

Know2Act: What Knowledge Reaches a Tool-Using Agent’s Actions?

Yiwen Lu

University of Notre Dame
ylu37@nd.edu

Abstract

Recent work trains language agents to predict not only their own actions but also the observations that the environment returns, and reports that this makes later reinforcement learning (RL) more effective because the agent learns the consequences of its actions. We ask whether knowledge learned in this way reaches the agent’s decisions. In Know2Act, a pre-registered study of a multi-turn text-to-SQL agent trained with supervised fine-tuning and then GRPO, we measure what each model learns and how it acts. Observation supervision is learned: on unseen databases the agent predicts the format and headers of result tables far better. Yet it changes the agent’s actions less than reshuffling the training data does, and it brings no gain after RL. We then vary what the agent predicts and where, up to a contract that states the shape of the correct answer and is checked against every query result. Every prediction is learned, but none changes when the agent submits an answer or keeps querying. The knowledge is usable from outside: selecting among sampled answers with the agent’s own contract improves accuracy. RL, which rewards only the final answer, removes even this gain and makes the agent more willing to submit after its own check has failed. Learning to predict the environment is therefore not the same as acting on the prediction. We also find that 44% of the training tasks in a public SQL RL environment reward an empty answer.

1 Introduction

The proposal we test. A tool-using agent’s trajectory interleaves decisions with their consequences. The agent issues a query, the environment returns a result, and the agent decides what to do next. Standard supervised fine-tuning (SFT) learns from one side of this exchange. The loss covers the agent’s actions, while observations stay in the context but are not prediction targets.

Zhang et al. (2026) propose removing this mask. Their method, ActObs, also supervises the observation tokens. It reports that, although the SFT checkpoints perform alike, GRPO from ActObs reaches higher pass@ k on Terminal-Bench. The authors attribute this to the policy learning what its actions do to the environment. Related work adds observation losses during RL (Shrivastava et al., 2026; Lu et al., 2026).

What has not been tested. For a predictive objective to help, what the model learns must change the actions it takes, and this link has not been measured. Two other explanations also remain open. A pooled loss over actions and observations shrinks the share of each update that goes to actions, and less action fitting could preserve plasticity for RL on its own (Liu et al., 2026a). A pre-registered study also found that content-free auxiliary losses match real observation losses (Wang, 2026), which suggests that some reported gains reflect regularisation rather than content. A third question is specific to tools that are cheap to call, such as an SQL engine. The agent can always run its query and see the result, so which knowledge about such an environment is still worth learning?

Our study. We test these questions in Know2Act (from *knowledge to action*), a study of a multi-turn text-to-SQL agent that writes SQL, runs it, reads the table or error, and answers (NovaSky-AI, 2025) (Figure 1). We use a Qwen3-4B student, GRPO, and the Spider and BIRD development sets. The study was pre-registered before any main run. It is token-faithful, in that SFT sequences match, token for token, the stream that RL trains on. It also includes controls that reproduce the observation-supervised model’s action share and action movement.

What we find. As in prior work, the SFT checkpoints are indistinguishable, but here GRPO does

not separate them either. To find out why, we measure what each model learned and whether it reached the actions. Observation supervision is learned: on unseen databases the model predicts the output template and result header far better. Its action distribution, however, differs from action-only SFT by less than two action-only models trained on reshuffled data differ from each other, and its behaviour after errors and empty results is unchanged. What it learns about row values is memorised and does not transfer to unseen databases.

We then change what the agent predicts and where. An outcome line such as “Expected result: 3 rows, 2 columns” is learned equally well as an auxiliary target and as part of the action, but only the latter moves the policy. An answer contract such as “Expected answer: 1 row, 1 column”, followed by a check after every observation, is learned on unseen databases and works as an external selector before RL. Yet no variant changes when the agent submits or keeps querying by more than retraining on a reshuffled data order does, and neither observation supervision nor the contract improves accuracy after RL.

Contributions. Our contributions are threefold. (1) We give a pre-registered, token-faithful test of observation supervision for a multi-turn SQL agent with matched controls, which finds no RL gain on the pre-registered outcome. Along the way, our audit of the public RL environment finds that 44% of its training tasks reward an empty answer. (2) We diagnose the null at the action level, calibrated against a data-order noise floor. Observation knowledge is learned, as transferable structure and memorised content, but it reaches neither the action policy nor behaviour. (3) Across five supervision targets and two placements, every target is learned and transfers to unseen databases, including the decision-relevant shape of the correct answer placed in the action stream. Yet no variant changes submit-or-revise decisions beyond retraining noise or improves accuracy after RL. Learned knowledge helps only as an external selector, and only before RL, in line with knowing–doing gaps reported for LLMs.

2 Observation Supervision, Its Alternatives, and Where Predictions Go

Every agent trajectory pairs what the agent did with what the environment returned. All the arms below train on the same trajectories in the same

order with the same hyper-parameters. They differ only in which tokens carry loss, or in a short line inserted into the text.

Objective. Let a trajectory be $\tau = (x, a_1, o_1, \dots, a_T, o_T)$, with task prompt x , agent turns a_t , and observations o_t . Here each o_t is an SQL result table or error message. Let A and O index the action and observation tokens. Following Zhang et al. (2026), we minimise

$$\mathcal{L}_\lambda = -\frac{S_A + \lambda S_O}{|A| + \lambda|O|}, \quad S_X = \sum_{i \in X} \log p_\theta(y_i | y_{<i}). \quad (1)$$

$\lambda = 0$ gives ActionSFT, and $\lambda = 1$ gives ActObs.

Two readings of a gain. If ActObs helps RL, there are two ways to explain it. On the world-model reading, predicting o_t requires modelling what a_t does, and that knowledge shapes later actions. On the dilution reading, Eq. (1) shrinks the action term’s share of each update to $|A|/(|A| + |O|)$, which is 0.69 in our data, and fitting actions less could by itself preserve plasticity for RL (Liu et al., 2026a). We pre-registered two controls that reproduce the dilution without any environment prediction. ActionSFT-LR is action-only SFT whose learning rate is chosen so that its held-out action KL to the base model matches ActObs’s; a frozen rule selected $0.85\times$. ActionSFT+Wiki replaces the observation term with the same number of unrelated Wikipedia tokens.

Where the prediction goes. Unmasking observations asks the model to predict them, but nothing at test time conditions on that prediction. Two exploratory arms separate learning a prediction from acting on it.

Both insert an outcome line derived from the real observation, such as “Expected result: 3 rows, 2 columns.” or “Expected result: error (no such column: GPA).” Outcome-Aux places the line after `</sql>` as an auxiliary target ($\lambda = 1$), leaves the real observation unsupervised, and never generates the line at test time. Outcome-in-Action places the same line before `<sql>`, inside the action, so that the query that follows is conditioned on it. The two arms supervise identical lines with identical weights.

What the prediction is about. An outcome line predicts the environment’s response to one’s own query. When running the query is free, that is information the agent will receive anyway. What

Task: "How many people live in Gelderland district?" (Spider-dev, real rollout)

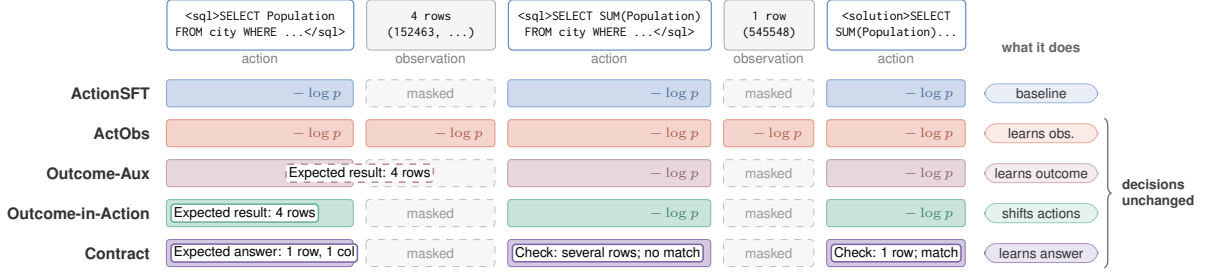


Figure 1: One real trajectory of our SQL agent (top) and where each SFT variant places supervision, in the style of Zhang et al. (2026). $-\log p$ marks tokens that carry loss; masked tokens are context only. Boxes are inserted lines: dashed means an auxiliary target that is never generated at test time, solid means part of the action. Every variant learns its target on unseen databases. Only the in-action line moves the action policy, and no variant changes when the agent submits or keeps querying (§5.3).

the executor cannot provide is a description of the correct answer.

Contract therefore begins each trajectory with an *answer contract*: the coarse shape of the gold result (no rows, one row, or several rows, plus the number of columns), e.g. “Expected answer: 1 row, 1 column.” After each observation, the next action begins with a check, e.g. “Check: got several rows, 2 columns; this does not match the expected answer.” The check depends only on text the model can see, so it is a skill to compute rather than a label.

Contract-Placebo is a placebo with the same format, using contracts taken from other questions.

Contract differs from Outcome-in-Action in three ways at once: the target, the label source (gold execution vs. hindsight), and the explicit check. We test this bundle as a whole. It also receives information that ActionSFT lacks, namely the gold result shapes of the 6,294 training questions. There is no test-time leakage, because the evaluation databases are unseen.

After SFT. All arms run the same GRPO. No prediction is rewarded, because rewarding prediction accuracy can collapse GRPO (Wang, 2026).

The outcome and contract arms were added after the first results were seen. Their predictions were written down before training (App. M).

3 Main Results

3.1 Setup

Environment. We use the text-to-SQL recipe of SkyRL (NovaSky-AI, 2025) at a pinned commit, without modification. Each turn the agent either issues `<sql> . . . </sql>`, whose result table or er-

ror message is returned as the observation, or ends with `<solution> . . . </solution>`. The reward is 1 when the solution’s result set equals the gold query’s. The agent has at most six turns.

Qwen3’s thinking mode breaks this environment. With thinking on, the teacher rarely uses the tool. With thinking off, it never writes the `<think>` tags that the format rules require. We therefore disable thinking and drop only these two rules (App. L).

Training. Qwen3-32B (Yang et al., 2025) generated 4 trajectories for each of 6,294 Spider training questions (Yu et al., 2018; Li et al., 2025), and 24,043 well-formed trajectories were kept; observations make up 31% of their supervised tokens. A further 1,149 trajectories on 12 databases that SFT never sees are used only for measurement (App. B). We fine-tune all of Qwen3-4B for one epoch (375 steps, batch 64, cosine schedule, peak learning rate 10^{-5}), rebuilding each SFT sequence segment by segment so that it matches, token for token, the stream RL trains on. GRPO (Shao et al., 2024) then runs for 135 steps with 32 tasks $\times$ 16 roll-outs, learning rate 10^{-6} and token-level truncated importance sampling.

Environment audit. In 290 of the 653 public RL training tasks (44.4%), the gold query returns no rows. A constant empty query therefore earns full reward on these tasks; for 18 of them the outcome also depends on the system clock. We train RL on the 363 tasks with non-empty gold results.

Evaluation. We evaluate on Spider-dev and BIRD-dev (Li et al., 2023), sampling 16 attempts per question at temperature 0.6. $\text{pass}@k$ uses the unbiased estimator.

Model	Spider-dev		BIRD-dev	
	p@1	p@16	p@1	p@16
Qwen3-4B (base)	48.5	82.6	32.9	66.1
<i>After SFT</i>				
ActionSFT	71.4	84.1	49.7	73.2
ActObs	71.7	83.7	50.2	73.4
ActionSFT-LR	71.3	84.3	50.3	73.8
ActionSFT+Wiki	71.8	84.2	50.4	73.1
Outcome-Aux	71.0	84.2	49.8	73.6
Outcome-in-Action	71.2	83.9	50.0	73.8
Contract	71.8	84.6	49.5	74.0
<i>After GRPO</i>				
ActionSFT, seed 1	80.5	88.8	58.1	76.6
ActionSFT, seed 2	80.3	88.4	58.2	76.8
ActObs, seed 1	80.4	88.9	57.2	76.4
ActObs, seed 2	81.4	88.9	57.6	77.5
ActionSFT-LR, seed 1	80.8	88.4	58.7	77.4
Outcome-in-Action, seed 1	[pending]	[pending]	[pending]	[pending]
Contract, seed 1	80.8	88.9	57.5	76.4

Table 1: Main results: pass@ k (%) on the primary sets, with 16 attempts per question. All SFT initialisations are closely matched. After the same GRPO, observation supervision (ActObs) does not improve on ActionSFT.

The *primary set* excludes questions whose gold query fails or returns nothing: 983 of 1,034 on Spider and 1,532 of 1,534 on BIRD. Differences between arms are reported with paired bootstrap 95% confidence intervals over questions.

The pre-registered primary outcome is post-GRPO pass@16, pooled over both primary sets.

3.2 Observation supervision does not strengthen downstream RL

Table 1 summarises the results. As in Zhang et al. (2026), all SFT initialisations are closely matched before RL. Unlike in their study, the same GRPO does not separate them (Figure 2).

The primary outcome. Averaged over seeds, as pre-registered, ActObs minus ActionSFT on pooled pass@16 is +0.26 points (95% CI $[-0.38, 0.91]$; 2,515 questions). The two ActObs seeds give -0.06 and $+0.58$ against the mean of the ActionSFT seeds, and both CIs cover zero. On each dataset separately, all eight pass@16 differences between a ActObs seed and an ActionSFT seed have CIs that cover zero; they range from -0.39 to $+0.85$.

This null has a limited resolution. Question-level CIs span about ± 1.2 points, and after RL, RL seeds and SFT checkpoints move pass@1 by up to about 1.0 and 1.5 points (below). An effect of the size Zhang et al. (2026) report for a 4B model ($+1.1$ pass@16) would therefore go undetected. Much larger effects are unlikely in this setting.

Pass@1 gaps are within training variance. At pass@1, ActObs’s first seed is 0.98 points below

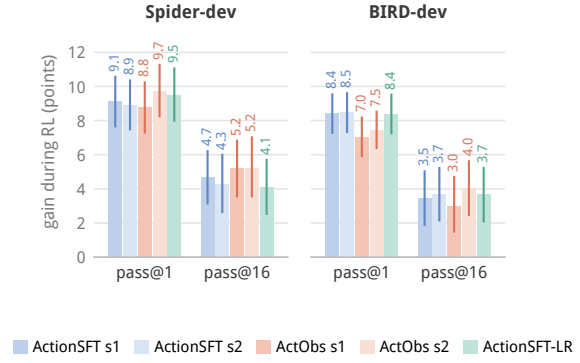


Figure 2: Gain during RL from each SFT initialisation: post-GRPO minus pre-GRPO pass@1 and pass@16 of the same initialisation, with 95% bootstrap CIs over primary questions. Unlike on Terminal-Bench (Zhang et al., 2026), GRPO extracts no consistently larger gain from ActObs. In both seeds its BIRD pass@1 gain is 1.0–1.4 points smaller than ActionSFT’s; elsewhere its two seeds lie on either side of the ActionSFT seeds or above them by less than one point, with overlapping CIs.

ActionSFT on BIRD (CI $[-1.69, -0.24]$ against ActionSFT’s seed 2; -0.94 against seed 1). Its second seed, trained from the same SFT checkpoint, is 0.50 and 0.55 points below, with CIs that cover zero, and on Spider it is 0.90 and 1.13 points above ActionSFT (CIs $[0.33, 1.49]$ and $[0.52, 1.76]$). The two ActObs seeds thus differ by 0.97 points on Spider pass@1 (CI $[0.50, 1.47]$), although the two ActionSFT seeds agree within 0.23 points.

ActionSFT-LR and ActObs are action-equivalent before RL (§4.1). Yet after GRPO ActionSFT-LR is 1.48 points above ActObs’s first seed on BIRD pass@1 (CI $[0.79, 2.17]$). RL seeds and SFT checkpoints therefore move pass@1 by up to about 1.0 and 1.5 points, more than the question-level CIs capture. We do not read either seed’s gap as an effect of observation supervision.

RL movement and entropy do not single out ActObs either (App. F).

The pre-registered replication test also required the same sign in two of three seeds. The two seeds that were run have opposite signs on pooled pass@16, and seed 3 was put on hold once seed 1 showed no gain (App. M), so the criterion is not met.

4 Observation Knowledge Is Learned but Not Used

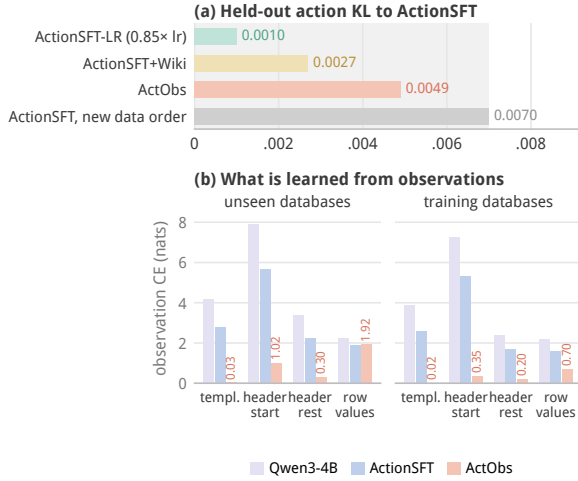


Figure 3: **(a)** Held-out action KL of each SFT arm to ActionSFT. The grey bar and dotted line mark the data-order noise floor: an action-only model trained on the same data in a different order. **(b)** Observation cross-entropy (nats) by part, on unseen (held-out) databases and on training databases. ActObs learns the template and the result header everywhere, but predicts row values better only on databases it was trained on.

4.1 The action policy is unchanged

We measure the mean per-token KL between action distributions on 300 held-out trajectories from databases that SFT never saw. For scale, SFT moves the policy far from the base model: the KL from Qwen3-4B is 0.364 for ActionSFT and 0.363 for ActObs. Against that, ActObs stays very close to ActionSFT, with $\text{KL}(\text{ActObs} \parallel \text{ActionSFT}) = 0.0049$ (Figure 3a).

This is *less* than the KL between two action-only models trained on the same data in a different order (0.0070). It is of the same order as the effect of a 15% lower learning rate (ActionSFT-LR, 0.0010) or of Wikipedia dilution (ActionSFT+Wiki, 0.0027).

Behaviour after observations agrees. On BIRD before RL, ActionSFT and ActObs produce a similar share of trajectories with an SQL error (16.5% vs. 15.7%). They also solve the same share of those trajectories (30.3% in both). After GRPO the figures are 13.5% vs. 13.6%, and 37.7% vs. 37.5%.

GRPO does change how the agent uses observations. Re-issuing a query that has just failed drops from 23–26% of failures to 6–7%. But it changes ActionSFT and ActObs by the same amount (App. G).

4.2 What the model learns

ActObs does learn to predict observations. Its observation cross-entropy on training trajectories falls

from 3.57 to 0.25 within the epoch. Most of this learning, however, does not transfer.

We split observation tokens into three parts: the fixed template (`<observation>` tags and the turn reminder), the result header (the first line of a result table, which encodes its column count), and the row values. We then compare databases that SFT never saw with databases it trained on (Figure 3b; full numbers in App. E). Structure transfers: on unseen databases ActObs predicts the template almost perfectly (CE 0.03 vs. 2.79 for ActionSFT), and its CE on the result header falls from 5.64 to 1.02 on the first tokens and from 2.22 to 0.30 on the rest of the line. Content does not. On unseen databases, row values are predicted no better than by ActionSFT (1.92 vs. 1.88 over 35,845 tokens), whereas on training databases ActObs fits them closely (0.70 vs. 1.60). Error messages and empty results are too rare in held-out teacher trajectories (5 and 3 segments) to analyse.

ActObs thus gains two kinds of knowledge. One is structure it can predict but has no reason to act on, since the template and header follow from the query it has just written. The other is database content that cannot transfer to the unseen databases it is evaluated on. Neither tells the agent whether its current result answers the question.

The gradients agree. Action and observation gradients are nearly orthogonal at the end of SFT (cosine 0.007), as Zhang et al. (2026) also report. On training trajectories, the observation gradient is no larger than the action gradient: the norm ratio is 1.09, against 5.49 on held-out trajectories.

The pre-registered dilution account also fails at the SFT stage: ActObs moves the action policy as much as ActionSFT does, although its pooled loss gives actions only 69% of each update. App. D.1 discusses why.

5 Knowledge Is Learned but Not Used for Decisions

§4 showed that observation knowledge is learned but leaves the policy unchanged. Two explanations were open: the knowledge may be the wrong kind, or it may sit in the wrong place. The exploratory arms of §2 vary both. Each arm can fail at three points: the knowledge may not be learned, it may not move the action policy, or it may not change the agent’s decisions. We test each in turn.

5.1 Every variant learns its target on unseen databases

All the knowledge we supervise is acquired, and it transfers to the 12 databases that SFT never saw. ActObs learns the observations’ structure (§4.2). The outcome arms learn their line with CE 0.20 (Outcome-Aux) and 0.26 (Outcome-in-Action), against 3.8–4.3 for models not trained on it. Contract states the correct coarse shape of the answer in 89% (Spider) and 82% (BIRD) of its rollouts. For comparison, always writing the most frequent shape is right 42% and 60% of the time, a naive-Bayes classifier on the question words 71% and 49%, and the placebo arm 30% and 35%. The checks that Contract writes after each observation are computed correctly 99.7–99.97% of the time. The models therefore know the outcome of their queries and what a correct answer should look like.

5.2 Placement decides whether actions move

Outcome-Aux and Outcome-in-Action supervise the same line with the same weight. Only Outcome-in-Action, where the line precedes the query, changes the actions. In the shared context that contains the line, $\text{KL}(\cdot \| \text{ActionSFT})$ is 0.0162 for Outcome-in-Action, about 2.3 times the data-order noise floor in that context (0.0070). For Outcome-Aux it is 0.0043, and for ActObs 0.0058.

The predictions inside Outcome-in-Action’s actions are, however, blind to failure. In its own rollouts, Outcome-in-Action writes “error” before 9 of the 8,311 calls that do fail (0.1%), and “no rows” before 9% of the calls that return nothing. Two explanations fit, and our design cannot separate them. One is the labels: they come from teacher trajectories in which only 2.8% of calls fail, whereas successful predict-then-act methods use failure-rich, on-policy labels (Chen et al., 2025; Zhang et al., 2025). The other is the target: predicting one’s own query’s result adds little when executing it is free (FAIR CodeGen team et al., 2025; Fu et al., 2024).

5.3 No variant changes the agent’s decisions

The decision a SQL agent faces after every observation is whether to submit or keep querying. An agent that used answer knowledge would submit when the result has the right shape and keep querying when it does not. We therefore label every observation in the evaluation rollouts by whether its coarse shape matches the gold result. We define

the *decision sensitivity* S as $\text{P}(\text{submit next} \mid \text{match})$ minus $\text{P}(\text{submit next} \mid \text{mismatch})$. We compare each arm with ActionSFT using a bootstrap over questions. As noise references we use ActionSFT trained on a different data order (before RL) and a second RL seed (after RL).

The result is uniform (Figure 4). The asymmetry predates the contract. ActionSFT already submits far more often after a matching result than after a mismatching one ($S = 0.43$ on Spider and 0.46 on BIRD), a tendency inherited from the teacher. No SFT variant moves S by more than the data-order reference does: $|\Delta S| \leq 0.016$ against 0.033 on Spider, and ≤ 0.010 against 0.020 on BIRD. This holds even for Contract, whose own checks report each mismatch explicitly. It submits after 55% and 31% of the mismatches it reports, the same as ActionSFT after mismatches judged against gold shapes (48% and 29%).

After GRPO the arms remain inseparable. Two RL seeds of the same checkpoint already differ in S by 0.052 and 0.019 (ActionSFT) and by 0.022 and 0.029 (ActObs) on Spider and BIRD. Differences between arms are of this size and have no consistent sign: the largest come from an action-only control (ActionSFT-LR: $+0.071$ and $+0.032$) and from ActObs’s second seed (-0.025 and -0.031). What RL does change is shared by every arm: it raises the rate of submitting after a mismatch, on BIRD from 0.29 to 0.41 for ActionSFT and from 0.28 to 0.41 for Contract. Measured with Contract’s own checks, the rate rises from 31% to 46%. Per-arm numbers, including first and later observations, are in App. H.

5.4 The knowledge is usable from outside, until RL

What the agent does not use itself, a selector can. For each question we take the contract Contract writes most often across its samples, and pick among the 16 samples one whose final result has that shape (Table 2). Before RL the contract helps selection. On Contract’s own samples it beats an execution-validity filter by 4.3 (Spider) and 3.4 (BIRD) points. The placebo’s own contracts add only about one point (73.7 and 54.5, against 72.5 and 53.4 for the filter). The contract also transfers to another model’s samples: applied to ActionSFT it gives 76.9 and 56.5, and combined with majority voting over execution results it adds 2.6 and 1.4 points to voting alone.

After GRPO the contract no longer helps. On

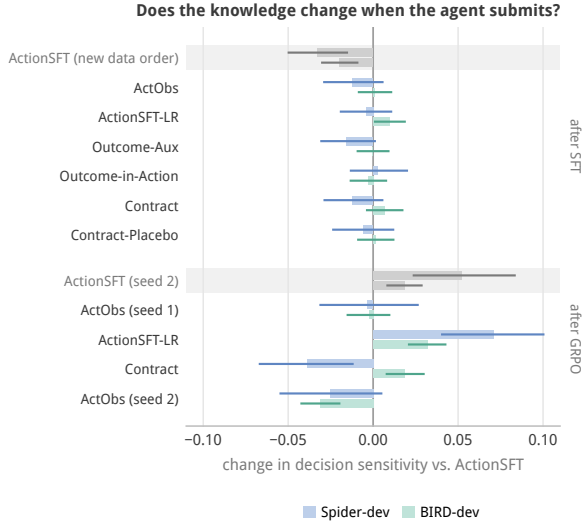


Figure 4: Decision sensitivity, $P(\text{submit next} \mid \text{result has the gold shape}) - P(\text{submit next} \mid \text{it does not})$, for each arm relative to ActionSFT, with 95% question-bootstrap CIs. Grey rows are noise references: ActionSFT trained on another data order (after SFT), and a second RL seed (after GRPO). After SFT no variant moves beyond the data-order reference; after GRPO the differences are of the size of RL-seed variation and have no consistent sign.

GRPO-trained ActionSFT, contract plus voting does not beat voting alone (82.5 vs. 82.6 on Spider; 63.3 vs. 63.9 on BIRD), and on Contract’s own GRPO samples its contract only matches the filter (81.3 vs. 81.2; 59.4 vs. 59.6). RL has already removed most of the shape errors that a contract would catch: among wrong answers on Spider, the share with the wrong number of columns falls from 36% to 12% (App. I).

5.5 No gain in accuracy

Our pre-registered prediction for Contract was a BIRD pass@1 gain of at least one point after GRPO. It is refuted. Contract reaches 57.5, against 58.1 and 58.2 for the two ActionSFT seeds (differences -0.60 $[-1.17, -0.03]$ and -0.65 $[-1.22, -0.07]$). On Spider it reaches 80.8, which is $+0.3$ and $+0.5$ points above the two seeds. On the pooled primary outcome, pass@16, Contract minus ActionSFT is -0.06 $[-0.78, 0.66]$. Whether GRPO changes this for Outcome-in-Action is **[pending]**.

6 Related Work

Learned but not used. Our central observation, that knowledge acquired in training does not reach decisions, has precedents. LLMs that write a cor-

rect exploration rationale still act greedily on bandits (Schmied et al., 2025). In tool use, whether a model needs a tool and whether it calls one disagree in 27–54% of cases, and the two signals are nearly orthogonal inside the model (Cheng et al., 2026). Models encode correct answers they do not generate, and a probe can use them to select among samples (Orgad et al., 2024), as our contract does before RL. Fine-tuning on correction traces rarely produces self-correction (Kumar et al., 2024), and models resist even ground-truth feedback (Jiang et al., 2025).

Two features of our setting stand out. First, the gap is inherited: the action-only imitator shows the same submit asymmetry, and none of the added knowledge changes it. Second, RL does not close it. Outcome-only GRPO is known to reduce tool use (Wei et al., 2025) and to concentrate samples (Yue et al., 2025); consistent with this, here it makes every variant submit more readily and removes the selector’s advantage. Predicted knowledge does reach decisions in methods that either train predictions on the agent’s own interaction (Chen et al., 2025; Zhang et al., 2025) or reward them (Wang et al., 2025). Our labels come from teacher trajectories and our reward ignores them.

Observation and world-model losses. Agent pipelines usually mask tool outputs during training (Jin et al., 2025). Recent work instead trains on them, through next-observation prediction in SFT (Zhang et al., 2026), observation losses during RL (Shrivastava et al., 2026), policy and world-model co-training (Lu et al., 2026; Li et al., 2026), a separate world-model RL stage (Yu et al., 2026), or decision-oriented targets (Cai et al., 2026). These works report gains in task success, but content-free placebo losses can match real observation losses (Wang, 2026). We ask instead whether the learned knowledge reaches actions and decisions.

Predictions inside reasoning, and execution knowledge. Agents can also write predictions into their own text, with rewarded state and transition prediction (Wang et al., 2025), self-experience SFT (Chen et al., 2025), simulation inside reasoning (Yu et al., 2025b,a), prompted outcome prediction (Fu et al., 2024) or model-based planning (Gu et al., 2024). In code generation, training on execution traces improves execution prediction more than generation (FAIR CodeGen team et al., 2025; Armengol-Estap   et al., 2025; Ni et al., 2024), and specification-first methods check can-

Selection among 16 samples	ActionSFT after SFT		ActionSFT after GRPO		Contract after SFT		Contract after GRPO	
	Spider	BIRD	Spider	BIRD	Spider	BIRD	Spider	BIRD
Random sample (= pass@1)	71.4	49.7	80.5	58.1	71.8	49.5	80.8	57.5
Free rule: no error, non-empty	72.5	53.5	81.1	60.5	72.7	53.6	81.2	59.6
Execution majority vote	75.8	58.3	82.6	63.9	75.9	58.0	83.1	63.4
Contract’s contract (modal over samples)	76.9	56.5	81.3	60.6	77.0	57.0	81.3	59.4
Contract’s contract + vote	78.4	59.7	82.5	63.3	78.2	60.1	83.0	62.5
<i>Oracle</i> : gold coarse shape	78.2	59.5	82.6	63.4	78.6	60.3	83.1	62.1

Table 2: Choosing one of the 16 samples per question (primary questions). The oracle uses the gold result’s shape. The contract rows use Contract’s most frequent contract per question: for ActionSFT they apply the SFT Contract’s contracts to another model’s samples, and Contract after GRPO uses its own.

didates against generated tests or properties (Chen et al., 2022; Prasad et al., 2025; Lee and Huang, 2026). Question answering has long predicted the expected answer type (Li and Roth, 2002). SQL systems check or rerank candidates after execution by column count (Chen et al., 2023), by rules (Tian and Zhang, 2026; Gong et al., 2025), with predicted results (Li and Xie, 2024), with trained verifiers (Ni et al., 2023; Pourreza et al., 2024), or by discarding failing and empty queries (Wang et al., 2018). To our knowledge these checks are prompted or rule-based; our contract is learned inside a multi-turn agent, and we ask whether the agent uses it.

SFT to RL. Over-trained SFT checkpoints can lose plasticity for RL (Liu et al., 2026a). Acting on execution feedback usually needs on-policy RL (Gehring et al., 2024; Kumar et al., 2024). Multi-turn RL for text-to-SQL is an active area (Xu et al., 2025; Liu et al., 2026b). Adam is partly invariant to scaling a task’s loss (Elich et al., 2023; Kurin et al., 2022).

7 Conclusion

Supervising a tool-using agent on its observations teaches it their structure and the training databases’ contents. It does not teach anything the agent uses when acting: the action policy is unchanged, behaviour after errors is unchanged, and RL gains nothing.

We then changed what the agent predicts and where. Placement matters for the action tokens: the same outcome line moves the policy only once it is part of the action. It does not matter for decisions. Even the shape of the correct answer is learned on unseen databases, written into the action stream and checked against every observation. Still, the agent submits after a mismatching result as often as an action-only model does. RL makes all variants submit more readily, and the contract then stops

helping even as an external selector.

Learning about the environment, or even about the answer, is therefore not enough. Decisions follow imitation and the outcome reward. Making knowledge reach them probably requires rewarding its use, or training on the agent’s own failure-rich interaction.

Limitations

Scope. We study one student (Qwen3-4B), one teacher, one domain (SQL), and one RL algorithm. Our setting differs from Zhang et al. (2026) in ways that may matter. Terminal observations are longer and richer in failures, whereas in our teacher data only 2.8% of SQL calls fail. Our negative result for observation supervision should therefore not be read as a refutation of their Terminal-Bench result.

Seeds and variance. GRPO results rest on two seeds for ActionSFT and ActObs and on one seed for the other arms. After RL, two seeds of the same checkpoint differ by up to about one point and SFT checkpoints by about 1.5 points (§3.2), more than our question-level confidence intervals capture. Effects of about one point should be treated as unresolved.

Exploratory arms. The outcome and contract arms were added after the first results were seen. Their predictions were written down before training, but they are not part of the original pre-registration.

Fairness of the contract arm. Contract receives the gold result shapes of the training questions, which ActionSFT does not. Contract-Placebo controls for the format of the inserted text, not for this information. A gold-matched control, SFT on correct trajectories only, is missing. Two further caveats apply to the placebo. Because the shapes are coarse, 24% of the placebo contracts coincide

with the gold shape. Most placebo checks also report a mismatch, which may bias Contract-Placebo as a control for behaviour, and Contract-Placebo was not trained with RL.

Mechanism. The optimiser account in §D.1 is suggestive. Action KL to the base model is an insensitive metric, and we did not ablate the optimiser.

Ethics Statement

This work uses public text-to-SQL benchmarks and open-weight models. The environment audit identifies a reward loophole in a public RL environment. We reported it to the maintainers together with a reproduction script (SkyRL issue #2451). We see no direct risks beyond those of the underlying models.

References

- Jordi Armengol-Estapé, Quentin Carbonneaux, Tianjun Zhang, Aram H. Markosyan, Volker Seeker, Chris Cummins, Melanie Kambadur, Michael F. P. O’Boyle, Sida Wang, Gabriel Synnaeve, and Hugh James Leather. 2025. [What i cannot execute, i do not understand: Training and evaluating llms on program execution traces](#). *arXiv preprint*.
- Guangfeng Cai, Kaibing Yang, Shuo He, Yu Li, Shengtian Yang, Jiaqi Lv, and Lei Feng. 2026. [Beyond next-observation prediction: Agent-authored world modeling for sequential decision making](#). *arXiv preprint*.
- Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2022. [Codet: Code generation with generated tests](#). *arXiv preprint*.
- Shiqi Chen, Tongyao Zhu, Zian Wang, Jinghan Zhang, Kangrui Wang, Ruochen Zhou, Siyang Gao, Teng Xiao, Yee Whye Teh, Junxian He, and Manling Li. 2025. [Why do llm agents fail in exploring new environments? a world-modeling perspective](#). *arXiv preprint*.
- Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. [Teaching large language models to self-debug](#). *arXiv preprint*.
- Yize Cheng, Chenrui Fan, Mahdi JafariRaviz, Keivan Rezaei, and Soheil Feizi. 2026. [Model-adaptive tool necessity reveals the knowing-doing gap in llm tool use](#). *arXiv preprint*.
- Cathrin Elich, Lukas Kirchdorfer, Jan M. Köhler, and Lukas Schott. 2023. [Examining common paradigms in multi-task learning](#). *arXiv preprint*.
- FAIR CodeGen team, Jade Copet, Quentin Carbonneaux, Gal Cohen, Jonas Gehring, Jacob Kahn, Jan-nik Kossen, Felix Kreuk, Emily McMilin, Michel Meyer, Yuxiang Wei, David Zhang, Kunhao Zheng, Jordi Armengol-Estapé, Pedram Bashiri, Maximilian Beck, Pierre Chambon, Abhishek Charnalia, Chris Cummins, and 32 others. 2025. [Cwm: An open-weights llm for research on code generation with world models](#). *arXiv preprint*.
- Dayuan Fu, Jianzhao Huang, Siyuan Lu, Guanting Dong, Yejie Wang, Keqing He, and Weiran Xu. 2024. [Preact: Prediction enhances agent’s planning ability](#). *arXiv preprint*.
- Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. 2024. [Rlef: Grounding code llms in execution feedback with reinforcement learning](#). *arXiv preprint*.
- Yue Gong, Chuan Lei, Xiao Qin, Kapil Vaidya, Balakrishnan Narayanaswamy, and Tim Kraska. 2025. [Sqlens: An end-to-end framework for error detection and correction in text-to-sql](#). *arXiv preprint*.
- Yu Gu, Kai Zhang, Yuting Ning, Boyuan Zheng, Boyu Gou, Tianci Xue, Cheng Chang, Sanjari Srivastava, Yanan Xie, Peng Qi, Huan Sun, and Yu Su. 2024. [Is your llm secretly a world model of the internet? model-based planning for web agents](#). *arXiv preprint*.
- Dongwei Jiang, Alvin Zhang, Andrew Wang, Nicholas Andrews, and Daniel Khashabi. 2025. [Feedback friction: Llms struggle to fully incorporate external feedback](#). *arXiv preprint*.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. [Search-rl: Training llms to reason and leverage search engines with reinforcement learning](#). *arXiv preprint*.
- 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. 2024. [Training language models to self-correct via reinforcement learning](#). *arXiv preprint*.
- Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. 2022. [In defense of the unitary scalarization for deep multi-task learning](#). *arXiv preprint*.
- Woojin Lee and Jin-Xia Huang. 2026. [Solidcoder: Bridging the mental-reality gap in llm code generation through concrete execution](#). *arXiv preprint*.
- Cong Li, Peixi Peng, Yisen Zhao, Xinyu Hu, Shudong Liu, Zhan Su, and Zhuojian Li. 2026. [Tapo: Transition-aware policy optimization for llm agents](#). *arXiv preprint*.

- Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Teying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. 2025. [Omnisql: Synthesizing high-quality text-to-sql data at scale](#). *arXiv preprint*.
- Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. [Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls](#). *arXiv preprint*.
- Xin Li and Dan Roth. 2002. [Learning question classifiers](#). In *COLING 2002: The 19th International Conference on Computational Linguistics*.
- Zhenwen Li and Tao Xie. 2024. [Using llm to select the right sql query from candidates](#). *arXiv preprint*.
- Runze Liu, Jiashun Liu, Xu Wan, Yuqian Fu, and Ling Pan. 2026a. [When rl fails after sft: Rejuvenating model plasticity for robust sft-to-rl handoff](#). *arXiv preprint*.
- Tao Liu, Tao Feng, Xiangheng Li, Jinwang Song, Yifan Li, Xiaoqing Cheng, Dixuan Zhang, Siquan Li, Lin Lan, Hongying Zan, Kunli Zhang, and Chao Wu. 2026b. [Serl-sql: Selective hindsight distillation for text-to-sql reinforcement agentic learning](#). *arXiv preprint*.
- Ning Lu, Baijiong Lin, Shengcai Liu, Jiahao Wu, Haoze Lv, Yanbin Wei, Lingting Zhu, Shengju Qian, Xin Wang, Ying-Cong Chen, Qi Wang, and Ke Tang. 2026. [Policy and world modeling co-training for language agents](#). *arXiv preprint*.
- Ansong Ni, Miltiadis Allamanis, Arman Cohan, Yinlin Deng, Kensen Shi, Charles Sutton, and Pengcheng Yin. 2024. [Next: Teaching large language models to reason about code execution](#). *arXiv preprint*.
- Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I. Wang, and Xi Victoria Lin. 2023. [Lever: Learning to verify language-to-code generation with execution](#). *arXiv preprint*.
- NovaSky-AI. 2025. SkyRL: A modular full-stack RL library for LLMs. <https://github.com/NovaSky-AI/SkyRL>. Commit 99824f0, text-to-SQL recipe.
- Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. 2024. [Llms know more than they show: On the intrinsic representation of llm hallucinations](#). *arXiv preprint*.
- Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. 2024. [Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql](#). *arXiv preprint*.
- Archiki Prasad, Elias Stengel-Eskin, Justin Chih-Yao Chen, Zaid Khan, and Mohit Bansal. 2025. [Learning to generate unit tests for automated debugging](#). *arXiv preprint*.
- Thomas Schmied, Jörg Bornschein, Jordi Grau-Moya, Markus Wulfmeier, and Razvan Pascanu. 2025. [Llms are greedy agents: Effects of rl fine-tuning on decision-making abilities](#). *arXiv preprint*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [Deepseekmath: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint*.
- Vaishnavi Shrivastava, Piero Kauffmann, Ahmed Awadallah, and Dimitris Papailiopoulos. 2026. [Echo: Terminal agents learn world models for free](#). *arXiv preprint*.
- Yuan Tian and Tianyi Zhang. 2026. [Pv-sql: Synergizing database probing and rule-based verification for text-to-sql agents](#). *arXiv preprint*.
- Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018. [Robust text-to-sql generation with execution-guided decoding](#). *arXiv preprint*.
- Kangrui Wang, Pingyue Zhang, Zihan Wang, Yaning Gao, Linjie Li, Qineng Wang, Hanyang Chen, Chi Wan, Yiping Lu, Zhengyuan Yang, Lijuan Wang, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, Yejin Choi, and Manling Li. 2025. [Vagen: Reinforcing world model reasoning for multi-turn vlm agents](#). *arXiv preprint*.
- Yu Wang. 2026. [The dark room in the reward channel: Dense prediction rewards collapse grpo-trained llm agents – and the channel, not the content, decides what works](#). *arXiv preprint*.
- Quan Wei, Siliang Zeng, Chenliang Li, Zhongruo Wang, William Brown, Oana Frunza, Wei Deng, Anderson Schneider, Yuriy Nevmyvaka, Yang Katie Zhao, Alfredo Garcia, and Mingyi Hong. 2025. [Reinforcing multi-turn reasoning in llm agents via fine-grained reward structure and credit assignment](#). *arXiv preprint*.
- Zekun Xu, Siyu Xia, Chuhuai Yue, Jiajun Chai, Mingxue Tian, Xiaohan Wang, Wei Lin, Haoxuan Li, and Guojun Yin. 2025. [Mtr-sql: Multi-turn tool-integrated reasoning reinforcement learning for text-to-sql](#). *arXiv preprint*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *arXiv preprint*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. [Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task](#). *arXiv preprint*.

Xiao Yu, Baolin Peng, Michel Galley, Hao Cheng, Qianhui Wu, Janardhan Kulkarni, Suman Nath, Zhou Yu, and Jianfeng Gao. 2025a. [Dyna-mind: Learning to simulate from experience for better ai agents](#). *arXiv preprint*.

Xiao Yu, Baolin Peng, Ruize Xu, Michel Galley, Hao Cheng, Suman Nath, Jianfeng Gao, and Zhou Yu. 2025b. [Dyna-think: Synergizing reasoning, acting, and world model simulation in ai agents](#). *arXiv preprint*.

Xiao Yu, Baolin Peng, Ruize Xu, Yelong Shen, Pengcheng He, Suman Nath, Nikhil Singh, Jianfeng Gao, and Zhou Yu. 2026. [Reinforcement world model learning for llm-based agents](#). *arXiv preprint*.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. 2025. [Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?](#) *arXiv preprint*.

Juzheng Zhang, Disha Makhija, Manoj Ghuhan Arivazhagan, Vinayshekhar Bannihatti Kumar, and Rashmi Gangadharaiyah. 2026. [Don’t mask the environment: Observation supervision changes how agents explore under rl](#). *arXiv preprint*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, and 11 others. 2025. [Agent learning via early experience](#). *arXiv preprint*.

A Evaluation Protocol and Statistics

Sampling. Every model is evaluated with 16 attempts per question at temperature 0.6, top- p 0.95 and top- k 20, with thinking disabled. Evaluation uses the multi-turn stream that RL trains on: at most six turns, at most 1,024 generated tokens per turn, and at most 8,192 input tokens. Our evaluator re-implements SkyRL’s generation loop with vLLM. On a shared checkpoint it gives Spider-dev pass@1 0.540, against 0.551 from SkyRL’s built-in evaluator with one sample (App. M).

Metrics. Consider a question with $n = 16$ attempts, c of which are correct. Its pass@ k is the unbiased estimator $1 - \binom{n-c}{k} / \binom{n}{k}$, and we average this over questions. An attempt is correct when the result set of its final <solution> query equals that

of the gold query, compared as sets of rows, exactly as in the training reward. An attempt without a well-formed solution scores zero.

Primary sets. Any query that returns nothing matches a gold query that returns nothing, so questions with empty gold results reward empty answers (§3.1). The primary sets therefore exclude questions whose gold query fails or returns no rows. That leaves 983 of 1,034 Spider-dev questions and 1,532 of 1,534 BIRD-dev questions, 2,515 when pooled. Every analysis uses these sets. Gold queries are executed once, with a 30 s limit, and their results are cached, so that analyses run at different times cannot drop questions whose gold query happens to time out.

Confidence intervals. Paired differences between models use a bootstrap over questions that scores the same resampled questions for both models. We use 10,000 resamples for pre-registered contrasts and 2,000 otherwise. Behavioural rates use a question-cluster bootstrap, which keeps all observations of a resampled question together. These intervals capture question sampling, not training variance. We estimate training variance separately from three sources: a second RL seed, a second SFT data order, and two SFT checkpoints that are action-equivalent before RL (§3.2).

Selection rules. The selection analyses pick one of the 16 samples for each question. From weakest to strongest, the *random* rule takes a uniformly random sample, which equals pass@1 in expectation. The *validity* rule takes a random sample among those whose final result is neither an error nor empty. *Majority vote* takes the largest cluster of identical final results, averaging over ties. The *contract* rule takes a random valid sample whose result matches Contract’s most frequent contract for that question, and *contract+vote* applies majority vote within those samples. The oracle instead picks a random sample matching the gold coarse shape. When no sample qualifies, a rule falls back to the next weaker one.

B Data, Prompt and Token Stream

Teacher trajectories. The teacher, Qwen3-32B, sampled 4 trajectories for each of 6,294 Spider training questions with temperature 0.6, top- p 0.95, top- k 20 and thinking disabled. This gave 25,176 trajectories. We dropped 44 that reached the length

limit and 1,089 with a malformed final turn, keeping 24,043, of which 12,171 are correct. The kept trajectories contain 7.19M action tokens, 2.45M observation-payload tokens and 0.77M observation-template tokens, so observations make up 31.0% of supervised tokens. We removed from the SFT pool the 114 questions that also appear in the RL task set.

Measurement and RL sets. The 1,149 measurement trajectories are teacher rollouts on 592 questions from 12 Spider databases that the SFT data never uses. We sampled 2 per question (1,184 trajectories) and kept 1,149; observations make up 35.5% of their supervised tokens. RL uses the 363 of the 653 public SkyRL-SQL training tasks whose gold query returns at least one row (App. L).

Token classes and stream. Each SFT sequence is one assistant turn into which the tool observations are inserted verbatim, which is SkyRL’s recipe mode for this environment. Every token belongs to one of four classes: the prompt, which is never supervised; actions; the observation payload; and the observation template, which consists of the `<observation>` tags and the turn reminder. We rebuild each trajectory segment by segment from the token ids that generation and the environment produced, so the SFT stream matches the RL stream token for token. Unit tests check this. They also check the loss masks, the $\text{pass}@k$ estimator, and that the chunked LM head gives the same loss and gradients as an unchunked one.

Prompt. All stages use SkyRL’s system prompt with one added format line, which asks the agent to execute and check a candidate query before answering (Figure 5).

C Full Results

Table 3 lists $\text{pass}@k$ for every model at $k \in \{1, 2, 4, 8, 16\}$, and Table 4 lists every paired contrast. Three patterns hold across the table. All SFT variants lie within about one point of each other at every k , and this includes ActionSFT trained on a different data order. GRPO adds 7–10 points at $\text{pass}@1$ and 3–5 points at $\text{pass}@16$ for every initialisation. No SFT objective changes how much GRPO adds by more than the spread between initialisations.

D SFT-Stage Measurements

Table 5 reports the action KL to the base model, the cross-entropy of each token class and gradient statistics, measured on 300 held-out trajectories from 12 unseen databases. The learning-rate candidates for ActionSFT-LR were fixed in advance at $\{0.85, 0.70, 0.55, 0.40\} \times$. The frozen rule picks the candidate whose action KL to the base model is closest to ActObs’s, and it picked $0.85 \times$ (0.358 against 0.363).

The ActionSFT+Wiki_{obs} row dilutes the action loss with observation text taken from other trajectories. This text stays harder to predict than in-context observations: training CE goes from 3.20 to 1.09, against 3.57 to 0.25 for ActObs. Its KL is nevertheless indistinguishable from ActionSFT-LR’s.

D.1 Why pooled dilution vanishes

The dilution reading predicts that ActObs moves the action policy as if its learning rate were scaled by 0.69. It does not: ActObs’s action KL to the base model (0.363) equals ActionSFT’s (0.364). Lowering the learning rate to $0.4 \times$ gives 0.334, and Wikipedia dilution at the same token share gives 0.352, which is close to a $0.7 \times$ learning rate.

One account consistent with the training logs is that Adam normalises each coordinate’s update by a running estimate of its gradient magnitude. Down-weighting the action loss uniformly therefore changes little unless the extra tokens keep producing large gradients in the same coordinates (Elich et al., 2023). ActObs’s observations are fitted early and their gradients shrink, whereas Wikipedia stays hard to predict (training CE from 3.18 to 2.44) and keeps diluting.

This account is only suggestive. KL to the base model changes little across learning rates and differs by 0.003 between two data orders, and we did not run an optimiser ablation.

E Observation Cross-Entropy by Part

Table 6 gives the full breakdown behind Figure 3b. On unseen databases, ActObs’s advantage over ActionSFT lies entirely in the template and the result header; on row values it is no better than ActionSFT or ActionSFT+Wiki. On row values, ActObs’s CE is 0.70 on training databases and 1.92 on unseen ones, the signature of memorised content.

Task Overview: You are a data science expert. Below, you are provided with a database schema and a natural language question. Your task is to understand the schema and generate a valid SQL query to answer the question within limited turns. [...]

Format:

- Before giving the final solution, you MUST execute at least one SQL query with the SQL tool (for example, your candidate query) and check its result in the observation.
- You can use SQL tool written within a single `<sql>your sql</sql>` block to explore or verify. SQL tool output will be shown as dataframe inside `<observation>...</observation>`. [...]
- If you find no further exploration is needed or reaches max turns, you MUST directly provide the final SQL query solution inside `<solution>...</solution>`.

Figure 5: Excerpt of the system prompt; the first *Format* line is the one we added (“verify” instruction).

Model	Spider-dev (983 questions)					BIRD-dev (1,532 questions)				
	p@1	p@2	p@4	p@8	p@16	p@1	p@2	p@4	p@8	p@16
Qwen3-4B (base)	48.5	63.7	74.0	79.6	82.6	32.9	44.9	54.7	61.5	66.1
ActionSFT	71.4	76.6	80.0	82.3	84.1	49.7	58.1	64.5	69.3	73.2
ActionSFT, data order 2	69.9	75.4	79.2	81.8	83.6	49.2	57.3	63.5	68.5	72.7
ActObs	71.7	76.9	80.2	82.3	83.7	50.2	58.5	64.7	69.5	73.4
ActionSFT-LR	71.3	76.7	80.2	82.6	84.3	50.3	58.6	64.8	69.6	73.8
ActionSFT+Wiki	71.8	76.9	80.2	82.5	84.2	50.4	58.4	64.5	69.2	73.1
Outcome-Aux	71.0	76.4	80.0	82.6	84.2	49.8	58.2	64.5	69.4	73.6
Outcome-in-Action	71.2	76.6	80.1	82.4	83.9	50.0	58.5	64.9	69.8	73.8
Contract	71.8	77.2	80.7	83.1	84.6	49.5	58.0	64.6	69.8	74.0
Contract-Placebo	71.5	77.0	80.5	82.8	84.4	49.7	58.1	64.4	69.2	73.2
GRPO ActionSFT, seed 1	80.5	83.8	85.9	87.5	88.8	58.1	65.0	69.8	73.4	76.6
GRPO ActionSFT, seed 2	80.3	83.5	85.6	87.1	88.4	58.2	65.1	69.9	73.7	76.8
GRPO ActObs, seed 1	80.4	83.7	85.9	87.4	88.9	57.2	64.2	69.5	73.5	76.4
GRPO ActObs, seed 2	81.4	84.6	86.5	87.7	88.9	57.6	64.8	70.1	74.1	77.5
GRPO ActionSFT-LR, seed 1	80.8	84.1	86.1	87.4	88.4	58.7	65.7	70.6	74.3	77.4
GRPO Outcome-in-Action, seed 1	[pending]	[pending]	[pending]	[pending]	[pending]	[pending]	[pending]	[pending]	[pending]	[pending]
GRPO Contract, seed 1	80.8	84.0	86.0	87.5	88.9	57.5	64.6	69.7	73.3	76.4

Table 3: pass@ k (%) on the primary sets for every model, 16 attempts per question (unbiased estimator).

F RL Training Dynamics

Training reward rises the same way from every initialisation (Table 7), reaching 0.69–0.70 over the last 20 steps.

Zhang et al. (2026) report that ActObs retains more entropy with less policy movement. Here both ActObs seeds (KL 0.107 and 0.100) and the action-only ActionSFT-LR (0.118) move *more* than either ActionSFT seed (0.067 and 0.082). They also keep more entropy: 0.47, 0.47 and 0.49, against 0.43 and 0.45. Because an action-only arm reproduces ActObs’s pattern, these quantities do not identify observation supervision in our setting. Note that ActionSFT seed 1 is the only run on 4 GPUs.

G Behaviour After Observations

Table 8 compares how the agents react to SQL errors and empty results. Before RL the SFT variants are indistinguishable. GRPO then changes every one of them the same way: they take fewer turns and re-issue far fewer failed queries (23–28% of

failures on BIRD before RL, 6–9% after).

H Decision Sensitivity

Table 9 gives the numbers behind Figure 4. Two further checks support the conclusion.

Position in the trajectory. The result does not depend on where in the trajectory the observation falls. We recomputed the measure for ActObs, ActionSFT-LR, Outcome-in-Action and Contract, separately for the first observation of each rollout and for later ones. Each stays within 0.02 of ActionSFT on both datasets.

Contract’s own checks. These lead to the same conclusion as the gold-shape labels. After a mismatch that it reports itself, Contract submits 55% (Spider) and 31% (BIRD) of the time before RL, and 79% and 46% after GRPO. After RL it is therefore more likely to submit an answer it has just labelled as wrong.

Contrast (after GRPO)	Spider p@1	Spider p@16	BIRD p@1	BIRD p@16	Pooled p@16
ActionSFT s2 – ActionSFT s1	-0.23 [-0.70, +0.24]	-0.41 [-1.53, +0.61]	+0.04 [-0.45, +0.55]	+0.20 [-0.91, +1.31]	-0.04 [-0.83, +0.80]
ActObs s1 – ActionSFT s1	-0.07 [-0.67, +0.52]	+0.10 [-0.71, +1.02]	-0.94 [-1.62, -0.25]	-0.20 [-1.44, +1.11]	-0.08 [-0.91, +0.72]
ActObs s1 – ActionSFT s2	+0.16 [-0.46, +0.79]	+0.51 [-0.61, +1.63]	-0.98 [-1.69, -0.27]	-0.39 [-1.57, +0.85]	-0.04 [-0.91, +0.83]
ActObs s2 – ActionSFT s1	+0.90 [+0.33, +1.49]	+0.10 [-0.71, +0.92]	-0.50 [-1.15, +0.16]	+0.85 [-0.39, +2.09]	–
ActObs s2 – ActionSFT s2	+1.13 [+0.52, +1.76]	+0.51 [-0.51, +1.63]	-0.55 [-1.24, +0.13]	+0.65 [-0.72, +2.02]	–
ActObs s2 – ActObs s1	+0.97 [+0.50, +1.47]	+0.00 [-1.02, +1.02]	+0.44 [-0.12, +1.01]	+1.04 [-0.20, +2.28]	+0.64 [-0.24, +1.47]
ActionSFT-LR s1 – ActObs s1	+0.40 [-0.11, +0.93]	-0.51 [-1.53, +0.41]	+1.48 [+0.79, +2.17]	+0.98 [-0.20, +2.15]	+0.40 [-0.44, +1.19]
Contract s1 – ActionSFT s1	+0.29 [-0.17, +0.74]	+0.10 [-1.02, +1.12]	-0.60 [-1.17, -0.03]	-0.20 [-1.37, +0.98]	–
Contract s1 – ActionSFT s2	+0.52 [+0.04, +1.00]	+0.51 [-0.61, +1.63]	-0.65 [-1.22, -0.07]	-0.39 [-1.57, +0.78]	–
mean(ActObs) – mean(ActionSFT) (pooled)		pass@1: -0.24 [-0.65, +0.16]			+0.26 [-0.38, +0.91]
ActObs s1 – mean(ActionSFT) (pooled)		pass@1: -0.57 [-1.04, -0.11]			-0.06 [-0.82, +0.68]
ActObs s2 – mean(ActionSFT) (pooled)		pass@1: +0.08 [-0.35, +0.52]			+0.58 [-0.22, +1.35]
Contract s1 – mean(ActionSFT) (pooled)		pass@1: -0.22 [-0.58, +0.12]			-0.06 [-0.78, +0.66]
ActionSFT-LR s1 – mean(ActionSFT) (pooled)		pass@1: +0.49 [+0.14, +0.83]			+0.34 [-0.38, +1.05]

Table 4: Paired differences in points with 95% bootstrap CIs over primary questions (10,000 resamples; 2,000 for ActionSFT-LR – ActObs). Pooled rows combine Spider and BIRD (2,515 questions); the first is the pre-registered primary outcome.

Arm	KL	CE _{act}	CE _{pay}	CE _{templ}	cos	ratio
ActionSFT	.364	.302	2.02	2.78	-.010	6.15
ActionSFT, data order 2	.361	.302	2.02	2.80		
ActObs	.363	.303	1.80	0.03	.007	5.49
ActionSFT-LR (0.85×)	.358	.301	2.01	2.74		
lr 0.70×	.351	.302	2.02	2.76		
lr 0.55×	.344	.303	2.02	2.79		
lr 0.40×	.334	.305	2.02	2.83		
ActionSFT+Wiki	.352	.301	1.97	2.77	-.005	5.99
ActionSFT+Wiki _{obs}	.358	.303	1.76	0.91		
Outcome-Aux	.362	.302	2.07	3.61		

Table 5: SFT-stage measurements on held-out trajectories. Columns: action KL to Qwen3-4B; CE on actions, observation payload and observation template; cosine and norm ratio of the observation and action gradients, summed over tokens. On training trajectories the ratio is 1.09 for ActObs and 5.87 for ActionSFT.

I Selection and Headroom

Table 10 extends Table 2 to every model. Before RL, majority voting is the strongest rule that uses no knowledge. For every SFT model, Contract’s contract adds 0.9–2.6 points on top of it. After GRPO it no longer helps: contract plus vote is between 1.1 and 0.0 points below voting alone.

Table 11 breaks the wrong answers down by how their result differs from the gold result. A shape check can catch only wrong answers whose result has a different shape from the gold result. Before RL these are about 63–65% of wrong answers. After GRPO they are 44% on Spider and 52% on BIRD, mostly because GRPO already fixes the column count.

J Construction of the Inserted Lines

Outcome lines. Each outcome line is derived from the observation that follows the call. A result table gives “Expected result: N rows, M columns”, or “more than 50 rows” when SkyRL truncates the table. Other observations give “no rows”, “error

(message)” with the database path removed from the message, or “timeout”. Turns whose action is not a valid SQL call get no line.

Contracts and checks. A contract states the coarse shape of the gold result: one row or several rows together with the number of columns, or no rows. A check compares the coarse shape of the observed result with the contract; errors and timeouts never match. The placebo assigns each question the contract of another question through a seeded derangement, which preserves the distribution of contracts.

Re-tokenisation. We re-tokenise only the edited action segments. We then verify that removing the inserted lines restores the original text exactly, and no trajectory failed this check.

Statistics. Of the 6,294 SFT questions, 2,843 have a one-row gold result, 3,207 have several rows, 243 have no rows, and one gold query fails. The 24,043 SFT trajectories receive 24,039 contracts. Under Contract, 18,895 of the inserted checks report a match and 13,089 a mismatch. Under Contract-Placebo the derangement turns most checks into mismatches (6,035 matches and 25,949 mismatches), and 24.3% of the placebo contracts still coincide with the gold shape. The outcome lines shared by Outcome-Aux and Outcome-in-Action describe 28,216 results with rows, 2,883 empty results, 893 errors and 2 timeouts; 3,752 turns with an invalid action get no line. Table 12 compares Outcome-in-Action’s predicted outcome with the actual one in its own rollouts.

K Representative Transcripts

The three rollouts below come from Spider-dev; observation tags are abbreviated.

Part	Unseen (held-out) databases					Training databases				
	base	ActionSFT	ActObs	ActionSFT+Wiki	tokens	base	ActionSFT	ActObs	ActionSFT+Wiki	tokens
template	4.19	2.79	0.03	2.78	8,000	3.88	2.57	0.02	2.57	8,194
header (first 3 tokens)	7.89	5.64	1.02	5.71	1,101	7.26	5.31	0.35	5.38	1,128
header (rest of line)	3.40	2.22	0.30	2.18	1,986	2.38	1.69	0.20	1.65	1,924
row values	2.26	1.88	1.92	1.83	35,845	2.18	1.60	0.70	1.55	17,550
error (first 3 tokens)	10.15	6.73	0.53	6.33	15	6.61	4.48	0.41	4.22	57
error message	5.44	3.93	0.23	3.95	213	3.66	2.69	0.07	2.68	728
empty result (first 3 tokens)	4.28	5.21	1.21	4.49	9	3.34	4.97	0.37	4.45	129

Table 6: Observation cross-entropy (nats per token) by part on 300 held-out and 300 training trajectories. The “rows” parts also include invalid-action notices, which the breakdown does not separate.

GRPO run	Reward, steps				KL to	
	1–10	31–40	61–70	116–135	SFT	Entropy
ActionSFT, seed 1	0.436	0.577	0.627	0.693	0.067	0.426
ActionSFT, seed 2	0.436	0.566	0.629	0.688	0.082	0.446
ActObs, seed 1	0.445	0.576	0.623	0.692	0.107	0.474
ActObs, seed 2	0.440	0.592	0.634	0.700	0.100	0.472
ActionSFT-LR, seed 1	0.444	0.583	0.630	0.701	0.118	0.486
Contract, seed 1	0.430	0.569	0.625	0.696	0.072	0.436

Table 7: Mean training reward over step windows, and action KL to the run’s own SFT checkpoint and action entropy on 300 held-out trajectories. ActionSFT seed 1 is the only run on 4 GPUs.

Contract before RL revises after a mismatch. Question 728 (world_1) asks “How many people live in Gelderland district?”. The contract expects one row and the first result has four, so the agent switches to a sum and answers correctly. See Figure 6.

Contract after GRPO submits despite a mismatch. Question 361 (cre_Doc_Template_Mgt) asks to “show paragraph details for paragraph with text ‘Korea’”. The contract is correct: one row, one column. The agent nevertheless submits a query whose result it has just checked as not matching, and the answer is wrong. See Figure 7.

Outcome-in-Action predicts rows and gets an error. Question 660 (poker_player) asks to “return the names of all the poker players”. The prediction written before each call ignores what the call will do: the first call fails and the second returns 5 rows, yet both predictions say 10 rows. See Figure 8.

L Environment Audit

Empty-answer reward. We ran the unmodified environment, data and databases with the constant answer `<solution>SELECT 1 WHERE 0;</solution>`. It earns reward 1.0 on 290 of the 653 training tasks: 283 of the 540 SynSQL tasks and 7 of the 113 Spider tasks. The reward compares frozensets of rows, so an empty predic-

tion matches any empty gold result. The empty gold results arise in several ways. Some queries filter on a combination that does not exist in the data, such as `entry_level = 'H' AND time = '2023-2024'` together with a HAVING clause. Others are five-way joins whose filter matches no row of the synthetic database. Others again filter on `year > strftime('%Y', 'now') - 5`, which returns nothing at the time of writing; their result depends on the date on which the environment runs. Of the 22 SynSQL tasks that use dates relative to ‘now’, 18 currently return no rows. The Spider-dev split shipped with the same release has 49 empty golds among its 1,034 questions. We reported it to the maintainers with a reproduction script and the list of affected tasks (SkyRL issue #2451).

Thinking mode. Table 13 compares teacher pilots of 100 trajectories each, scored without the `<think>` rules. Under the shipped rules both thinking-off rows score zero, and the thinking-on row keeps only single-turn answers. With thinking on, the teacher mostly answers in one turn. When it does call the tool, it continues after the observation with a stray `</think>` or no tool tag, which the format check rejects.

M Pre-registration, Predictions and Deviations

The plan was frozen before any main SFT or GRPO run. It fixed the arms ActionSFT–

Model	Spider-dev					BIRD-dev				
	turns	err. traj.	solved	retry	empty→ans.	turns	err. traj.	solved	retry	empty→ans.
ActionSFT SFT	2.31	3.3	33.7	35.8	49.3	2.86	16.5	30.3	23.2	38.3
ActObs SFT	2.30	3.0	38.2	34.2	48.7	2.84	15.7	30.3	25.7	38.0
Outcome-Aux SFT	2.30	2.9	37.1	32.8	54.3	2.85	16.2	27.6	26.6	40.5
Outcome-in-Action SFT	2.33	3.3	36.3	38.2	38.8	2.88	16.7	29.4	27.8	33.7
Contract SFT	2.31	2.7	40.1	29.7	43.3	2.88	16.4	28.6	23.4	36.3
ActionSFT GRPO s1	2.09	2.4	52.8	6.0	52.4	2.34	13.5	37.7	7.3	37.5
ActionSFT GRPO s2	2.10	2.3	56.2	4.3	47.8	2.32	12.7	37.2	8.6	33.8
ActObs GRPO	2.08	2.5	55.2	4.7	49.6	2.30	13.6	37.5	6.0	38.4
ActionSFT-LR GRPO	2.10	2.8	56.7	3.9	43.0	2.33	13.9	38.8	8.1	31.0

Table 8: Behaviour after observations (all questions, 16 samples). Columns give mean turns; % of trajectories with an SQL error; % of those solved; % of failed queries re-issued unchanged; % of empty results followed directly by an answer.

Model	Spider-dev				BIRD-dev			
	P(sub m)	P(sub mm)	S	ΔS vs. ActionSFT	P(sub m)	P(sub mm)	S	ΔS vs. ActionSFT
<i>after SFT</i>								
ActionSFT	0.900	0.475	0.425	ref.	0.755	0.292	0.463	ref.
ActionSFT (new data order)	0.895	0.502	0.393	-0.033 [-0.050, -0.015]	0.739	0.296	0.443	-0.020 [-0.030, -0.009]
ActObs	0.900	0.486	0.413	-0.012 [-0.029, +0.005]	0.758	0.294	0.464	+0.001 [-0.008, +0.011]
ActionSFT-LR	0.899	0.477	0.421	-0.004 [-0.019, +0.011]	0.758	0.286	0.472	+0.009 [+0.001, +0.019]
Outcome-Aux	0.898	0.488	0.410	-0.016 [-0.030, +0.001]	0.758	0.296	0.462	-0.000 [-0.009, +0.009]
Outcome-in-Action	0.896	0.468	0.428	+0.003 [-0.013, +0.020]	0.739	0.279	0.460	-0.003 [-0.013, +0.008]
Contract	0.904	0.491	0.413	-0.012 [-0.029, +0.005]	0.753	0.283	0.469	+0.007 [-0.004, +0.017]
Contract-Placebo	0.901	0.481	0.420	-0.006 [-0.024, +0.012]	0.752	0.288	0.464	+0.001 [-0.009, +0.012]
<i>after GRPO</i>								
ActionSFT	0.982	0.643	0.339	ref.	0.936	0.408	0.529	ref.
ActionSFT (seed 2)	0.988	0.597	0.390	+0.052 [+0.024, +0.083]	0.949	0.402	0.547	+0.019 [+0.008, +0.029]
ActObs	0.990	0.654	0.336	-0.003 [-0.031, +0.026]	0.960	0.433	0.526	-0.002 [-0.015, +0.009]
ActionSFT-LR	0.986	0.577	0.410	+0.071 [+0.041, +0.100]	0.953	0.393	0.560	+0.032 [+0.021, +0.043]
Contract	0.990	0.691	0.300	-0.039 [-0.067, -0.012]	0.961	0.414	0.547	+0.019 [+0.008, +0.030]
ActObs (seed 2)	0.992	0.679	0.313	-0.025 [-0.054, +0.005]	0.961	0.464	0.497	-0.031 [-0.042, -0.020]

Table 9: Decision sensitivity. P(sub|m) and P(sub|mm) are the probabilities of submitting right after an observation whose coarse shape matches or does not match the gold result. S is their difference, and ΔS the difference from ActionSFT at the same stage, with 95% question-cluster bootstrap CIs (2,000 resamples).

ActionSFT+Wiki, the selection rule for ActionSFT-LR, the primary outcome, the hypotheses and the evaluation protocol. Every later change was recorded as a dated addendum, together with what had been seen at the time.

Three deviations concern implementation only. We changed the SFT implementation so that it fits in memory (a chunked LM head and ZeRO-1) without changing the objective, accumulated gradients on the CPU in the measurement code, and reran jobs with checkpointing disabled after the node’s local disk filled. Jobs were also limited to 2 GPUs, after ActionSFT seed 1 had already run on 4. Two deviations change the design. After seed 1 we put ActionSFT+Wiki and the third seeds on hold and added the outcome arms, the contract arms and a data-order seed for ActionSFT. Following a literature review, we queued the Outcome-in-Action GRPO run after Contract’s. The replication hypothesis required the same sign in two of three seeds. The two seeds that were run have opposite signs, and seed 3 was put on hold. Table 14 lists every

written prediction and its outcome.

N Systems Notes and Compute

Three engineering problems could have distorted the measurements. We describe each and how we fixed it.

Padding. With flash attention disabled, GRPO padded every batch to 20,350 tokens and took 53 minutes per step. Enabling flash attention, removing micro-batch padding and capping turns at 1,024 tokens brought this down to about 5–7 minutes on 2 GPUs. Evaluation scores did not change (Spider pass@1 0.551 vs. 0.550).

Memory. SFT ran out of memory on long trajectories. fp32 parameters, gradients and AdamW state took about 64 GB per GPU, and full-vocabulary logits took another 10 GB. ZeRO-1 and a chunked, recomputed LM head brought the worst-case peak down to 62–65 GB. A unit test checks that the chunked head gives the same loss and gra-

Samples from	Spider-dev						BIRD-dev					
	rand.	rule	vote	contr.	c.+v.	oracle	rand.	rule	vote	contr.	c.+v.	oracle
ActionSFT SFT	71.4	72.5	75.8	76.9	78.4	78.2	49.7	53.5	58.3	56.5	59.7	59.5
ActObs SFT	71.7	72.4	76.0	76.2	78.2	77.8	50.2	53.6	57.8	56.1	59.4	58.9
ActionSFT-LR SFT	71.3	72.3	75.3	76.6	78.0	78.1	50.3	53.5	58.0	56.8	59.9	59.5
Outcome-in-Action SFT	71.2	72.4	75.2	76.2	77.5	77.9	50.0	53.7	58.5	56.2	59.3	59.7
Contract SFT	71.8	72.7	75.9	77.0	78.2	78.6	49.5	53.6	58.0	57.0	60.1	60.3
Contract-Placebo SFT	71.5	72.5	75.7	76.6	77.9	78.1	49.6	53.4	58.3	56.9	60.0	60.3
ActionSFT GRPO s1	80.5	81.1	82.6	81.3	82.5	82.6	58.1	60.5	63.9	60.6	63.3	63.4
ActionSFT GRPO s2	80.3	80.7	82.3	80.9	82.2	82.0	58.2	60.3	63.4	60.2	62.7	62.5
ActObs GRPO	80.4	81.0	83.1	80.9	82.6	82.5	57.2	59.5	62.2	59.4	61.7	61.9
ActionSFT-LR GRPO	80.8	81.2	83.2	81.2	82.8	82.1	58.7	61.0	64.2	60.8	63.2	63.8
Contract GRPO	80.8	81.2	83.1	81.3	83.0	83.1	57.5	59.6	63.4	59.2	62.3	62.1

Table 10: Selecting one of 16 samples per question by: a random pick (= pass@1); the validity rule (no error, non-empty); execution majority vote; Contract’s modal contract; contract then vote; and the gold coarse shape (oracle). BIRD uses all 1,532 primary questions.

Model	Spider-dev							BIRD-dev						
	error	no answer	empty	#cols differ	1 vs. many rows	row count differs	same shape	error	no answer	empty	#cols differ	1 vs. many rows	row count differs	same shape
ActionSFT SFT	1.6	2.6	3.6	35.6	6.6	13.0	36.9	6.5	10.0	7.8	25.8	6.6	8.0	35.4
Outcome-in-Action SFT	1.7	3.7	3.5	34.9	7.0	12.7	36.6	5.8	9.9	7.8	25.5	6.9	8.8	35.4
ActionSFT GRPO s1	1.2	0.0	4.7	12.3	9.5	16.6	55.7	5.3	0.8	9.4	15.0	9.6	12.3	47.6
ActObs GRPO s1	1.4	0.0	3.5	11.9	8.8	17.8	56.5	5.6	0.5	9.8	14.4	9.1	12.2	48.4

Table 11: Wrong final answers by how their result differs from the gold result (% of wrong answers). Only “same shape” errors are invisible to any shape check.

	actual outcome			
	rows	no rows	error	timeout
predicts rows	54,497	4,051	8,026	425
predicts empty	689	408	276	36
predicts error	20	3	9	2
predicts timeout	0	0	0	0

Table 12: Outcome-in-Action’s predicted vs. actual outcome of each SQL call in its own pre-RL rollouts (Spider-dev and BIRD-dev).

Teacher pilot	Acc.	Multi-turn	Obs. share
Thinking on	0.52	19/100	1.4%
Thinking off	0.52	81/94	21.6%
Thinking off + verify prompt	0.49	96/97	26.9%

Table 13: Teacher pilots (Qwen3-32B). Accuracy is scored without the <think> rules; multi-turn counts are among well-formed trajectories.

dients as an unchunked one (App. B).

Disk and evaluator. SkyRL saves a full checkpoint at every epoch end whenever a checkpoint interval is set. Together with per-job environments, this filled a 1.8 TB local disk, and we reran the affected jobs with checkpointing disabled. Our evaluator agrees with SkyRL’s built-in one on a shared checkpoint.

Compute. All runs used NVIDIA RTX PRO 6000 Blackwell GPUs (96 GB). Excluding failed attempts, the study used about 280 GPU-hours. Most of it went to the 7 GRPO runs (about 185) and the 13 SFT runs (about 47). Evaluation took about 35 GPU-hours over roughly 25 evaluations, teacher generation and pilots about 8, and the measurements a few more.

O Hyper-parameters

SFT. We use AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$ and no weight decay. The learning rate is 10^{-5} with cosine decay and 3% warm-up. Each step uses a global batch of 64 trajectories (micro-batch 1 with gradient accumulation), and training runs for one epoch (375 steps) with gradient clipping at 1.0. The maximum length is 12,288 tokens. Weights are kept in fp32 with bf16 autocast, and seed 0 fixes the example order for every arm.

GRPO. We use SkyRL’s GRPO for 135 steps with 32 tasks and 16 samples per step, learning rate 10^{-6} and no KL penalty. Importance sampling is truncated at the token level with a cap of 2.0. Rollouts are limited to 6 turns, 1,024 tokens per turn and 8,192 input tokens, and are sampled at temperature 0.6 and top- p 0.95. Training runs FSDP on 2 GPUs with 2 vLLM engines.

Evaluation. See App. A.

```
Expected answer: 1 row, 1 column.
<sql>SELECT Population FROM city WHERE District = 'Gelderland';</sql>
[obs] 152463 / 101574 / 138020 / 153491
Check: got several rows, 1 column; this does not match the expected
answer. [...] Since the question asks for the total number of people
living in the district, we need to sum these values.
<sql>SELECT SUM(Population) FROM city WHERE District = 'Gelderland';</sql>
[obs] 545548
Check: got 1 row, 1 column; this matches the expected answer.
<solution>SELECT SUM(Population) FROM city WHERE District = 'Gelderland';
</solution>
```

Figure 6: Contract before RL (question 728): the check reports a mismatch and the agent revises.

```
Expected answer: 1 row, 1 column.
<sql>SELECT * FROM Paragraphs WHERE Paragraph_Text = 'Korea';</sql>
[obs] 510442723 2394 Korea None / 7 2394 Korea None
Check: got several rows, 4 columns; this does not match the expected
answer.
<solution>SELECT * FROM Paragraphs WHERE Paragraph_Text = 'Korea';
</solution>
```

Figure 7: Contract after GRPO (question 361): the contract is correct and the check reports a mismatch, yet the agent submits the wrong query.

P Artifacts and Licenses

Spider and BIRD are distributed under CC BY-SA 4.0. We obtained them, along with the SynSQL tasks in SkyRL’s training set, through the OmniSQL release. Qwen3-4B and Qwen3-32B are released under Apache 2.0, as is SkyRL. Our use is consistent with these licenses and with the research purpose of the datasets. We will release our code, analysis scripts, pre-registration and the list of affected environment tasks under Apache 2.0.

```

Expected result: 10 rows, 1 column.
<sql>SELECT p.Name FROM poker_player p JOIN people p2 ON ...;</sql>
[obs] Error executing SQL: no such column: p.Name
Expected result: 10 rows, 1 column.
<sql>SELECT p2.Name FROM poker_player p JOIN people p2 ON ...;</sql>
[obs] Aleksey Ostapenko / Yevgeni Sivozhelez / ... (5 rows)
<solution>SELECT p2.Name FROM poker_player p JOIN people p2 ON ...;
</solution>

```

Figure 8: Outcome-in-Action before RL (question 660): the in-action prediction ignores what the call will do.

Prediction (written before the result)	Outcome	Evidence
ActObs improves post-GRPO pooled pass@16 over ActionSFT	not found	+0.26 [−0.38, 0.91], mean of two seeds
$KL(\text{ActObs} \parallel \text{ActionSFT}) \leq KL(\text{ActionSFT-LR} \parallel \text{ActionSFT})$	refuted	0.0049 vs. 0.0010; both below data-order noise 0.0070
ActObs learns observation structure but not values on unseen DBs	supported	Table 6
ActObs’s observation gradient is smaller on training than on held-out data	supported	ratio 1.09 vs. 5.49
ActionSFT+Wiki _{obs} dilutes like ActionSFT+Wiki	not supported	KL 0.358, the same as ActionSFT-LR
Outcome-Aux leaves actions unchanged; Outcome-in-Action moves them	supported	KL to ActionSFT 0.0043 vs. 0.0162 (noise 0.0070)
Contract’s contract beats the majority shape and Contract-Placebo on unseen DBs	supported	89/82% vs. 42/60% and 30/35%
Contract submits less after a reported mismatch	supported, not attributable	ActionSFT shows the same gold-shape asymmetry (Table 9)
Contract’s pre-RL pass@1 equals ActionSFT’s	supported	71.8/49.5 vs. 71.4/49.7
Contract’s own-contract selection beats the validity rule on BIRD	supported	57.4 vs. 53.9
Contract gains at least 1 point of BIRD pass@1 over ActionSFT after GRPO	refuted	−0.60 [−1.17, −0.03]
GRPO lowers Contract’s submission after a mismatch	refuted	rises from 31% to 46% (BIRD)

Table 14: Predictions written before each result, and their outcomes.