COMPOSITION LIVES AT THE SEAMS: ON-POLICY CORRECTION FOR COMPOSING INTERNALIZED SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

A skill that lives in a prompt is paid for on every call, and two skill documents pasted into one prompt do not teach a model to chain the skills they describe. Distilling the documents into the weights looks like the cure, and at first it is. A 3B student fine-tuned on flawless demonstrations of 27 documented text skills executes them from memory more reliably than a model over ten times its size reading the documentation. The cure fails where nobody demonstrated anything, on chains one step deeper than the training data. The failure sits at the seams, the handoffs where one skill's output becomes the next skill's input; the states that need repairing arise only in the student's own imperfect attempts, beyond off-policy reach. We therefore let the student attempt compositions, catch the first botched seam with a programmatic verifier, and train on a correct continuation spliced onto the student's own prefix. This lifts deep chains to the level of a 32B model reading the documentation, at no in-distribution cost. Two controls show what the gain rests on. Freeze the rollouts and it collapses; take corrections from a teacher weaker than the student and it reverses. Composition lives at the seams, in the states where the student stumbles.

1 INTRODUCTION

A language model can acquire a skill in two ways. It can read the skill's documentation at inference time, as in in-context learning (Brown et al., 2020) and in the skill libraries from which agents load instructions on demand (Anthropic, 2025; Wang et al., 2024). Or it can internalize the skill, distilling the documented behavior into its weights so that it executes the skill with an empty context (Snell et al., 2022; Askell et al., 2021). Reading is flexible but costly, since every prompt carries the documents and the cost grows with the library. Internalization removes that cost, and on individual skills it works well.

Agents, however, rarely use one skill at a time; they chain them, and composition is where internalization fails. On a testbed of 27 documented text-transformation skills, a 3B student fine-tuned on gold traces of chains up to depth three reaches 96% accuracy in distribution, above a 32B model that reads the documentation. On chains of depth five the same student falls to 64% (Figure 1). Step-level grading places these failures at the *seams* of a composition, the handoffs where one skill's output becomes the next skill's input, and their number grows with every additional handoff. In one real failure, the student reaches the second step of an unseen pairing, writes out the bookkeeping of a different skill under the header of the one it was asked to apply, and never recovers (Appendix E).

The cause is familiar from imitation learning. Gold traces contain no errors, so they never show the student what to do after its own slip, and the states that need supervision arise only in the student's own rollouts. This is the exposure-bias problem (Ranzato et al., 2016; Bengio et al., 2015), which imitation learning addresses by querying an expert on the learner's own trajectory (Ross et al., 2011) and language modeling addresses by on-policy distillation, in which a teacher scores the student's own samples (Gu et al., 2024; Agarwal et al., 2024). On-policy distillation spreads supervision over every sampled token. In a composition, most tokens execute skills the student already has, and the tokens that decide the task sit at the seams.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

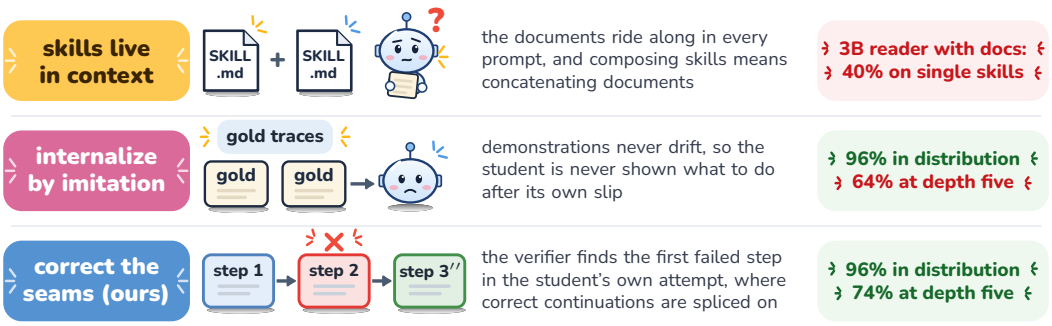

Figure 1: **Three ways to give a model a skill library.** Keeping documents in context costs tokens on every call and helps small models little; imitating flawless demonstrations internalizes the skills but not their composition; correcting the student's own seams keeps in-distribution accuracy and extends composition to deeper chains.

We therefore propose *seam-corrective on-policy learning*. The student attempts compositions, a programmatic verifier finds the first step that fails, and training supervises a correct continuation spliced onto the student's own prefix at that step, with extra weight on the tokens that follow each handoff. Unlike on-policy distillation, it spends supervision only on failed rollouts, and within them only from the failed seam onward. Our contributions are as follows.

- **A checkable testbed.** Twenty-seven documented skills with executable reference implementations, so that every intermediate state of every composition is graded by code, without a judge model (Section 2).
- **A correction method.** Seam-corrective learning in an oracle form that splices reference continuations, and a distilled form in which a stronger model reading the documentation scores the failed steps (Section 3).
- **Evidence for the mechanism.** On unseen depths four and five the method raises accuracy from 73.8% to 81.5%, significant in each of three seeds, with no in-distribution cost. The gain disappears when the rollouts are frozen, when the up-weighting moves off the seams, or when the corrector is weaker than the student (Sections 5 and 6).
- **A measured cost.** Skills that never appear in training compositions lose accuracy, and the loss traces to the splices that continue from the student's own errors; a fix-only variant roughly halves it (Section 6).

2 Problem Setup

Skills, chains, and seams. A skill σ is a deterministic function from word sequences to word sequences, described to the model by a documentation file. A task of depth k is a chain $\sigma_{1..k}$ with input u; its gold states are $s_0 = u$ and $s_i = \sigma_i(s_{i-1})$, and its answer is s_k. A model answers in a fixed format, writing for each step i a work line and a result line that states its intermediate sequence o_i, then a final line. Step i is *correct* if $o_i = s_i$ and *conditionally correct* if $o_i = \sigma_i(o_{i-1})$, with $o_0 = u$, which means that the skill was applied correctly to whatever the model itself produced one step earlier. We call the handoff from step $i{-}1$ to step i a *seam*, and the first step of a rollout that is not conditionally correct, $f = \min\{\, i : o_i \neq \sigma_i(o_{i-1}) \,\}$, its first failed seam. Because every skill is executable, both checks are computed by code for any sequence the model writes, and seams are located exactly rather than inferred.

Testbed. We build the testbed in the spirit of controlled compositionality studies such as SCAN, COGS, and CFQ (Lake & Baroni, 2018; Kim & Linzen, 2020; Keysers et al., 2020) and the systematicity taxonomy of Hupkes et al. (2020), adapted to instruction-following models. Each of the 27 skills transforms a sequence of lowercase words, for example by reversing it, rotating it, keeping every second word, or sorting it alphabetically. Each ships as a documentation file in the style of an agent skill library, with rules, a bookkeeping protocol for the work line, and two worked exam-

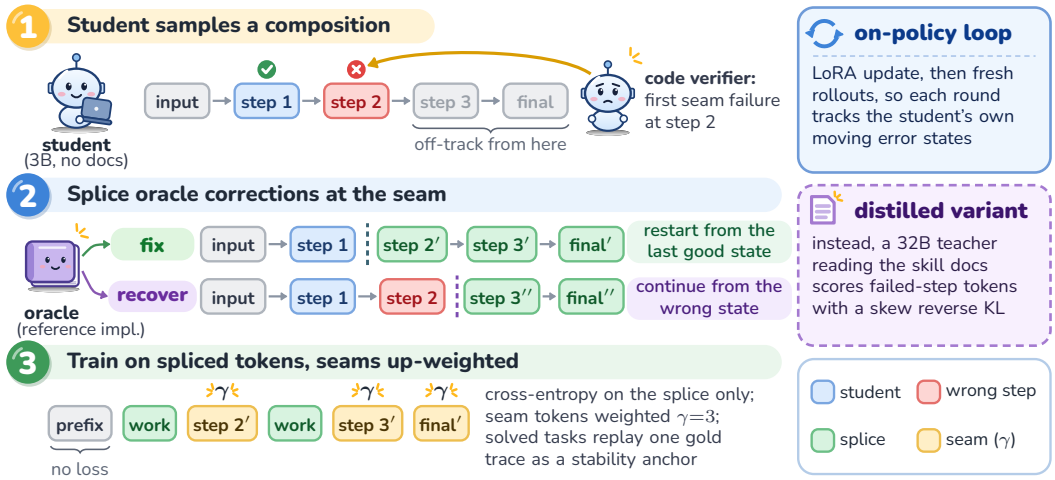

Figure 2: **One round of seam-corrective on-policy learning.** The verifier locates the first seam failure in a student rollout; fix and recover splices attach correct continuations to the student's own prefix, and the loss covers the spliced tokens together with gold anchors from solved tasks.

ples. A depth-two task, with `mirror-run` reversing the sequence and `moss-cull` keeping odd positions, looks as follows.

```
work: 1:ash 2:bay 3:elm 4:fern
      | read 4,3,2,1 -> fern elm bay ash
Step 1 [mirror-run]: fern elm bay ash
work: 1:fern 2:elm 3:bay 4:ash
      | keep 1:fern 3:bay
Step 2 [moss-cull]: fern bay
FINAL: fern bay
```

Skill names are codenames such as `vapor-knot` or `willow-gate`, so the name alone reveals little; a 3B model without documentation solves 8% of atomic tasks and at most 5% of any composed cell (Table 2). The bookkeeping protocol came out of calibration. Unstructured chain-of-thought (Wei et al., 2022; Zhou et al., 2023) is not enough here. A 7B reader solved 23.4% of atomic screening tasks when answering directly and 55.0% when allowed to reason first, and the documented protocol makes teacher-grade execution reliable enough to gate the study (Appendix C).

Splits and evaluation. Nineteen core skills appear in training compositions. Eight *held-out* skills are trained only atomically and never inside a training chain. Training chains have depth $k \in \{1, 2, 3\}$ and use 40% of the ordered core pairs. Evaluation covers nine cells, each probing a different kind of generalization. Atomic tasks and seen pairs at depths two and three measure in-distribution accuracy; unseen pairs of trained skills measure recombination at a trained depth; pairs involving held-out skills ask whether composition extends to skills never exercised in a chain; and depths four and five, which no training regime visits, ask whether it extends along its own axis. Trained models are evaluated greedily without documentation, 200 tasks per cell.

The off-policy baseline. Every trained model in this paper starts from the same initialization, the 3B student fine-tuned to convergence on gold traces of the training distribution, which we call SFT-gold. It is the strongest off-policy baseline we could construct, since the rigid output format makes the correct trace unique and verifier-filtered teacher demonstrations therefore coincide with gold.

3 Seam-Corrective On-Policy Learning

Seam-corrective learning starts from SFT-gold and runs in rounds. Each round samples rollouts, turns every failed rollout into corrected sequences at its first failed seam, and takes gradient steps on those sequences together with gold anchors (Figure 2 and Algorithm 1).

Algorithm 1 Splice construction for one failed rollout

Require: chain $\sigma_{1..k}$, input u, rollout tokens $\hat{y}$ with a wrong final answer, parsed step outputs $o_{1..k}$, conditional checks $v_{1..k}$

$f \leftarrow$ first index with $v_f \neq$ PASS; **return** $\emptyset$ if none
$S \leftarrow \emptyset$
if outputs $o_{1..f-1}$ are all valid word sequences **then**
  $s \leftarrow o_{f-1}$, or u if $f{=}1$ {on-chain by induction}
  $S \leftarrow S \cup \{\, \hat{y}_{1..\mathrm{end}(f-1)} \oplus \textsc{GoldCont}(\sigma, s, f) \,\}$ {fix}
end if
if $f < k$ and o_f is a valid word sequence **then**
  $S \leftarrow S \cup \{\, \hat{y}_{1..\mathrm{end}(f)} \oplus \textsc{GoldCont}(\sigma, o_f, f{+}1) \,\}$ {recover}
end if
return S, which enters $B_{\mathcal{C}}$ of Equation 2 with c_y at the first $\oplus$-appended token

Rollouts and seam detection. The student samples 192 compositions from the training distribution at temperature 0.7, prompted with x, which names the skills but contains no documentation. The verifier grades every rollout. A rollout with a correct final answer contributes the gold trace of its task as an anchor. A rollout with a wrong final answer is corrected at its first failed seam f.

Fix and recover splices. Let $\hat{y}$ be the tokens of a failed rollout and $\hat{y}_{\le i}$ its prefix through the result line of step i. Let $\textsc{Gold}(\sigma, s, j)$ denote the reference continuation that applies $\sigma_j, \ldots, \sigma_k$ to a sequence s in the trace format, work lines included. The two splices are

$$y^{\mathrm{fix}} = \hat{y}_{\le f-1} \oplus \textsc{Gold}(\sigma, o_{f-1}, f), \qquad y^{\mathrm{rec}} = \hat{y}_{\le f} \oplus \textsc{Gold}(\sigma, o_f, f{+}1). \tag{1}$$

Every step before f is conditionally correct, so $o_{f-1} = s_{f-1}$ and the fix splice supplies gold content for exactly the seam the student failed, attached to the student's own formatting of the prefix. The recover splice keeps the wrong state o_f and continues correctly from it, which is possible because the skills are functions on any word sequence. It supervises the property a composition operator needs, applying the next skill to whatever the previous step produced, on states that gold data never contains; it exists only when $f < k$.

Objective. Let c_y be the position of the first spliced token of y. Seam weights w_t are $\gamma{=}3$ on the result lines of steps two onward and on the final line, the states written right after each handoff, and 1 elsewhere. A minibatch B minimizes

$$\mathcal{L}(B) \;=\; \frac{1}{Z_B}\Big(\sum_{y \in B_{\mathcal{C}}} \sum_{t \ge c_y} w_t \, \ell_t(y) \;+\; 0.3 \sum_{y \in B_{\mathcal{G}}} \sum_{t} \ell_t(y) \Big), \qquad \ell_t(y) = -\log \pi_\theta(y_t \mid x, y_{<t}), \tag{2}$$

where $B_{\mathcal{C}}$ holds the fix and recover splices in B, $B_{\mathcal{G}}$ the gold anchors, and Z_B the summed weight of all trained tokens. The anchor term was added after a 1.5B pilot in which training on self-sampled targets collapsed, and fixed gold targets prevented the collapse (Appendix I). Training updates a rank-32 LoRA adapter (Hu et al., 2022) for 40 rounds at learning rate 2×10^{-5}. In a typical round about seventeen of the 192 rollouts fail, yielding some twenty splices against 175 anchors.

Distilled variant. The oracle above is the reference implementation, which establishes what seam correction can achieve. Many domains, however, can check a step without being able to solve it. In the distilled variant, a 32B instruction model (Qwen, 2024) reads the full documentation and scores the student's own tokens in each conditionally failed step with a per-token skew reverse KL, in the manner of on-policy distillation (Agarwal et al., 2024; Gu et al., 2024; Ko et al., 2024). Nothing is spliced; the gate, the seam weights, and the anchor are unchanged.

4 Experimental Setup

Models. The student is Qwen2.5-3B-Instruct throughout. Teachers, where used, are Qwen2.5-14B and Qwen2.5-32B reading the documentation in context; they share the student's tokenizer, so per-token scores align exactly. Evaluation is greedy, and trained models never see documentation. All training fits on single 32GB V100 GPUs except the 32B teacher, which is sharded across three.

Arm	Rollouts	Signal on failed rollouts	Weighting
SFT-gold	none	gold traces only	uniform
STaR	on-policy	none; CE on own solved rollouts	uniform
Seam-corrective, oracle (ours)	on-policy	fix and recover splices	seam
fix only, recover only	on-policy	one splice type	seam
frozen rollouts	fixed at initialization	fix and recover splices	seam
uniform, anti-seam	on-policy	fix and recover splices	uniform, mid-line
Seam-corrective, 14B or 32B	on-policy	skew reverse KL on failed steps	seam

Table 1: **Trained arms.** Every arm starts from SFT-gold; all on-policy arms except STaR also train on gold anchors of solved tasks.

	In-distribution			Unseen pairs		Held-out skills		Deeper chains	
Training	k=1	k=2	k=3	k=2	k=3	k=2	k=3	k=4	k=5
Base (no context)	8.3	3.3	3.3	5.0	1.7	0.0	0.0	1.7	0.0
Base + docs in context	40.0	18.3	1.7	10.0	1.7	11.7	3.3	3.3	5.0
14B teacher + docs	95.0	83.3	56.7	83.3	48.3	71.7	71.7	50.0	50.0
32B teacher + docs	93.3	91.7	93.3	93.3	86.7	95.0	91.7	85.0	76.7
SFT on gold traces	94.5	97.5	**96.5**	96.5	90.0	**59.0**	**47.0**	83.5	64.0
STaR (self-CE)	58.0	57.5	40.0	50.0	37.0	16.5	15.0	35.5	20.5
Seam-corrective, 14B teacher	93.0	95.5	89.5	92.5	89.5	55.0	39.5	82.5	46.5
Frozen-rollout corrections	**96.5**	**98.0**	93.5	95.0	**93.5**	57.0	36.5	88.5	58.0
Seam-corrective, 32B teacher	93.2	97.5	93.5	95.8	92.5	54.2	39.0	88.8	72.5
Seam-corrective, oracle (ours)	94.8	97.7	96.3	**96.7**	93.3	53.2	38.7	**89.2**	**73.8**

Table 2: **Main results.** Task success (%) on nine cells of 200 greedy tasks; trained models see no documentation. The top block holds untrained reference points on 60 tasks per cell. Multi-seed rows report seed means, and bold marks the best trained model per column.

Baselines and controls. We compare against SFT-gold itself; STaR-style self-training (Zelikman et al., 2022; Yuan et al., 2023), which trains with cross-entropy on the student's own verifier-correct rollouts; and a frozen-rollout control, which applies the same correction pipeline to rollouts generated once from the initialization and never refreshed. The frozen control isolates the effect of on-policy data, in the spirit of the controls advocated for distillation (Agarwal et al., 2024). Ablations replace seam weighting with uniform or mid-line anti-seam weighting, drop one splice type, or swap the corrector; Table 1 lists every arm.

Statistics. A paired bootstrap resamples the 200 shared tasks of a cell, or of a pooled family of cells, 10,000 times and reports the probability that the baseline matches or beats the treatment, so smaller values are stronger evidence. The main method and its fix-only variant run with three seeds and the 32B variant with two; tables report seed means for these and single runs for every other arm. A claim counts as established only if it holds in every seed.

Teacher screening. Before a teacher may supervise, it must pass a per-skill screen of eight atomic tasks per skill with documentation in context. The 32B model passes with a mean of 0.986 and no skill below 0.875; the 14B model reaches 0.958 on the same 27 skills, a difference whose consequences Section 6 measures.

5 Results

Internalization succeeds in distribution and fails with depth. The reference rows of Table 2 establish the starting point. The untrained student is near the floor, at 8% atomic accuracy and at most 5% on any chain, so every gain reported below comes from training. With the documentation in context it reaches only 40% atomic accuracy and stays below 20% on every chain; a small model makes little use of a skill document it is given. SFT-gold internalizes the library almost perfectly

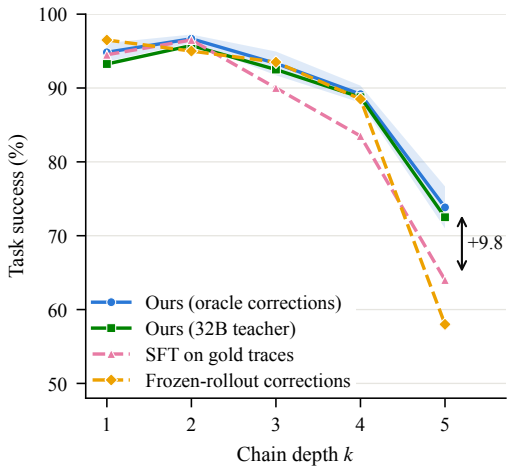

Figure 3: **Success by chain depth** on atomic tasks, unseen pairs, and deeper chains (seed means; the band spans one standard deviation). Training chains stop at k=3 for every arm.

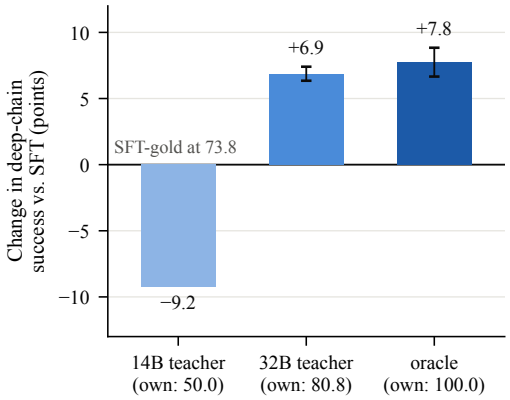

Figure 4: **Corrector quality is a dose-response axis.** Change in pooled deep-chain success over SFT-gold; each label gives the corrector's own deep-chain success.

Variant	In-dist.	Unseen	Held-out	Depth
SFT on gold traces	96.2	93.2	53.0	73.8
Seam correction (ours)	96.3±0.8	95.0±0.8	45.9±1.9	81.5±1.1
frozen rollouts	96.0	94.2	46.8	73.2
anti-seam weights	93.3	94.2	49.2	72.5
uniform weights	95.8	94.8	40.5	79.8
fix only	96.5±0.3	95.2±0.4	49.8±3.4	78.6±2.8
recover only	94.5	93.5	35.5	81.2

Table 3: **Ablations.** Pooled success (%) by evaluation family, ± std over seeds.

in distribution, at 94.5–97.5%, above the 32B teacher reading the documentation on those cells. Accuracy then drops with depth, to 84% at k=4 and 64% at k=5, a loss of thirteen points at the first seam past the training horizon and twenty more at the second, on compositions of skills that the model executes reliably on their own. The conditional accuracies in Figure 6 show the same decline step by step, although the student solves every skill in these chains without error in isolation.

Seam correction improves deeper chains. Forty rounds of seam-corrective training raise pooled accuracy on the deep cells from 73.8% to 81.5% (Figure 3; k=4: 83.5 → 89.2, k=5: 64.0 → 73.8). The paired bootstrap gives $p \leq 0.0003$ for the pooled gain in each of three seeds, and the standard deviation across seeds is 1.1 points against a mean gain of 7.8. Unseen pairings at trained depths improve in all three seeds, from 93.2% to 95.0 ± 0.8%, although only the first seed is significant on its own (p=0.024, against 0.09 and 0.24). In-distribution cells are statistically unchanged at 96.3 ± 0.8%. Held-out pairings drop from 53.0% to 45.9 ± 1.9%, a cost that Table 3 traces to one component and Section 6 analyzes. The 32B-teacher variant reproduces the depth gain (80.6%, $p \leq 0.0002$ in both seeds) at a small in-distribution cost. This matters in practice, because the distilled variant needs no solver, only a step check and a corrector that is stronger than the student where the student fails.

Compared with the reference rows, the corrected 3B student outperforms the 32B teacher reading the documentation on every in-distribution cell (95 to 98 against 92 to 93) and at depth four (89.2 against 85.0), and trails it only at depth five (73.8 against 76.7) and on held-out pairings. A model one tenth the size, with a prompt one quarter as long, thus saves context without giving up accuracy.

Contrast	$k1$	$k2$s	$k3$s	$k2$u	$k3$u	$k2$h	$k3$h	$k4$	$k5$
Oracle vs. SFT	.167	.367	.223	.427	.017	.994	.976	.003	.006
32B vs. SFT	.977	.751	.996	.926	.172	.898	.979	.031	.002
Oracle vs. frozen	.755	.658	.010	.101	.310	.961	.075	.231	$<$.001

Table 4: **Per-cell significance.** Paired-bootstrap probability that the second arm matches or beats the first, from 10,000 resamples at seed 0, so small values favor the first arm. Columns s, u, and h hold seen, unseen, and held-out pairs.

Cost. Forty rounds of oracle-corrected training take ninety minutes on one V100 and build 857 correction splices; the oracle variant requires no teacher forward pass, and the 32B variant scores 537K teacher tokens, all on failed rollouts. At inference the internalized student reads 187 to 296 prompt tokens per task, against 348 to 1,118 for the 32B teacher, which must carry the documentation on every call.

Where errors occur. The failure anatomy identifies depth as the main factor. SFT-gold makes at least one wrong step in 75 of its 200 depth-five chains, including chains that still reach the correct final answer. Fifty-four of these first errors occur at steps three and four, where drift has had the most room to accumulate, and eleven at steps one and two, which resemble the training data. Seam correction removes errors only where they concentrate. Averaged over seeds, the number of chains with a wrong step falls from 75 to 57 and the number of late first errors from 54 to 36, while steps one, two, and five are unchanged; Figure 6 shows the corresponding change in conditional accuracy. STaR-style self-training fails differently. Of the 178 depth-five chains in which it errs, 105 err at step one, so it loses the skills themselves, and it drops thirty to nearly sixty points per cell from the same initialization (Table 2).

Controls. Table 3 reports the controls. The frozen-rollout arm uses the same splices, seam weights, and anchors, but computes its corrections on rollouts from the initial policy that are never refreshed. Its depth score is 73.2%, against 81.5% for the on-policy method, and at $k{=}5$ the gap is 13.5 points (58.0 against 71.5 in seed 0, $p{=}0.0001$). At that cell the frozen arm falls below the uncorrected baseline, 58.0 against 64.0. Corrections computed for a forty-round-old snapshot target errors the student no longer makes, in states it no longer visits, and the effect is largest at the deepest cell. Moving the up-weighting onto mid-line anti-seam tokens costs 9.0 pooled depth points (8.5 in the paired seed-0 contrast, $p{=}0.0001$), and uniform weighting falls in between, so the useful signal is carried by the seam tokens. STaR, which only reinforces what the student already does well, shows that on-policy *data* without corrective *content* is not sufficient.

The weighting ablations also affect held-out skills. Uniform weighting, which spreads the gradient over whole splices, gives the lowest held-out score of the three weightings, 40.5%; seam weighting raises it to 45.9%; and anti-seam weighting, which places little weight on the handoff tokens, reaches 49.2% but loses the depth gain. Seam weighting is the only placement that improves on uniform weighting in both depth and held-out accuracy, so where the loss is placed determines what the correction costs as well as what it gains.

Per-cell significance. Table 4 gives the significance structure across all nine cells. Both corrected variants separate decisively from SFT-gold on the two depth cells and on no other cell, and the held-out columns lean toward the baseline for both, with three of the four entries above 0.97. The frozen contrast is strongest where the mechanism predicts, at the deepest cell ($p{<}0.001$).

6 Anatomy of the Correction Signal

We examine the corrector, the two splice types, the gold anchor, and the held-out cells that corrections never reach during training.

Corrector quality. Figure 4 varies the source of the corrections. Replacing the oracle with a 14B teacher that reads the documentation, which solves half of the deep chains in context where the SFT student solves three quarters (Table 2), drives depth to 64.5% under the distilled pipeline,

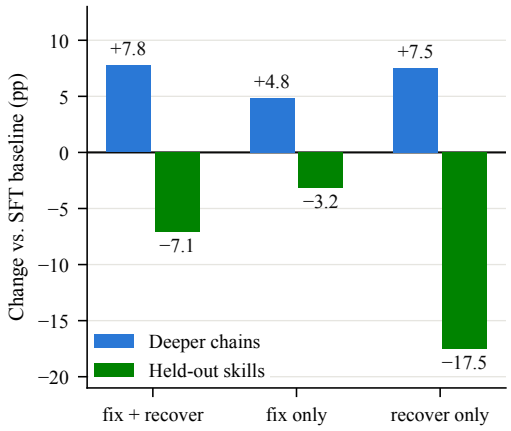

Figure 5: **Decomposing the correction.** Change against the SFT baseline in pooled success for each splice type.

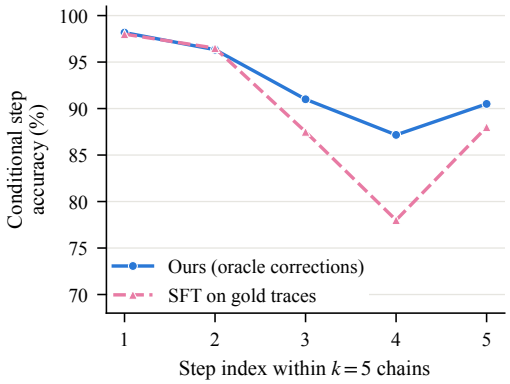

Figure 6: **Conditional step accuracy within k=5 chains.** Whether step i correctly transforms the student's own output from step $i-1$.

nine points *below* the baseline. Per-token supervision at failed steps pulls the student toward the corrector's distribution whether or not that distribution is better.

The 1.5B pilot exhibited the same effect. There, light corrective KL from a 14B teacher tied with the SFT baseline, and we first attributed the tie to a lack of signal, since nine rollouts in ten passed the verifier. Sampling four attempts per task and keeping every failure quadrupled the corrective mass and made the results significantly worse, 7.5 pooled out-of-distribution points below the baseline (Appendix I), so the diagnosis was wrong. Corrective KL toward that teacher was harmful, and only its scarcity had limited the damage, because a 14B model reading documentation is weaker than a gold-trained student on most cells. Scaling the student to 3B widened the gap and made even the light dose harmful, the 64.5% bar in Figure 4, while a 32B teacher, stronger than the student on every deep cell, made it beneficial again. A corrector should therefore be verified to outperform the student on the distribution being corrected before it is used.

Fix and recover splices. Figure 5 separates the two splice types. Fix-only training largely spares held-out skills (-3.2 points on the three-seed mean, against -7.1 for the full method) and still gains 4.8 depth points, but this gain varies across seeds (± 2.8). Recover-only training gains 7.5 depth points in its single run, close to the full method, and lowers held-out accuracy by 17.5 points. The full method combines both, with a robust depth gain and a held-out cost of 7.1 points. The input states explain the difference. Recover splices are the only supervision whose input states lie off the gold trajectory, which is what makes the composition robust to accumulated drift. Conditional accuracy at step four of k=5 chains shows this directly, reaching 90.5% under recover-only and 87.2% under both splices (Figure 6), against 81.7% under fix-only and 78.0% for SFT-gold. Held-out skills, however, were trained only atomically from gold states, and supervision that teaches the student to continue from arbitrary preceding states on core skills interferes with the behavior those skills learned from clean inputs. The trade-off can be controlled, with fix-only splices when held-out accuracy matters and both splices when depth does, but no configuration we tested removes it.

The gold anchor. The anchor term of Equation 2 was introduced because training collapsed without it. In the 1.5B pilot, every arm that trained on its own sampled outputs declined monotonically, and STaR fell to 2% atomic accuracy. Probes with a zero learning rate and with full fp32 ruled out generation, measurement, and numerics (Appendix I), leaving the training loop itself. Cross-entropy toward self-sampled, temperature-drawn targets raises entropy, which degrades the next round's samples, which become the next targets. Fixed gold anchors break this cycle because the target distribution no longer depends on the student. The 3B runs inherit the anchor, and the 3B STaR row of Table 2 shows the same collapse at larger scale. Appendix K summarizes these findings as a decision procedure for practitioners.

Held-out skills. Held-out pairings resist every variant, including ours, for a structural reason. The on-policy training distribution contains no compositions involving those skills, so no rollout fails there and no correction is generated there. The gate that makes the method cheap, correcting only where the student fails, is also what excludes the cells the student never visits. The 32B teacher solves these pairings in context at 95.0% and 91.7%, so the knowledge is available on the supervision side, yet its distilled student reaches only 46.6% pooled, half of its accuracy on unseen pairings of core skills, where corrections do occur. Closing this gap is therefore a curriculum problem, because the student must attempt those pairings before any corrector can help.

7 Related Work

Distillation, off-policy and on. Knowledge distillation (Hinton et al., 2015; Ba & Caruana, 2014), its sequence-level extension (Kim & Rush, 2016), trace-level imitation of stronger models (West et al., 2022; Hsieh et al., 2023; Mukherjee et al., 2023), and instruction self-generation (Wang et al., 2023b) supply the machinery of our off-policy baselines. On-policy distillation instead trains a student on its own samples under dense teacher scoring, from reverse-KL formulations (Gu et al., 2024), generalized on-policy objectives (Agarwal et al., 2024), and skew divergences (Ko et al., 2024; 2025) to speculative interleaving (Xu et al., 2025), and concurrent work intervenes inside the rollout itself, relaying failed reasoning prefixes to a brief teacher takeover (Xu et al., 2026) or routing supervision through structured protocols when the teacher is closed (Liu et al., 2026). Our corrections share the intervene-at-failure instinct of that line. Here a programmatic verifier decides where to intervene, so no teacher pass is spent on detection, the correction is spliced onto a completed rollout, and what gets repaired is the seam between internalized skills. We accordingly measure composition beyond the training horizon, where that work measures answer accuracy on a fixed task distribution, and we find that teacher quality decides whether distillation helps at all.

Learning on the learner's own states. The correction scheme is a DAgger-style interactive imitation (Ross et al., 2011), in a lineage running from behavior cloning's earliest drift problems (Pomerleau, 1989) through search-based structured prediction (Daumé III et al., 2009), here adapted to token-level splices inside a structured trace, with exposure bias (Ranzato et al., 2016; Bengio et al., 2015) as the reason off-policy data cannot substitute. STaR-style self-training (Zelikman et al., 2022; Yuan et al., 2023) is the boundary case of on-policy data with no external correction. Verifier-guided training more broadly ranges from outcome and process reward models (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024; Hosseini et al., 2024) to models trained to revise their own output (Welleck et al., 2023; Kumar et al., 2025; An et al., 2023); our recover splices share the revision instinct but supervise the continuation after a mistake instead of its retraction.

Skills and their composition. Context distillation internalizes prompts into weights (Askell et al., 2021; Snell et al., 2022), skill libraries keep them external (Anthropic, 2025; Wang et al., 2024), and tool-using agents acquire skills through prompting and demonstrations (Yao et al., 2023; Schick et al., 2023). Skill-Mix evaluates in-context skill composition with model graders (Yu et al., 2024a); our testbed instead studies internalized skills under mechanical grading, in the lineage of SCAN, COGS, and Dziri et al. (2023). The held-out interference we report connects to continual-learning work on protecting prior abilities during new training (Li & Hoiem, 2018; Wang et al., 2023a), and weight-space merging (Yadav et al., 2023; Yu et al., 2024b) offers a training-free way to combine skills, though nothing in it targets the seams. Appendix L widens this survey.

8 Limitations and Conclusion

Our experiments use synthetic skills with exactly measurable seams, one model family, and one trace format. Within this setting the results are consistent. An internalized skill library composes only as well as its seams, off-policy imitation does not supervise seams it never visits, and a small amount of on-policy correction, applied at the seams by a corrector stronger than the student, raises a 3B student's accuracy on deep chains to that of a 32B model reading the documentation, with no added cost at inference time.

Ethics statement

This work uses only synthetic word-sequence tasks and publicly released model checkpoints; no human subjects, personal data, or sensitive content are involved, and we see no ethical concerns beyond those generic to language-model training research.

Reproducibility statement

All evaluation results in this paper derive from per-task records that accompany the submission as supplementary material, together with the full source code: the testbed generator, the skill library with its documentation files and reference implementations, every training and evaluation script, and the programs that regenerate each table and figure directly from the records. Hyperparameters and the training protocol appear in Appendix D, the evaluation protocol in Appendix J, and seed-level results in Appendix F.

AI use statement

During the experiments, we used large language models (LLMs) as an auxiliary tool for experiment monitoring and management. Specifically, the LLMs were used to monitor experiment logs and runtime information, identify potential execution issues or anomalies, and assist in reporting the status of ongoing experiments. The LLMs did not determine the research questions, experimental methodology, hyperparameter settings, or final experimental conclusions. All experimental configurations, result verification, analysis, and scientific conclusions were determined and validated by the authors. We take full responsibility for the final content and results of this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.13649.

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes LLM better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. arXiv:2207.04901.

Anthropic. Equipping agents for the real world with Agent Skills. `https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills`, 2025. Anthropic Engineering Blog. Accessed 2026-07-20.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, 2014. arXiv:1312.6184.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. arXiv:1506.03099.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, 2009.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine Learning*, 2009. arXiv:0907.0786.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. 2025. arXiv:2501.12948.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jerry A. Fodor and Zenon W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born-again neural networks. In *International Conference on Machine Learning*, 2018. arXiv:1805.04770.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.08543.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2402.06457.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019. arXiv:1902.00751.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! Outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023. arXiv:2305.02301.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020. arXiv:1908.08351.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. arXiv:2212.04089.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:1912.09713.

Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. arXiv:1606.07947.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.03898.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2503.07067.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. arXiv:2409.12917.

Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. Conference version in ECCV 2016. arXiv:1606.09282.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

Junlin Liu, Jiangwang Chen, Zixin Song, Shuaiyu Zhou, Chunji Lv, Hank Wu, Kailin Jiang, Jinyang Wu, Bohan Yu, and Chenxi Zhou. From proprietary to open-source: Bridging the distribution gap via multi-agent protocol distillation in agentic search. *arXiv preprint arXiv:2607.24280*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. arXiv:2303.17651.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.02155.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 1, 1989.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023. arXiv:2210.03350.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. arXiv:2307.16789.

Qwen. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2016. arXiv:1511.06732.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. arXiv:1011.0686.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2302.04761.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. arXiv:2303.11366.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. arXiv:2312.06585.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. *arXiv preprint arXiv:2209.15189*, 2022.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. arXiv:2305.16291.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023a. arXiv:2310.14152.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2201.11903.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.00053.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4602–4625, 2022. arXiv:2110.07178.

Sam Wiseman and Alexander M. Rush. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016. arXiv:1606.02960.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: Averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2203.05482.

Haolei Xu, Xiaowen Xu, Haiwen Hong, Zixuan Ni, Hongxing Li, Yiwen Qiu, Weiming Lu, and Yongliang Shen. Pass the baton: Trajectory-relayed on-policy distillation. *arXiv preprint arXiv:2607.26057*, 2026.

Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.11325.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

Dingli Yu, Simran Kaur, Arushi Gupta, Jonah Brown-Cohen, Anirudh Goyal, and Sanjeev Arora. Skill-Mix: a flexible and expandable family of evaluations for AI models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024a. arXiv:2310.17567.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are Super Mario: Absorbing abilities from homologous models as a free lunch. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024b. arXiv:2311.03099.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2203.14465.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2205.10625.

A NOTATION

Table 5 collects the symbols of Sections 2 and 3.

Symbol	Meaning
σ, $\sigma_{1..k}$	a skill; the chain of a depth-k task
u	input word sequence of a task
s_i	gold state after step i, with $s_0 = u$ and $s_i = \sigma_i(s_{i-1})$
o_i	sequence the model writes at step i, with $o_0 = u$
f	first failed seam, the first i with $o_i \neq \sigma_i(o_{i-1})$
x	student prompt, which names the skills without documentation
$\hat{y}$, $\hat{y}_{\leq i}$	tokens of a rollout; its prefix through the result line of step i
$\textsc{Gold}(\sigma, s, j)$	reference continuation applying $\sigma_j, \ldots, \sigma_k$ to the sequence s
y^{fix}, y^{rec}	fix and recover splices of Equation 1
c_y	position of the first spliced token of y
w_t, γ	loss weight of token t; its value on seam tokens, γ=3
$B_{\mathcal{C}}$, $B_{\mathcal{G}}$	splices and gold anchors in a minibatch B
Z_B	summed weight of all trained tokens in B
π_θ, π_T	student and teacher next-token distributions

Table 5: **Notation** used in the main text.

B THE SKILL LIBRARY

Table 6 lists all 27 skills with a one-line summary of each transformation. Every skill ships as a documentation file containing the rules, a worked bookkeeping protocol, and two worked examples; the reference implementation that grades all evaluations is the same code that generates gold traces. Inputs are sequences of three to six tokens drawn from a fixed pool of 77 lowercase words, ten of which, such as `k9` and `opal5`, carry digits for the digit-sensitive skills. Chain construction verifies that every intermediate state remains in the domain of the next skill, so every task in every cell has a well-defined gold trajectory. Table 7 adds the per-skill atomic picture after training. Core skills, the ones every deep chain is built from, are solved without error by SFT-gold and, apart from `hollow-mark` at 87.5%, by ours; the held-out block is where atomic mastery is thinner, which bounds every held-out composition cell from above before any seam enters the story.

A complete documentation file. Every documentation file follows the same template, with a one-line summary, numbered rules, a work protocol that prescribes the bookkeeping line, and two worked examples in the trace format. The file below belongs to `quartz-anchor`, the skill at which the student of the worked example in Appendix E fails.

Documentation file of `quartz-anchor`

```
## SKILL: quartz-anchor

Move the longest word to the front (earliest wins ties).

Rules:
1. Find the maximum word length.
2. The EARLIEST word with that length moves to the front.
3. All other words keep their relative order.
Work protocol: Write each word with its length, pick the earliest longest, then move
    it to the front.
Example 1: input `ox stone elk petal` | work: ox(2) stone(5) elk(3) petal(5) | max=5
    earliest=stone | move to front | output `stone ox elk petal`
Example 2: input `mud fig log` | work: mud(3) fig(3) log(3) | max=3 earliest=mud |
    move to front | output `mud fig log`
```

Prompts. Trained models and the untrained base row receive the student prompt, which names the skills in order but contains no documentation. Teachers and the documentation-reading reference rows receive the teacher prompt, which adds the documentation file of every skill in the chain. Both prompts below are for the task of the worked example in Appendix E.

Skill	Transformation
amber-pairs	Swap adjacent pairs of words; with an odd word count, the last word stays in place.
briar-tag	Prefix v- to every word that starts with a vowel.
cedar-crest	Append one word at the end: the first letters of all words joined together.
drift-left	Rotate left by one: the first word moves to the end.
drift-seed	Insert ash as the new first word and oak as the new last word.
echo-tail	Append a copy of the last word.
ember-line	Sort the words alphabetically (a to z).
fern-cut	Delete the last word.
frost-split	Split every hyphenated word into its parts (in order).
gale-stitch	Join adjacent pairs with a hyphen; an odd trailing word stays alone.
harbor-right	Rotate right by one: the last word moves to the front.
hollow-mark	Replace the words at positions 3, 6, 9, ... with the word gap.
iron-echo	Duplicate every word at an odd position (1st, 3rd, ...), copy placed right after it.
lunar-swap	Swap the first word and the last word.
mirror-run	Reverse the order of the words.
moss-cull	Keep the words at odd positions (1st, 3rd, 5th, ...); delete the rest.
nimbus-veil	Replace the middle word with veil; for an even count the middle is position n/2.
opal-thread	Insert the word sep between every two adjacent words.
pearl-count	Append one word at the end: the letter n followed by the word count.
quartz-anchor	Move the longest word to the front (earliest wins ties).
rune-prune	Delete the longest word (earliest wins ties).
sable-drop	Delete the first word.
star-tally	Append even or odd according to the word count.
tide-halves	Move the back half of the sequence to the front.
tundra-two	Swap the 2nd word with the last word (needs at least 3 words).
vapor-knot	Move every word that contains a digit to the end (order kept).
willow-gate	Delete every word that contains a digit.

Table 6: **The skill library.** Names are codenames; behavior is defined entirely by the documentation file and the matching reference implementation.

Student prompt (no documentation)

```
[system] You are a precise skill-execution engine. You have internalized a library of
    named text-transformation skills that operate on sequences of words. Execute the
    requested skills exactly, in order, in the requested format. Output nothing except
    the required lines.

[user] Task: apply 3 skills to the input word sequence, in order.
Skills: 1. harbor-right 2. quartz-anchor 3. opal-thread
Input: ink atlas ore

For EACH step, first write one 'work:' line where you carefully carry out the skill's
    rules on the current sequence, then write the step's result line. Use exactly this
    format (each result is a space-separated word sequence after the colon):
work: <apply the rules>
Step 1 [harbor-right]: <result after this skill>
work: <apply the rules>
Step 2 [quartz-anchor]: <result after this skill>
work: <apply the rules>
Step 3 [opal-thread]: <result after this skill>
FINAL: <the final result>
The FINAL line must repeat the result of the last step exactly.
```

Teacher prompt (documentation in context)

```
[system] You are a precise skill-execution engine. Documentation for the required
    skills is provided. Follow the documentation exactly and execute the requested
    skills, in order, in the requested format. Output nothing except the required
    lines.

[user] Skill documentation:

## SKILL: harbor-right

Rotate right by one: the last word moves to the front.
```

Skill	Split	Tasks	SFT-gold	Ours
drift-seed	core	11	100.0	100.0
echo-tail	core	5	100.0	100.0
ember-line	core	5	100.0	100.0
fern-cut	core	5	100.0	100.0
frost-split	core	5	100.0	100.0
gale-stitch	core	5	100.0	100.0
harbor-right	core	6	100.0	100.0
hollow-mark	core	8	100.0	87.5
iron-echo	core	7	100.0	100.0
lunar-swap	core	13	100.0	100.0
moss-cull	core	3	100.0	100.0
nimbus-veil	core	8	100.0	100.0
opal-thread	core	10	100.0	100.0
quartz-anchor	core	11	100.0	100.0
rune-prune	core	5	100.0	100.0
sable-drop	core	10	100.0	100.0
tide-halves	core	7	100.0	100.0
vapor-knot	core	11	100.0	100.0
willow-gate	core	6	100.0	100.0
amber-pairs	held-out	3	100.0	100.0
briar-tag	held-out	6	50.0	83.3
cedar-crest	held-out	13	46.2	38.5
drift-left	held-out	5	100.0	100.0
mirror-run	held-out	5	100.0	100.0
pearl-count	held-out	10	100.0	100.0
star-tally	held-out	7	100.0	100.0
tundra-two	held-out	10	90.0	96.7

Table 7: **Per-skill atomic accuracy** (%) on the k=1 cell, with the number of tasks per skill. Ours averages three seeds, and held-out skills are shaded.

```
Rules:
1. Remove the last word and put it at the front.
2. A single word stays unchanged.
Work protocol: Index the words, name the last word, then write it followed by indices
    1..n-1.
Example 1: input `ash2 elm ivy` | work: 1:ash2 2:elm 3:ivy | last=3:ivy | ivy + ash2
    elm | output `ivy ash2 elm`
Example 2: input `net rug` | work: 1:net 2:rug | last=2:rug | rug + net | output `rug
    net`

## SKILL: quartz-anchor

Move the longest word to the front (earliest wins ties).

Rules:
1. Find the maximum word length.
2. The EARLIEST word with that length moves to the front.
3. All other words keep their relative order.
Work protocol: Write each word with its length, pick the earliest longest, then move
    it to the front.
Example 1: input `ox stone elk petal` | work: ox(2) stone(5) elk(3) petal(5) | max=5
    earliest=stone | move to front | output `stone ox elk petal`
Example 2: input `mud fig log` | work: mud(3) fig(3) log(3) | max=3 earliest=mud |
    move to front | output `mud fig log`

## SKILL: opal-thread

Insert the word `sep` between every two adjacent words.

Rules:
1. Between each pair of neighboring words, insert the word `sep`.
2. Nothing is added before the first word or after the last.
3. A single word stays unchanged.
Work protocol: Write the words joined by ` sep `.
Example 1: input `dew sun fog` | work: dew sep sun sep fog | output `dew sep sun sep
    fog`
Example 2: input `elk` | work: single word, unchanged | output `elk`
```

```
Task: apply 3 skills to the input word sequence, in order.
Skills: 1. harbor-right 2. quartz-anchor 3. opal-thread
Input: ink atlas ore

For EACH step, first write one 'work:' line where you carefully carry out the skill's
    rules on the current sequence, then write the step's result line. Use exactly this
    format (each result is a space-separated word sequence after the colon):
work: <apply the rules>
Step 1 [harbor-right]: <result after this skill>
work: <apply the rules>
Step 2 [quartz-anchor]: <result after this skill>
work: <apply the rules>
Step 3 [opal-thread]: <result after this skill>
FINAL: <the final result>
The FINAL line must repeat the result of the last step exactly.
```

C Format Calibration

The output format gates the whole study. In early calibration with a direct-answer format, a 7B documentation-reading model solved 23.4% of atomic screening tasks. Allowing free-form reasoning before the answer raised this to 55.0%. Requiring the documented bookkeeping protocol, index the words, then apply the rules on the indexed sequence, raised it to 74.5% on the final library, and the same protocol brings the 14B model to 95.8% and the 32B model to 98.6% there, with no 32B skill below 0.875. Small models do not spontaneously externalize the intermediate computation these transformations need; told exactly how to write it down, they can. The library was also simplified during calibration from 32 candidate skills to the final 27, removing character-level operations that no in-context reader could execute reliably, and each screening mean above uses its own version of the library.

D Training Details

All arms initialize from the same SFT-gold adapter: rank-32 LoRA (α=64, all attention and MLP projections), three epochs over 3,648 gold traces of the training distribution, learning rate 10^{-4}, effective batch 64. Seam-corrective training then runs 40 rounds. Each round samples 192 tasks from the training distribution (depth mix 10/30/60% for k=1/2/3), generates one rollout per task at temperature 0.7 with a 560-token cap, grades all rollouts, builds fix and recover splices for failed rollouts, and takes gradient steps with effective batch 16 at learning rate 2×10^{-5} with 16 warmup updates, gradient clipping at 1.0, and fp16 compute with fp32 LoRA master weights. Seam tokens are the tokens of every result line from step two on and of the final line, up to the first 40 characters after the colon, which covers the whole line in almost every task. They receive weight γ=3, and each completion's weights are rescaled to mean one before the student prefix is masked, so the normalizer Z_B of Equation 2 counts these rescaled weights plus one unit per anchor token. The anti-seam control places the same γ on tokens from the middle thirds of work and result lines, and the uniform control sets every weight to one. Solved tasks contribute one gold-trace anchor each at weight 0.3. The 32B-teacher variant builds no splices. On every conditionally failed step of a failed rollout it applies the per-token skew reverse KL $\mathrm{KL}\big(\pi_\theta \,\|\, 0.9\,\pi_T + 0.1\,\pi_\theta\big)$ to the student's own tokens, where π_T is the teacher reading the documentation and sharing the Qwen2.5 tokenizer; the seam weights and the anchor are as above. The oracle run takes roughly 90 minutes on one V100; the 32B variant shards its teacher across three additional V100s and scores 537K teacher tokens. Table 8 collects these settings.

E Splice Construction

Algorithm 1 in Section 3 gives the exact construction that turns one failed rollout into training material, matching the released code. A correction is skipped when the student's intermediate state leaves the domain of a later skill, which the reference implementation reports by raising an error; on the released evaluation rollouts of the SFT-gold student, none of 504 candidate splices triggers it. The worked example below takes one real failed rollout of that student on an unseen depth-three pairing and shows both splices the reference implementation builds from it.

Stage	Setting	Value
SFT-gold	adapter	LoRA, rank 32, α=64, all attention and MLP projections
	data and schedule	3,648 gold traces, 3 epochs, learning rate 10^{-4}
	batch	effective 64 (micro-batch 4, 16 accumulation steps)
Seam-corrective	rounds and rollouts	40 rounds of 192 rollouts
	depth mix, k=1/2/3	10%, 30%, 60%
	sampling	temperature 0.7, at most 560 new tokens
	optimization	learning rate 2×10^{-5}, 16 warmup updates, clipping 1.0
	batch	effective 16 (micro-batch 2, 8 accumulation steps)
	loss	seam weight γ=3, anchor weight 0.3
	precision	fp16 compute, fp32 LoRA master weights
Distilled variant	teacher	Qwen2.5-32B-Instruct with documentation, three V100s
	divergence	$\mathrm{KL}(\pi_\theta \,\|\, 0.9\,\pi_T + 0.1\,\pi_\theta)$ on failed steps
Evaluation	decoding	greedy, at most 768 new tokens, 200 tasks per cell

Table 8: **Hyperparameters** of every trained arm.

In the rollout below, the student writes the bookkeeping of `vapor-knot` under the `quartz-anchor` header at step 2 and leaves the words unchanged; step 3 fails as well.

Student rollout (SFT-gold, first failed seam f=2)

```
work: 1:ink 2:atlas 3:ore | last=3:ore | ore + ink atlas
Step 1 [harbor-right]: ore ink atlas
work: ore:n ink:n atlas:n | front ore ink atlas | end (none)
Step 2 [quartz-anchor]: ore ink atlas
work: ore:n ink:n atlas:n | join ore+ink+atlas
Step 3 [opal-thread]: ore ink atlas
FINAL: ore ink atlas
```

The fix splice keeps the student's own tokens through step 1 and appends the reference continuation from the last on-track state. Loss applies only below the marker.

Fix splice

```
work: 1:ink 2:atlas 3:ore | last=3:ore | ore + ink atlas
Step 1 [harbor-right]: ore ink atlas
--- loss starts here ---
work: ore(3) ink(3) atlas(5) | max=5 earliest=atlas | move to front
Step 2 [quartz-anchor]: atlas ore ink
work: atlas sep ore sep ink
Step 3 [opal-thread]: atlas sep ore sep ink
FINAL: atlas sep ore sep ink
```

The recover splice keeps the wrong step 2 and continues correctly from its output.

Recover splice

```
work: 1:ink 2:atlas 3:ore | last=3:ore | ore + ink atlas
Step 1 [harbor-right]: ore ink atlas
work: ore:n ink:n atlas:n | front ore ink atlas | end (none)
Step 2 [quartz-anchor]: ore ink atlas
--- loss starts here ---
work: ore sep ink sep atlas
Step 3 [opal-thread]: ore sep ink sep atlas
FINAL: ore sep ink sep atlas
```

F Full 3B Results

Table 9 reports all nine cells for every trained arm, with every seed of the multi-seed methods on its own row, and Table 10 gives the pooled families per seed. The main paper's multi-seed rows are means over the seeds shown here, and Table 11 extends the main paper's significance table to every paired contrast run in this study, including each seed against the baseline and the 14B control family.

Arm	$k1$	$k2$s	$k3$s	$k2$u	$k3$u	$k2$h	$k3$h	$k4$	$k5$
SFT on gold traces	94.5	97.5	96.5	96.5	90.0	59.0	47.0	83.5	64.0
STaR	58.0	57.5	40.0	50.0	37.0	16.5	15.0	35.5	20.5
Seam-corrective, 14B	93.0	95.5	89.5	92.5	89.5	55.0	39.5	82.5	46.5
no seam weight	92.0	94.5	82.0	90.5	86.5	52.5	35.5	77.0	60.0
anti-seam	92.5	97.5	91.0	94.0	90.0	49.0	35.5	83.0	60.5
frozen rollouts	91.0	95.0	89.5	92.5	86.5	45.0	27.0	78.0	47.5
Seam-corrective, 32B (s0)	92.5	97.0	92.5	94.5	92.0	56.0	41.5	88.5	73.5
Seam-corrective, 32B (s1)	94.0	98.0	94.5	97.0	93.0	52.5	36.5	89.0	71.5
Oracle fix+recover (s0)	96.0	98.0	97.5	97.0	94.5	53.5	41.0	90.5	71.5
Oracle fix+recover (s1)	93.5	98.0	96.0	96.0	94.0	52.0	35.5	88.5	73.0
Oracle fix+recover (s2)	95.0	97.0	95.5	97.0	91.5	54.0	39.5	88.5	77.0
frozen rollouts	96.5	98.0	93.5	95.0	93.5	57.0	36.5	88.5	58.0
no seam weight	93.0	98.0	96.5	96.5	93.0	48.0	33.0	90.0	69.5
anti-seam	92.5	96.5	91.0	95.5	93.0	57.0	41.5	83.0	62.0
Oracle fix only (s0)	95.0	98.0	96.5	95.0	94.5	60.0	45.0	91.5	72.0
Oracle fix only (s1)	96.0	97.5	97.0	97.0	93.5	59.5	42.5	87.5	66.0
Oracle fix only (s2)	94.0	98.0	96.5	96.5	94.5	55.0	37.0	87.0	67.5
frozen rollouts	95.5	98.0	97.0	96.5	94.0	54.0	37.0	88.0	70.5
Oracle recover only	92.0	96.5	95.0	95.5	91.5	45.0	26.0	86.0	76.5

Table 9: **Full 3B results.** Task success (%) for every trained arm and seed, 200 tasks per cell; s, u, and h mark seen, unseen, and held-out pairs.

Method	Seed	In-dist.	Unseen	Held-out	Depth
Oracle fix+recover	0	97.2	95.8	47.2	81.0
	1	95.8	95.0	43.8	80.8
	2	95.8	94.2	46.8	82.8
Seam-corrective, 32B	0	94.0	93.2	48.8	81.0
	1	95.5	95.0	44.5	80.2
Oracle fix only	0	96.5	94.8	52.5	81.8
	1	96.8	95.2	51.0	76.8
	2	96.2	95.5	46.0	77.2

Table 10: **Per-seed pooled results** for the multi-seed methods.

G Step-Level Accuracy and a Side-by-Side Example

Table 12 extends Figures 6 and 9 to every trained arm, reporting the conditional accuracy of each step of k=4 and k=5 chains, averaged over seeds for the multi-seed methods. Steps within the training horizon stay close to ceiling for every arm except STaR, whose collapse is visible from the first step. The weakest step of SFT-gold at depth five, step four at 78.0%, is also where the arms differ most. Recover-only reaches 90.5%, the full method 87.2%, and the 32B-corrected arm 85.5%, while the 14B-corrected arm falls to 61.0%, the step-level counterpart of the dose-response result in Section 6. At depth four the weakest step is step three, where seam correction raises SFT-gold from 90.0% to 94.2%.

A side-by-side example. The two traces below answer the same depth-five task. Both models execute the first three steps identically. At step four, `ember-line`, which sorts the words alphabetically, SFT-gold misorders the sequence it receives. Its step five is conditionally correct, applying `hollow-mark` faithfully to the wrong state, so the chain fails at a single seam. The corrected model sorts the same state correctly and completes the chain.

SFT-gold (wrong from step 4)

```
work: 1:ant 2:pine 3:toad 4:vole | n=4 back=2 words (indices 3..4) | toad vole + ant
    pine
Step 1 [tide-halves]: toad vole ant pine
work: toad:n vole:n ant:n pine:n | front toad vole ant pine | end (none)
Step 2 [vapor-knot]: toad vole ant pine
```

Contrast	$k1$	$k2$s	$k3$s	$k2$u	$k3$u	$k2$h	$k3$h	$k4$	$k5$
Oracle vs. SFT	.167	.367	.223	.427	.017	.994	.976	.003	.006
Oracle s1 vs. SFT	.849	.372	.707	.689	.026	.998	1.000	.033	.005
Oracle s2 vs. SFT	.421	.757	.821	.416	.293	.980	.997	.038	<.001
32B vs. SFT	.977	.751	.996	.926	.172	.898	.979	.031	.002
32B s1 vs. SFT	.687	.372	.949	.390	.097	.992	1.000	.003	.006
Oracle vs. 32B	.019	.225	.006	.056	.102	.828	.594	.226	.760
Oracle vs. frozen	.755	.658	.010	.101	.310	.961	.075	.231	<.001
Frozen vs. SFT	.100	.393	.986	.857	.055	.827	1.000	.009	.962
Oracle vs. anti-seam	.008	.125	<.001	.198	.243	.960	.599	.006	.006
Oracle vs. uniform	.037	.648	.275	.412	.199	.023	.004	.460	.286
Oracle vs. fix-only	.271	.650	.271	.065	.561	.998	.938	.733	.597
Oracle vs. recover-only	.005	.048	.037	.216	.069	<.001	<.001	.063	.944
Fix-only vs. SFT	.415	.396	.571	.859	.009	.386	.797	<.001	.008
Fix-only s1 vs. SFT	.126	.661	.403	.407	.052	.456	.954	.034	.275
Fix-only s2 vs. SFT	.721	.400	.573	.574	.022	.894	.999	.095	.184
Fix-only vs. its frozen	.755	1.000	.817	.953	.455	.013	.006	.065	.342
Recover-only vs. SFT	.973	1.000	.947	.819	.235	1.000	1.000	.190	<.001
14B vs. SFT	.884	.953	1.000	.994	.606	.934	.997	.688	1.000
14B vs. its frozen	.175	.442	.542	.566	.152	<.001	<.001	.079	.634
14B vs. its anti-seam	.430	.980	.887	.855	.612	.036	.111	.600	1.000
14B vs. its uniform	.344	.314	.001	.193	.140	.253	.147	.033	1.000

Table 11: **Complete per-cell significance map.** Paired-bootstrap probability that the second arm matches or beats the first, from 10,000 resamples; s, u, and h mark seen, unseen, and held-out pairs.

	k=4 chains				k=5 chains				
Arm	1	2	3	4	1	2	3	4	5
SFT on gold traces	99.0	96.0	90.0	96.0	98.0	96.5	87.5	78.0	88.0
STaR	53.0	60.0	63.0	59.0	47.5	50.0	49.5	41.5	44.5
Seam-corrective, 14B	99.5	96.0	89.0	95.0	98.5	94.5	82.5	61.0	79.0
Oracle, frozen rollouts	99.0	97.5	90.0	95.5	99.0	93.0	86.0	70.5	85.5
Oracle, uniform weights	100.0	97.5	96.0	96.0	98.5	96.5	90.0	80.0	89.5
Oracle, anti-seam weights	99.5	96.0	94.0	91.5	98.0	92.5	88.0	86.0	82.5
Oracle, fix only	99.5	98.2	93.5	96.5	98.7	96.0	89.3	81.7	89.7
Oracle, recover only	99.5	96.0	91.0	96.5	98.5	94.0	91.0	90.5	87.0
Seam-corrective, 32B	99.8	97.5	93.5	96.5	98.8	97.2	90.8	85.5	90.5
Seam-corrective, oracle (ours)	99.5	97.2	94.2	96.8	98.2	96.3	91.0	87.2	90.5

Table 12: **Conditional step accuracy** (%) of every trained arm, as a seed mean for the multi-seed methods, at each step of k=4 and k=5 chains.

```
work: 1:toad 2:vole 3:ant 4:pine | last=4:pine | pine + toad vole ant
Step 3 [harbor-right]: pine toad vole ant
work: picked in order: ant , toad , pine , vole
Step 4 [ember-line]: ant toad pine vole
work: 1:ant 2:toad 3:pine 4:vole | multiples of 3: 3 -> ant toad gap vole
Step 5 [hollow-mark]: ant toad gap vole
FINAL: ant toad gap vole
```

Seam-corrective, oracle, seed 0 (correct)

```
work: 1:ant 2:pine 3:toad 4:vole | n=4 back=2 words (indices 3..4) | toad vole + ant
    pine
Step 1 [tide-halves]: toad vole ant pine
work: toad:n vole:n ant:n pine:n | front toad vole ant pine | end (none)
Step 2 [vapor-knot]: toad vole ant pine
work: 1:toad 2:vole 3:ant 4:pine | last=4:pine | pine + toad vole ant
Step 3 [harbor-right]: pine toad vole ant
work: picked in order: ant , pine , toad , vole
Step 4 [ember-line]: ant pine toad vole
```

```
work: 1:ant 2:pine 3:toad 4:vole | multiples of 3: 3 -> ant pine gap vole
Step 5 [hollow-mark]: ant pine gap vole
FINAL: ant pine gap vole
```

H Failure Anatomy Across Depths

Figure 7 gives the appendix view of the failure anatomy discussed in the main paper. Among depth-five chains that slip, first errors concentrate at steps three and four for both models, and seam correction removes errors only there, from 54 to 36 on the seed mean, while steps one, two, and five keep 11 and 10 first errors under both models. The depth-four picture in Figures 8 and 9 repeats the pattern one seam earlier. First errors pile up at step three, and the conditional-accuracy lift concentrates at that weakest handoff, from 90.0% to 94.2%, so the k=5 story is not special to that depth.

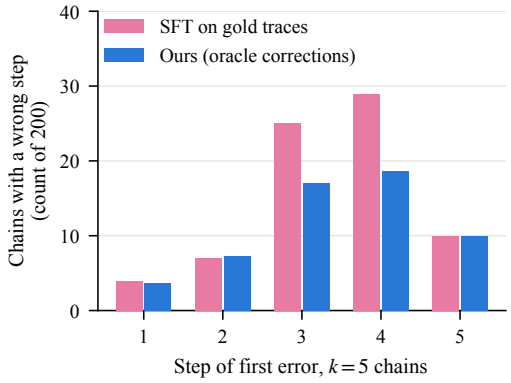

Figure 7: **First-error positions** in k=5 chains with a wrong step; ours is a three-seed mean.

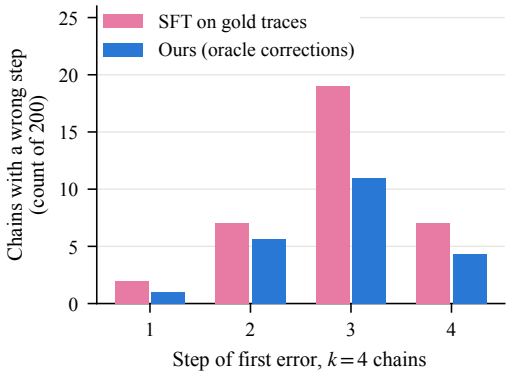

Figure 8: **First-error positions** in k=4 chains with a wrong step; ours is a three-seed mean.

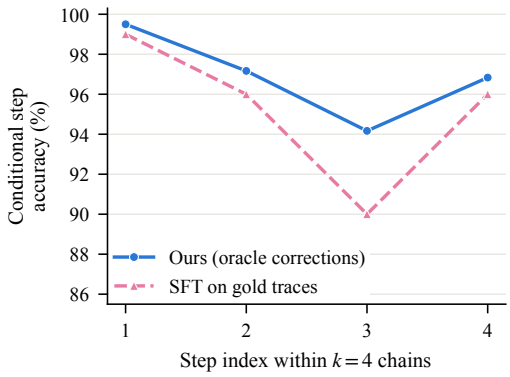

Figure 9: **Conditional step accuracy within k=4 chains**, the depth-four companion to the main paper's Figure 6.

I The 1.5B Pilot: Collapse, Dose, and Diagnosis

The final method grew out of a 1.5B pilot, whose full results appear in Table 13, and three episodes from it shaped the design.

Self-training collapse. The main paper recounts the collapse mechanism; here are the probe settings behind it. Probe one set the learning rate to zero and ran eight full rounds of the otherwise identical loop, generation at temperature 0.7 and grading included, and rollout accuracy held flat, clearing the generation and measurement path. Probe two switched the student to full fp32 with the

Arm	$k1$	$k2$s	$k3$s	$k2$u	$k3$u	$k2$h	$k3$h	$k4$	$k5$
SFT on gold traces	93.5	96.0	91.5	92.0	86.0	50.0	29.0	79.5	54.5
STaR	2.0	1.0	0.5	1.5	0.0	0.0	0.0	0.0	0.5
Corrective KL, 14B (light)	92.0	96.5	92.5	94.0	90.5	45.0	30.0	82.0	54.5
no seam weight	94.0	97.5	87.5	92.0	87.0	43.0	22.0	75.5	52.0
anti-seam	87.0	95.0	87.5	92.0	84.5	33.0	18.5	72.5	54.0
frozen rollouts	93.0	96.0	91.0	92.5	88.5	39.0	19.5	79.0	46.5
Corrective KL, 14B ($4\times$ dose)	91.5	92.0	81.0	88.0	79.5	43.0	21.0	69.0	45.5
conservative dose variant	90.0	91.5	82.5	88.5	83.5	41.0	26.0	73.5	46.0

Table 13: **The 1.5B pilot** on the nine cells of the main paper, 200 tasks per cell. Escalated-dose rows keep every failure from four rollouts per task.

gradient scaler disabled, and the decay reproduced exactly, clearing numerics. STaR's endpoint under the uncorrected loop is the 2.0% atomic row of Table 13; every arm trained after the gold-anchor repair holds stable through all 40 rounds.

Light corrective KL from a 14B teacher. With the anchor in place, corrective KL from a 14B teacher at temperature 0.7 landed in a statistical tie with SFT-gold on most cells, with one pooled exception: unseen pairings improved from 89.0% to 92.2% (p=0.013, paired bootstrap over 400 tasks). In-distribution cells were unchanged (93.7% both), and depth moved by 1.3 points (p=0.31). This result motivated the 3B experiments.

Controls at 1.5B. The three controls that localize the 3B depth gain already separated from the light arm at 1.5B (Table 13). On the pooled out-of-distribution cells the light arm scored 66.0%, against 60.8% with frozen rollouts, 59.1% with anti-seam weights, and 61.9% with uniform weights, each paired contrast at $p<0.001$. All three also fell below SFT-gold at 65.2%, so at this scale the controls show how much misplaced correction hurts, and the light arm's advantage over them motivated the final design.

Dose escalation. Attributing the tie to scarce corrections, we sampled four rollouts per task and kept every failure, quadrupling corrective mass (Table 13, last two rows). Pooled out-of-distribution accuracy fell 7.5 points below baseline ($p \approx 1.0$ that the escalated arm fails to beat baseline), in-distribution fell 5.5, and a conservative variant of the escalation showed the same signs. The correct reading, confirmed at 3B in the main paper, is that corrective KL toward a teacher weaker than the student is harmful in proportion to its mass. Sixty-task probes of mid-run checkpoints also drifted into false alarms in both directions, so every result in this paper comes from the final full evaluation.

J Evaluation Protocol

Every evaluation of a trained arm is zero-context and greedy: the prompt contains the task instruction, the skill names, and the input sequence, never documentation. The untrained reference rows are the deliberate exception, measuring the effect of documentation in context without training. Cells hold 200 tasks except the untrained reference rows, which use the first 60 tasks of every cell; the two base-student rows were decoded greedily in bf16 with MLX on Apple silicon, on the same prompts. The same task lists are shared across all arms, which is what makes the paired bootstrap paired. Reported significance always resamples task indices within a cell or pooled family, 10,000 draws, and compares the two arms' resampled means on identical tasks.

K Practitioner Guidance

The results assemble into a short decision procedure. If the domain has an executable reference, use oracle splices, the cheapest and safest supervision in this study. If steps can be checked but no solver exists, distill a stronger model's corrections, but gate the pairing by screening the teacher per skill and refusing it unless it beats the student where it will teach. If the library contains skills that training will not compose, decide what they are worth, since fix-only splices roughly halve their

loss at about five points of depth, while adding recover splices gains about eight points at a larger held-out cost. Whatever the configuration, keep a gold anchor in the loss, sample fresh rollouts every round, and spend the weighting budget at the seams; each of these three choices accounted for several points in our experiments.

Two follow-up directions are immediate. Closing the held-out gap needs a curriculum that lets training compositions touch every skill while preserving an untouched split for evaluation, which the task generator already supports. The held-out cost should also be measured in agentic tool chains, where the skills are real tools and the seams are tool boundaries, so that an eroded skill corresponds to a broken workflow.

L Extended Related Work

This appendix extends the discussion of Section 7.

Distillation, from compression to behavior transfer. Knowledge distillation began as model compression by matching soft targets (Ba & Caruana, 2014; Hinton et al., 2015), was later applied to pretraining compact students (Sanh et al., 2019), and was shown to let students of the same size as their teachers outperform them (Furlanello et al., 2018). For large language models the transferred object shifted from logits to behavior, with commonsense corpora (West et al., 2022), rationales (Hsieh et al., 2023), and explanation traces (Mukherjee et al., 2023) distilled as text. On-policy distillation (Gu et al., 2024; Agarwal et al., 2024; Ko et al., 2024; 2025; Xu et al., 2025) moved supervision onto the student's own samples, and this paper extends that line to composition seams.

Learning on the learner's own states. Imitation learning has long recognized that a learner must be supervised on the states it visits itself. ALVINN's synthetic recovery frames (Pomerleau, 1989), SEARN's learned roll-ins (Daumé III et al., 2009), DAgger's expert relabeling (Ross et al., 2011), and sequence training that accounts for search or exposure bias (Wiseman & Rush, 2016; Ranzato et al., 2016; Bengio et al., 2015) all address drift by supervising visited states. Our splices apply the same principle inside a single structured generation, with a verifier selecting the states.

Self-improvement and its verifiers. Between imitation and reinforcement learning lies a family of methods that filter or edit a model's own outputs before training on them. It includes STaR and rejection-sampling fine-tuning (Zelikman et al., 2022; Yuan et al., 2023), iterated offline variants (Gulcehre et al., 2023; Singh et al., 2024), verifier and process-reward training (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024; Hosseini et al., 2024), and explicit self-correction (Welleck et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Kumar et al., 2025; An et al., 2023). Our results add two observations to this literature. Filtering alone collapses in our setting, and correction helps only when the corrector outperforms the student, which few of these pipelines verify.

Compositional generalization. Whether trained systems recombine what they know is a classical question (Fodor & Pylyshyn, 1988). SCAN, COGS, and CFQ made it measurable for neural sequence models (Lake & Baroni, 2018; Kim & Linzen, 2020; Keysers et al., 2020), within the taxonomy of Hupkes et al. (2020), and it reappears in large language models as compositionality gaps (Press et al., 2023), limits on length generalization (Anil et al., 2022), and the compounding of errors analyzed by Dziri et al. (2023). Decomposition prompting (Wei et al., 2022; Zhou et al., 2023) addresses these limits in context, whereas our method addresses them in the weights.

Skills as artifacts and as parameters. Skill libraries treat a capability as a document or tool to be loaded (Anthropic, 2025; Wang et al., 2024; Yao et al., 2023; Schick et al., 2023; Qin et al., 2024), while post-training treats it as a parameter update (Ouyang et al., 2022; Houlsby et al., 2019; Hu et al., 2022), and merging methods combine such updates after training (Wortsman et al., 2022; Ilharco et al., 2023; Yadav et al., 2023; Yu et al., 2024b). Context distillation connects the two views (Askell et al., 2021; Snell et al., 2022), curriculum learning orders what is learned when (Bengio et al., 2009), and reasoning-oriented post-training (DeepSeek-AI, 2025) shows how far on-policy signals can scale. Our contribution is a seam-level account of what internalization does to the composition of skills.