WHAT DO CURRENT SYSTEMATIC GENERALIZATION TASKS MISS? A REASONING-CENTERED ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Systematic generalization, the ability to solve novel problems by recombining known atomic elements, is central to human intelligence but difficult to study rigorously under controlled settings. Existing studies therefore rely on simplifications such as elemental composition, productivity-based tests, and action-explicit goals, which make systematic generalization easier to study but omit some essential aspects of this capability. To characterize what these simplifications miss, we adopt a reasoning-centered lens and introduce TranSGrid, a testbed that brings deductive, inductive, and abductive reasoning together within a unified task. Experiments with seven Transformer models on 4,800 TranSGrid instances show that all models perform much worse on TranSGrid than on a held-out test set: the largest model solves 79.6% of the test set, but only 55.3% of TranSGrid and 15.8% of the hardest subset. The gap remains within the training length range, showing that productivity alone is not sufficient to evaluate systematic generalization. Additionally, we reintroduce the other two simplifications into TranSGrid: one variant limits interactions among action effects to approximate elemental composition (reducing the inductive demand); the other makes goals action-explicit (reducing the abductive one). In both, solve rates return to roughly the test set level, showing that either simplification alone is enough to reduce TranSGrid to an ordinary held-out test set. Together, our results show that existing tasks reduce either or both of the inductive and abductive demands of systematic generalization, and that comprehensively measuring this capability requires a task that involves all three forms of reasoning.

1 INTRODUCTION

Systematic generalization, which refers to the ability to systematically combine learned actions or skills to address novel problems, is a fundamental aspect of human intelligence (Lake & Baroni, 2018; 2023). This capability is essential across diverse domains, including grounded navigation (Ruis et al., 2020; Sikarwar et al., 2022; Spilsbury et al., 2024; Chen et al., 2026a), semantic parsing (Kim & Linzen, 2020; Wu et al., 2023; Jabbar et al., 2025), text-to-image generation (Han et al., 2025; Huang et al., 2025; Dat et al., 2025), facial recognition (Rotshtein et al., 2007; Leong et al., 2023), AI safety (Addepalli et al., 2025; Chen et al., 2026b), and AI for Science (Ji et al., 2023; Chen et al., 2025; Li et al., 2026a;b). Across these domains, success depends not merely on mastering individual components, but also on understanding how they interact and recombining them appropriately under novel goals, contexts, and constraints.

A rigorous study of systematic generalization therefore requires (1) a *clearly defined space of atomic actions*, (2) *meaningful, interaction-rich compositions*, (3) *scalable generation* of instances that probe models' systematicity, and (4) *automatic evaluation* of multiple valid solutions. In practice, however, satisfying all these requirements within a single task is difficult, leading existing studies to rely on simplifications in task design. One common simplification is elemental composition[1], in which interactions among atomic elements are limited (Lake & Baroni, 2018; 2023). A second is to assess systematicity primarily through productivity, typically by testing whether models extrapolate

[1]This terminology is adapted from studies of human and animal learning (Devaud et al., 2015; Duncan et al., 2018): elemental learning predicts a combination's outcome from its parts alone, whereas configural learning also uses predictive information from how those parts are combined.

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.

to sequences longer than those observed during training (Hupkes et al., 2020; Fu & Liu, 2026). A third is to use action-explicit goals, where the input already specifies the actions and their order, largely reducing the task to interpreting a given composition rather than finding a valid action sequence within a vast combinatorial space (Wu et al., 2023; Li et al., 2023; Mondorf et al., 2026). Although these simplifications make systematic generalization easier to study, they remove some of its core challenges and thus provide an incomplete picture of the capability.

To examine what these simplifications ignore, **we employ a reasoning-centered lens to study systematic generalization** and introduce **TranS**formation on a **Grid** (**TranSGrid**), a controllable and flexible testbed. Inspired by cognitive science research highlighting the interplay of deduction, induction, and abduction in human intelligence (Peirce, 1934; Shank, 1998; Chemero, 2026), TranSGrid is designed to operationalize this interplay within a unified task. Specifically, TranSGrid requires a model to transform an initial board into a target board by generating an action sequence drawn from ten atomic operations on rows, columns, or local 2×2 blocks. The task involves deductive reasoning to derive and track the board state after each action. Its inductive challenge lies in discovering reusable composition rules from observed examples of how action effects combine, cancel, or obscure one another, and applying these rules to novel instances. Abductive reasoning involves inferring a suitable action sequence from the desired outcome. To accommodate multiple valid solutions, TranSGrid evaluates the correctness of each generated sequence by executing it on the initial board and checking whether the resulting board matches the target. Any sequence that produces the target board is considered correct; it need neither be a shortest solution nor match the reference sequence.

Experiments with seven Transformer models ranging from 0.96M to 88.43M parameters on 4,800 TranSGrid instances show that all models perform substantially worse on TranSGrid than on a held-out Test set. Under greedy decoding, the largest model's solve rate falls from 79.63% on Test to 55.31% on TranSGrid overall, reaching only 15.75% on TranSGrid$_{\mathrm{Hard}}$. Crucially, this gap persists even when reference answer lengths fall within the training range, showing that productivity alone is insufficient to evaluate systematic generalization. We then reintroduce the other two simplifications through controlled variants. TranSGrid (Decoupled) approximates elemental composition by limiting interactions among action effects, thus reducing the inductive demand of inferring reusable interaction rules from examples. TranSGrid (SCAN) makes goals largely action-explicit by providing all but the final gold action, reducing the abductive demand of inferring a suitable action sequence from the desired outcome. Solve rates in both variants recover to roughly the held-out test set level, indicating that either simplification substantially reduces the challenge posed by TranSGrid. Taken together, these findings show that existing tasks overlook essential inductive and abductive reasoning demands of systematic generalization, and that a comprehensive evaluation of this capability requires tasks that involve all three forms of reasoning: deduction, induction, and abduction.

2 Related Work

Existing benchmarks for systematic generalization can be broadly grouped into three categories. First, action or solution-explicit tasks, such as SCAN (Lake & Baroni, 2018), PCFG (Hupkes et al., 2020), COGS (Kim & Linzen, 2020), and HINT (Li et al., 2023), provide the action sequence directly in the input. Models therefore need only to parse or execute the given composition rather than infer one from a goal. Second, grounded navigation benchmarks, including gSCAN (Ruis et al., 2020) and ReaSCAN (Wu et al., 2021), require models to generate an action sequence in a structured world. However, once the target is identified, the route can usually be derived directly from the world, so these benchmarks focus more on language grounding and target localization than on abductive reasoning. Action-action interactions are also present, but are usually simple. Across these two lines, many evaluations emphasize productivity by testing sequences longer or deeper than those seen during training. Finally, rule-changing tasks such as *Baba Is You* provide richer action interactions by requiring agents to create, break, and combine rules to reach a goal (Cloos et al., 2024). However, its visual input and various distractors make it difficult to determine whether failures result from weak systematic generalization or poor scene understanding. TranSGrid addresses these limitations by requiring models to infer action sequences from goals and reason about interacting action effects in a simple, controllable setting, allowing systematic generalization to be studied more reliably and with fewer confounding factors.

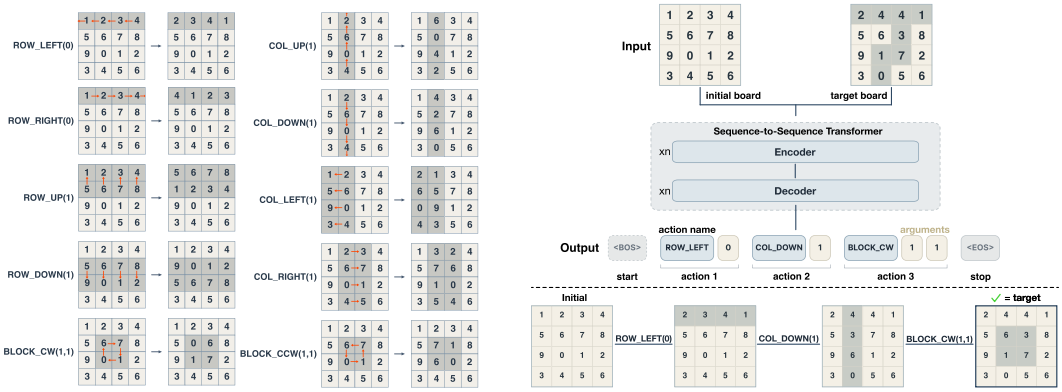
3 A GRID-TRANSFORMATION TASK FOR SYSTEMATIC GENERALIZATION

3.1 CHALLENGES IN MEASURING SYSTEMATIC GENERALIZATION

Although systematic generalization is essential across many domains, constructing a rigorous and scalable benchmark remains difficult. Such a benchmark needs to combine a well-defined atomic action space, automatic generation of meaningful compositions, scalable data construction, and automatic evaluation of multiple valid solutions. Yet most existing task settings satisfy only a subset of these requirements. For example, there is no precise definition of atomic actions in code generation; meaningful action interactions in spreadsheet editing often require substantial human effort; and tasks such as *Baba Is You* introduce perceptual confounds unrelated to systematic generalization. Beyond these construction challenges, there is also no consensus on how to measure systematic generalization: most criteria are task-specific, with productivity being the only one widely shared across tasks. However, productivity is not a necessary condition for systematic generalization, as generalization within the training length range can also be systematic. Together, these construction and evaluation challenges help explain why prior work relies on simplifications such as elemental composition, productivity-based evaluation, and action-explicit goals. They also motivate a testbed that addresses these construction challenges and the limitations of productivity-based evaluation.

3.2 DESIGN OF THE TRANSGRID TASK

To address these challenges, we introduce **TranS**formation on a **Grid** (**TranSGrid**), a controllable and flexible grid-transformation testbed for systematic generalization. Figure 1 provides an overview of the task, with its atomic action space and task formulation detailed below.



(a) (b)

Figure 1: The proposed TranSGrid task with its atomic actions. (a) The ten types of atomic actions defined for grid manipulation, including row-wise shifts, column-wise shifts, and rotations of local blocks. Red arrows indicate the direction of each transformation. (b) Illustration of the TranSGrid task. Given an initial board and a target board, the models need to generate an action sequence that, when executed sequentially, can transform the initial board into the target board.

Atomic action space As illustrated in Figure 1a, TranSGrid defines ten types of atomic actions. `ROW_LEFT` and `ROW_RIGHT` cyclically shift the elements of a selected row, while `COL_UP` and `COL_DOWN` perform the analogous operation on a column. `ROW_UP` and `ROW_DOWN` move a selected row by swapping it with an adjacent row, and `COL_LEFT` and `COL_RIGHT` similarly move a selected column. Finally, `BLOCK_CW` and `BLOCK_CCW` rotate a 2×2 subgrid 90° in either direction.

By design, each atomic action modifies multiple cells at once, operating on an entire row, column, or 2×2 block. Because actions may affect overlapping regions and are executed sequentially, their effects are both state-dependent and interdependent: a later action may reinforce, overwrite, or reverse the effect of an earlier one. Solving TranSGrid therefore requires models to track the evolving board and reason about both action-action and action-environment interactions, providing a controlled, interaction-rich setting for studying systematic generalization.

Task formulation A TranSGrid instance consists of an initial board and a target board, both of size $N \times N$, as shown in Figure 1b.[2] Let

$$B^{(0)}, B^{\star} \in \mathcal{V}^{N \times N}, \quad \mathcal{V} = \{0, 1, \ldots, 9\},$$

denote the initial and target boards, respectively, where $B^{(0)}_{i,j}$ is the value in row i and column j of the initial board. TranSGrid provides ten atomic operation types,

$$\mathcal{A} = \{\alpha_1, \ldots, \alpha_{10}\},$$

each of which performs a deterministic transformation on the board, as illustrated in Figure 1a. Let

$$T : \mathcal{A} \times \mathcal{V}^{N \times N} \to \mathcal{V}^{N \times N}$$

denote the transition function, such that $T(a, B)$ is the board obtained by applying action a to board B. Composition in TranSGrid is sequential. An action sequence of length L is defined as

$$\pi = (a_1, \ldots, a_L) \in \mathcal{A}^L.$$

Executing π from $B^{(0)}$ yields

$$B^{(t)} = T\big(a_t, B^{(t-1)}\big), \qquad t = 1, \ldots, L.$$

The final board after executing π is therefore $B^{(L)}$. Given an instance $(B^{(0)}, B^{\star})$, the model is required to generate a sequence π such that $B^{(L)} = B^{\star}$. Any sequence satisfying this condition is considered a valid solution; it need neither be a shortest solution nor match the reference sequence.

3.3 Controllability, Flexibility, and Scalability of TranSGrid

TranSGrid provides explicit control over task properties through adjustable parameters, including grid size N, digit distribution, reference action-sequence length, and composition patterns. These parameters allow task structure and difficulty to be varied systematically. Its flexible design can also reproduce two common simplifications in existing tasks: elemental composition and action-explicit goals (see Figures 3c and 4c, respectively, for further details).

Moreover, TranSGrid provides a vast combinatorial task space. For example, with $N = 6$ and ten possible cell values, there are 10^{36} possible initial boards. Because the atomic actions can be applied to different rows, columns, or blocks, a 6×6 board admits

$$4 \cdot 6 + 2 \cdot 6 + 2 \cdot 5^2 = 86$$

distinct board transformations.[3] Restricting the reference action-sequence length to the range 1–9 therefore yields approximately

$$10^{36} \times \sum_{k=1}^{9} 86^k \approx 10^{36} \times 2.60 \times 10^{17} \approx 2.60 \times 10^{53}$$

possible pairs of initial boards and reference action sequences. Although different action sequences may produce the same target board from a given initial board when their effects overlap or cancel, the task space of TranSGrid remains extremely large. Instances can be generated automatically by sampling an initial board and executing a reference action sequence to obtain the target board, guaranteeing a valid solution by construction. Together with its controllability and flexibility, this scale makes TranSGrid an ideal platform for developing new methods to improve models' systematic generalization performance and investigating the mechanisms underlying this capability.

4 Experimental Results on TranSGrid

4.1 The three reasoning forms in TranSGrid

TranSGrid is designed to jointly engage deductive, inductive, and abductive reasoning within a unified task. Although each form corresponds to a distinct aspect of the task, the associated reasoning

[2] All experiments in this paper use 6×6 boards, but for visual clarity, we use 4×4 boards in the figures.

[3] The ten types of atomic actions can be applied in $8 \cdot 6 + 2 \cdot 5^2 = 98$ ways. Among them, `ROW_UP`/`ROW_DOWN` and `COL_LEFT`/`COL_RIGHT` form 12 equivalent pairs, leaving 86 distinct transformations.

demands are inherently interdependent. We next describe how these forms manifest in TranSGrid and introduce the corresponding metrics.

Deductive Reasoning Deductive reasoning in TranSGrid involves inferring the next board state from the current state and action. Since each additional action demands one more step of forward inference over the board state, we use the reference sequence length as a proxy for deductive difficulty, with longer sequences indicating greater deductive load.

Inductive Reasoning The atomic actions in TranSGrid operate on multiple cells. However, when combined in specific ways, their intermediate effects can cancel out, leaving only a few cells permuted. The inductive challenge lies in discovering reusable composition rules from observed examples of such interactions and applying these rules to new instances. We refer to action combinations that realize these localized permutations as induction rules and manually define 12 such rules for constructing TranSGrid instances (see Appendix A for details). Accordingly, we use the number of induction rules instantiated in the reference sequence as a proxy for inductive difficulty, with larger counts representing higher inductive load.

Abductive Reasoning The task formulation of TranSGrid inherently involves abductive reasoning, as the model needs to infer a plausible sequence of actions from the observed initial and target boards. When actions overwrite or cancel one another's effects, the resulting boards can provide fewer observable cues about the actions involved. To quantify this concealment, we let D denote the number of cells that differ between the initial and target boards. We then map D to an effective action length L_{eff}, which estimates how many random actions would typically produce the same amount of observable change (see Appendix B). Given the reference sequence length L, we define $H_{\text{act}} = L - L_{\text{eff}}$ as a proxy for action concealment: larger values indicate that the reference sequence contains more actions than its observable effect would typically suggest. We then normalize this proxy to obtain an abductive score:

$$A_{\text{abd}} = \sigma\left(\frac{H_{\text{act}} - 6}{2}\right),$$

where $\sigma(x) = 1/(1 + e^{-x})$. Using the maximum reference sequence length $L_{\max} = 12$, we set the sigmoid midpoint to $L_{\max}/2 = 6$ and its scale to $L_{\max}/6 = 2$. This calibration gives $A_{\text{abd}} = 0.5$ at $H_{\text{act}} = 6$, with scores of approximately 0.05 and 0.95 at $H_{\text{act}} = 0$ and 12, respectively. For analysis, we discretize A_{abd} into five equal-width levels (1–5), defined by the intervals $[0, 0.2), [0.2, 0.4), \ldots, [0.8, 1)$, with higher levels indicating greater abductive load.

4.2 Experimental Setup

Model Following prior work on systematic generalization (Kim & Linzen, 2020; Lake & Baroni, 2023; Li et al., 2023; Kumon & Yanaka, 2025), we train standard encoder-decoder Transformers to map an initial–target board pair to an action sequence. The encoder jointly processes both boards, while the decoder autoregressively generates action-name and argument tokens one by one (Figure 1b). We evaluate seven models ranging from 0.96M to 88.43M parameters, indexed from 1 to 7 by increasing parameter count. Model configurations, including the numbers of encoder and decoder layers, hidden dimensions, and attention heads, are detailed in Appendix C.

Dataset All experiments in this paper use 6×6 boards with cell values independently sampled from $\{0, \ldots, 9\}$. For clarity, however, we use 4×4 boards in all figures. Each instance contains an initial board, a target board, and a reference sequence that transforms the former into the latter. We construct a TranSGrid evaluation set of 4,800 instances, marginally balanced across reference sequence lengths $L \in [1, 12]$ and numbers of induction rules $K \in [0, 3]$. Based on L, we divide the set into Easy ($L \in [1, 4]$), Medium ($L \in [5, 8]$), and Hard ($L \in [9, 12]$), each containing 1,600 instances. Appendix D provides details on dataset construction and the joint L–K distribution.

Implementation details The two boards and their spatial relations can be encoded in multiple ways. We therefore evaluate four combinations of board and positional encodings and use the best overall configuration, `GRID+PAIR`, in all experiments (see Appendix E for a detailed comparison). All seven models are trained for eight epochs on 30M instances with reference sequence lengths ranging from 1 to 9. For each model, we select the checkpoint with the highest solve rate on a 2,700 instance development set. Appendix F details the generation procedure for the training and development sets. Using the same procedure, we construct a 4,800 instance Test set. Its reference

sequence lengths are uniformly distributed from 1 to 12, and no Test instance appears in either the training or development set. We report results under greedy decoding (Greedy) and beam search (Top@8); for the latter, an instance is considered solved if any of the eight candidates is correct.

4.3 Main Results

Table 1: Solve rates (%) on the standard Test set and TranSGrid. The overall TranSGrid score is the unweighted mean over the Easy, Medium, and Hard subsets. Top@8 counts an instance as solved if any of its eight beam-search candidates succeeds upon execution.

		Test		TranSGrid		$\text{TranSGrid}_{\text{Easy}}$		$\text{TranSGrid}_{\text{Medium}}$		$\text{TranSGrid}_{\text{Hard}}$	
ID	Params (M)	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8
1	0.96	25.81	36.38	16.10	20.25	46.63	57.06	1.44	3.06	0.25	0.63
2	1.89	35.79	51.54	18.10	23.77	48.94	61.56	4.88	8.63	0.50	1.13
3	7.45	61.85	64.12	26.83	33.02	64.56	76.31	12.88	18.38	3.06	4.38
4	11.13	71.27	72.50	32.06	40.85	71.13	86.69	19.69	28.19	5.38	7.69
5	44.29	73.90	75.15	37.73	47.90	78.13	94.00	27.94	39.13	7.13	10.56
6	59.00	78.33	78.90	50.40	58.56	92.06	97.94	45.00	59.75	14.13	18.00
7	88.43	79.63	80.17	55.31	61.60	95.94	98.38	54.25	65.19	15.75	21.25

As shown in Table 1, models with IDs 4–7 achieve greedy solve rates of 71.27%–79.63% on Test, suggesting strong generalization under standard held-out evaluation. Across all seven models and both decoding strategies, however, TranSGrid solve rates are significantly lower than those on Test and decline sharply from Easy to Hard. Larger models and Top@8 improve solve rates, but performance remains limited on the more difficult subsets: under Top@8, even ID 7 achieves only 65.19% on $\text{TranSGrid}_{\text{Medium}}$ and 21.25% on $\text{TranSGrid}_{\text{Hard}}$.

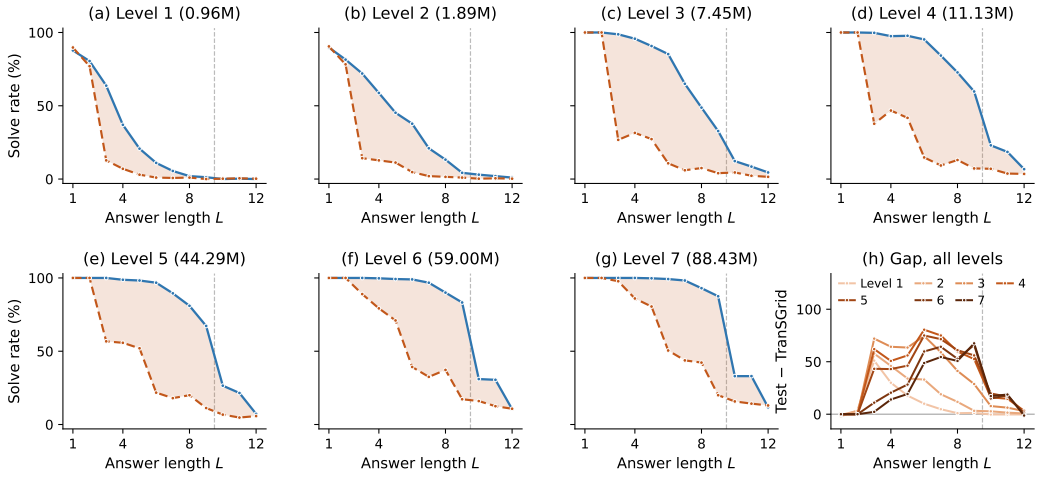


Figure 2: Solve rate (%) versus reference sequence length L under greedy decoding. Panels (a)–(g) show results for model IDs 1–7, respectively. In each panel, — and - - show performance on Test and TranSGrid, respectively, while ▭ marks the gap between them. ⋮ marks the maximum training length, $L = 9$, with larger values requiring length extrapolation. Panel (h) summarizes the Test–TranSGrid gaps for all seven models, with darker curves indicating larger models.

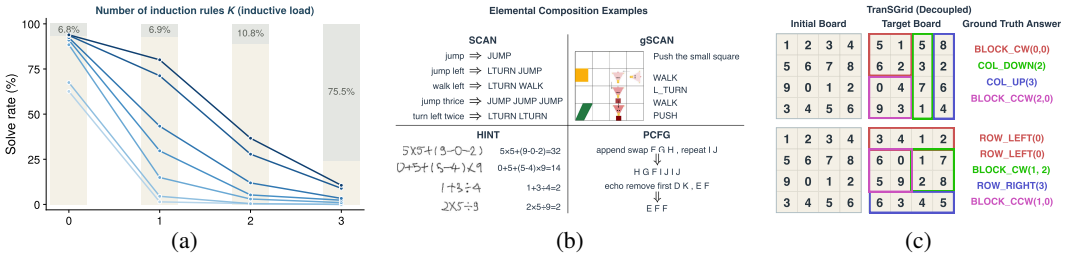
To examine whether this gap persists at matched reference sequence lengths, we compare Test and TranSGrid solve rates in Figure 2. A substantial gap remains for $L \leq 9$, even though this range is fully covered during training and requires no length extrapolation. Across most of this range, larger models maintain high solve rates on Test, whereas their TranSGrid performance declines sharply as L increases. Panel (h) further shows that increasing model capacity shifts the largest gap toward longer sequences but does not eliminate it within the training-length range. For $L > 9$, the gap narrows as Test performance also declines under length extrapolation. Together, these results

demonstrate the value of TranSGrid as a task for systematic generalization, while also highlighting its unique advantage of not relying on productivity.

5 What Existing Metrics Capture and Miss: A Reasoning-Centered Analysis

5.1 Inductive Reasoning Is Suppressed by Elemental Composition and Not Guaranteed by Productivity

Inductive reasoning involves discovering reusable composition rules from observed examples of how action effects interact and applying these rules to new instances. This demand is largely suppressed by elemental composition, where compound outcomes are explained mainly by individual action effects rather than interaction effects. Nor is it guaranteed by productivity: longer sequences may require only repeated applications of atomic actions, without requiring models to infer new composition rules. To test whether inductive reasoning contributes to systematic generalization beyond productivity and whether elemental composition suppresses this demand, we first analyze performance across inductive loads relative to L and then compare TranSGrid with a decoupled variant.



(a) (b) (c)

Figure 3: (a) Solve rates on TranSGrid as inductive load K increases. Curves from light to dark denote model IDs 1–7, and the shaded bars indicate the proportion of instances requiring length extrapolation. (b) Examples of elemental composition in prior benchmarks, where interactions among atomic operations are limited. (c) Examples from TranSGrid (Decoupled), where actions operate on separate parts of the board to reduce action interactions and approximate elemental composition.

Inductive reasoning presents a distinct challenge for systematic generalization that productivity alone does not capture. As shown in Figure 3a, solve rates decline sharply across all seven models as inductive load K increases. More importantly, this decline is already pronounced even when productivity demands remain limited: as indicated by the shaded bars, only 10.8% of the instances at $K = 2$ require length extrapolation, yet solve rates are below 40% for all models.

Table 2: Comparison results on Test, TranSGrid, and TranSGrid (Decoupled). Δ denotes the absolute gain over the corresponding TranSGrid results.

		Test		TranSGrid		TranSGrid (Decoupled)		Δ	
ID	Params (M)	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8
1	0.96	25.81	36.38	16.10	20.25	23.23	46.77	7.13	26.52
2	1.89	35.79	51.54	18.10	23.77	26.54	58.37	8.44	34.60
3	7.45	61.85	64.12	26.83	33.02	70.79	73.06	43.96	40.04
4	11.13	71.27	72.50	32.06	40.85	74.90	76.00	42.84	35.15
5	44.29	73.90	75.15	37.73	47.90	75.90	76.98	38.17	29.08
6	59.00	78.33	78.90	50.40	58.56	77.96	79.04	27.56	20.48
7	88.43	79.63	80.17	55.31	61.60	78.42	79.87	23.11	18.27

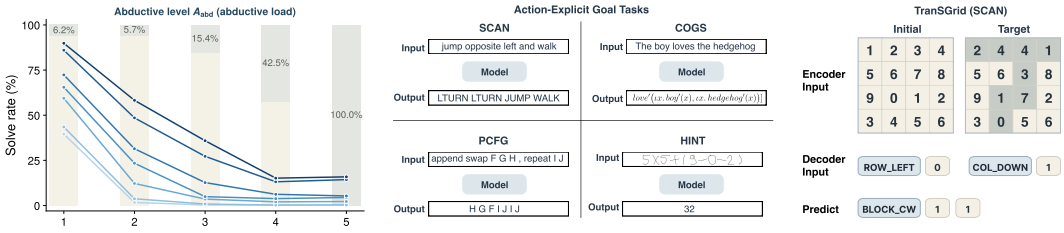
However, elemental composition largely suppresses this inductive demand by limiting interactions among action effects. Figure 3b illustrates this structure in SCAN (Lake & Baroni, 2018), gSCAN (Ruis et al., 2020), HINT (Li et al., 2023), and PCFG (Hupkes et al., 2020). To approximate elemental composition in a controlled setting, we construct TranSGrid (Decoupled), as shown in Figure 3c. Specifically, TranSGrid (Decoupled) is constructed so that actions operate on separate

parts of the board, thereby minimizing interference among their effects. We apply this decoupling to instances with $L \leq 9$, which require no length extrapolation, while leaving instances with $L \geq 10$ unchanged and preserving the initial boards and per-length counts.

As shown in Table 2, decoupling consistently improves solve rates across all model sizes and both decoding strategies. Gains reach 43.96 percentage points under greedy decoding and 40.04 points under Top@8. Performance recovers to roughly the Test level or higher for model IDs 3–7 under both decoding strategies. For example, the largest model improves from 55.31% to 78.42% under greedy decoding and from 61.60% to 79.87% under Top@8, approaching its Test solve rates of 79.63% and 80.17%, respectively. Because the reference action sequence length distribution remains unchanged, this recovery reflects reduced action interactions rather than reduced productivity demands. Overall, these results show that elemental composition suppresses this inductive demand, and that productivity alone does not guarantee this form of reasoning.

5.2 Action-Explicit Goals Largely Bypass Abductive Reasoning

Abductive reasoning in TranSGrid involves inferring, from the initial and target boards, which actions could have produced the observed transformation and in what order. Action-explicit goals largely suppress this demand by providing the complete solution sequence, including the actions and their order, in the input. Although the model still needs to generate the complete sequence, its task is largely reduced to mapping the input-specified composition to the corresponding output rather than inferring a suitable composition from scratch. To examine the role of abductive reasoning in systematic generalization and the effect of action-explicit goals, we first analyze how model performance varies with abductive load and then compare TranSGrid with an action-explicit variant.



(a) (b) (c)

Figure 4: (a) Solve rates on TranSGrid as abductive load A_{abd} increases. Curves from light to dark denote model IDs 1–7, and the shaded bars indicate the proportion of instances requiring length extrapolation. (b) Examples of action-explicit tasks in prior work, where the input specifies the complete action sequence to parse or sovle. (c) TranSGrid (SCAN), which likewise requires the model to parse or execute an almost complete action sequence before producing the final answer.

As presented in Figure 4a, solve rates decline sharply across all seven models as abductive load increases. Notably, between abductive levels 1 and 2, solve rates fall substantially even as the proportion of instances requiring length extrapolation decreases from 6.2% to 5.7%. These results indicate that abductive reasoning is an important dimension of systematic generalization that productivity alone does not fully capture.

Despite its importance, many prior benchmarks, such as SCAN (Lake & Baroni, 2018), COGS (Kim & Linzen, 2020), PCFG (Hupkes et al., 2020), and HINT (Li et al., 2023), evaluate systematic generalization through action-explicit tasks, in which the input specifies the complete action sequence (Figure 4b). Such tasks directly evaluate whether a model can parse or execute a novel composition, but have the side effect of reducing the need for abductive reasoning in systematic generalization. To examine the effect of this design in a controlled setting, we construct an action-explicit variant of TranSGrid, denoted TranSGrid (SCAN), as shown in Figure 4c. TranSGrid (SCAN) leaves the encoder input unchanged but provides the decoder with the first $L - 1$ gold actions, requiring the model to parse or execute an almost complete action sequence before producing the final answer, thereby substantially reducing the abductive demand.

Table 3 shows that providing the gold action prefix consistently improves solve rates across all model sizes and both decoding strategies. The gains reach 23.06 percentage points for model ID 5 under

Table 3: Comparison results on Test, TranSGrid, and TranSGrid (SCAN). Δ denotes the absolute gain over the corresponding TranSGrid result.

		Test		TranSGrid		TranSGrid (SCAN)		Δ	
ID	Params (M)	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8	Greedy	Top@8
1	0.96	25.81	36.38	16.10	20.25	32.69	62.00	16.59	41.75
2	1.89	35.79	51.54	18.10	23.77	37.90	69.65	19.80	45.88
3	7.45	61.85	64.12	26.83	33.02	45.21	73.62	18.38	40.60
4	11.13	71.27	72.50	32.06	40.85	53.02	74.73	20.96	33.88
5	44.29	73.90	75.15	37.73	47.90	60.79	72.81	23.06	24.91
6	59.00	78.33	78.90	50.40	58.56	62.31	72.85	11.91	14.29
7	88.43	79.63	80.17	55.31	61.60	65.29	78.35	9.98	16.75

greedy decoding and 45.88 points for model ID 2 under Top@8. Under Top@8, the smallest model improves from 20.25% to 62.00%, surpassing its Test result of 36.38%, while the largest model improves from 61.60% to 78.35%, approaching its Test result of 80.17%. This recovery arises primarily from reduced abductive demand rather than from the shorter output length: although the model only needs to predict the final action, it still needs to execute the given prefix and track intermediate states before producing the answer, as in prior action-explicit tasks. Taken together, our results show that tasks with action-explicit goals largely reduce the need for abductive reasoning and thus provide an incomplete picture of systematic generalization.

5.3 A Reasoning-Centered Lens on Systematic Generalization

The preceding analyses reveal two important blind spots in existing work on systematic generalization. Elemental composition suppresses the inductive demand of generalizing from observed action interactions to reusable composition patterns applicable to new task instances, whereas action-explicit goals bypass the abductive demand of inferring a solution sequence from the desired outcome. Many prior studies rely on productivity as primary evidence of systematic generalization. Our results show, however, that productivity is neither necessary for systematic generalization nor sufficient to capture its full reasoning demands.

A comprehensive evaluation should therefore go beyond productivity and engage all three forms of reasoning. Specifically, deduction derives consequences from given rules and states; induction discovers reusable rules from observed examples; and abduction infers plausible action sequences that can achieve the desired outcome. Despite their distinct roles, these reasoning forms are closely coupled: each composition rule involves multiple actions, while longer sequences provide more opportunities for action interactions and concealment. This interplay is reflected in TranSGrid: longer sequences increase deductive load and provide more opportunities to introduce inductive and abductive demands, while reducing either of the latter makes the task much easier. We hope that, by capturing these coupled reasoning demands, TranSGrid could advance research on the mechanisms of systematic generalization and support the development of algorithms to improve this capability.

6 Conclusion

This paper examines what common simplifications leave untested in systematic generalization through a reasoning-centered lens. To support this analysis, we introduce TranSGrid, a controllable testbed that jointly engages deduction, induction, and abduction within a unified task. Experiments with seven Transformer models on TranSGrid show that solve rates are much lower than those on the standard held-out test set. This gap persists on instances whose reference sequence lengths are within the training range, indicating that productivity alone is insufficient to evaluate systematic generalization. We further introduce TranSGrid (Decoupled) to approximate elemental composition and TranSGrid (SCAN) to make goals largely action-explicit. Both variants improve performance to roughly the held-out test level, suggesting that these simplifications reduce the inductive and abductive demands of systematic generalization. Therefore, a comprehensive evaluation of systematic generalization should go beyond productivity and jointly engage all three forms of reasoning.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, and assisting with translation. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results, and providing critical ingredients for proving mathematical claims and assisting in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, drafting parts of a research paper, summarizing or analyzing existing literature, sourcing/searching for information, editing the paper to improve readability, and proposing a title or keywords for the paper. We have reviewed all AI-assisted work: LLM-generated code was checked and verified for correctness by the authors; all reported results come from experiments we ran and inspected ourselves; AI-polished text was manually reviewed for factual accuracy, originality, and consistency with our intended claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All datasets used in this work are generated programmatically from numerical grids and predefined transformation rules. The study does not involve human participants or the collection or use of personal or sensitive data. Our experiments are conducted entirely within an abstract grid-transformation environment and aim to advance the understanding of systematic generalization. Given the scope of the data, task, and experiments, we do not identify any specific ethical concerns arising from this work. We plan to release the datasets and code for data generation, training, and evaluation to support reproducibility and further research.

REPRODUCIBILITY STATEMENT

For our submission, we have uploaded the entirety of the source code as a zipped file that has been properly anonymized. The source code contains inline documentation that details purpose and usage of different parts of the codebase. In addition, we also include the full set of model responses and the evaluation script. As discussed in the ethics statement, we plan to more formally release TranSGrid to the public as an open source repository with thorough details that describes the framework, outlines the code, and details its usage.

REFERENCES

Sravanti Addepalli, Yerram Varun, Arun Suggala, Karthikeyan Shanmugam, and Prateek Jain. Does safety training of llms generalize to semantically related natural prompts? In *International Conference on Learning Representations*, volume 2025, pp. 43611–43631, 2025.

Anthony Chemero. Abduction and deduction in dynamical cognitive science. *Topics in Cognitive Science*, 18(3):e12692, 2026.

Yongqiang Chen, Quanming Yao, Juzheng Zhang, James Cheng, and Yatao Bian. Hierarchical graph tokenization for molecule-language alignment. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=wpbNczwAwV.

Yu Chen, Ting Lei, Yaoyi Li, Jia Cai, Zhecen Wu, and Yang Liu. Rule-compliant visual spatial planning for multimodal large language models. *arXiv preprint arXiv:2608.20237*, 2026a.

Yueh-Han Chen, Guy Davidson, and Brenden Lake. Sage-eval: Evaluating llms for systematic generalizations of safety facts. *Advances in Neural Information Processing Systems*, 38, 2026b.

Nathan Cloos, Meagan Jens, Michelangelo Naim, Yen-Ling Kuo, Ignacio Cases, Andrei Barbu, and Christopher J Cueva. Baba is ai: Break the rules to beat the benchmark. *arXiv preprint arXiv:2407.13729*, 2024.

Do Huu Dat, Nam Hyeon-Woo, Po-Yuan Mao, and Tae-Hyun Oh. Vsc: Visual search compositional text-to-image diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19153–19162, 2025.

Jean-Marc Devaud, Thomas Papouin, Julie Carcaud, Jean-Christophe Sandoz, Bernd Grünewald, and Martin Giurfa. Neural substrate for higher-order learning in an insect: mushroom bodies are necessary for configural discriminations. *Proceedings of the National Academy of Sciences*, 112 (43):E5854–E5862, 2015.

Katherine Duncan, Bradley B Doll, Nathaniel D Daw, and Daphna Shohamy. More than the sum of its parts: a role for the hippocampus in configural reinforcement learning. *Neuron*, 98(3): 645–657, 2018.

Xiyan Fu and Wei Liu. Reinforcement learning for compositional generalization with outcome-level optimization. *arXiv preprint arXiv:2605.04920*, 2026.

Xu Han, Linghao Jin, Xiaofeng Liu, and Paul Pu Liang. Progressive compositionality in text-to-image generative models. In *International Conference on Learning Representations*, volume 2025, pp. 83268–83290, 2025.

Kaiyi Huang, Chengqi Duan, Kaiyue Sun, Enze Xie, Zhenguo Li, and Xihui Liu. T2i-compbench++: An enhanced and comprehensive benchmark for compositional text-to-image generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(5):3563–3579, 2025.

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

Ahmad Jabbar, Cleo Condoravdi, and Christopher Potts. Distinguishing fair from unfair compositional generalization tasks. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 20796–20807, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1133. URL https://aclanthology.org/2025.findings-emnlp.1133/.

Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Lanqing Li, Long-Kai Huang, Tingyang Xu, Yu Rong, Jie Ren, Ding Xue, et al. Drugood: Out-of-distribution dataset curator and benchmark for ai-aided drug discovery–a focus on affinity prediction problems with noise annotations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8023–8031, 2023.

Najoung Kim and Tal Linzen. Cogs: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)*, pp. 9087–9105, 2020.

Ryoma Kumon and Hitomi Yanaka. Analyzing the inner workings of transformers in compositional generalization. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8529–8540, 2025.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pp. 2873–2882. PMLR, 2018.

Brenden M Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, 2023.

Bryan Qi Zheng Leong, Alejandro J Estudillo, and Ahamed Miflah Hussain Ismail. Holistic and featural processing's link to face recognition varies by individual and task. *Scientific Reports*, 13 (1):16869, 2023.

Jiatong Li, Junxian Li, Weida Wang, Yunqing Liu, Changmeng Zheng, Yatao Bian, Dongzhan Zhou, Xiao-Yong Wei, and Qing Li. Speak-to-structure: Evaluating llms in open-domain natural language-driven molecule generation. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 9314–9325, 2026a.

Jiatong Li, Weida Wang, Changmeng Zheng, Shufei Zhang, Yatao Bian, Xiao-yong Wei, and Qing Li. Do llms truly generalize in the molecular domain? a perturbation-based analysis. *arXiv preprint arXiv:2607.01800*, 2026b.

Qing Li, Siyuan Huang, Yining Hong, Yixin Zhu, Ying Nian Wu, and Song-Chun Zhu. A minimalist dataset for systematic generalization of perception, syntax, and semantics. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=kIPyTuEZuAK.

Philipp Mondorf, Shijia Zhou, Monica Riedler, and Barbara Plank. Compositional-ARC: Assessing systematic generalization in abstract spatial reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=h497VpgFKd.

Charles Sanders Peirce. *Collected papers of charles sanders peirce*, volume 5. Harvard University Press, 1934.

Pia Rotshtein, Joy J Geng, Jon Driver, and Raymond J Dolan. Role of features and second-order spatial relations in face discrimination, face recognition, and individual face skills: Behavioral and functional magnetic resonance imaging data. *Journal of Cognitive Neuroscience*, 19(9):1435–1452, 2007.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. *Advances in neural information processing systems*, 33:19861–19872, 2020.

Gary Shank. The extraordinary ordinary powers of abductive reasoning. *Theory & psychology*, 8 (6):841–860, 1998.

Ankur Sikarwar, Arkil Patel, and Navin Goyal. When can transformers ground and compose: Insights from compositional generalization benchmarks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 648–669, 2022.

Sam Spilsbury, Pekka Marttinen, and Alexander Ilin. Generating demonstrations for in-context compositional generalization in grounded language learning. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 15960–15991, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.893. URL https://aclanthology.org/2024.emnlp-main.893/.

Zhengxuan Wu, Elisa Kreiss, Desmond Ong, and Christopher Potts. ReaSCAN: Compositional reasoning in language grounding. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. URL https://openreview.net/forum?id=Rtquf4Jk0jN.

Zhengxuan Wu, Christopher D Manning, and Christopher Potts. Recogs: How incidental details of a logical form overshadow an evaluation of semantic interpretation. *Transactions of the Association for Computational Linguistics*, 11:1719–1733, 2023.

A Induction Rules

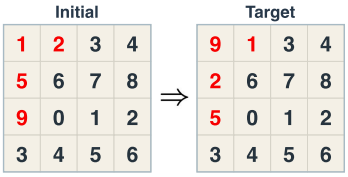
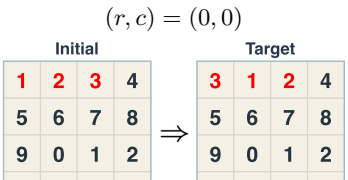
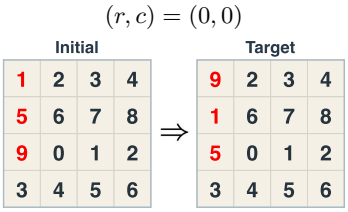
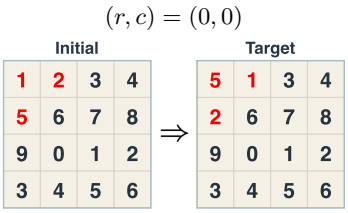
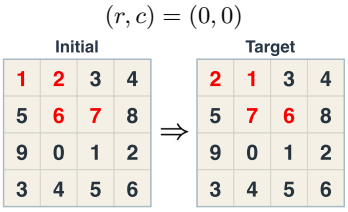
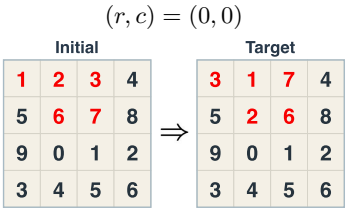
An induction rule is a short program of three to five atomic actions whose effects cancel or overwrite one another, leaving a net transformation that permutes only a few cells within a small local window. Table 4 lists the twelve induction rules used in this paper, together with their constituent action sequences and net effects. Here, (r, c) denotes the top-left corner of the local window. For a block operation such as BLOCK_CW$(r{+}i, c{+}j)$, the two arguments specify the top-left cell of the 2×2 block to be rotated, where i and j are offsets from the rule origin. For row and column operations, $r{+}i$ and $c{+}j$ specify the corresponding row and column indices, respectively. Collectively, the twelve rules realize localized permutations of two to five cells, including pairwise swaps, three to five cell cycles, and two disjoint swaps arranged in horizontal, vertical, diagonal, L-shaped, or bent configurations. The actions in each program are executed from top to bottom. The final column

illustrates the resulting net effect using a concrete example. For each example, the values of r and c are provided above the boards, and the cells involved in the permutation are highlighted in red.

Table 4: The twelve induction rules used in this paper, together with their constituent action sequences and illustrative net effects. In each illustration, the rule origin (r, c) is indicated above the boards, and the cells permuted by the rule are highlighted in red.

Induction Rule	Action sequence	Illustrative net effect
`horizontal_swap_mixed`	`BLOCK_CW`(r, c) `COL_DOWN`(c) `ROW_LEFT`$(r{+}1)$ `COL_UP`(c) `ROW_RIGHT`$(r{+}1)$	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 ⇒ Target: 2 1 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6
`diagonal_swap_blocks`	`BLOCK_CW`$(r, c{+}1)$ `BLOCK_CCW`$(r{+}1, c)$ `BLOCK_CCW`(r, c) `BLOCK_CCW`$(r, c{+}1)$ `BLOCK_CW`$(r{+}1, c)$	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 ⇒ Target: 6 2 3 4 / 5 1 7 8 / 9 0 1 2 / 3 4 5 6
`bent_four_cycle`	`BLOCK_CW`(r, c) `BLOCK_CW`$(r, c{+}1)$ `BLOCK_CCW`(r, c)	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 ⇒ Target: 2 7 1 4 / 5 6 3 8 / 9 0 1 2 / 3 4 5 6
`wide_bent_four_cycle`	`BLOCK_CW`$(r, c{+}1)$ `BLOCK_CW`(r, c) `BLOCK_CCW`$(r, c{+}1)$	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 ⇒ Target: 5 2 3 4 / 7 1 6 8 / 9 0 1 2 / 3 4 5 6
`tall_bent_four_cycle`	`BLOCK_CW`(r, c) `BLOCK_CW`$(r{+}1, c)$ `BLOCK_CCW`(r, c)	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 ⇒ Target: 1 6 3 4 / 5 9 7 8 / 0 2 1 2 / 3 4 5 6

continued on the next page

Induction Rule	Action sequence	Illustrative net effect
`tall_bent_flipped`	BLOCK_CW$(r{+}1, c)$ BLOCK_CW(r, c) BLOCK_CCW$(r{+}1, c)$	$(r, c) = (0, 0)$ 
`horizontal_three_cycle`	ROW_LEFT(r) COL_LEFT$(c{+}1)$ ROW_RIGHT(r) COL_LEFT$(c{+}1)$	$(r, c) = (0, 0)$ 
`vertical_three_cycle`	COL_UP(c) ROW_DOWN(r) COL_DOWN(c) ROW_DOWN(r)	$(r, c) = (0, 0)$ 
`l_three_cycle`	COL_UP(c) ROW_LEFT(r) COL_DOWN(c) ROW_RIGHT(r)	$(r, c) = (0, 0)$ 
`double_horizontal_swap`	BLOCK_CW(r, c) BLOCK_CW$(r, c{+}1)$ BLOCK_CCW(r, c) BLOCK_CCW$(r, c{+}1)$	$(r, c) = (0, 0)$ 
`five_cycle_2x3`	BLOCK_CW(r, c) COL_RIGHT$(c{+}1)$ BLOCK_CCW(r, c) COL_RIGHT$(c{+}1)$	$(r, c) = (0, 0)$ 

continued on the next page

Induction Rule	Action sequence	Illustrative net effect
`vertical_swap_blocks`	`BLOCK_CW`(r, c) `BLOCK_CW`$(r{+}1, c)$ `BLOCK_CCW`$(r, c{+}1)$ `BLOCK_CCW`$(r{+}1, c)$ `BLOCK_CW`$(r, c{+}1)$	$(r, c) = (0, 0)$ Initial: 1 2 3 4 / 5 6 7 8 / 9 0 1 2 / 3 4 5 6 $\Rightarrow$ Target: 5 2 3 4 / 1 6 7 8 / 9 0 1 2 / 3 4 5 6

B Estimating the Effective Action Length

Let D denote the number of cells whose values differ between the initial and target boards. To convert D into an effective action length, we first estimate the expected footprint of a random trajectory:

$$g(\ell) = \mathbb{E}[D \mid \ell],$$

where ℓ denotes the action-sequence length. For each $\ell \in \{1, \ldots, 12\}$, we sample 400 random trajectories of length ℓ on 6×6 boards whose cell values are drawn independently from $\{0, \ldots, 9\}$. We execute each trajectory, count the cells whose values differ between its initial and final boards, and average these counts. The resulting footprint curve is reported in Table 5.

Table 5: The estimated footprint curve. Each value is averaged over 400 random trajectories of the corresponding length.

Length ℓ	1	2	3	4	5	6	7	8	9	10	11	12
Footprint $g(\ell)$	7.2	12.5	16.0	19.1	21.3	22.9	24.4	25.0	26.3	27.0	27.3	27.8

We obtain L_{eff} by inverting this curve through linear interpolation:

$$L_{\text{eff}}(D) = \begin{cases} \dfrac{D}{g(1)}, & 0 \le D \le g(1), \\ \ell + \dfrac{D - g(\ell)}{g(\ell+1) - g(\ell)}, & g(\ell) < D \le g(\ell+1), \quad \ell \in \{1, \ldots, 11\}, \\ 12, & D > g(12). \end{cases}$$

For $D \le g(1)$, the effective length is interpolated between $(D, L_{\text{eff}}) = (0, 0)$ and $(g(1), 1)$. Values between adjacent entries of the footprint curve are linearly interpolated, while values above $g(12)$ are assigned an effective action length of 12.

C Model Architectures

We evaluate seven encoder-decoder Transformer models that vary in depth and hidden dimensions. In every configuration, the encoder and decoder contain the same number of layers. The encoder jointly processes the initial and target boards, while the decoder generates the action sequence autoregressively. Each action is represented by an action-name token followed by its argument tokens, and decoding terminates when the model generates the `EOS` token.

Table 6 summarizes the architectural configurations of the seven models. Here, d_{model} is the number of units in each Transformer bottleneck layer, $d_{\text{ff}} = 4d_{\text{model}}$ is the number of units in each feed-forward sublayer, and Heads is the number of attention heads. The reported parameter counts include all trainable parameters under the `GRID+PAIR` encoding used in the main experiments.

D Dataset Construction and Composition

Instance construction. Each instance begins with an initial board $B^{(0)} \in \mathcal{V}^{6\times6}$, whose cells are sampled independently from $\mathcal{V} = \{0, \ldots, 9\}$. We generate a reference action sequence $\pi =$

Table 6: Details of the seven Transformer models used in this paper. Model IDs are ordered by the total number of trainable parameters.

ID	Enc. layers	Dec. layers	d_{model}	Heads	d_{ff}	Params (M)
1	2	2	128	2	512	0.96
2	4	4	128	2	512	1.89
3	4	4	256	4	1024	7.45
4	6	6	256	4	1024	11.13
5	6	6	512	8	2048	44.29
6	8	8	512	8	2048	59.00
7	12	12	512	8	2048	88.43

$(a_1, \ldots, a_L)$ and execute it to obtain the target board:

$$B^\star = T(\pi, B^{(0)}).$$

The reference sequence guarantees the existence of a valid solution, but it need not be the shortest or unique solution. To construct an instance with inductive load K, we sample K induction rules with replacement, subject to $\sum_{k=1}^{K} |\rho_k| \leq L$, and instantiate each rule at a valid board location. We fill the remaining $L - \sum_{k=1}^{K} |\rho_k|$ positions with randomly sampled atomic actions. The instantiated rules and individual filler actions are then shuffled as blocks, preserving the internal action order of each rule. The induction rules are defined in Appendix A.

Dataset composition. The TranSGrid evaluation set contains 4,800 instances and is marginally balanced with respect to reference sequence length L and inductive load K. Each length $L \in \{1, \ldots, 12\}$ contains 400 instances, and each load $K \in \{0, \ldots, 3\}$ contains 1,200 instances. Because every induction rule contains at least three actions, a feasible pair must satisfy $L \geq 3K$. Table 7 reports the resulting joint distribution.

Table 7: Joint distribution of the $4{,}800$ TranSGrid evaluation instances by reference sequence length L and inductive load K.

Subset	Reference length L	Inductive load K				Total
		0	1	2	3	
Easy	1	400	–	–	–	400
	2	400	–	–	–	400
	3	66	334	–	–	400
	4	65	335	–	–	400
Medium	5	71	329	–	–	400
	6	36	31	333	–	400
	7	16	26	358	–	400
	8	30	34	336	–	400
Hard	9	35	28	43	294	400
	10	34	30	39	297	400
	11	24	27	46	303	400
	12	23	26	45	306	400
Total		1,200	1,200	1,200	1,200	4,800

E Input Encoding Variants

Encoding variants. We compare two board encodings and two positional encoding schemes. Under the `PAIR` encoding, the values at corresponding locations of the initial and target boards are represented jointly by a single token, producing an input sequence of 36 tokens. Under the `BOARD` encoding, the two boards are serialized separately, producing an input sequence of 86 tokens. For positional encoding, `GRID` represents each position as the sum of learned embeddings for its board,

row, and column, whereas FLAT assigns a learned embedding to each absolute sequence position. Combining these choices yields four variants: GRID+PAIR, FLAT+PAIR, GRID+BOARD, and FLAT+BOARD.

Table 8: Development-set solve rates (%) for the four input encoding variants across the seven models. A dagger indicates that the run had not fully converged by the end of training.

Model ID	GRID+PAIR	FLAT+PAIR	GRID+BOARD	FLAT+BOARD
1	32.93	**36.11**	20.78†	33.22
2	**45.89**	44.63	41.93	45.63
3	79.70	**82.70**	79.78	81.74
4	**89.33**	89.04	87.44	88.22
5	91.85	92.26	91.96	**95.00**
6	96.70	96.48	**97.19**	44.96†
7	**97.70**	97.59	97.37	80.22†

Evaluation protocol. We evaluate all four variants using each of the seven model architectures in Appendix C, resulting in 28 runs. All runs use the same 30M training instances, eight training epochs, a batch size of 512, and a random seed of 42. We use AdamW with a peak learning rate of 1.5×10^{-4}, 1,000 warmup steps, weight decay of 0.01, gradient clipping at 1.0, and dropout of 0.1. Table 8 reports greedy solve rates on the held-out development set after the final epoch.

Encoding selection. For model ID 7, the three converged variants differ by only 0.33 percentage points, indicating that their final solve rates are effectively comparable. The PAIR encoding nevertheless provides a shorter input sequence and more stable optimization: all 14 PAIR runs converged within the training budget, whereas three BOARD runs did not. Within the PAIR encoding, GRID and FLAT achieve nearly identical solve rates at the largest model sizes. Nevertheless, across all seven model sizes, GRID+PAIR achieves the highest solve rate in three cases, more than any other variant, and is therefore selected for the main experiments.

F Training and Development Sets

The training set contains 30M instances with reference sequence lengths $L \in \{1, \ldots, 9\}$. For each instance, we sample L using the weights in Table 9, and then sample L actions independently from the ten atomic action types, with their arguments randomly sampled from valid locations. Sequences whose combined effect is the identity transformation are resampled. Each accepted sequence is executed on an initial board to obtain the target board. If the target board is identical to the initial board, the initial board is resampled. These checks exclude instances that can be solved without performing any action, ensuring that every training instance requires at least one action.

Table 9: Sampling weights, counts, and actual shares of the 30M training instances by reference sequence length L. Actual shares differ from the weights by at most 0.015 points.

Reference length L	Sampling weight (%)	Training instances	Actual share (%)
1	4.000	1,197,610	3.992
2	4.000	1,195,963	3.987
3	5.000	1,498,398	4.995
4	6.000	1,797,360	5.991
5	9.000	2,699,700	8.999
6	13.000	3,904,518	13.015
7	17.000	5,101,514	17.005
8	20.000	6,002,666	20.009
9	22.000	6,602,271	22.008
Total	100.000	30,000,000	100.000

Beyond these validity checks, we use a non-uniform length distribution to help the models learn the task more effectively. Preliminary experiments showed that using the same number of instances

at each length led to lower solve rates at lengths 7–9, while increasing the proportion of longer sequences improved performance at these lengths and maintained performance at shorter lengths. We therefore generate the training set using the sampling weights reported in Table 9.

The development set contains 2,700 instances, with 300 instances sampled for each reference sequence length $L \in \{1, \ldots, 9\}$. Unlike the training set, the development set is balanced across reference sequence lengths. Duplicate instances with the same initial and target boards are removed before splitting, and the training and development sets are verified to contain no shared instance.