RELATIONAL LEARNING AS WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Relational learning typically predicts from a database state, but decision making asks a different question: how will that state change under an action? In operational systems, the answer depends on system-specific dynamics that may not be visible in the database itself. We view each such system as a relational "world" and ask whether a model can infer how it behaves from observed executions. We introduce **TabWM** (**Tab**ular **W**orld **M**odel) and WORLDREL, a generator of executable relational worlds for pretraining. Given executions from an unseen world, TabWM uses them as context to predict the sparse structured edits caused by a new action without parameter updates. Across 512 held-out worlds, complete-edit F1 rises from 0.016 with no observed executions to 0.685 with 64, where a correct edit must match its operation, relational location, and value. Because these predictions are explicit relational edits, they can be applied recursively to imagine candidate futures and compare actions before they are taken. Broader search over these predicted futures improves the realized return of selected action sequences. Together, these results suggest that relational dynamics can be learned in context, broadening relational learning from modeling system state to modeling how that state changes under action.

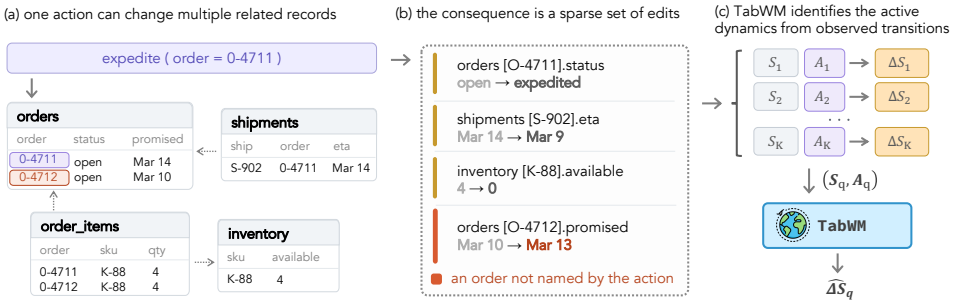

Figure 1: Relational learning as world modeling. **(a)** An action can affect both referenced and linked records. **(b)** Its consequences form a sparse edit set. **(c)** TabWM uses observed executions from the same world to predict edits for a new action without parameter updates.

1 INTRODUCTION

Relational foundation models are rapidly expanding what a single pretrained model can learn from databases. Recent models transfer across unseen schemas and tasks (Ranjan et al., 2026a; Hudovernik et al., 2026), scale through synthetic relational pretraining (Kothapalli et al., 2026), and adapt to new prediction problems from context (Wang et al., 2026; Ranjan et al., 2026b). Yet these advances largely share the same learning interface, using a relational database state S to predict a target y. This formulation treats the database primarily as a dataset. In many applications, however, the database is also the observable state of a running system, shaped by users, software, and operational processes (Klein & Hoffart, 2025; 2026). Choosing an action requires asking how it would change that state. This calls for an action-conditioned transition model from (S_t, A_t) to S_{t+1}, the core abstraction of world modeling that allows a model to imagine how a system evolves under candidate actions (Ha & Schmidhuber, 2018; Ding et al., 2025; Hafner et al., 2025). We ask whether relational learning can make this shift from prediction to *world modeling*.

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.

Consider an order that is predicted to arrive late. Knowing that it will be late does not tell us what will change if we expedite it. In Figure 1a, expediting O-4711 updates its order and shipment, consumes the remaining inventory, and pushes back O-4712 although the action does not name it. Similar what-if questions arise in prescriptive process monitoring and business-process simulation (Dasht Bozorgi et al., 2023; Camargo et al., 2020; 2022; Chapela-Campa et al., 2025). We represent these direct and propagated changes as a sparse set of structured edits ΔS_t (Figure 1b). Each edit specifies an operation, its relational location, and a value when needed. Applying the edits updates the affected records without regenerating unchanged data. The action specifies what is done, but the system dynamics determine where its effects propagate and what values result.

These dynamics depend on operational rules that may not be apparent from the database records (Klein & Hoffart, 2025; 2026). We call a relational system with a particular schema, set of actions, and transition mechanism a *world*. Its executions are observable, but we assume its mechanism is hidden. In a controlled experiment across 200 pairs of worlds, executing the same action from identical relational states produces different structured edits in 65% of probes. The current state and action therefore leave the consequence ambiguous across worlds. Past executions provide evidence about how actions behave in the current world. Intuitively, *states show what a system looks like, while transitions show how it behaves*.

Learning from these executions is often approached by discovering or calibrating a simulator for a particular process (Camargo et al., 2020; 2022; Chapela-Campa et al., 2025). In-context system identification offers another route. A model trained across a class of dynamical systems can infer the behavior of a new one from its observed history (Forgione et al., 2023; Rufolo et al., 2025). We ask whether the same idea can be extended to relational systems. Instead of fitting a separate model for every new world, a single pretrained model would use observed executions as context to predict the outcome of a new action in that world (Figure 1c).

We realize this idea with **TabWM**[1] (**Tab**ular **W**orld **M**odel), trained across executable relational worlds generated by WORLDREL. WORLDREL extends the PLUREL relational prior (Kothapalli et al., 2026) with executable actions and world-specific transition mechanisms. Within a task, the observed executions and query share the same dynamics, while those dynamics vary across tasks. This extends prior-data fitted learning (Müller et al., 2022; Wang et al., 2026) from static relational prediction to action-conditioned transitions. TabWM preserves relational structure and the correspondence between actions and their observed consequences. It then predicts the consequence of a new action as a sparse set of edits that specify what changes, where it changes, and what value is written. At test time, the model adapts to a new world through observed executions without updating its parameters.

Across 512 held-out worlds, complete-edit F1 rises from 0.016 with no behavioral context to 0.685 with 64 observed executions, where a correct edit must match its operation, relational location, and value. Context remains useful when action mappings, schemas, and mechanism compositions change, and the same pattern persists when relational structure and values are derived from real databases. A single pretrained model can therefore improve its predictions from experience in a new world without updating its parameters.

A world model is useful not only for predicting what happens next, but for deciding what to do next. Because TabWM predicts explicit edits, it can roll candidate actions forward into future relational states and compare the resulting trajectories through model-based planning (Hafner et al., 2019; Chua et al., 2018). At a fixed three-step horizon, expanding the candidate pool from 16 to 64 improves the selected return in 30 of 40 held-out worlds. This gives TabWM two ways to improve at test time. More observed executions improve consequence prediction, while broader search over predicted futures improves action selection.

2 PROBLEM SETUP

2.1 RELATIONAL WORLDS

We study a relational system whose execution rules are hidden from the learner. Its structure and dynamics define a world

$$w = (\Sigma_w, \mathcal{A}_w, \rho_w, T_w), \tag{1}$$

[1]Anonymous code is available at `https://anonymous.4open.science/r/TabWM-4EF2/`.

where Σ_w specifies the tables, columns, and relational links, $\mathcal{A}_w$ the available actions, ρ_w the distribution of initial execution states, and T_w the transition mechanism. These components remain fixed while the database state changes through execution.

An action A_t specifies its type and arguments, which can reference records or supply values (Pasula et al., 2007; Martínez et al., 2016). The database is the learner's observation of the system, not its complete execution state. We write $S_t = \mathrm{Obs}(X_t)$, where X_t also contains any unobserved information used during execution. Starting from $X_0 \sim \rho_w$, executing an action produces

$$\Delta S_t \sim T_w(\cdot \mid X_t, A_t). \tag{2}$$

The consequence ΔS_t is a variable-size set of edits that update numeric or categorical attributes, insert records, or delete records. Each edit specifies an operation, its relational target, and any required values or record contents. Numeric updates specify resulting values, not increments. The target records the changes caused during action execution, excluding separately generated external changes. It is empty when execution produces no edits.

2.2 Predicting from Behavioral Context

The learner receives a behavioral context $\mathcal{C}_w^K = \{(S_i, A_i, \Delta S_i)\}_{i=1}^K$ of K observed transitions and a query (S_q, A_q) from the same world. The demonstrations include observed consequences, while the query consequence ΔS_q is withheld. The learner predicts

$$\widehat{\Delta S}_q = f_\theta(\mathcal{C}_w^K, S_q, A_q). \tag{3}$$

Demonstrations and queries share dynamics but come from separate episodes, each starting from a fresh initial state $X_0 \sim \rho_w$. Context therefore provides evidence across executions, not the query's own history. The parameters θ are learned across worlds and held fixed for an unseen test world (Forgione et al., 2023; Wang et al., 2026). Adaptation occurs through the observed transitions, without access to executable rules or complete execution states. We first test why this evidence is needed by holding the query state and action fixed while changing the execution rules.

3 Same State, Different Consequences

Similar database states can arise under different operational rules (Klein & Hoffart, 2026). We test whether this ambiguity remains when the observed state and action are both identical.

We pair worlds that share a schema, initial-state distribution, observable action definitions, and set of transition mechanisms. We change only which mechanism each action invokes. We sample 200 pairs without filtering and draw 64 matched initial states per pair, checking that all database tables agree elementwise. Each probe executes the same valid action, including its arguments, in both worlds with the same random seed, giving $(S_0, A) \xrightarrow{w_0} \Delta S^{(0)}$ and $(S_0, A) \xrightarrow{w_1} \Delta S^{(1)}$. We compare the complete edit sets, including each edit's operation, relational target, and required values or record contents. The observed states match in all 12,800 paired probes, yet the edit sets differ in 65.0% of them. Figure 2 shows how this disagreement rate varies across world pairs.

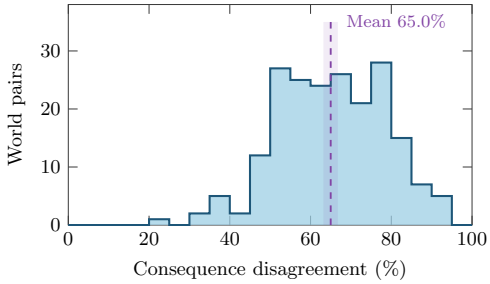

Figure 2: Edit-set disagreement under identical states and actions. The dashed line marks the mean across world pairs (65.0%), with shading indicating bootstrap uncertainty in the mean.

Observed transitions link actions to the changes they produced in the current world. They provide evidence about the execution rules that the state alone does not reveal. We next build a model that learns across worlds to use this evidence when predicting a new execution.

4 TabWM: A Tabular World Model for Relational Systems

The ambiguity in §3 turns consequence prediction into an identification problem: the same observed state and action can behave differently across worlds, so a model must use previous executions to

infer which dynamics are active before predicting a new one. TabWM learns this inference across relational worlds while predicting the consequence itself rather than the hidden transition mechanism. Its training distribution varies dynamics across worlds, its encoder preserves the behavioral evidence within each observed execution, and its decoder converts that evidence into a sparse set of structured edits. Because these edits define the resulting relational state, the same model can also be applied recursively to imagine subsequent actions.

4.1 Learning Across Relational Worlds

Learning to identify a new world's dynamics from context requires training tasks in which behavior remains consistent within a world and varies across worlds. We construct such tasks with WorldRel, which extends the relational prior of PluRel (Kothapalli et al., 2026) with executable actions and world-specific transition mechanisms. Each sampled world fixes a relational schema, an action interface, an initial-state distribution, and a transition mechanism. Observed executions and the query are drawn from separate episodes of the same world, while worlds are resampled across tasks. § E describes this executable-world prior in detail.

The query consequence provides the prediction target, while the observed executions reveal which behavior applies in the current world. Following prior-data fitted learning (Müller et al., 2022), training across these tasks teaches a single model to use behavioral context instead of relying on one fixed action–consequence mapping. This amortizes action-conditioned system identification across worlds (Forgione et al., 2023). At test time, TabWM adapts to an unseen world through its observed executions without updating its parameters.

4.2 Encoding Relational Transitions

Using executions as evidence requires preserving more than the database state alone. The representation must retain how records are linked, which action and consequence belong to the same execution, and how an observed value changed before and after that action. We initialize TabWM's relational encoder from the pretrained Relational Transformer (Ranjan et al., 2026a), using weights learned from PluRel-generated relational databases (Kothapalli et al., 2026). This provides an initial representation of relational structure, which TabWM adapts together with its transition-specific components to action-conditioned prediction.

For each execution, we collect an action-centered neighborhood by following primary- and foreign-key links from the records referenced by the action, subject to a cell budget. Cells retain their column identity, type, and value, while textual identifiers use deterministic hash embeddings rather than a pretrained language encoder. The action is represented as a pseudo-row linked to the records bound by its arguments, with scalar and categorical arguments represented as action cells.

We encode the correspondence within each execution. Demonstrated edits are attached to the records they modify. Role embeddings distinguish state, action, and consequence cells, while transition-index embeddings mark which elements belong to the same execution. For observed updates, shared table, record, and column identities align the before-state cell with its demonstrated edit. Together, these connections preserve relational structure, execution membership, and the before-and-after evidence needed to interpret the query. Packing the behavioral context and query gives

$$\mathcal{U}_q = \mathrm{Pack}(\mathcal{C}_w^K, S_q, A_q), \qquad H_q = \mathrm{Enc}_\theta(\mathcal{U}_q), \tag{4}$$

where the query consequence is withheld and H_q contains the resulting contextual representations.

4.3 Decoding Action Consequences

The encoder represents the query together with executions that may explain its behavior; the remaining problem is to predict a variable-size consequence. An action may produce several edits or none, so TabWM predicts its consequence as an unordered set. Following the set-prediction approach of DETR (Carion et al., 2020), the decoder uses M learned edit queries. Each query attends to H_q and predicts an existence indicator $\hat{z}_j$, an operation $\hat{o}_j$, a relational target $\hat{\ell}_j$, and any required value or record contents $\hat{v}_j$. The active slots form

$$\widehat{\Delta S}_q = \left\{ (\hat{o}_j, \hat{\ell}_j, \hat{v}_j) \mid 1 \le j \le M,\ \hat{z}_j = 1 \right\}. \tag{5}$$

Pointers select affected cells, rows, or tables as required by the operation, while typed output heads produce categorical values and insertion contents.

Locating an edit does not yet determine the value to write. For a predicted numeric target, TabWM reads visible before-and-after pairs (x_r, y_r) from previous executions, conditioned on the query action and target. Each pair links a value before an observed action to the value written by its consequence. A neural head produces a learned estimate $\hat{y}_{\text{learned}} = \hat{y}_0 + r_\theta$, while the same evidence is also used to infer transformations that can transfer to a query with a different starting value.

For a target whose current value is x_q, the decoder constructs additive and affine candidates,

$$h_{\text{add}} = x_q + \sum_r \alpha_r (y_r - x_r), \qquad h_{\text{aff}} = \bar{y} + \beta(x_q - \bar{x}), \tag{6}$$

where α_r are learned attention weights over relevant before-and-after pairs, $\bar{x}$ and $\bar{y}$ are their weighted means, and β is an affine slope estimated from the selected evidence. The additive candidate transfers an observed increment, while the affine candidate estimates how the outcome varies with the starting value. Both transfer a transformation to the query rather than copying a demonstrated outcome.

Learned gates combine these candidates with the neural prediction,

$$\hat{y} = \hat{y}_{\text{learned}} + g_{\text{add}}(h_{\text{add}} - \hat{y}_{\text{learned}}) + g_{\text{aff}}(h_{\text{aff}} - \hat{y}_{\text{learned}}). \tag{7}$$

The resulting value completes the predicted edit without changing its operation or target. This separation lets us isolate numeric transfer from relational effect localization in §F.5. Further details on evidence selection and candidate construction appear in §C.3.

Minimum-cost bipartite matching associates predicted slots with reference edits during training (Carion et al., 2020). Matched slots are supervised on their operations, targets, and contents, while unmatched slots learn to emit no edit.

Explicit edit predictions are also directly composable. After an imagined action, the predicted edits are applied to obtain the next relational state, which can then serve as the input to another action query:

$$\widehat{\Delta S}_t = f_\theta(\mathcal{C}_w^K, \widehat{S}_t, A_t), \qquad \widehat{S}_{t+1} = \text{Apply}(\widehat{S}_t, \widehat{\Delta S}_t), \qquad \widehat{S}_0 = S_0. \tag{8}$$

The behavioral context remains fixed while the predicted state evolves along the action sequence. These rollouts produce the candidate futures used for action selection in §8.

5 Inferring Dynamics in Unseen Worlds

If observed executions reveal how actions behave in a world, prediction should improve as more executions from that world enter context, without changing the model parameters. We test this by varying the amount of behavioral context available to TabWM.

Evaluation. Each task provides K observed state–action–consequence transitions and a separate query from a held-out WorldRel world. The context sweep uses 250 worlds and three queries per world at each $K \in \{0, 1, 2, 4, 8, 16, 32, 64\}$. Support F1 measures whether the predicted edits have the correct operations and relational locations, while Edit F1 additionally requires their written values to be correct. Value MAE measures numeric error once the correct location has been identified. The numeric tolerance and remaining scoring details are given in §B.

Observed executions improve both effect localization and complete-edit prediction, while numeric error on correctly located edits falls (Figure 3). Edit F1 reaches 0.705 with 64 executions, up from 0.029 without behavioral context. Edit F1 still improves between 32 and 64 observations, so the context gain has not saturated by 32 executions.

We next ask whether other model classes can exploit the same evidence. Flat-GBM and Flat-TabPFN reduce each task to fixed-width candidate-cell features and summaries of observed changes, then predict edits using gradient boosting (Friedman, 2001) or TabPFN (Hollmann et al., 2025). Qwen3 (Yang et al., 2025a) instead receives a textual serialization of the observed executions and query, testing whether a general-purpose LLM can infer the current dynamics from the same evidence.

Readable identifiers provide Qwen3 with an additional source of information because table, column, categorical, and action names may carry familiar linguistic cues. We therefore evaluate matched readable and opaque renderings of the same tasks. In the *opaque* view, these identifiers are replaced consistently with arbitrary labels while preserving relational structure, row identities, values, types, and action arguments. The *readable* view restores the original identifiers. TabWM uses deterministic hash embeddings, so readable strings do not provide pretrained linguistic semantics. All methods are evaluated on the same 60 held-out worlds with one query per world at each $K \in \{0, 4, 16, 64\}$, and Qwen3 and TabWM receive the same action-centered cell budget.

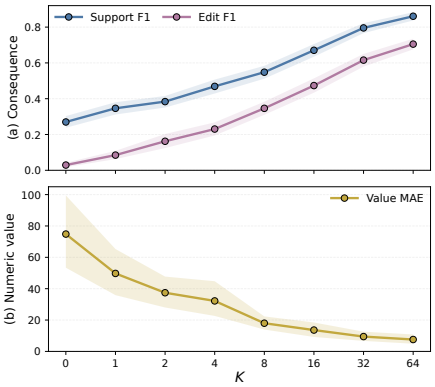

Figure 3: More observed behavior improves action–consequence prediction. **(a)** Support F1 and Edit F1 with increasing behavioral context. **(b)** Numeric MAE on correctly located edits. Shading shows 95% world-bootstrap confidence intervals.

Under opaque naming, TabWM leads at every context size and reaches 0.687 Edit F1 with 64 observed executions, compared with 0.407 for Qwen3-32B (Table 1). The advantage is already visible in effect localization, where Support F1 is 0.887 for TabWM and 0.753 for Qwen3-32B under the same condition. Restoring readable names benefits Qwen3 substantially more than TabWM, but the complete-edit gap remains. Readable semantics therefore narrow the gap for the LLM, but do not account for TabWM's improvement from behavioral context.

Table 1: Complete-edit prediction (Edit F1, ↑) under opaque and readable naming cues. Readable names expose schema and action semantics. Best scores in each column are bold. †Flat baselines do not consume table or column names, so their inputs are unchanged between the two views.

Model	# Params	Opaque names				Readable names			
		K=0	K=4	K=16	K=64	K=0	K=4	K=16	K=64
Flat-GBM†	–	0.000	0.005	0.000	0.020	0.000	0.005	0.000	0.020
Flat-TabPFN†	7M	0.000	0.016	0.039	0.106	0.000	0.016	0.039	0.106
Qwen3	8B	0.000	0.043	0.096	0.191	0.020	0.140	0.311	0.373
Qwen3	14B	0.000	0.067	0.230	0.266	0.011	0.128	0.357	0.398
Qwen3	32B	0.012	0.127	0.320	0.407	0.028	0.187	0.436	0.527
TabWM	≈24.7M	**0.043**	**0.302**	**0.524**	**0.687**	**0.029**	**0.302**	**0.518**	**0.678**

6 Identifying the Active Dynamics

Improvement from demonstrations alone does not establish that a model has inferred how actions behave in the current world. In-context learners can benefit from demonstrations even when their labels are randomized (Min et al., 2022). If TabWM is using executions as behavioral evidence, however, preserving which action produced which consequence should be essential.

We test this using the paired worlds from §3, where an identical observed state and action can produce different consequences under different hidden dynamics. TabWM receives either intact transitions or corrupted versions of the same observed executions. Shuffling the actions preserves the states and consequences but assigns each consequence to a different action. Shuffling the consequences preserves the states and actions but attaches the wrong response to each execution. Both interventions retain the observed components while breaking their correspondence.

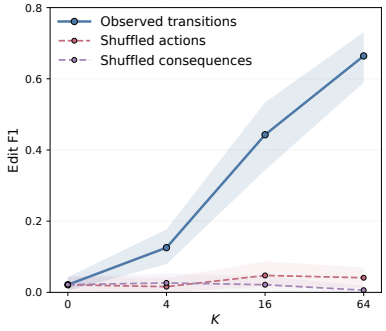

Figure 4: Behavioral context helps only when action–consequence correspondence is preserved.

With 64 observed executions, intact transitions yield 0.664 Edit F1, compared with 0.041 after shuffling actions and 0.006 after shuffling consequences (Figure 4). The context gain therefore depends on observing which consequences followed which actions, rather than simply on seeing additional database states or edit sets.

Which executions are most informative for a query? At fixed $K = 16$, Edit F1 rises from 0.048 when the queried action is absent from context to 0.403 with one execution of that action and 0.700 with four. This gain is not explained by replaying previous consequences. An explicit per-action template-transfer baseline reaches only 0.169 and 0.216 with one and four matching executions. TabWM therefore uses repeated executions of the queried action to infer how its effect transfers to a new relational state (§ F.3).

We next vary which part of the world changes relative to the pretraining setting. A *held-out state* changes the database state while leaving the surrounding world construction unchanged. A *permuted action map* retains the action vocabulary but reassigns which transition mechanism each action invokes. An *unseen schema* changes the relational structure, while a *held-out mechanism composition* changes which familiar mechanisms occur together in one world.

The permuted action map is particularly diagnostic because the action identities remain unchanged while their effects do not. Edit F1 rises from 0.378 with 16 executions to 0.604 with 64, showing that behavioral context can override the mapping suggested by prior experience and recover the remapped action behavior. Additional executions also improve prediction under the other shifts (Table 2).

Table 2: Edit F1 under changes to the evaluated world.

	$K = 16$	$K = 64$
Held-out state	0.483	0.644
Permuted action map	0.378	0.604
Unseen schema	0.485	0.687
Held-out composition	0.465	0.698

These shifts still draw their transition mechanisms from the pretraining library. Appendix F.4 tests a stricter boundary with mechanisms introduced only at evaluation time. Behavioral context continues to improve effect localization, while complete-edit transfer depends more strongly on the quantitative transformation.

7 Transfer to Real-Derived Structure and Values

The experiments above vary the active dynamics. We next vary the relational observations through which those dynamics must be inferred. Relational topology and value distributions are derived from real databases while the action and transition mechanism remain fixed.

We first replace the relational structure. Table and primary- and foreign-key topology are imported from 139 databases in THE JOIN (Ranjan et al., 2026b) and paired with 139 size-matched synthetic references under the same protocol. With 64 observed executions, Edit F1 is 0.657 on the imported topologies and 0.671 on the matched synthetic worlds. Replacing relational connectivity alone therefore causes little loss.

We then change the values populating those structures. We derive empirical value distributions from 241 columns across six RelBench databases (Robinson et al., 2024) and map their marginal shapes into the ranges supported by the generator. A separate 200-world evaluation compares synthetic and real-derived values, first under synthetic topology and then with real-derived topology as well.

The larger transfer gap appears when the value distributions change (Table 3). Edit F1 decreases under real-derived values, and Support F1 falls as well, showing that the shift affects where consequences are predicted to occur rather than only the reconstruction of numeric values. More behavioral context remains beneficial in every condition. Adding real-derived topology to the value shift produces little further degradation. Across these observation shifts, value distributions therefore matter more than relational connectivity.

Table 3: Prediction under real-derived value distributions and relational topology.

Topology	Values	Edit F1		Support F1	
		$K = 16$	$K = 64$	$K = 16$	$K = 64$
Synthetic	Synthetic	0.481	0.696	0.675	0.875
Synthetic	Real-derived	0.355	0.522	0.614	0.789
Real-derived	Real-derived	0.365	0.497	0.661	0.767

8 From Predicted Consequences to Decisions

Comparing actions requires more than predicting one consequence at a time. A useful world model must imagine possible futures and distinguish which are preferable before acting. We therefore use TabWM as a rollout model: predicted edits are applied recursively along candidate action sequences, and the resulting trajectories are scored with the known task objective (Hafner et al., 2019; Chua et al., 2018). We first ask whether these imagined futures preserve the ordering induced by the true system, and then whether that ordering leads to better choices when more alternatives are available.

8.1 Ranking Candidate Futures

At each decision state, we sample a shared pool of 64 legal three-step action sequences. TabWM rolls each sequence forward and scores the imagined trajectory using the known world-specific objective, which combines changes in state potential with action costs (§B.4). We execute the same sequences in the true simulator to obtain their realized returns. The evaluation contains 160 decision states from 40 held-out worlds, with states and candidate pools shared across context sizes. Spearman correlation is computed within each 64-sequence pool and averaged across decision states. Normalized regret measures, within the same pool, the realized-return gap between the selected candidate and the best available candidate, divided by the pool's return span.

To separate information about state changes from simpler sources of ranking signal, we use three references. The *simulator oracle* rolls candidates forward with the true transition mechanism rather than TabWM. The *shuffled-context* control mismatches demonstrated actions with their observed consequences before prediction, preserving the amount of context while breaking its behavioral correspondence. The *no-change* control keeps the predicted state fixed, leaving action costs as the only source of variation between sequences.

With intact behavioral context, TabWM's imagined ranking becomes increasingly consistent with realized returns (Table 4). With 64 observed executions, its Spearman correlation reaches 0.622, compared with 0.338 after shuffling the action–consequence correspondence and 0.345 when state changes are ignored altogether. The simulator oracle reaches 0.732. The contrast with the two controls shows that the ranking gain comes from information about how candidate actions change the current world, rather than only from their immediate costs.

Table 4: Candidate ranking under TabWM and reference rollouts.

Scorer	K	Spearman ρ	Normalized regret
Simulator oracle	–	0.732	0.107
No-change	–	0.345	0.393
Shuffled context	64	0.338	0.392
TabWM	4	0.444	0.298
	16	0.590	0.208
	64	0.622	0.207

The full ranking continues to improve beyond 16 executions, but most of the gain in the selected candidate has already appeared. Normalized regret falls from 0.298 with four executions to 0.208 with 16, and changes little thereafter. One-step consequence prediction, by contrast, continues to improve at larger context sizes. In this planning task, the behavioral evidence needed to distinguish good candidates therefore appears to saturate earlier than the evidence needed to reconstruct their consequences in detail. This separation echoes prior observations that predictive fidelity and control utility need not improve in lockstep (Lambert et al., 2020; Grimm et al., 2020; Zhang et al., 2026).

8.2 Choosing Among More Futures

Once candidate futures can be ranked meaningfully, search determines how much that ranking can be exploited. We fix the behavioral context and reveal progressively larger nested subsets of the same 64 legal sequences, so broader search adds choices without changing those already available.

Table 5: Median realized return of selected sequences as the candidate budget grows at $K = 16$.

	$N = 1$	$N = 4$	$N = 16$	$N = 64$
Simulator oracle	−1.019	−0.348	−0.078	0.285
TabWM	−1.019	−0.462	−0.323	−0.165
Shuffled context	−1.019	−0.668	−0.513	−0.412
No-change	−1.019	−0.646	−0.508	−0.398

The common starting point in Table 5 is a useful sanity check: before search creates a choice, every scorer must agree. As the candidate pool grows, the rankings begin to matter. TabWM consistently selects higher-return sequences than either control, while the simulator oracle shows the

remaining value of more accurate dynamics. The controls also improve because action costs alone induce a weak preference over candidates.

This advantage is not driven only by a few large gains. Across the final expansion of the candidate pool, TabWM improves the selected return in 30 of 40 held-out worlds, compared with 21 under shuffled context and 22 under no-change dynamics. Taken together, the two planning experiments show that behavioral context and search play complementary roles. Executions provide an informative ordering over candidate futures, while broader search gives that ordering more opportunities to improve the decision.

9 Related Work

Relational foundation models. Relational learning has recently moved from task-specific models toward reusable models that transfer across databases and prediction tasks. Griffin (Wang et al., 2025), the Relational Transformer (Ranjan et al., 2026a), and KumoRFM-2 (Hudovernik et al., 2026) learn transferable representations directly over relational data, while synthetic pretraining and in-context formulations further support adaptation across databases (Kothapalli et al., 2026; Wang et al., 2026; Hayler et al., 2026). Despite these different approaches, the learning interface remains largely predictive, with relational context used to infer targets or properties of database records. TabWM changes what is inferred from context: observed executions provide evidence about how actions change those records.

Learning dynamics from behavioral context. Once the target becomes an action-conditioned transition, our setting connects to work on learning action effects and identifying dynamics from experience. Relational action models learn structured or stochastic effects for planning (Pasula et al., 2007; Martínez et al., 2016), while context-conditioned dynamics models adapt prediction from recent transitions (Lee et al., 2020). Prior-data fitted learning provides a broader route to amortizing such inference across tasks (Müller et al., 2022), including models trained across synthetic dynamical systems that infer new behavior from input-output histories (Forgione et al., 2023; Rufolo et al., 2025). TabWM brings this form of identification to relational systems, where an action can induce sparse changes that propagate across linked records.

World models for operational systems. In operational systems, the transition mechanism itself may not be apparent from the values stored in the database (Klein & Hoffart, 2025; 2026). Enterprise workflow benchmarks expose cascading action effects (Gupta et al., 2026), while process simulators recover executable models from event logs for what-if analysis (Chapela-Campa et al., 2025). When the active configuration is readable, runtime discovery provides another way to ground predictions in the current system (Nair et al., 2026). TabWM studies the complementary case in which the mechanism is hidden but its executions are observable, allowing behavior in the current world to be inferred from context.

10 Conclusion

Relational databases can describe not only what a system looks like, but also how it changes. TabWM learns from observed executions to predict sparse relational consequences of new actions, adapting through context rather than parameter updates. Because these predictions are explicit edits, they can be composed into candidate futures and used to compare actions before they are taken. More broadly, our results suggest that relational dynamics can be learned in context. Pretraining across executable worlds allows a single model to infer action consequences from executions in a new world.

Limitations & Future Work. Moving from relational prediction to relational world modeling also changes what is required from an evaluation benchmark. Studying behavior from experience requires aligned observations of the relational state before an action, the action itself, and the resulting changes, which current relational benchmarks rarely expose. This motivates our use of executable worlds and suggests a natural next step in building datasets around real operational execution traces. Such benchmarks would enable broader evaluation across systems and open the study of how relational models infer, track, and use dynamics from experience.

AI USE STATEMENT

In this work, we used generative AI tools primarily for coding assistance and editing the paper to improve clarity and readability. For tasks with required disclosure, we used generative AI tools to provide feedback on experimental design and presentation. We have not used generative AI tools to propose or refine hypotheses, develop research methodology, implement methods, clean or reformat datasets, or interpret results, and tasks involving the generation of synthetic datasets, development of theoretical models or conceptual frameworks, formulation or proof of mathematical claims, translation, and qualitative or thematic data analysis are not applicable to this work. We have not used generative AI tools for other tasks with recommended disclosure. We have reviewed all AI-assisted work. AI-assisted code was reviewed and tested by the authors, and all AI-assisted edits and feedback were reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Automated discovery of business process simulation models from event logs. *Decision Support Systems*, 134:113284, 2020.

Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. Learning accurate business process simulation models from event logs via automated process discovery and deep learning. In *International Conference on Advanced Information Systems Engineering*, pp. 55–71. Springer, 2022.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

David Chapela-Campa, Orlenys López-Pintado, Ihar Suvorau, and Marlon Dumas. Simod: automated discovery of business process simulation models. *SoftwareX*, 30:102157, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Zahra Dasht Bozorgi, Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Prescriptive process monitoring based on causal effect estimation. *Inf. Syst.*, 116(C), June 2023. ISSN 0306-4379. doi: 10.1016/j.is.2023.102198. URL https://doi.org/10.1016/j.is.2023.102198.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, Fengli Xu, and Yong Li. Understanding world or predicting future? a comprehensive survey of world models. *ACM Comput. Surv.*, 58(3), September 2025. ISSN 0360-0300. doi: 10.1145/3746449. URL https://doi.org/10.1145/3746449.

Marco Forgione, Filippo Pura, and Dario Piga. From system models to class models: An in-context learning paradigm. *IEEE Control Systems Letters*, 7:3513–3518, 2023. doi: 10.1109/LCSYS.2023.3335036.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in neural information processing systems*, 33: 5541–5552, 2020.

Lakshya Gupta, Litao Li, Yizhe Liu, Sriram Ganapathi Subramanian, Kaheer Suleman, Zichen Zhang, Haoye Lu, and Sumit Pasupalak. World of workflows: a benchmark for bringing world models to enterprise systems. *arXiv preprint arXiv:2601.22130*, 2026.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/hafner19a.html.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025.

Adrian Hayler, Klemens Flöge, Alan Arazi, Rishabh Ranjan, Jure Leskovec, Felix Birkel, Brendan Roof, Anurag Garg, Kristina Collins, Lydia Sidhoum, Jonas Kübler, Siyuan Guo, Oscar Key, Jan Hendrik Metzen, Rylee Grace, David Salinas, Arthur Cahu, Simon Bing, Benjamin Jäger, Tuana Çelik, Mihir Manium, Vitor Monteiro, Jake Robertson, Jerry Chen, Eliott Kalfon, Tomás Pereda, Lilly Wehrhahn, Dominik Safaric, Tobias Schroeder, Georg Grab, Diana Kriuchkova, Clara Cornu, Philipp Singer, Nick Erickson, Vahid Balazadeh, Marie Salmon, Simone Alessi, Kürşat Kaya, Philipp Jund, Léo Grinsztajn, Yann LeCun, Bernhard Schölkopf, Madelon Hulsebos, Lennart Purucker, Sauraj Gambhir, Frank Hutter, and Noah Hollmann. Advancing open and reproducible relational learning: Relarena-α, tabpfn-rel and rpi, 2026. URL https://arxiv.org/abs/2608.16319.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Valter Hudovernik, Federico López, Vid Kocijan, Akihiro Nitta, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. Kumorfm-2: Scaling foundation models for relational learning. *arXiv preprint arXiv:2604.12596*, 2026.

Tassilo Klein and Johannes Hoffart. Position: Foundation models for tabular data within systemic contexts need grounding. *arXiv preprint arXiv:2505.19825*, 2025.

Tassilo Klein and Johannes Hoffart. Statistically indistinguishable, operationally distinct: A formal barrier for tabular foundation models. In *2nd ICML Workshop on Foundation Models for Structured Data*, 2026. URL https://openreview.net/forum?id=TUYc2XUdwz.

Vignesh Kothapalli, Rishabh Ranjan, Valter Hudovernik, Vijay Prakash Dwivedi, Johannes Hoffart, Carlos Guestrin, and Jure Leskovec. Plurel: Synthetic data unlocks scaling laws for relational foundation models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=rM9r9qyRSA.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In Alexandre M. Bayen, Ali Jadbabaie, George Pappas, Pablo A. Parrilo, Benjamin Recht, Claire Tomlin, and Melanie Zeilinger (eds.), *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, pp. 761–770. PMLR, 10–11 Jun 2020. URL https://proceedings.mlr.press/v120/lambert20a.html.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5757–5766. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/lee20g.html.

David Martínez, Guillem Alenya, Carme Torras, Tony Ribeiro, and Katsumi Inoue. Learning relational dynamics of stochastic domains for planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 26, pp. 235–243, 2016.

Ali Mesbah. Stochastic model predictive control with active uncertainty learning: A survey on dual control. *Annual Reviews in Control*, 45:107–117, 2018.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In

Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11048–11064, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.759. URL https://aclanthology.org/2022.emnlp-main.759/.

Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KSugKcbNf9.

Jishnu Sethumadhavan Nair, Patrice Bechard, Rishabh Maheshwary, SRAVAN RAMACHANDRAN, Surajit Dasgupta, Aakash Bhagat, Shruthan Radhakrishna, Pulkit Pattnaik, Johan Obando-Ceron, Shiva Krishna Reddy Malay, Sagar Davasam, Seganrasan Subramanian, Vipul Mittal, Sridhar Krishna Nemala, Christopher Pal, Srinivas Sunkara, and Sai Rajeswar. Do enterprise systems need learned world models? the importance of context to infer dynamics. In *Workshop on Failure Modes of Agentic AI at ICML 2026*, 2026. URL https://openreview.net/forum?id=LKR0BIqTRQ.

Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

Rishabh Ranjan, Valter Hudovernik, Mark Znidar, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Mahmoud Mohammadi, Joe Meyer, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Relational transformer: Toward zero-shot foundation models for relational data. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=rpPtgMC5s9.

Rishabh Ranjan, Vignesh Kothapalli, Harshvardhan Agarwal, Charilaos I. Kanatsoulis, Roshan Reddy Upendra, Tom Palczewski, Carlos Guestrin, and Jure Leskovec. Large-scale pretraining unlocks few-shot prediction for relational data. In *2nd ICML Workshop on Foundation Models for Structured Data*, 2026b. URL https://openreview.net/forum?id=oQINTd9din.

Joshua Robinson, Rishabh Ranjan, Weihua Hu, Kexin Huang, Jiaqi Han, Alejandro Dobles, Matthias Fey, Jan Eric Lenssen, Yiwen Yuan, Zecheng Zhang, Xinwei He, and Jure Leskovec. Relbench: A benchmark for deep learning on relational databases. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=WEFxOm3Aez.

Matteo Rufolo, Dario Piga, Gabriele Maroni, and Marco Forgione. Enhanced transformer architecture for in-context learning of dynamical systems. In *2025 European Control Conference (ECC)*, pp. 819–824. IEEE, 2025.

Yanbo Wang, Xiyuan Wang, Quan Gan, Minjie Wang, Qibin Yang, David Wipf, and Muhan Zhang. Griffin: Towards a graph-centric relational database foundation model. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64604–64627. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wang25da.html.

Yanbo Wang, Jiaxuan You, Chuan Shi, and Muhan Zhang. Relational in-context learning via synthetic pre-training with structural prior. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kiSm94YKor.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Shuo Yang, Zheyu Zhang, Bardh Prenkaj, and Gjergji Kasneci. Doubling your data in minutes: Ultra-fast tabular data generation via LLM-induced dependency graphs. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 10337–10358, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.525. URL https://aclanthology.org/2025.emnlp-main.525/.

Shuo Yang, Zheyu Zhang, Bardh Prenkaj, and Gjergji Kasneci. SAGE: Sparse adaptive guidance for dependency-aware tabular data generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3792–3807, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.174. URL https://aclanthology.org/2026.acl-long.174/.

Zheyu Zhang, Shuo Yang, Bardh Prenkaj, and Gjergji Kasneci. Not all features deserve attention: Graph-guided dependency learning for tabular data generation with language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 6217–6242, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.330. URL https://aclanthology.org/2025.findings-emnlp.330/.

Zheyu Zhang, Shuo Yang, Bardh Prenkaj, and Gjergji Kasneci. Active tabular augmentation via policy-guided diffusion inpainting. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=oa9VyvHw8K.

A Extended Discussion

Relational foundation models have largely expanded pretraining along the diversity of databases, schemas, and prediction tasks (Ranjan et al., 2026a; Kothapalli et al., 2026; Wang et al., 2026). World modeling introduces another axis: the diversity of behaviors that relate actions to future relational states. Scaling along this axis is not simply a matter of observing more database instances. It requires a prior that captures reusable patterns of change while allowing their concrete realization to vary across systems. This suggests behavioral diversity as a distinct dimension of relational pretraining alongside structural and semantic diversity.

A broad prior describes which behaviors are plausible, but evidence from the current system is still needed to determine which are active. Observed executions provide one such source by directly connecting actions to realized consequences (Forgione et al., 2023; Rufolo et al., 2025). Runtime-readable configuration can provide another by exposing rules instantiated in the current system (Nair et al., 2026). These signals constrain the dynamics in different ways. A pretrained prior captures regularities across systems, configuration describes explicitly available rules, and executions reveal their realized behavior under the current state. Combining these sources raises a broader question of how a relational world model should update its view of the current system when evidence differs in specificity or completeness, and how such evidence should be exposed by operational environments in the first place.

Once a model can act, the evidence available for identification is no longer fixed. Actions can change both the system and what can be learned about it, connecting planning with ideas from dual control and active identification (Mesbah, 2018). The relevant uncertainty, however, need not be the uncertainty that contributes most to average transition error. Our planning results show that consequence prediction and action selection improve at different rates, consistent with prior observations that predictive fidelity and control utility need not align perfectly (Lambert et al., 2020; Grimm et al., 2020). This suggests a decision-directed view of identification in which a model seeks evidence that can change which futures should be preferred.

B Experimental Details

B.1 Metrics

An action consequence is represented as a set of typed edits. For an edit e, let

$$\mathrm{supp}(e) = (o_e, t_e, r_e, c_e)$$

denote its structural support, consisting of the operation, table, row, and column whenever the corresponding fields apply. This representation separates identifying *what changes* from predicting the value produced by that change.

Support F1. Two edits match in support when their operations and relational targets agree. Support F1 therefore measures whether the model locates the consequence of an action, independently of the value written at each location. This distinction is useful in relational systems because an action may affect several records reached through different relational paths.

Edit F1. Complete-edit matching additionally requires any new value or inserted record contents to be correct. Categorical values match exactly. Numeric values use

$$|\hat{v} - v| \le \max\left(10^{-3},\, 0.02|v|\right). \tag{9}$$

Edit F1 is computed from the matched and unmatched edits after duplicate predictions at the same structural support are collapsed.

Value MAE. For numeric updates whose structural support is matched, we additionally report

$$\mathrm{MAE}_{\mathrm{value}} = \frac{1}{|\mathcal{M}_{\mathrm{num}}|} \sum_{(e,\hat{e})\in\mathcal{M}_{\mathrm{num}}} |v_e - \hat{v}_e|, \tag{10}$$

where $\mathcal{M}_{\mathrm{num}}$ is the set of correctly located numeric edits. Value MAE is therefore conditional on localization and is used as a diagnostic of numeric reconstruction rather than as the primary cross-model metric.

Confidence intervals. Queries drawn from the same world share its transition dynamics. Bootstrap estimates therefore resample worlds and retain the queries associated with each sampled world. The main context sweep uses 2,000 world-bootstrap draws.

B.2 Baselines

All baselines are evaluated on the same sampled tasks and behavioral context as TabWM. They differ in how the relational execution is represented and in the model used to predict its consequence.

Flat baselines. Each candidate query cell becomes one row in a fixed-width feature table. Its features describe the current value and type, column-level statistics, whether the row is bound to or linked from the queried action, and summaries of demonstrated changes to the same column. Flat-GBM applies gradient-boosted trees (Friedman, 2001) to this representation, while Flat-TabPFN uses TabPFN (Hollmann et al., 2025). The representation retains local values and coarse relational evidence but does not preserve each execution as a structured relational object. Table and column names are excluded, so the flat baselines are unchanged by the naming intervention.

LLM baselines. Qwen3 (Yang et al., 2025a) receives a textual serialization of the same observed executions and query under the same action-centered cell budget as TabWM. The readable and opaque conditions differ only in the identifiers assigned to schema elements, categorical values, and actions. Relational links, values, row identities, and action bindings remain unchanged.

B.3 LLM Prompt Construction

Recent LLMs have demonstrated substantial ability to work with structured data (Zhang et al., 2025; Yang et al., 2026; 2025b). Following the relational serialization used for LLM baselines in prior relational foundation-model evaluations (Ranjan et al., 2026a), the prompt states the schema and action signatures before presenting the observed executions. Each demonstration contains a pre-action relational neighborhood, the executed action, and the resulting edit set. The final query uses the same format with its edits withheld.

Figure 5 shows an abridged prompt in the readable naming condition.

B.4 Candidate-Future Scoring

Each world defines a state potential $\Phi(S)$ and an action cost $c(A)$. For a horizon-H candidate sequence, the predicted return is

$$\widehat{R}(A_{0:H-1}) = \sum_{h=0}^{H-1} \gamma^h \left[\Phi(\widehat{S}_{h+1}) - \Phi(\widehat{S}_h) - c(A_h)\right], \qquad H = 3,\ \gamma = 0.95. \tag{11}$$

The predicted state is updated after every imagined action using Equation 8, while the behavioral context remains fixed throughout the rollout.

The evaluation contains four decision states from each of 40 held-out worlds. Decision states are reached through actions sampled independently of all scorers. At each state, we sample one shared pool of 64 legal action sequences, which is reused across context sizes and scoring methods. The candidate-budget experiment uses nested prefixes of this same pool for $N \in \{1, 4, 16, 64\}$.

The simulator oracle replaces TabWM with the true transition mechanism. Shuffled context breaks the action–consequence correspondence in the observed executions, while the no-change scorer leaves the predicted state fixed and therefore uses only action costs. Spearman correlation is computed within each 64-sequence pool and averaged across the 160 decision states. Normalized regret divides the selected candidate's true-return shortfall from the best candidate by the true-return span of that pool.

C Method Details

TabWM must preserve two kinds of structure at once: the relational structure of the database and the correspondence between each observed action and its consequence (Ranjan et al., 2026a). The

```
You are a strict prediction assistant. Follow the instructions exactly.

# System
A relational database whose state is changed by actions. The rules governing
what an action changes are not given. Infer them from the demonstrations.

# Schema
  customers(id, stage, risk, quantity, score, units, stock, credit): 18 rows
  accounts(id, warehouses_id, customers_id, is_flagged, risk, tier): 11 rows
  warehouses(id, amount, priority, state, is_active, is_locked): 17 rows

Foreign keys:
  accounts.warehouses_id --> warehouses.id
  accounts.customers_id --> customers.id

Action signatures:
  restock_customer(target: customers.id, amount: number)
  reserve_warehouse(target: warehouses.id, mode: category)
  ...

# Task
Name: state-transition prediction

Description: Given an action and the records near it, predict the complete set
of cell-level edits that the action causes.

# Demonstrations

--- demonstration 1 ---
state before:
[customers]
  id=9, stage=stage_2, risk=39.4568, quantity=334, score=45.043, units=8, ...
[accounts]
  id=3, warehouses_id=2, customers_id=9, is_flagged=0, risk=120.856, tier=tier_0
[warehouses]
  id=2, amount=121, priority=terminated, state=queued, is_active=1, is_locked=1

action: restock_customer(target=9, amount=43)

edits:
  {"op": "UPDATE_VALUE", "table": "customers", "row_id": 9,
   "column": "risk", "value": 5.40528}

<demonstrations 2--3 omitted for display>

--- demonstration 4 ---
state before:
[warehouses]
  id=14, amount=493, priority=new, state=queued, is_active=0, is_locked=0
[accounts]
  id=9, warehouses_id=14, customers_id=4, is_flagged=0, risk=492.989, tier=tier_0
[customers]
  id=4, stage=stage_2, risk=16.9865, quantity=0, score=27.294, units=11, ...

action: reserve_warehouse(target=14, mode=3)

edits:
  {"op": "EMIT_EVENT", "table": "accounts"}
  {"op": "UPDATE_VALUE", "table": "warehouses", "row_id": 14,
   "column": "amount", "value": 496}

# Query
state before:
[customers]
  id=8, stage=stage_0, risk=21.7258, quantity=197, score=13.9966, units=36, ...
[accounts]
  id=10, warehouses_id=15, customers_id=0, is_flagged=0, risk=247.983, tier=tier_1

action: reserve_customer(target=8, amount=9.7493)

# Output
Output JSON only, in exactly this form:
{"edits": [{"op": "...", "table": "...", "row_id": 0,
            "column": "...", "value": ...}]}

Valid operations are UPDATE_VALUE, UPDATE_STATE, EMIT_EVENT, and DELETE_ROW.
Use identifiers exactly as they appear in the task and output no text outside JSON.
```

Figure 5: Abridged prompt template used in the readable naming condition for LLMs. Rows and two demonstrations are shortened only for display. The opaque condition preserves the same structure and values while consistently replacing schema, categorical, and action identifiers.

encoder represents both in a shared context, while the decoder predicts the query consequence as a set of typed edits and transfers numeric transformations across executions.

C.1 Relational Transition Encoding

The relational state is represented with the pretrained Relational Transformer backbone (Ranjan et al., 2026a), using weights learned from PluRel-generated relational databases (Kothapalli et al., 2026). TabWM extends this representation from a static relational neighborhood to an execution by placing the state, action, and observed consequence in the same relational context.

For each execution, an action-centered neighborhood is collected by following primary- and foreign-key links from the records referenced by the action until the cell budget is reached. Cells retain their column identity, type, and value. Textual identifiers use deterministic hash embeddings rather than a pretrained language encoder. This lets the model preserve identity within a task without introducing external linguistic semantics.

The action is represented as a distinguished pseudo-row linked to the records bound by its entity arguments. Scalar and categorical arguments are represented as action cells. Demonstrated edits are attached to the records they modify. Role embeddings distinguish state, action, and consequence cells, while transition identifiers mark which elements belong to the same observed execution. For an updated cell, shared table, row, and column identities align its before-action representation with the edit that follows the action.

These connections preserve both relational structure and execution membership. When the demonstrations and query are packed as in Equation 4, the query consequence is withheld while its state and action share attention with the observed executions. The encoder can therefore relate a queried action to the current relational state and to before-and-after evidence from the same world.

C.2 Structured Edit Prediction

An action may change several relational targets or none, and the order of those changes has no meaning. TabWM therefore predicts the consequence as an unordered set using learned edit queries, following DETR-style set prediction (Carion et al., 2020). Each query attends to the contextual representation and predicts whether an edit exists, which operation it performs, where it acts, and any value or record contents required by that operation.

The predicted operation determines the form of the relational target. Value and state updates point to affected cells, while row- and table-level operations select the corresponding relational objects. Typed output heads predict categorical and numeric contents. The decoder therefore separates identifying *what changes* from predicting *what value results*, while keeping both parts within the same edit representation.

There is no canonical ordering between predicted slots and reference edits. During training, minimum-cost bipartite matching assigns predicted slots to reference edits using their operation, relational target, and value when applicable. Matched slots receive supervision for the corresponding edit fields, while unmatched slots learn to emit no edit. The supervision is applied to the observed consequence itself. TabWM is not given the hidden transition program that produced it.

C.3 Numeric Value Transfer

Once a numeric target has been identified, the remaining question is how its value should change. The decoder uses visible before-and-after pairs (x_r, y_r) from previous executions that modify the same relational role. Here x_r is the value before the demonstrated action and y_r is the value written by its consequence. The edit query, target representation, and query action condition attention over these pairs, allowing the model to select changes that are relevant to the current query rather than treating all observed numeric edits equally.

A learned path first predicts a numeric estimate without assuming a particular transformation form. It produces an initial estimate $\hat{y}_0$ together with a residual correction,

$$\hat{y}_{\text{learned}} = \hat{y}_0 + r_\theta. \tag{12}$$

The contextual representation used for this prediction includes the selected before-and-after evidence and the actions under which those changes were observed.

Alongside this learned estimate, TabWM constructs two transformation candidates from the same evidence. Let α_r denote normalized attention weights over usable pairs and let $\bar{x}$ and $\bar{y}$ be their weighted means. For a query value x_q,

$$h_{\text{add}} = x_q + \sum_r \alpha_r (y_r - x_r), \qquad h_{\text{aff}} = \bar{y} + \beta(x_q - \bar{x}), \tag{13}$$

where β is a bounded weighted least-squares slope. The additive candidate transfers an observed change, while the affine candidate estimates how the outcome varies with the starting value. Both therefore apply a transformation inferred from the demonstrations to the query value rather than copying a demonstrated outcome.

Learned gates combine these candidates with the neural estimate as in Equation 7. This gives the decoder several ways to use the same behavioral evidence. The learned path can represent changes that are not captured by a simple arithmetic form, while the additive and affine paths provide direct routes for regularities that transfer cleanly across starting values.

Candidate construction uses only visible before-and-after pairs that remain informative after value-domain clipping. The affine candidate additionally requires at least two usable pairs with variation in their starting values. When a candidate is unavailable, it contributes no correction to $\hat{y}_{\text{learned}}$. If no usable numeric evidence is present, prediction falls back to the learned estimate. These choices affect only the value written by a numeric edit and do not alter its predicted operation or relational target. The arithmetic paths only modify the value attached to an already selected numeric edit. § F.5 isolates their contribution under matched reader training.

C.4 Training Details

TabWM initializes its relational encoder from the pretrained Relational Transformer used in §4.2. The encoder, structured edit decoder, and numeric value decoder are then optimized jointly on tasks sampled from WorldRel. Each task contains a behavioral context and a separate query from the same world, with the context size sampled from $K \in \{0, 1, 2, 4, 8, 16, 32, 64\}$. Training, development, and evaluation use disjoint world seed ranges.

For numeric edits, training uses a routing curriculum to provide a stable target representation while the edit pointers are still inaccurate. After set matching, a matched numeric slot may initially read the reference target cell when constructing the value-decoder input. This probability is annealed to zero during training, after which the model uses its own predicted target throughout. Matching is computed before the route is chosen, and validation, evaluation, and rollout always use predicted targets.

Table 6: Training details for TabWM.

Initialization	Relational Transformer pretrained on PluRel (Kothapalli et al., 2026)
Trainable components	Relational encoder, edit decoder, numeric value decoder
Context sizes	$K \in \{0, 1, 2, 4, 8, 16, 32, 64\}$
Optimizer	AdamW
Learning rates	7.5×10^{-5} encoder; 3×10^{-4} transition components
Training length	200k optimizer updates
Routing curriculum	Reference-to-predicted target over the first 50k updates
Checkpoint selection	Highest pooled development Edit F1 at $K \in \{16, 64\}$

D A Formal View of Behavioral Identification

Observed executions can be interpreted as evidence about a latent transition rule T_w. This connects TabWM to in-context system identification (Forgione et al., 2023; Rufolo et al., 2025) and to prior-data fitted learning, where inference over a task-specific latent structure is amortized across a pretraining

distribution (Müller et al., 2022). This section makes that interpretation explicit without assuming that TabWM recovers a symbolic transition program.

D.1 Posterior-Predictive Interpretation

The behavioral context

$$\mathcal{C}_w^K = \{(S_i, A_i, \Delta S_i)\}_{i=1}^K$$

provides evidence about T_w. A posterior-predictive interpretation of Equation 3 is

$$p(\Delta S_q \mid S_q, A_q, \mathcal{C}_w^K) = \mathbb{E}_{T \sim p(T \mid \mathcal{C}_w^K)} \left[p(\Delta S_q \mid S_q, A_q, T) \right]. \tag{14}$$

TabWM does not explicitly recover or parameterize the posterior over symbolic transition programs. Instead, pretraining across worlds learns the resulting predictive map from behavioral context to query consequences.

Under this view, readable identifiers may affect the prior over plausible dynamics, while observed executions constrain that prior through realized action–consequence pairs. The naming intervention in §5 changes the former while preserving the latter.

D.2 Identification Through Correspondence

For intuition, suppose the dynamics are deterministic and fully determined by the observed state and action, and define

$$\mathcal{T}(\mathcal{C}_w^K) = \left\{ T : T(S_i, A_i) = \Delta S_i \quad \forall (S_i, A_i, \Delta S_i) \in \mathcal{C}_w^K \right\} \tag{15}$$

as the set of transition rules compatible with the observed executions. A query is identified whenever all compatible rules agree on its consequence,

$$\left| \left\{ T(S_q, A_q) : T \in \mathcal{T}(\mathcal{C}_w^K) \right\} \right| = 1. \tag{16}$$

Each aligned action–consequence pair rules out transition functions that are inconsistent with the observed execution. Breaking this correspondence removes those constraints because the resulting pairs need not be jointly explained by a single T. The shuffling intervention in §6 tests this consequence of the identification view directly.

E The WorldRel Prior

A relational world is defined not only by what its database looks like, but by how that database changes under actions. WorldRel turns this distinction into a generative prior. Starting from relational databases sampled with PluRel (Kothapalli et al., 2026), it adds executable actions and world-specific transition mechanisms whose effects remain consistent within a world and vary across worlds. The resulting tasks expose TabWM to different relational structures and different ways in which those structures can evolve.

E.1 Generating Executable Worlds

Each sampled world fixes four components: a relational schema, an action interface, an initial-state distribution, and a transition mechanism. The base relational schema contains between 2 and 12 sampled tables with typed attributes and sampled foreign-key structure. Each world also contains between 4 and 24 actions that bind a primary entity and may additionally take entity, scalar, or categorical arguments. Actions can carry state-dependent preconditions and costs. Resetting the world draws a fresh relational state while preserving its action interface and dynamics.

Actions are governed by compositions of executable effects. The transition library spans local numeric updates and state transitions, propagation through foreign-key links, multi-row cascades, aggregate recomputation, threshold-triggered changes, stochastic and regime-dependent behavior, event creation, deletion, and delayed consequences. Relational effects can follow foreign-key paths for up to three hops and may fan out across several linked records. Delayed effects can arrive one to four steps after the action that scheduled them.

Table 7: Sources of variation in a WORLDREL world.

Component	Variation within the prior
Relational structure	2–12 tables, typed columns, and sampled foreign-key topology
Initial state	Fresh relational states drawn when an episode is reset
Action interface	4–24 actions with entity, scalar, and categorical arguments
Transition mechanism	Ordered compositions drawn from 13 executable effect classes

Several effects may occur within the same action. They are applied in sequence to the evolving state, allowing one change to alter the conditions or inputs of another. Their target columns, coefficients, thresholds, transition maps, delays, and stochastic parameters remain fixed within a world and are resampled across worlds. Direct numeric updates include additive, multiplicative, reset, and action-conditioned cross-table transformations, while aggregate effects derive values from linked records through `sum`, `mean`, `max`, or `count`. A consequence can therefore combine value changes with relational propagation, state transitions, event creation, and structural effects.

A realized sample of $1{,}500$ worlds from the training seed range contains 14.0 actions and 2.17 effects per action on average, producing 1.38 edits per transition. Importantly, 38.1% of realized edits occur on a table other than the one directly bound by the action. A separate 500-world sample from the evaluation range gives nearly identical statistics. The executable prior therefore produces substantial relational propagation in practice rather than only through rare constructed cases.

E.2 VARIATION ACROSS WORLDS AND TASKS

Sampling a new world changes several parts of the system at once. Its relational schema and initial-state distribution are resampled together with the action interface, the assignment from actions to mechanisms, and the parameters and composition of those mechanisms. The mechanism families and structured edit vocabulary provide a common behavioral prior across worlds.

Table 8: Variation between pretraining worlds and held-out WORLDREL worlds.

World component	Held-out world
Schema and foreign-key graph	Resampled
Initial-state distribution	Resampled
Table, column, category, and action identities	Resampled
Action argument signatures	Resampled
Action-to-mechanism assignment	Resampled
Mechanism parameters	Resampled
Mechanism composition	Resampled
Mechanism families	Shared
Edit vocabulary	Shared

Observed executions distinguish among the behaviors made possible by this prior. An action can therefore invoke different effects in different worlds, while related mechanisms can appear under different schemas, parameters, and relational bindings. The shifts in §6 probe these sources of variation separately by changing the action mapping, schema, or mechanism composition. Throughout the main evaluation, an unseen world is a fresh instantiation of this executable-world prior.

A task draws its observed executions and query from independently reset episodes of the same world. Each world provides 20 episodes of four steps, giving up to 80 transitions from the same underlying dynamics. During pretraining, the amount of behavioral context is sampled from $K \in \{0, 1, 2, 4, 8, 16, 32, 64\}$. Drawing context across episodes exposes repeated behavior from the same world without giving the query its own execution history.

World generation is also separated from rendering. Schema, categorical, and action identifiers can be replaced consistently with opaque labels while preserving the relational structure, values, row identities, action bindings, and consequences. The naming experiments in §5 use this separation to vary semantic cues without changing the underlying behavioral task.

E.3 Behavior Transfer Across Executions

The same world-level behavior can produce different concrete edits when an action is applied in a different relational state. Figure 6 shows one observed execution and a query from the same world. They share an action type and transition mechanism, but bind to different records and begin from different values.

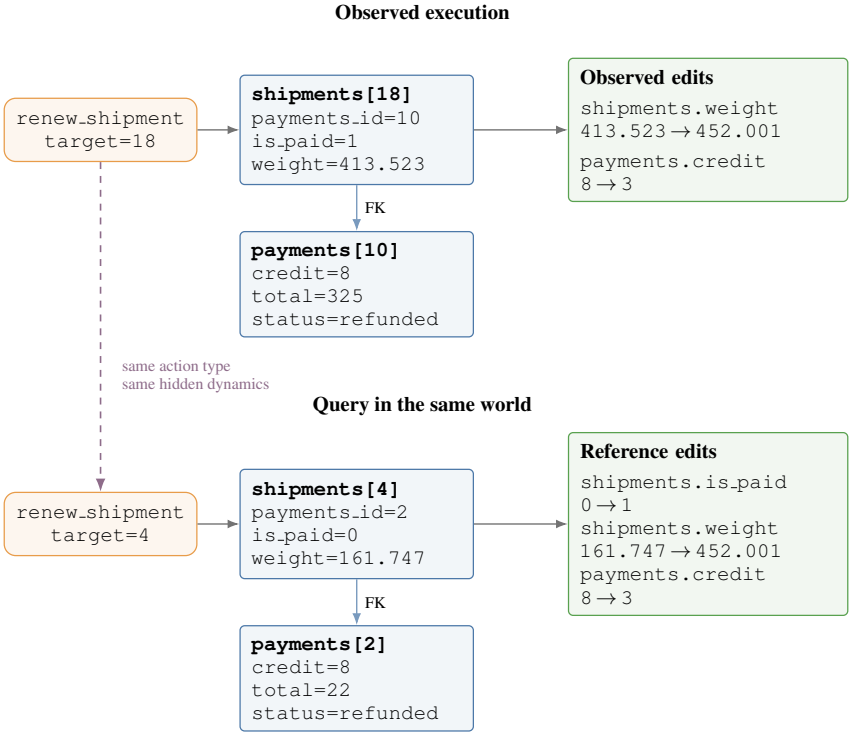

Figure 6: Transferring behavior across executions. The demonstrated action and the query bind to different shipment and payment rows, so the relational targets must be transferred by role rather than by row identity. The query additionally changes `is_paid` because its starting value differs from the demonstration. Replaying the demonstrated edit list would therefore be insufficient.

The observed action changes both the targeted shipment and its linked payment. In the query, the same relational roles are occupied by different rows, so replaying the demonstrated row identities would fail. The starting state also differs, producing an additional edit to `is_paid`. What transfers across executions is therefore not a literal edit list, but the relationship between an action, the current relational state, and the records and values it changes.

F Additional Analyses

F.1 Held-Out Confirmation

A larger confirmation set of 512 unseen worlds reproduces the main context-scaling pattern. Support and complete-edit prediction both improve as more executions become available, while conditional numeric error decreases.

F.2 Separating Naming Cues

Schema and action names can be changed independently because task generation is separated from rendering. Their effect on TabWM is small once behavioral context is available.

Table 9: Held-out confirmation on 512 unseen worlds.

K	Support F1 ↑	Edit F1 ↑	Value MAE ↓
0	0.263	0.016	73.820
4	0.530	0.237	22.010
16	0.750	0.466	14.303
64	**0.875**	**0.685**	**6.200**

Table 10: Separating schema and action naming cues. Values are Edit F1.

Naming cues		Observed executions		
Schema	Action	$K = 4$	$K = 16$	$K = 64$
Readable	Readable	0.194	0.498	0.684
Opaque	Readable	0.178	0.494	0.681
Readable	Opaque	0.195	0.498	0.682
Opaque	Opaque	0.188	0.487	0.674

With more observed executions, the four conditions converge to a narrow range. Together with the Qwen3 comparison in Table 1, this separates semantic naming cues from the behavioral evidence TabWM uses for adaptation.

F.3 Action-Specific Behavioral Evidence

The context-size sweep increases both the amount of behavioral evidence and the chance that the queried action has been observed before. We separate these effects by fixing the total context at $K = 16$ and varying the number m of demonstrations that execute the queried action. The remaining demonstrations use other actions from the same world. We evaluate the paired subset for which all values of $m \in \{0, 1, 2, 4\}$ are available. As a direct retrieval control, an action-template baseline transfers the relational edit pattern observed for matching actions to the query.

Table 11: Edit F1 at fixed $K = 16$ as evidence for the queried action increases.

Matching executions m	TabWM	Template transfer
0	0.048	0.003
1	0.403	0.169
2	0.599	0.188
4	**0.700**	0.216

Direct evidence about the queried action is highly informative, but its benefit is not explained by replaying a previous consequence. The explicit template baseline improves little after the first matching execution, while TabWM continues to improve substantially as additional executions of the same action become available. TabWM therefore uses repeated executions to infer how an action's effect transfers across relational states rather than simply retrieving an observed edit pattern.

F.4 Beyond the Pretraining Mechanism Prior

The shifts in Table 2 vary how mechanisms already represented during pretraining are assigned or combined. We next evaluate TabWM on four transition mechanisms introduced only at evaluation time. They retain the same relational state, action, and structured-edit interface, while their transition rules are absent from the WorldRel pretraining library.

Relational copy transfers a value across a foreign-key link. Cross-cell product makes an update depend multiplicatively on another cell. Redistribution spreads a quantity across related rows. Linked predicate gate conditions an effect on the state of a linked record. Table 12 separates effect localization from complete-edit prediction.

Behavioral context improves effect localization across all four mechanisms. Complete-edit prediction transfers less uniformly when the written value depends on a transformation not represented during pretraining. Redistribution is the strongest complete-edit case, consistent with its use of additive or

Table 12: Behavioral transfer to evaluation-only transition mechanisms.

Mechanism	Support F1 $K = 0$	Support F1 $K = 64$	Edit F1 $K = 64$
Relational copy	0.006	0.702	0.089
Cross-cell product	0.376	0.910	0.031
Redistribution	0.133	0.332	0.267
Linked predicate gate	0.000	0.561	0.077

multiplicative per-row arithmetic already supported by the numeric decoder. These results indicate that behavioral evidence transfers the relational footprint of an action farther than its exact quantitative transformation.

F.5 Numeric Value Transfer Ablation

The numeric decoder in §4.3 combines a learned prediction with explicit additive and affine candidates. To isolate their contribution, we compare two numeric-decoder variants with the same initialization, training tasks, and optimization budget. The neural-only variant removes the explicit arithmetic candidates, while the full variant retains them.

Table 13: Matched ablation of numeric value transfer. Values are Edit F1.

Variant	$K = 16$	$K = 64$
Neural-only	0.380	0.476
+ additive / affine	**0.436**	**0.616**

The arithmetic candidates improve complete-edit prediction, with a larger difference as more behavioral evidence becomes available. Support F1 is unchanged across the two variants. The explicit arithmetic paths therefore improve numeric value transfer without changing which relational edits are predicted.

F.6 Sensitivity to Context Sampling and Numeric Tolerance

Two checks probe whether the main pattern depends on which executions are sampled or on the numeric matching threshold. Repeatedly sampling tasks from the same worlds produces little variation relative to the gain from additional behavioral context. Rescoring the same predictions under different numeric tolerances gives a smooth accuracy curve rather than a discontinuity around the default threshold. Figure 7 summarizes both checks.

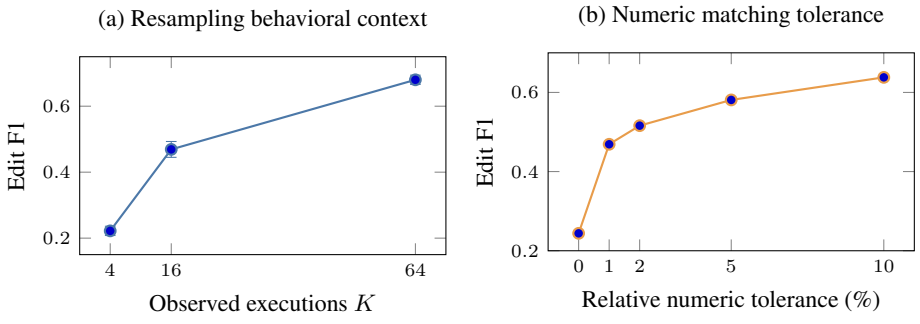

Figure 7: Sensitivity of complete-edit prediction. (a) Mean Edit F1 across eight independently sampled tasks on the same worlds; error bars show the standard deviation across draws. (b) The same $K = 16$ predictions rescored under increasingly permissive numeric tolerances.