Ex 2.2.8: a survey of the intervention operator on the stored ex-2.2.3 checkpoints

> This is a survey: no training and no hypothesis gates. We scored 84 intervention operators on stored checkpoints from ex-2.2.3 (the adopted point at twenty seeds, and `t00` at five). We were looking for one that removes *red* as fully as the plain projection while staying as selective as the operand-only one.
>
> On the adopted point the plain projection is inside the selectivity gate on every op, so the frozen rule proposes it. Setting a threshold on the anchor alignment above the range the non-red lines occupy removes less *red*, at no cost the survey can resolve. Setting it inside that range costs more than projecting everything. On `t00` the syntax embeddings carry the axis, and there only the operand-only edits are inside the gate.

Observations

- **The reference rows reproduce.** The `projection` and `operands` rows match their stored ex-2.2.3 values on every seed, op, and group (table).
- **Noise floors.** Under `projection` at the twenty seeds of `recipe-short`, two seed means can be told apart when they differ by more than 0.013 in red accuracy or 0.013 in non-red deficit. We cannot resolve smaller differences.
- **On the adopted point, the plain projection is inside the gate.** Its worst op is `mix`: non-red deficit 0.040 at red accuracy 0.015. The operand-only projection is at 0.012 and 0.025. We expected to see the whole-sequence cost that ex-2.2.3 found on the first points its rule proposed, but at twenty seeds the adopted point does not show it. Both projections are feasible, and the plain one removes more *red*, so the rule proposes it. It clears the gate by +0.010, less than one band.
- **A threshold inside the non-red range costs more than projecting everything.** On the non-red `mix` lines the clean alignment reaches 0.26 (99th percentile over slices and positions). Applied at every position, the steps at a ≤ 0.3 cost `shaped-a0.1-p0` 0.230, `shaped-a0.2-p0` 0.176, `shaped-a0.3-p0` 0.070. That is above the 0.040 of the projection, and they remove about as much *red* (0.008, 0.014, 0.031). So zeroing the axis on some states of a line while leaving their neighbors alone costs that line more than zeroing all of them. These 3 trials are the only ones that cost more than the projection (figure).
- **Above the non-red range the cost vanishes, and what is left of *red* follows the landing.** 38 of the 41 tuned trials applied at every position are inside the gate on every op, and 36 of those cost within a band of zero. Of those 36, `shaped-a0.4-p0` removes the most, leaving red accuracy 0.045. Red accuracy rises with the threshold, the ramp, and the landing, up to 0.714 at `shaped-a0.7-p2` (figure). The shaped suppression scored in ex-2.2.1 (`shaped-a0.5-p1`) is at 0.292 and 0.000. A state left part-way along the axis keeps part of the color it held: at the last slice the alignment of the red operand settles at the threshold or the landing, as designed (figure).
- **The position mask only matters where the threshold is low.** For 33 of the 41 tuned trials, the whole-sequence row and its operand-only copy agree on both `mix` reads to within a band. The 8 that differ are `shaped-a0.1-p0`, `shaped-a0.1-p0.5`, `shaped-a0.1-p1`, `shaped-a0.1-p2`, `shaped-a0.2-p0`, `shaped-a0.2-p0.5`, `shaped-a0.2-p1`, `shaped-a0.3-p0`. Where a threshold already leaves the non-red states alone, the position mask changes nothing.
- **On `t00` only operand-only edits are feasible.** There the syntax embeddings hold the axis, and the clean alignment of the non-red lines reaches 0.93. Projecting everything costs 0.623 on `mix`. Every tuned trial applied at every position costs between 0.324 and 0.950, and the steps cost 0.948 to 0.950. So partial removal costs more than full removal there too. All 41 tuned operand-only trials are inside the gate, as is `operands`, at 0.078 and 0.024. At five seeds we cannot tell the operand-only rows near it apart.
- **The front is short.** 81 of 84 trials are feasible on every op: both projections, 38 of the 41 tuned trials at every position, and 41 of the 41 at the operands. The `mix` front has 6 trials, from the gentlest edit to the most complete (table).
- **Proposed operator: `projection`,** the plain projection at every position. Its worst deficit over the six ops is 0.040, on `mix`, a margin of +0.010 (proposal). On `t00` the same rule picks `shaped-a0.1-p0-operands` (red accuracy 0.071, deficit 0.004 on `mix`).

> **How to read this report**
>
> This is a survey, so it scores no hypothesis. What it does preregister is a search plan: the trial list, the objective, its constraint, and the noise trials. All of those were frozen in `experiment.py` before the run.
>
> Every trial is published, and nothing here may be quoted as a result. The anchored-op prereg adopts the proposed operator and re-measures it at fresh seeds, reporting the survey value beside the confirmed one. The checkpoints are stored, so these are the same seeds ex-2.2.3 scored, and the operator choice is the part that is not fresh: this survey makes it after seeing those seeds.
>
> The plan was rehearsed once on the dev storage pair before this production run. That rehearsal also scored a Bézier-mapped repulsion; the production plan leaves it out (search plan).

Why, and what we ran

Ex-2.2.1 left three operators, none of them both complete and selective. The plain projection removes *red* fully, but it costs the non-red lines. Ex-2.2.3 saw that cost on the syntax embeddings of the first points its selection rule proposed, where the `mix` deficits were well above the 0.05 gate, so it amended the rule with the selectivity gate and adopted `recipe-short`. On that point the projection's cost is inside the gate, with little to spare.

The other two each give something up. The operand-only projection avoids the cost, but it needs to know the syntax of the line. The shaped suppression at the threshold used in M1 (a = 0.5, b = 1, p = 1) removes about half of *red* at no non-red cost.

So the design called for this scoring-only pass on stored runs before the anchored-op prereg, and two backlog items name it as their closing move (shaped suppression, repulsion). Does any operator remove as much as the plain projection while keeping the margin of the operand-only edit? And does one of them do that with no position mask?

The design named the nine checkpoints from ex-2.2.1. D2.2 has since moved to the six-op grammar and adopted `recipe-short`, so this pass scores the stored runs of ex-2.2.3 instead: the adopted point at all twenty seeds, and `t00` at five. `t00` was the first point the rule proposed, and its syntax embeddings carry the axis at more than twice the level the recipe does.

Each checkpoint is scored on the six-op probe set from ex-2.2.3 through `sca.intervention.apply`. The lines of all six ops are concatenated, so each operator is one forward pass. The results are then read per op with the readout from ex-2.2.3, so every statistic means what it meant there.

The `repulsion` operator is new to the contract. The shaped suppression removes a fraction of the axis component; repulsion instead sets where the state lands, moving every state at or above the threshold *a* to alignment *b*. That is a step at the threshold, except when a = b, where it becomes the ceiling $\min(\alpha, b)$. The write is the angle between the arriving alignment and the landing one, and the scorer checks it against the measured rotation on every state.

▶ **Glossary…**

Search plan

Frozen in `experiment.py` before the run. The space is a grid: two families whose parameters mean something, so every corner is worth a look. At 84 operators on stored checkpoints, the whole grid costs less than one training run.

The space. 2 reference rows: the `projection` from ex-2.2.3, applied at every position and again at the operand positions. Then two families, each at every position and again at the operand positions only:

family	grid	trials
shaped suppression	a ∈ {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7} × p ∈ {0, 0.5, 1, 2}, b = 1	28
repulsion	(a, b) ∈	13

`shaped` at p = 0 is a thresholded projection, and repulsion at b = 0 does the same thing, so that edge is not repeated. Counting the operand-only copies, that is 84 trials in all.

What the rehearsal changed. The plan first ran on the dev storage pair, so it was run again here on production. The rehearsal also included the Bézier-mapped repulsion from M1, at eight points; that version is continuous at the threshold. Each of those eight matched the linear row at the same landing on both reads, so the production plan drops that family. Nothing else changed.

The objective. Removal against selectivity, as ex-2.2.3's H4 read them: seed-mean accuracy on the red lines (the dose above 0.8; lower is more removed) against the seed-mean non-red deficit (the drop in P(answer) on lines whose dose is at most 0.2; lower is more selective), on `mix` as the gated op and on every op beside it. Feasible: non-red deficit within 0.05 on every op. Proposed: the feasible trial with the lowest red accuracy on `mix`, with the front reported whole.

Noise floors. The per-run σ of each objective is read under `projection` at the twenty seeds of `recipe-short`. A difference between two seed means smaller than $2\sigma\cdot\sqrt{(2/20)}$ is unresolved. Every trial is read on the same twenty seeds, so this band is conservative for a paired comparison.

What is checked, per trial and seed. The assertions in the contract run on every state: the clean pass matches, the edit stays within the named positions, and where the write has a closed form (projection and repulsion) the measured rotation matches it to 2×10^{-3} rad. A trial whose write did not match would have failed the scoring task rather than being scored.

Not in the plan. We did not try positions other than the operands and all, and we did not edit only some of the slices. Operators fitted to the data (LEACE, diff-in-means) are left to the anchor-versus-fitted comparison in the D2.2 design. The readout still reports the redder-than-both lines, but this pass does not rank on them.

The reference rows reproduce

The two projection rows from ex-2.2.3, re-scored by this pass on the same checkpoints and probe lines, beside their stored values.

condition	row	red acc	non-red acc	non-red deficit	P(ans), all
`recipe-short`	`projection`	0	0	0	0
`recipe-short`	`operands`	0	0	0	0
`t00`	`projection`	0	0	0	0
`t00`	`operands`	0	0	0	0

Largest |re-scored − stored| per statistic. *Over the seeds of the condition, the six ops, and the group named. The floating-point path differs only in batch composition (the six ops are scored in one pass here).*

Noise floors

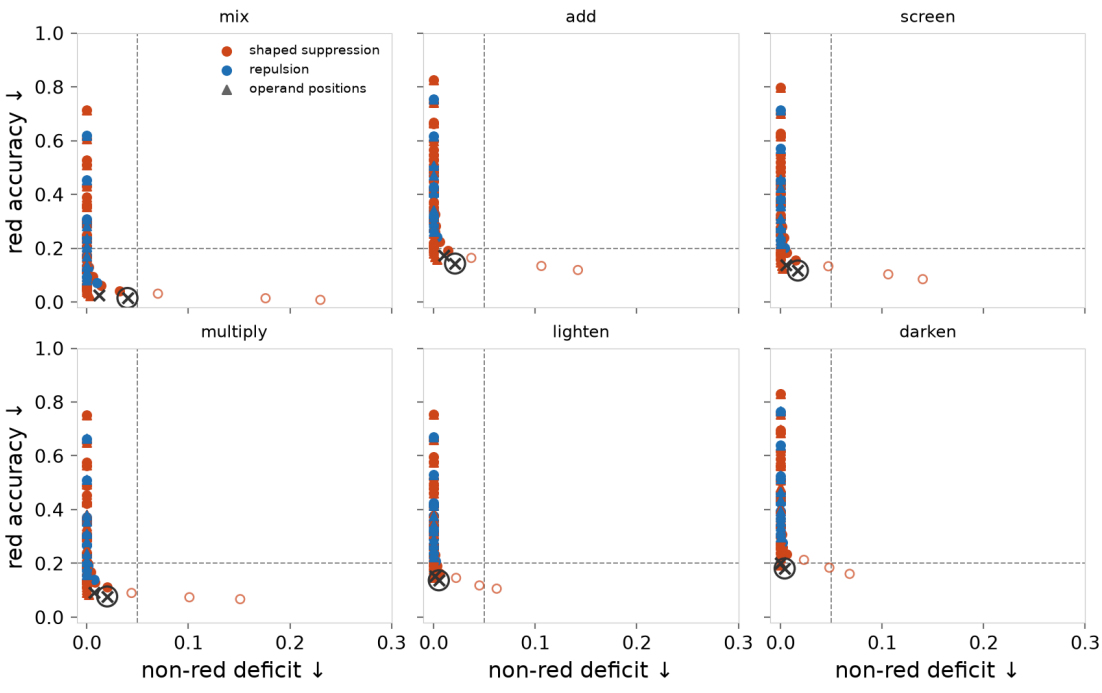
Measured under `projection` at the twenty seeds of `recipe-short`. The rule ranks on `mix`, and the band there is 0.013 on red accuracy and 0.013 on the non-red deficit.

op	red accuracy	σ	non-red deficit	σ
`mix`	0.015 ±0.042	0.021	0.040 ±0.036	0.021
`add`	0.143 ±0.178	0.075	0.021 ±0.021	0.013
`screen`	0.117 ±0.142	0.064	0.017 ±0.015	0.009
`multiply`	0.077 ±0.110	0.053	0.020 ±0.018	0.011
`lighten`	0.137 ±0.099	0.048	0.005 ±0.011	0.005
`darken`	0.181 ±0.181	0.085	0.004 ±0.013	0.006

Per-run spread of the two objectives, by op. *Seed mean ± half the seed range, and the per-run standard deviation the bands are built from.*

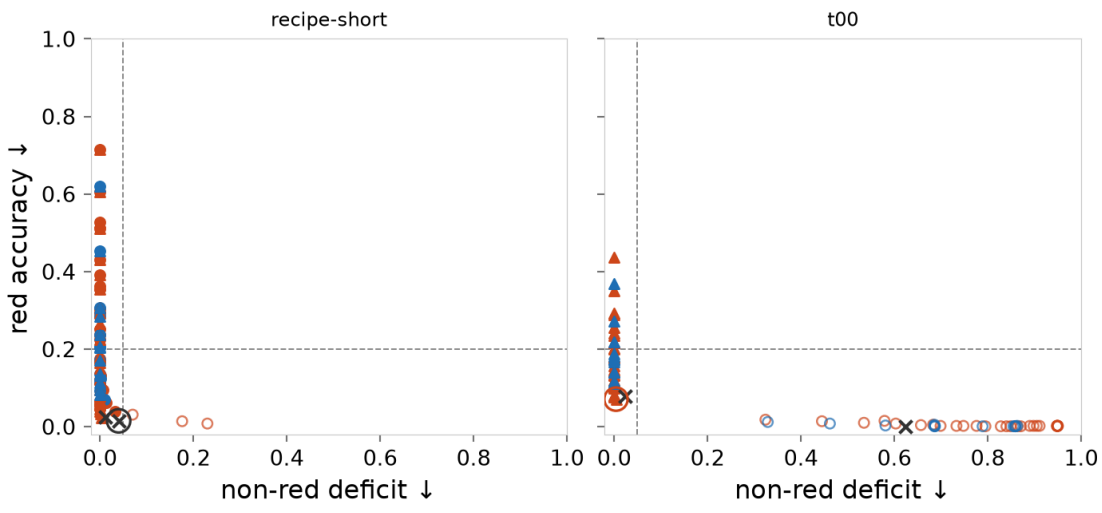
The landscape

Every trial on every op, on the two reads the objective uses. On the adopted point nearly every trial stands at zero deficit, and the position mask makes no difference; what separates the trials is how much red they leave. Two exceptions: the steps set inside the non-red range trail off to the right, and the plain projection sits just inside the gate.



The landscape: removal against selectivity, per op. *One mark per trial, seed means over the twenty seeds of* `recipe-short`*: ● at every position, ▲ at the operand positions, filled when the trial is inside the deficit gate on every op and open otherwise, in the family's ink. × marks the two reference projections. Dashed lines are ex-2.2.3's gates (0.05 on the deficit, 0.2 on red accuracy); the corner they enclose is where an operator is both selective and complete. The ring is the proposed trial. The deficit axis is zoomed to the range the adopted point uses; the figure below shows the full range beside* `t00`*.*

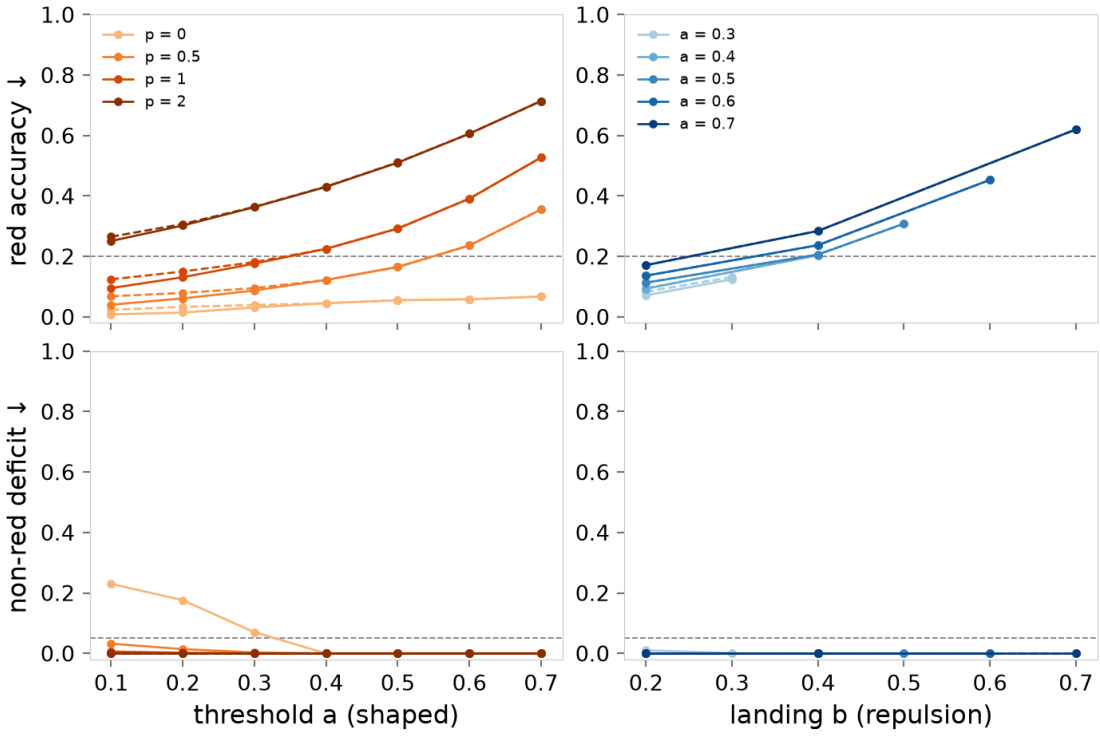
Which operator is best depends on the point. On `t00` the syntax embeddings carry the axis, and there every edit applied at every position costs most of the non-red lines, whatever its threshold, ramp, or landing. The operand-only edits are the only ones inside the gate.



The same landscape on `mix`***, at the adopted point and at*** `t00`***.*** *Seed means over twenty seeds (left) and five (right), the marks as above; the ring is the frozen rule's pick on each point. The axes run the full range on both panels so the two points can be compared; the per-op figure above zooms into the corner. Dashed lines are the gates (0.05, 0.2).*

The marginals

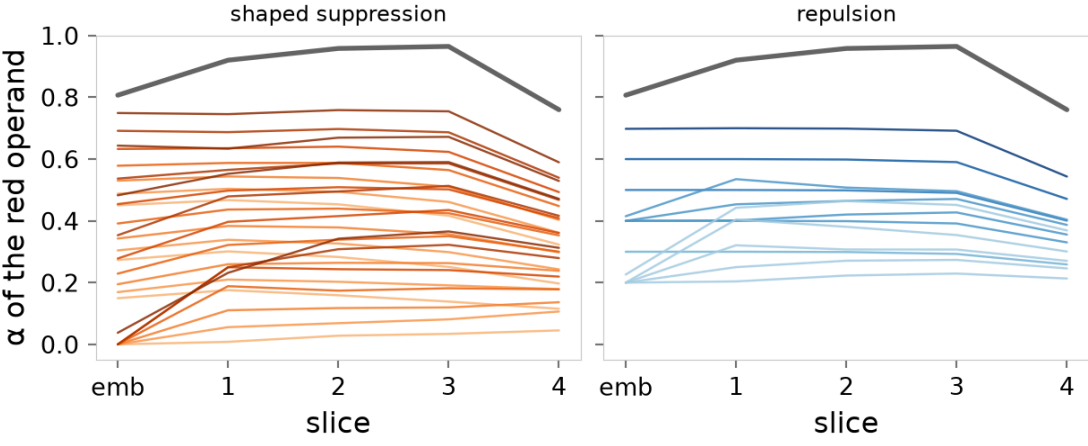
The same two reads on `mix`, now plotted against the parameters of each family. For the shaped suppression, the threshold and the ramp both set how much *red* is left. Its cost is zero everywhere except for the step at low thresholds applied at every position. For repulsion what matters is the landing: red accuracy follows *b*, and the threshold adds a little on top.



The marginals on `mix`. *Seed means over twenty seeds. Left: shaped suppression against its threshold a, one shade per ramp p. Right: repulsion against its landing b, one shade per threshold a. Both panels are keyed by their legends. Solid lines are the whole-sequence trials, dashed the operand-only ones. The dashed grey rule is the gate.*

Where the operators leave the state

In both families, how much *red* an operator leaves tracks where it leaves the state of the red operand. The shaped suppression lands each state wherever its ramp puts it, so the landing varies with where the state arrived. Repulsion lands every state it touches at one alignment, and the stream holds it there through slice 3. The last slice pulls every landing down, including the clean state.

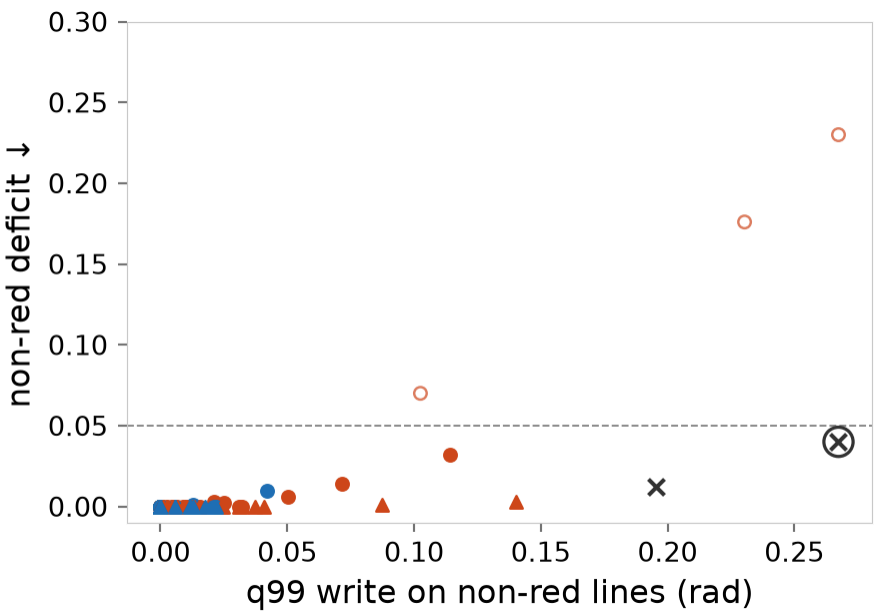


Where the operators leave the red operand, by slice, on the red `mix` lines.

Mean alignment of the dose-carrying operand's state after the edit, over lines and twenty seeds, for every whole-sequence trial; the bold grey line is the clean value. Shaped rows shade by threshold (faintest to boldest: 0.1 to 0.7), repulsion rows by landing b (faintest to boldest: 0.2 to 0.7).

The write and its cost

The write is the angle an operator turns a state through. M1 argued for the shaped operators on the grounds that keeping the write small keeps the side-effect small. That only holds if the two move together, and on the non-red lines of this point they do not. The plain projection and the step at a = 0.1 turn the non-red lines through the same angle, yet the step costs them several times as much. A step at a = 0.2 turns them less and still costs four times what the projection does. What the non-red lines pay depends more on which of their states are turned than on how far.



What the non-red lines pay for the write they receive, on `mix`. *The 99th-percentile write of each trial on the non-red lines (the largest over the five slices, seed mean) against its non-red deficit. Same marks as the landscape.*

Every trial

No omissions. Seed means over the twenty seeds of `recipe-short`. ⦿ is the proposed trial and ⋆ marks the `mix` front; bold values pass their gate. *Margin* is the gate minus the worst non-red deficit over the six ops, which is what the survey ranks on alongside the objective.

trial	family	parameters	positions	red acc `mix` ↓	deficit `mix` ↓	worst red acc ↓	worst deficit ↓	margin ↑	feasible
`projection` ● ⦿ ⋆	projection	–	all	**0.015**	**0.040**	**0.181**	**0.040**	+0.010	✓
`shaped-a0.1-p0` ● ⋆	shaped suppression	a 0.1, p 0	all	**0.008**	0.230	**0.161**	0.230	-0.180	
`shaped-a0.1-p0.5` ●	shaped suppression	a 0.1, p 0.5	all	**0.040**	**0.032**	0.235	**0.032**	+0.018	✓
`shaped-a0.1-p1` ●	shaped suppression	a 0.1, p 1	all	**0.095**	**0.006**	0.309	**0.006**	+0.044	✓
`shaped-a0.1-p2` ●	shaped suppression	a 0.1, p 2	all	0.251	**0.000**	0.461	**0.000**	+0.050	✓
`shaped-a0.2-p0` ● ⋆	shaped suppression	a 0.2, p 0	all	**0.014**	0.176	**0.184**	0.176	-0.126	
`shaped-a0.2-p0.5` ●	shaped suppression	a 0.2, p 0.5	all	**0.061**	**0.014**	0.259	**0.014**	+0.036	✓
`shaped-a0.2-p1` ●	shaped suppression	a 0.2, p 1	all	**0.131**	**0.002**	0.345	**0.002**	+0.048	✓
`shaped-a0.2-p2` ●	shaped suppression	a 0.2, p 2	all	0.302	**0.000**	0.511	**0.000**	+0.050	✓
`shaped-a0.3-p0` ●	shaped suppression	a 0.3, p 0	all	**0.031**	0.070	0.213	0.070	-0.020	
`shaped-a0.3-p0.5` ●	shaped suppression	a 0.3, p 0.5	all	**0.087**	**0.003**	0.297	**0.003**	+0.047	✓
`shaped-a0.3-p1` ●	shaped suppression	a 0.3, p 1	all	**0.176**	**0.000**	0.393	**0.000**	+0.050	✓
`shaped-a0.3-p2` ●	shaped suppression	a 0.3, p 2	all	0.363	**0.000**	0.561	**0.000**	+0.050	✓
`shaped-a0.4-p0` ●	shaped suppression	a 0.4, p 0	all	**0.045**	**0.000**	0.236	**0.000**	+0.050	✓
`shaped-a0.4-p0.5` ●	shaped suppression	a 0.4, p 0.5	all	**0.122**	**0.000**	0.339	**0.000**	+0.050	✓
`shaped-a0.4-p1` ●	shaped suppression	a 0.4, p 1	all	0.225	**0.000**	0.443	**0.000**	+0.050	✓
`shaped-a0.4-p2` ●	shaped suppression	a 0.4, p 2	all	0.430	**0.000**	0.617	**0.000**	+0.050	✓
`shaped-a0.5-p0` ●	shaped suppression	a 0.5, p 0	all	**0.055**	**0.000**	0.255	**0.000**	+0.050	✓
`shaped-a0.5-p0.5` ●	shaped suppression	a 0.5, p 0.5	all	**0.165**	**0.000**	0.390	**0.000**	+0.050	✓
`shaped-a0.5-p1` ●	shaped suppression	a 0.5, p 1	all	0.292	**0.000**	0.506	**0.000**	+0.050	✓
`shaped-a0.5-p2` ●	shaped suppression	a 0.5, p 2	all	0.510	**0.000**	0.685	**0.000**	+0.050	✓
`shaped-a0.6-p0` ●	shaped suppression	a 0.6, p 0	all	**0.058**	**0.000**	0.267	**0.000**	+0.050	✓
`shaped-a0.6-p0.5` ●	shaped suppression	a 0.6, p 0.5	all	0.236	**0.000**	0.462	**0.000**	+0.050	✓
`shaped-a0.6-p1` ●	shaped suppression	a 0.6, p 1	all	0.391	**0.000**	0.588	**0.000**	+0.050	✓
`shaped-a0.6-p2` ●	shaped suppression	a 0.6, p 2	all	0.606	**0.000**	0.753	**0.000**	+0.050	✓
`shaped-a0.7-p0` ●	shaped suppression	a 0.7, p 0	all	**0.067**	**0.000**	0.275	**0.000**	+0.050	✓
`shaped-a0.7-p0.5` ●	shaped suppression	a 0.7, p 0.5	all	0.355	**0.000**	0.564	**0.000**	+0.050	✓
`shaped-a0.7-p1` ●	shaped suppression	a 0.7, p 1	all	0.527	**0.000**	0.696	**0.000**	+0.050	✓
`shaped-a0.7-p2` ●	shaped suppression	a 0.7, p 2	all	0.714	**0.000**	0.831	**0.000**	+0.050	✓
`linear-a0.3-b0.2` ●	repulsion	a 0.3, b 0.2	all	**0.071**	**0.010**	0.277	**0.010**	+0.040	✓
`linear-a0.3-b0.3` ●	repulsion	a 0.3, b 0.3	all	**0.124**	**0.001**	0.340	**0.001**	+0.049	✓
`linear-a0.4-b0.2` ●	repulsion	a 0.4, b 0.2	all	**0.093**	**0.000**	0.307	**0.000**	+0.050	✓
`linear-a0.4-b0.4` ●	repulsion	a 0.4, b 0.4	all	0.204	**0.000**	0.426	**0.000**	+0.050	✓
`linear-a0.5-b0.2` ●	repulsion	a 0.5, b 0.2	all	**0.113**	**0.000**	0.333	**0.000**	+0.050	✓
`linear-a0.5-b0.4` ●	repulsion	a 0.5, b 0.4	all	0.206	**0.000**	0.432	**0.000**	+0.050	✓
`linear-a0.5-b0.5` ●	repulsion	a 0.5, b 0.5	all	0.308	**0.000**	0.525	**0.000**	+0.050	✓
`linear-a0.6-b0.2` ●	repulsion	a 0.6, b 0.2	all	**0.136**	**0.000**	0.367	**0.000**	+0.050	✓
`linear-a0.6-b0.4` ●	repulsion	a 0.6, b 0.4	all	0.237	**0.000**	0.462	**0.000**	+0.050	✓
`linear-a0.6-b0.6` ●	repulsion	a 0.6, b 0.6	all	0.453	**0.000**	0.639	**0.000**	+0.050	✓
`linear-a0.7-b0.2` ●	repulsion	a 0.7, b 0.2	all	**0.171**	**0.000**	0.395	**0.000**	+0.050	✓
`linear-a0.7-b0.4` ●	repulsion	a 0.7, b 0.4	all	0.284	**0.000**	0.513	**0.000**	+0.050	✓
`linear-a0.7-b0.7` ●	repulsion	a 0.7, b 0.7	all	0.620	**0.000**	0.765	**0.000**	+0.050	✓
`operands` ▲	projection	–	operands	**0.025**	**0.012**	0.201	**0.012**	+0.038	✓
`shaped-a0.1-p0-operands` ▲ ⋆	shaped suppression	a 0.1, p 0	operands	**0.023**	**0.003**	**0.194**	**0.003**	+0.047	✓
`shaped-a0.1-p0.5-operands` ▲	shaped suppression	a 0.1, p 0.5	operands	**0.068**	**0.000**	0.272	**0.000**	+0.050	✓
`shaped-a0.1-p1-operands` ▲	shaped suppression	a 0.1, p 1	operands	**0.124**	**0.000**	0.349	**0.000**	+0.050	✓
`shaped-a0.1-p2-operands` ▲	shaped suppression	a 0.1, p 2	operands	0.265	**0.000**	0.483	**0.000**	+0.050	✓
`shaped-a0.2-p0-operands` ▲ ⋆	shaped suppression	a 0.2, p 0	operands	**0.033**	**0.001**	0.205	**0.001**	+0.049	✓
`shaped-a0.2-p0.5-operands` ▲	shaped suppression	a 0.2, p 0.5	operands	**0.079**	**0.000**	0.290	**0.000**	+0.050	✓
`shaped-a0.2-p1-operands` ▲	shaped suppression	a 0.2, p 1	operands	**0.150**	**0.000**	0.373	**0.000**	+0.050	✓
`shaped-a0.2-p2-operands` ▲	shaped suppression	a 0.2, p 2	operands	0.307	**0.000**	0.523	**0.000**	+0.050	✓
`shaped-a0.3-p0-operands` ▲ ⋆	shaped suppression	a 0.3, p 0	operands	**0.039**	**0.000**	0.222	**0.000**	+0.050	✓
`shaped-a0.3-p0.5-operands` ▲	shaped suppression	a 0.3, p 0.5	operands	**0.095**	**0.000**	0.314	**0.000**	+0.050	✓
`shaped-a0.3-p1-operands` ▲	shaped suppression	a 0.3, p 1	operands	**0.181**	**0.000**	0.405	**0.000**	+0.050	✓
`shaped-a0.3-p2-operands` ▲	shaped suppression	a 0.3, p 2	operands	0.364	**0.000**	0.566	**0.000**	+0.050	✓
`shaped-a0.4-p0-operands` ▲	shaped suppression	a 0.4, p 0	operands	**0.045**	**0.000**	0.239	**0.000**	+0.050	✓
`shaped-a0.4-p0.5-operands` ▲	shaped suppression	a 0.4, p 0.5	operands	**0.122**	**0.000**	0.346	**0.000**	+0.050	✓
`shaped-a0.4-p1-operands` ▲	shaped suppression	a 0.4, p 1	operands	0.225	**0.000**	0.448	**0.000**	+0.050	✓
`shaped-a0.4-p2-operands` ▲	shaped suppression	a 0.4, p 2	operands	0.431	**0.000**	0.619	**0.000**	+0.050	✓
`shaped-a0.5-p0-operands` ▲	shaped suppression	a 0.5, p 0	operands	**0.055**	**0.000**	0.257	**0.000**	+0.050	✓
`shaped-a0.5-p0.5-operands` ▲	shaped suppression	a 0.5, p 0.5	operands	**0.165**	**0.000**	0.394	**0.000**	+0.050	✓
`shaped-a0.5-p1-operands` ▲	shaped suppression	a 0.5, p 1	operands	0.292	**0.000**	0.509	**0.000**	+0.050	✓
`shaped-a0.5-p2-operands` ▲	shaped suppression	a 0.5, p 2	operands	0.511	**0.000**	0.685	**0.000**	+0.050	✓
`shaped-a0.6-p0-operands` ▲	shaped suppression	a 0.6, p 0	operands	**0.058**	**0.000**	0.267	**0.000**	+0.050	✓
`shaped-a0.6-p0.5-operands` ▲	shaped suppression	a 0.6, p 0.5	operands	0.236	**0.000**	0.463	**0.000**	+0.050	✓
`shaped-a0.6-p1-operands` ▲	shaped suppression	a 0.6, p 1	operands	0.391	**0.000**	0.589	**0.000**	+0.050	✓
`shaped-a0.6-p2-operands` ▲	shaped suppression	a 0.6, p 2	operands	0.606	**0.000**	0.753	**0.000**	+0.050	✓
`shaped-a0.7-p0-operands` ▲	shaped suppression	a 0.7, p 0	operands	**0.067**	**0.000**	0.276	**0.000**	+0.050	✓
`shaped-a0.7-p0.5-operands` ▲	shaped suppression	a 0.7, p 0.5	operands	0.355	**0.000**	0.564	**0.000**	+0.050	✓
`shaped-a0.7-p1-operands` ▲	shaped suppression	a 0.7, p 1	operands	0.527	**0.000**	0.697	**0.000**	+0.050	✓
`shaped-a0.7-p2-operands` ▲	shaped suppression	a 0.7, p 2	operands	0.714	**0.000**	0.831	**0.000**	+0.050	✓
`linear-a0.3-b0.2-operands` ▲	repulsion	a 0.3, b 0.2	operands	**0.083**	**0.000**	0.297	**0.000**	+0.050	✓
`linear-a0.3-b0.3-operands` ▲	repulsion	a 0.3, b 0.3	operands	**0.132**	**0.000**	0.356	**0.000**	+0.050	✓
`linear-a0.4-b0.2-operands` ▲	repulsion	a 0.4, b 0.2	operands	**0.094**	**0.000**	0.314	**0.000**	+0.050	✓
`linear-a0.4-b0.4-operands` ▲	repulsion	a 0.4, b 0.4	operands	0.204	**0.000**	0.433	**0.000**	+0.050	✓
`linear-a0.5-b0.2-operands` ▲	repulsion	a 0.5, b 0.2	operands	**0.113**	**0.000**	0.339	**0.000**	+0.050	✓
`linear-a0.5-b0.4-operands` ▲	repulsion	a 0.5, b 0.4	operands	0.206	**0.000**	0.436	**0.000**	+0.050	✓
`linear-a0.5-b0.5-operands` ▲	repulsion	a 0.5, b 0.5	operands	0.308	**0.000**	0.528	**0.000**	+0.050	✓
`linear-a0.6-b0.2-operands` ▲	repulsion	a 0.6, b 0.2	operands	**0.136**	**0.000**	0.371	**0.000**	+0.050	✓
`linear-a0.6-b0.4-operands` ▲	repulsion	a 0.6, b 0.4	operands	0.237	**0.000**	0.465	**0.000**	+0.050	✓
`linear-a0.6-b0.6-operands` ▲	repulsion	a 0.6, b 0.6	operands	0.453	**0.000**	0.640	**0.000**	+0.050	✓
`linear-a0.7-b0.2-operands` ▲	repulsion	a 0.7, b 0.2	operands	**0.171**	**0.000**	0.396	**0.000**	+0.050	✓
`linear-a0.7-b0.4-operands` ▲	repulsion	a 0.7, b 0.4	operands	0.284	**0.000**	0.514	**0.000**	+0.050	✓
`linear-a0.7-b0.7-operands` ▲	repulsion	a 0.7, b 0.7	operands	0.620	**0.000**	0.766	**0.000**	+0.050	✓

Every trial on `recipe-short`. *Reference rows are shaded.*

The proposal

The frozen rule proposes **`projection`**, the plain projection at every position. Its margin to the gate is +0.010, less than the deficit band of 0.013. The margin of the operand-only projection is +0.038. 2 other feasible trials sit within one red-accuracy band of the proposal: `operands`, `shaped-a0.1-p0-operands`. The `mix` front, from the gentlest edit to the most complete, as (red accuracy, deficit): `shaped-a0.3-p0-operands` (0.039, 0.000), `shaped-a0.2-p0-operands` (0.033, 0.001), `shaped-a0.1-p0-operands` (0.023, 0.003), `projection` (0.015, 0.040), `shaped-a0.2-p0` (0.014, 0.176), `shaped-a0.1-p0` (0.008, 0.230).

On `t00` the same trial has a worst deficit of 0.623, outside the gate, and the rule picks `shaped-a0.1-p0-operands`, shaped suppression at threshold 0.1, ramp p = 0, at the operand positions there.

What the prereg should carry. The rule picks the plain projection, which is the row the anchored-op experiments already score, so on the adopted point this pass changes nothing about the primary intervention. What it adds is the margin: the projection is inside the gate by less than a band. So the prereg should keep the operand-only projection beside it as the selective reference.

Two things the rule did not rank are for the prereg to settle before its seeds are drawn. First, the answer depends on the point: on `t00` the same rule picks an operand-only step. A prereg that may run on a syntax-heavy point should name the operand-only operator as its intervention there, rather than choosing it after the read.

Second, there is a candidate that needs no syntax: applied at every position, `shaped-a0.4-p0` leaves red accuracy 0.045 at no cost the survey can resolve. That is close to the 0.025 of the operand-only projection, and it uses no position mask, which is what a prompt with unknown operand positions will need. Choosing it here would be post hoc, so it is recorded for the prereg to carry as a third row, re-scored at fresh seeds, if the syntax-free question is worth one.

Post hoc

Nothing was added after the run. The reference rows reproduced to the bit, and every contract check passed on every trial and seed; a failed check fails the scoring task, and none did. The plan allowed for the rule picking a reference row, since the references are trials and the objective ranks them with the rest.

One observation was not anticipated by the plan, and is recorded as exploratory. The plan expected a threshold to trade removal against cost monotonically, with the plain projection at the costly end. Instead, a step set inside the alignment range of the non-red lines costs more than the projection does, on the adopted point and on `t00` alike. A line whose states are zeroed on some tokens and kept on others is decoded worse than one zeroed throughout. That suggests a prediction the anchored-op prereg can carry: above the non-red range the whole-sequence cost is zero, and below it the cost rises as the threshold falls.