3

Figures. Figure 1 is the Perslis fail-first infographic. Figures 9 and 10 are single frames extracted from the recorded full consoles (the DOOM frame at 8 s, cropped below a header whose wording predates the current phrasing; the drone frame at 170 s, in the wiped-memory relearn). All other figures are drawn from the numbers in the tables they accompany.