LEARNING WHEN TO ABORT IN TRANSFORMER PLANNING: A MECHANISTIC STUDY

Anonymous authors
Paper under double-blind review

ABSTRACT

Planning does not by itself imply the ability to recognise errors, but recognising errors can improve planning capacity. In this paper, we study this phenomenon in Transformer planning. To analyze the inner mechanism, we follow the approach of Wang et al. (2024) and abstract planning as path-finding in graphs. In this setting, we train the model to detect the error of invalid transitions, i.e., after the model generates a next node that is not adjacent with the current node, the model should then generate a special abort token x. For expressiveness, we construct a Transformer with feed-forward width $O(|V|)$ for any fixed graph with node set V. It can detect all possible invalid transitions, and select a valid next node if the prefix is valid. We then propose a worst-case lower bound on feed-forward width, which makes the detector's linear width near-optimal up to logarithmic factors. In trained models, we demonstrate that invalid-transition detection increases previous-node attention, and most of the valid–invalid separation locates in the feed-forward network's output. This accords with our theoretical insights. Finally, we use this error detection to improve the model's performance: at inference, generating the abort token x will directly cause an external procedure to discard the attempt and restart from the original prompt. Experiments show that error detection combined with retry improves model's planning reliability on graphs, Countdown, and mazes.

1 INTRODUCTION

Planning involves choosing a sequence of steps that leads from an initial state to a goal, and is ubiquitous in daily life. In recent years, Transformer-based large language models (LLMs) are increasingly used for tasks requiring such multi-step planning, including mathematics and coding. Understanding how these models plan, and improving their planning capacity therefore becomes more and more important. In this paper, we study how learning to recognise invalid planning steps (or invalid transitions), could improve language-model planning. Traditional training on correct planning trajectories teaches a model which steps to take, but does not explicitly teach how to recognise invalid steps. In fact, planning ability does not need to bring failure recognition: a model may be able to find valid solutions yet fail to recognise that a problematic input contains invalid steps.

In human planning, however, failure recognition is also thought to play an important role in planning ability (Alexander & Brown, 2010; Fu et al., 2023).

Can a Transformer-based language model learn to detect invalid planning steps, what could be the underlying mechanism of this learning process, and how can this improve the model's performance?

We conduct a mechanistic study on this problem, based on the path planning setup of ALPINE (Wang et al., 2024). In ALPINE, the planning task is abstracted as looking for valid paths in a fixed graph. The graph is not supplied as an input, hence the model must learn the graph's structure from the training paths and find valid paths for unseen pairs. As for invalid planning steps, we focus on *invalid transition*, which is a step in the generated path from some node u to node v that has no corresponding edge in the underlying graph. To enable the Transformer to detect invalid transitions, we generate new training data containing valid path prefixes followed by an invalid transition, and then followed by a special abort token x. Only the abort token is supervised on these generated data, so that the Transformer could learn to flag invalid transitions without being trained to produce them. Intuitively, this is analogous to adding wrong planning steps and masking them in a planning task for the model to learn from mistakes.

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.

Graph path-finding lets us distinguish two computations. Selecting the next node uses the current node's outgoing edges and reachability to the target; detecting an invalid transition instead requires relating the previous node to the current one.

Learning from search traces and error markers can improve planning (Gandhi et al., 2024; Lehnert et al., 2024; Huang et al., 2025; Yang et al., 2026). Our focus is the mechanism: what computation invalid-transition detection requires, and whether abort supervision induces it in trained models. Theoretically, Proposition 1 of Section 4.1 shows that an additive score of the previous and current nodes with a threshold cannot detect arbitrary invalid transitions. In Section 4.2, we construct a one-layer Transformer that combines next-node selection with invalid-transition detection using the non-linear feed-forward layer with width linear in the number of nodes. The lower bound in Section 4.3 makes the detector's linear width near-optimal up to logarithmic factors within the specified class of ReLU detectors. Empirically, Section 5 connects the abort decision to previous-node information and feed-forward network (FFN) processing in real trained Transformers. Attention measurements show whether the model looks back, interventions test whether previous-node information controls the abort decision, and direct logit attribution measures the FFN's contribution to the abort signal. A fitted additive-logit alternative retains substantial detection error on the same supervised prefixes (Appendix B.2). These findings are consistent with the computational dependencies identified by the theory.

We then ask whether the learned signal improves planning. At inference, a *harness*, an external procedure surrounding the model, discards an attempt when $\mathtt{x}$ is emitted and restarts from the original prompt. Experiments on random graphs, Countdown and mazes show that this procedure improves planning reliability (Section 6). Unlike verifier-ranked best-of-N sampling (Cobbe et al., 2021; Lightman et al., 2024), it does not require a fixed set of complete candidates or a separate correctness verifier: the model supplies the signal for restarting an attempt.

Our contributions are threefold: a simple training method that turns local errors into abort supervision; a mechanistic account of invalid-transition detection, combining a one-layer Transformer construction, a capacity lower bound and experiments on trained models; and an inference-time restart harness that uses the learned signal to improve planning across tasks and models.

2 Related Work

Our mechanistic starting point is ALPINE (Wang et al., 2024), which explains how Transformers select the next node using adjacency and reachability information. We build on this account to study invalid-transition detection. Complementary work studies how models compose reachability information to select future steps (Zhong et al., 2025), represent and maintain the current planning state during search (Phua et al., 2026), and improve planning through reinforcement-learning fine-tuning (Wang et al., 2026).

Our training approach belongs to the line of work on learning from search and error traces. Gandhi et al. (2024) train on complete search trajectories containing exploratory mistakes and backtracking, while Lehnert et al. (2024) train models to imitate A* search dynamics. Closer to our setting, Huang et al. (2025) train on erroneous actions followed by in-context corrections. Their models recognise mistakes, but self-correction does not improve plan validity in their evaluation. We instead separate detection from recovery: the model learns to flag an invalid transition, and an external procedure handles retry.

Self-Backtracking (Yang et al., 2026) is the closest method. It learns a token that triggers rollback and further search from an earlier state; our harness restarts from the original prompt. Its error-example loss supervises both the correct prefix and the backtrack token, whereas ours supervises only the abort token. Our labels require checking one transition, rather than determining that a prefix admits no valid completion, which can require searching over future continuations (Bylander, 1994). In our small-model Countdown experiments, we cross the two objectives with restart and rollback decoders. Abort-only supervision performs better with either decoder, suggesting that the simpler objective contributes to the gain in this setting (Appendix B.4). Our mechanistic analysis further distinguishes this work by combining an explicit Transformer construction with experiments on the learned computation.

Additionally, some work uses separate signals to guide generation. State–action verifiers (Arora & Kambhampati, 2023) check whether an action is valid in the current state. Process-verifier approaches use information about whether a prefix admits a valid completion (Botta et al., 2025), or learned estimates of the quality of partial generations (Rohatgi et al., 2026), to guide sampling and backtracking. In our approach, the planner itself (rather than an independent verifier) learns to emit the abort signal during generation. Confidence-token methods (Chuang et al., 2024) are also related, but estimate the reliability of the overall answer rather than whether the latest step is invalid.

3 Model and Method

3.1 Transformer Architecture

We use the following one-layer architecture for the theoretical analysis, following Wang et al. (2024). Let N be the sequence length, d the embedding size, H the number of attention heads, $d_k = d/H$ the size of each head, and M the vocabulary size. A token sequence is a one-hot matrix $U \in \mathbb{R}^{N\times M}$, embedded as $H_0 = UW_t + W_p$, where $W_t \in \mathbb{R}^{M\times d}$ and $W_p \in \mathbb{R}^{N\times d}$ are the token and positional embedding matrices. The multi-head attention (MHA) sublayer is

$$\mathrm{MHA}(H_0) = \left[\mathrm{softmax}\left(\frac{(H_0W_Q^{(h)})(H_0W_K^{(h)})^\top}{\sqrt{d_k}}\right)H_0W_V^{(h)}\right]_{h=1}^{H} W^A, \tag{1}$$

where the softmax is causally masked and $W_Q^{(h)}$, $W_K^{(h)}$, and $W_V^{(h)}$ are the $d \times d_k$ query, key, and value projections for head h, and the brackets denote horizontal concatenation over heads. The attention output projection is $W^A \in \mathbb{R}^{d\times d}$. With $\tilde{H} = \mathrm{MHA}(H_0) + H_0$, the feed-forward network is $\mathrm{FFN}(\tilde{H}) = \mathrm{ReLU}(\tilde{H}W_1 + b_1)W_2 + b_2$, where $\mathrm{ReLU}(z) = \max(0, z)$ acts elementwise, and the layer output is $H_1 = \mathrm{FFN}(\tilde{H}) + \tilde{H}$. Here $W_1 \in \mathbb{R}^{d\times r}$, $b_1 \in \mathbb{R}^r$, $W_2 \in \mathbb{R}^{r\times d}$, and $b_2 \in \mathbb{R}^d$. The feed-forward width r is conventionally fixed at $4d$. We name r explicitly because it is the resource the abort detector uses. The vocabulary logits (next-token scores before softmax) are H_1W^U, where $W^U \in \mathbb{R}^{d\times M}$ is the unembedding matrix. Layer normalisation is omitted in the above equations.

3.2 Path-Planning Task, Data, and Training

We follow the graph path-finding setup of ALPINE (Wang et al., 2024). Fix a directed acyclic graph (DAG) $G = (V, E)$ with $n = |V|$ nodes. Given a source s and a reachable target t, the model receives the prompt `s t` and needs to generate a node sequence starting at s, ending at t, and following graph edges. We define a *valid path prefix* as a node sequence that starts at s and follows only graph edges, and write $u \rightsquigarrow t$ when u can reach t, including $t \rightsquigarrow t$.

There are two types of training trajectories. Correct training trajectories have the form `s t s` u_1 `...` `t`, where `s` u_1 `...` `t` is a valid path from s to t. These trajectories are used to train the usual sequence generation process, and thus receive the usual next-token loss. Error trajectories add supervision about invalid transitions. Each contains a valid prefix followed by one non-edge $p \to i$ and the token `x`. Only the final `x` is supervised; the prefix and invalid transition provide context, not prediction targets. We call this training signal *abort supervision*. We call the resulting model a $+$`x` model, and a model trained only on correct trajectories the *baseline*. The detailed loss formulation is given in (11) (Appendix B.1).

Our abort labels are local: in graph planning, determining whether a transition $p \to i$ is invalid only requires checking whether $(p, i) \in E$, without reasoning about whether the current prefix can still reach the target. Thus, constructing error trajectories requires only a local edge check, rather than solving the remaining planning problem.

For graph experiments, each edge $i \to j$ with $i < j$ is included independently with probability 0.1. Reachable source–target pairs are randomly split into training and test sets, except that direct-edge pairs always enter training. Each training pair contributes 20 valid paths. The $+$`x` dataset adds one error trajectory per correct trajectory, and minibatches sample uniformly from this mixture. The loss averages over supervised tokens: correct trajectories supply multiple labels, error trajectories only one. Thus, correct trajectories make up half the batch on average but supply most labels.

We denote an architecture by layer count/head count/embedding size and compare $1/1/120$, $1/2/120$ and $1/2/384$. All use feed-forward width $r = 4d$, where d is embedding size. The experiments use

GELUs (Gaussian error linear units), layer normalisation and tied embedding/unembedding weights. Models are trained for 10^4 updates with batch size 1,024 and learning rate 5×10^{-4}; Appendix B.1 gives the full protocol. Graph results use five seeds, each with an independently generated graph. Error bars are standard errors (s.e.) over seeds and gains are paired.

4 Theory of Invalid-Transition Detection in Path-Planning

Let $A^{\text{true}}_{(p,i)} = \mathbf{1}[(p,i) \in E]$ be the graph's adjacency matrix, where $\mathbf{1}[\cdot]$ is 1 when its condition holds and 0 otherwise. The true reachability matrix has entry $R^{\text{true}}_{(t,u)} = \mathbf{1}[u \rightsquigarrow t]$.

4.1 Additive Detection Is Insufficient

To output the abort token correctly, the model needs to determine whether the most recent transition $p \to i$ is valid, that is, whether $(p,i) \in E$. We next show that simply adding separate functions of p and i in a single thresholded score is not sufficient for general graphs.

Proposition 1. *If A^{true} has nodes $p \neq p'$ and $i \neq i'$ with $A^{\text{true}}_{(p,i)} = A^{\text{true}}_{(p',i')} = 1$ and $A^{\text{true}}_{(p,i')} = A^{\text{true}}_{(p',i)} = 0$, then no threshold on an additive function $f(p) + g(i)$ can correctly classify all four transitions as edges or non-edges, whether larger scores indicate validity or invalidity.*

To see why, consider the two edges (p,i) and (p',i'), whose crossed pairs (p,i') and (p',i) are non-edges. For an additive function $\phi(p,i) = f(p) + g(i)$, with $f, g : V \to \mathbb{R}$,

$$\phi(p,i) + \phi(p',i') = \phi(p,i') + \phi(p',i).$$

Thus, the two edge scores and the two non-edge scores have the same total, so no threshold can place both edges on one side and both non-edges on the other.

Thus, general invalid-transition detection with a scalar threshold requires a non-additive combination of previous-node and current-node information. In the next subsection, we show how a Transformer implements this check by selecting the current node's entry from the previous node's adjacency row.

4.2 A Gated Lookup for Invalid Transition Detection

In this subsection, we construct a one-layer, two-head Transformer that performs path planning and invalid-transition detection simultaneously. We consider input sequences of the form "$s\ t\ s\ u_1 \dots u_k$", where $s \rightsquigarrow t$.

Theorem 2. *Given a DAG G with adjacency matrix A^{true} and reachability matrix R^{true}, for every $\varepsilon > 0$ there is a 1-layer, 2-head Transformer with embedding size $O(|V|)$ and feed-forward width $O(|V|)$ that, for every input sequence "$s\ t\ s\ u_1 \dots u_k$" for which $(u_0, \dots, u_{k-1})$ is a valid path prefix, where $u_0 = s$, $k \geq 1$, and $u_k \neq t$, generates with probability at least $1 - \varepsilon$*

- *(i) the token x, if $A^{\text{true}}_{(u_{k-1},\, u_k)} = 0$, and*
- *(ii) otherwise a node u_{k+1} that is both an out-neighbour of u_k and able to reach t, whenever such a node exists.*

Write $p = u_{k-1}$ and $i = u_k$ for the previous and current nodes. One attention head retrieves the target's reachability row $R^{\text{true}}_{(t,:)}$; the other retrieves the previous node's adjacency row $A^{\text{true}}_{(p,:)}$. The residual connection retains the current node's one-hot vector e_i. Thus, at exact attention the relevant FFN input blocks are $R^{\text{true}}_{(t,:)} \oplus A^{\text{true}}_{(p,:)} \oplus e_i$, where $\oplus$ denotes concatenation.

The FFN's first linear map reads the current node's adjacency row $A^{\text{true}}_{(i,:)}$ from e_i, and thus the parameter becomes $R^{\text{true}}_{(t,:)} \oplus A^{\text{true}}_{(p,:)} \oplus e_i \oplus A^{\text{true}}_{(i,:)}$. Its first $|V|$ ReLU units use $R^{\text{true}}_{(t,:)}$ and $A^{\text{true}}_{(i,:)}$, compute $\max(0, R^{\text{true}}_{(t,v)} + A^{\text{true}}_{(i,v)}) = R^{\text{true}}_{(t,v)} + A^{\text{true}}_{(i,v)}$, selecting out-neighbours of i that can reach t (if such nodes exist), as in the adjacency–reachability account of Wang et al. (2024). The other $|V|$ units combine $A^{\text{true}}_{(p,:)}$ with e_i to check the preceding transition. For each node v, a unit computes

$$\max\left(0, \mathbf{1}[i = v] + A^{\text{true}}_{(p,v)} - 1\right) = \mathbf{1}[i = v]\, A^{\text{true}}_{(p,v)}. \tag{2}$$

Their output suppresses the abort logit when the preceding transition is valid (when the sum of detection units is 1), and otherwise do nothing, letting the abort logit be largest (when the sum of detection units is 0).

The detector uses $|V|$ units, while the complete FFN has width $2|V|$. Appendix A.1 gives the detailed architecture and weights and shows that the logit separation persists with finite attention scores.

Remark 1 (Observed reachability and retry). Let $R^{\text{obs}}_{(t,u)} = 1$ when a training path ending at t contains u. Using R^{obs} instead of R^{true} can affect next-node selection, but not invalid-transition detection. With exact checks of every transition (including the final one) and both endpoints, every accepted path is valid without a global ground-truth check. If independent, length-bounded attempts each succeed with fixed probability $\pi > 0$, retry succeeds within K attempts with probability $1-(1-\pi)^K$ and terminates almost surely without a fixed budget. Thus, retry can overcome incomplete observed reachability (Wang et al., 2024).

4.3 How Much Capacity Does the Detector Require?

Suppose that the previous node p, current node i, and target node t are represented at the FFN input by separate one-hot blocks, i.e., $x_{p,i,t} = e_p \oplus e_i \oplus e_t$. A width-$r$ ReLU FFN computes $\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$. Let $W_1^{(p)}, W_1^{(i)}, W_1^{(t)} \in \mathbb{R}^{n\times r}$ be the row blocks of W_1 acting on the three node inputs. The h-th FFN unit receives

$$W_1^{(p)}[p,h] + W_1^{(i)}[i,h] + W_1^{(t)}[t,h] + b_1[h].$$

The FFN's contribution to the abort logit is

$$z_{\mathrm{x}}^{\mathrm{FFN}}(p,i,t) = \big(\mathrm{ReLU}(x_{p,i,t}W_1 + b_1)W_2 + b_2\big)W^U[:,\mathrm{x}]. \tag{3}$$

Since transition validity depends only on (p,i), we can fix any target t_0. Then $W_1^{(t)}[t_0,:]$ is constant and can be absorbed into b_1. Thus, for a fixed target, invalid-transition detection reduces to a width-r ReLU network over the previous and current nodes.

Theorem 3. *Suppose that a width-r one-hidden-layer ReLU FFN receives the previous node p, current node i, and target t as the input $e_p \oplus e_i \oplus e_t$. If, for every DAG on $|V|$ nodes, there exist FFN parameters for which the sign of $z_{\mathrm{x}}^{\mathrm{FFN}}(p,i,t)$ correctly determines whether $(p,i) \in E$ for every $p,i,t \in V$, then*

$$r = \Omega\left(\frac{|V|}{\log^2 |V|}\right).$$

To prove the theorem, we bound the number of adjacency matrices representable by a width-r network using the VC dimension of piecewise-linear networks (Bartlett et al., 2019) and the Sauer–Shelah lemma, and compare it with the $2^{\binom{|V|}{2}}$ DAGs obtained by choosing arbitrary forward edges under a fixed node ordering (Appendix A.2). The detector in Theorem 2 uses $|V|$ ReLU units, matching this worst-case lower bound up to logarithmic factors in this class.

5 Mechanistic Analysis of Abort Supervision

In this section, we examine whether trained models implement the invalid-transition detection mechanism suggested by our theoretical analysis. We study previous-node attention and its causal role in the abort decision, then use direct logit attribution (DLA) to locate the abort signal. Additional analyses of learned adjacency and reachability representations are in Appendix B.2. We compare baseline and $+\mathrm{x}$ models with the 1/2/120 and 1/2/384 architectures. For each graph size and architecture, we train both conditions from scratch on five independently generated graphs (Section 3).

At the beginning, we check that these models can both find valid paths and recognise failures. The 1/2/384 $+\mathrm{x}$ models produce a valid path in more than 90% of single draws at every tested graph size, averaging over the five graphs at each size (Figure 2a). Their abort precision, the fraction of emitted aborts whose prefixes admit no valid completion, is 94–99% across these sizes (Appendix B.1). For the causal interventions and DLA, we use six independently generated $n = 300$ graphs, each with a trained 1/2/384 $+\mathrm{x}$ model. On held-out prefixes with an injected invalid transition, these models assign a mean probability of 99.7% to the abort token (Appendix B.2). This strong abort response gives us a clear setting in which to study how the decision is made.

5.1 Previous-Node Information Drives the Abort Decision

We first test whether abort supervision increases attention to the previous node. For each model, we feed 3,000 correct held-out paths as fixed input sequences and average the attention weights over the two heads and over sequences containing each position. We measure attention on the previous node (key $q-1$) and the mean attention on the other intermediate emitted nodes (keys $3,\dots,q-2$). We average each quantity over $q=5,\dots,8$, where this comparison set is nonempty, and take their ratio. A ratio of 1 means that the previous node receives the same attention as an average intermediate node. This ratio rises from 0.88 to 2.52 at $d=384$ (Figure 1), with an average paired increase of 1.65 ± 0.09 over five seeds and five graph sizes. At $d=120$, the corresponding increase is 0.39 ± 0.25.

The above measurement uses only correct paths and therefore captures the model's general tendency to look back. We then test whether attention depends on transition validity. On six $n=300$ graphs, the head with larger previous-node attention on injected invalid prefixes assigns 0.58 attention mass to the previous node after an invalid transition, compared with 0.17 after a valid transition.

We next test whether previous-node information causally affects the abort decision. We use the six $n=300$, 1/2/384 +x models described above. For each graph, we take 2,000 held-out correct paths, cut each path at a random interior node u_k, and append a node w that is not a successor of u_k. This produces an invalid prefix "$s\ t\ u_0\dots u_k\ w$", and we measure $P(\mathrm{x})$ at the position immediately after w. We retain prefixes for which a previous-node replacement can make both adjacent transitions valid, so the interventions use the same set of prefixes.

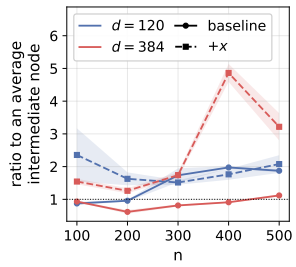


Figure 1: Previous-node attention ratio over five seeds; shaded bands are s.e.

Intervention	$P(\mathrm{x})$
None	0.987 ± 0.003
Remove previous-node attention (main head)	0.089 ± 0.086
Remove previous-node attention (both heads)	0.008 ± 0.002
Swap previous node (valid prefix)	0.001 ± 0.000
Remove FFN output	0.000 ± 0.000

Table 1: Interventions after an injected invalid transition, on prefixes whose current node can still reach the target. Mean $P(\mathrm{x})\pm$ s.e. over six $n=300$ graphs.

Table 1 reports interventions on the subset of prefixes for which w can still reach the target. For each graph, we call the head with more attention to the previous node on invalid prefixes the main head. Removing its attention to the previous node and renormalising lowers $P(\mathrm{x})$ from 0.99 to 0.09, while removing previous-node attention from both heads lowers it to 0.008. We also change only the previous node. Specifically, we replace u_k by a node p' such that $(u_{k-1},p')\in E$ and $(p',w)\in E$. The resulting prefix is valid and differs from the injected-invalid prefix only at the previous-node position. This change lowers $P(\mathrm{x})$ to 0.001. Finally, we set the FFN output at the final position to zero before the residual addition, and then apply the final layer normalisation and unembedding as usual. This totally removes the abort response.

All rows of Table 1 are restricted to prefixes whose injected node w can still reach the target. This avoids mixing invalid-transition detection with dead-end detection: after the previous-node swap, the prefix is valid and the target is still reachable, so the model no longer has a reason to abort. Appendix B.2 gives the full protocol, controls, and uncertainty estimates.

5.2 Direct Logit Attribution Locates the Abort Signal in the FFN

The interventions above show that previous-node information causally affects the abort decision. We next take a closer look at where this information is converted into an abort signal, using direct logit attribution (DLA). Let X be the sum of token and positional embeddings. For our one-layer Transformer, the residual stream before the final normalisation is

$$H = X + \mathrm{MHA}\big(\mathrm{LN}_1(X)\big) + \mathrm{FFN}\big(\mathrm{LN}_2\big(X + \mathrm{MHA}\big(\mathrm{LN}_1(X)\big)\big)\big)\,.$$

The first two terms contain the information retrieved by attention, while the last term is the output after this information passes through the nonlinear FFN. Here LN_1 and LN_2 are the layer normalisations before attention and before the FFN, respectively.

DLA measures how much each part directly contributes to the abort logit in the same forward pass. For example, let M denote the FFN output. Its contribution is $z_{\mathtt{x}}^{M} = w_{\mathtt{x}}^{\top}[g \odot (M - \mathrm{mean}(M))/\,\mathrm{std}(H)]$. All quantities here are evaluated at the final position: $w_{\mathtt{x}}$ is the abort unembedding vector, g is the final layer-normalisation scale vector, and $\mathrm{std}(H)$ includes the numerical stabiliser. Intuitively, this is the amount of abort logit that disappears if the FFN contribution is set to zero. It is an attribution, rather than rerunning the model with the FFN removed. We similarly compute the contribution before the FFN. Comparing these contributions between valid and invalid prefixes (as described in the last subsection) shows whether most of the valid–invalid separation is already written directly before the FFN or instead appears in the FFN output.

On target-reachable invalid prefixes, the direct contribution before the FFN changes little when the previous node is swapped to make the last transition valid (-0.68 to -1.15), while the FFN contribution drops from $+29.6$ to $+1.5$; the total abort logit falls from 29.0 to 0.3. Thus, little of the valid–invalid difference is written directly into the abort logit before the FFN; most of it appears in the FFN output. The same pattern holds for the abort margin, defined as the abort logit minus the largest node logit: the pre-FFN contribution changes from -1.0 to -1.7, while the FFN contribution changes from $+22.4$ to -10.2. Since the FFN receives only the pre-FFN residual stream, the information needed to distinguish the two prefixes must already be present there, but it is the FFN transforms it into a strong abort-versus-continue signal. This accords with our analysis, i.e., an additive score is not enough ($X + \mathrm{MHA}\big(\mathrm{LN}_1(X)\big)$), and the ReLU in FFN layer is critical for the abort signal. Appendix B.2 gives the full decomposition, including the unit-level analysis.

6 Planning with the Learned Abort Signal

We now ask whether the learned abort signal improves planning. We evaluate graphs, Countdown, and mazes; Countdown and maze experiments use $1/2/384$ models. All models are trained for 10^4 updates with batch size $1{,}024$ and learning rate 5×10^{-4}. As in Section 3, correct trajectories receive standard next-token supervision, while error trajectories supervise only the final abort token. All results use sampling at temperature 1.

We use a simple restart harness with draw budget K. If the $+\mathtt{x}$ model emits $\mathtt{x}$, the current draw is rejected and generation restarts from the original prompt. Otherwise, the completed output is checked by a domain-specific acceptance test $\mathcal{A}$, which provides a second reason to reject a draw without checking full correctness. In graphs, $\mathcal{A}$ requires at least two path-node tokens and a final node equal to the target, but does not check intermediate edges. In Countdown, it requires a nonempty, well-formed sequence of equations whose last stated result equals the target, without checking arithmetic or operand availability. In mazes, it accepts every non-aborting output. After any rejection, the model draws again until an output is accepted or the budget K is exhausted. The baseline uses the same acceptance rule and budget, but has no learned abort signal.

We report three metrics: single-shot validity is the valid-output rate of one draw; harness validity is the rate at which the harness returns a valid output, counting exhausted budgets as failures; and mean draw count is averaged over all evaluation instances, with an exhausted budget counted as K draws (Appendix B.1).

6.1 Graph Path-Finding

Abort supervision can reduce single-shot validity, especially at smaller model size: at $n = 500$, the two $d = 120$ models lose about 11 percentage points, whereas the loss for $d = 384$ is at most 2.8 points across the tested sizes. With retry, however, the $+\mathtt{x}$ model has higher harness validity than the baseline at every graph size for $d = 384$ (Figure 2b), reaching 98.9–99.7% compared with 95.0–98.3%. This gain requires only a small number of additional draws: averaged over all instances, the $+\mathtt{x}$ harness uses 1.03–1.09 draws per instance, compared with 1.01 for the baseline, an increase of only 0.02–0.08 draws (Figure 2d) in expectation.

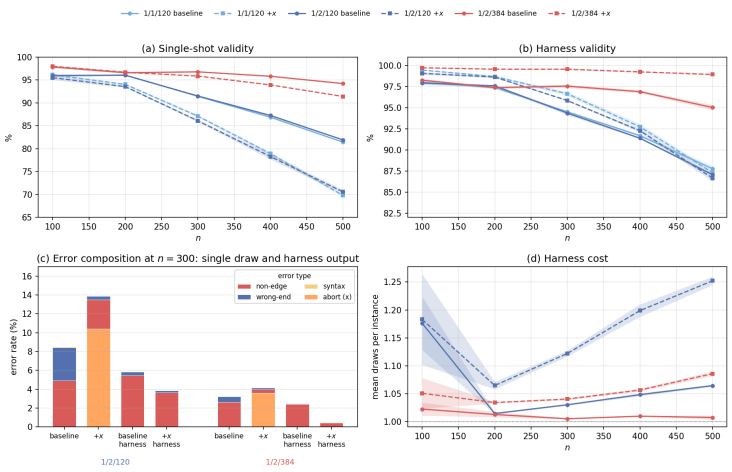


Figure 2: Graph domain over five seeds, each an independently drawn graph; solid is baseline, dashed $+\mathtt{x}$; harness budget $K = 100$. (a) Single-shot validity. (b) Harness validity. (c) Error composition at $n = 300$ for a single draw and for the harness output. (d) Mean draws per instance.

Figure 2c shows how the learned abort signal changes the error composition. In a single draw, many errors that would otherwise appear as non-edge or wrong-end outputs are replaced by aborts. At $d = 384$, the invalid transition rate falls from 2.4% for the baseline to 0.4% for $+\mathtt{x}$; at $d = 120$, it falls from 5.9% to 3.9%. The harness then rejects these aborting draws and resamples, so many of these errors disappear from the returned output. The gain persists when we match correct-trajectory exposure by doubling $+\mathtt{x}$ training updates (Appendix B.3).

6.2 Countdown

In this dataset, three numbers must be combined with $+, -, \times, \div$ to reach a target. Intermediate values and targets are restricted to 1–100. We enumerate all three-number multisets over 1–20 together with all targets reachable under these restrictions, giving 16,117 distinct instances, and hold out a disjoint tenth. A valid solution writes one equation per operation, `a op b = c`: the two operands are removed from the available numbers and the result is inserted. Every operation must be correct, and the final result must equal the target.

To construct an error trajectory, we corrupt the result after one `=`. For each trajectory, we choose one of two corruption types with equal probability: either adding or subtracting a small offset of magnitude 1–3, or replacing the result with a random value. Random replacements often produce obviously implausible results, whereas small offsets produce plausible-looking arithmetic errors that require checking the operation more carefully. Mixing the two reduces the chance that the model learns a shortcut based only on unusually large or small results. The corrupted equation is the invalid transition and is followed immediately by `x`; only this final `x` receives loss on the error trajectory. We denote by ρ the ratio of error trajectories to correct trajectories.

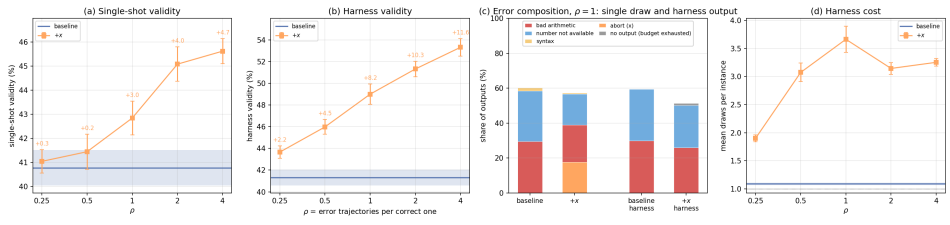


Figure 3: Countdown, $1/2/384$, five seeds per ρ. (a) Single-shot validity against ρ, with the paired gain over the baseline. (b) Harness validity. (c) Error composition at $\rho = 1$ for a single draw and for the harness output. (d) Mean draws per instance. In (a), (b) and (d) the baseline is a level band (mean $\pm$ s.e. over seeds) because its training data do not depend on ρ.

At $\rho = 1$, the $+\mathtt{x}$ model improves harness validity by $+\mathbf{8.2} \pm \mathbf{0.9}$ percentage points (Figure 3). This comes at a higher sampling cost: averaged over all instances, the $+\mathtt{x}$ harness uses 3.7 draws per instance, counting an exhausted budget of 100 in full, compared with 1.1 for the baseline. The $+\mathtt{x}$ harness fails on 51% of instances: it returns an output with incorrect arithmetic on 26% and one using an unavailable number on 24%, and it exhausts its budget without accepting an output on 1.2%. The baseline harness fails on 59% of instances (Figure 3c).

We next vary the amount of abort supervision. Averaging repeated baseline runs within each seed gives a harness validity of $41.3 \pm 0.7\%$ over five seeds. The gain from $+\mathtt{x}$ increases monotonically from $+2.2 \pm 0.6$ percentage points at $\rho = 0.25$ to $+11.6 \pm 1.3$ points at $\rho = 4$ (Figure 3b). Abort supervision also improves single-shot validity, by 3.0 ± 0.5 percentage points at $\rho = 1$ and 4.7 ± 0.8 points at $\rho = 4$ (Figure 3a). This differs from the graph setting, where most of the harness gain comes from retrying after an abort. On Countdown, validity improves even before retry, suggesting that abort supervision may also improve representations used to generate valid equations.

6.3 Mazes

In a 10×10 maze with random walls, the model starts facing East and emits relative-direction tokens: `F` moves forward, `L` and `R` turn left and right, and `T` turns around. Each token changes direction as needed and then advances one cell. For each of four independently generated mazes, we split reachable source–target pairs between training and test sets with probability 0.5 each. Training uses correct paths and an equal number of error trajectories.

A valid output reaches the target without crossing a wall. Error trajectories contain one wall-crossing move followed by the supervised abort token. For mazes, the acceptance test $\mathcal{A}$ accepts every non-aborting output. Thus, the harness retries only when the model emits `x`; wall crossings are checked only offline when computing validity.

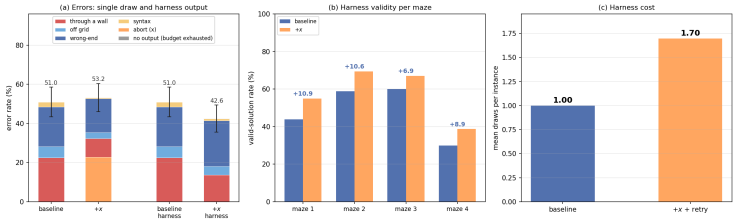


Figure 4: Maze task, $1/2/384$, four independently generated mazes. (a) Error composition for a single draw and the harness output. (b) Harness validity per maze, averaged over two runs. (c) Mean draws per instance.

Rejecting outputs that contain `x` and drawing again improves harness validity by $+\mathbf{9.3} \pm \mathbf{0.9}$ percentage points (Figure 4). This requires 1.70 draws per instance on average for the $+\mathtt{x}$ harness, while the baseline always accepts its first draw. In single-shot generation, abort supervision reduces the wall-crossing rate from 22.5% to 9.4%, with some of these failures replaced by aborts. The harness retries aborting draws, reducing wall crossings in returned outputs from 22.5% for the baseline to 13.7% for $+\mathtt{x}$. Wrong-end errors remain at 20–23%: neither the learned abort signal nor the acceptance test detects them (Figure 4a).

7 Conclusion

Error trajectories can teach a Transformer to detect invalid transitions by supervising only a final abort token. Our theoretical analysis characterises the required previous–current interaction and gives a one-layer construction in which attention retrieves information and the FFN checks transition validity. The width lower bound further characterises the capacity required within the specified readout class. In trained models, abort supervision increases previous-node attention, and the FFN produces most of the validity-dependent abort signal. Using this learned signal to reject and retry improves planning reliability on graphs, Countdown and mazes.

AI USE STATEMENT

Generative artificial intelligence (AI) tools were used to assist with parts of the research and writing process. Claude (Anthropic) was used for code debugging and as an aid in literature search, while Claude and Codex (OpenAI) were used for limited drafting and editing of the manuscript. The research questions, problem formulation, theorem statements, proof strategies, experimental design, and interpretation of results were developed by the authors. The authors reviewed and verified all AI-assisted content and take responsibility for the final manuscript and accompanying artifacts.

REPRODUCIBILITY STATEMENT

Reproducibility details are provided in the paper and appendices. Section 3 describes the architectures, training objective and graph data, and Section 6 describes the Countdown and maze tasks and the retry procedure. Appendix B.1 gives the training and evaluation protocols, Appendix B.2 details the mechanistic analyses, and Appendices B.3 and B.4 report additional experiments, controls and method comparisons. Appendix A provides the explicit Transformer construction and proofs. We will release the code publicly.

REFERENCES

William H Alexander and Joshua W Brown. Computational models of performance monitoring and cognitive control. *Topics in cognitive science*, 2(4):658–677, 2010.

Daman Arora and Subbarao Kambhampati. Learning and leveraging verifiers to improve planning capabilities of pre-trained language models. *arXiv preprint arXiv:2305.17077*, 2023.

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

Edoardo Botta, Yuchen Li, Aashay Mehta, Jordan T Ash, Cyril Zhang, and Andrej Risteski. On the query complexity of verifier-assisted language generation. *arXiv preprint arXiv:2502.12123*, 2025.

Tom Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

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

Zhongzheng Fu, Amirsaman Sajad, Steven P Errington, Jeffrey D Schall, and Ueli Rutishauser. Neurophysiological mechanisms of error monitoring in human and non-human primates. *Nature Reviews Neuroscience*, 24(3):153–172, 2023.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Sukai Huang, Trevor Cohn, and Nir Lipovetzky. Chasing progress, not perfection: Revisiting strategies for end-to-end llm plan generation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 35, pp. 204–212, 2025.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Yin Jun Phua, Tony Ribeiro, Tuan Nguyen, and Katsumi Inoue. Can transformers learn to verify during backtracking search? *arXiv preprint arXiv:2605.22221*, 2026.

Dhruv Rohatgi, Abhishek Shetty, Donya Saless, Yuchen Li, Ankur Moitra, Andrej Risteski, and Dylan Foster. Taming imperfect process verifiers: A sampling perspective on backtracking. In *International Conference on Learning Representations*, volume 2026, pp. 128647–128722, 2026.

Siwei Wang, Yifei Shen, Shi Feng, Haoran Sun, Shang-Hua Teng, and Wei Chen. Alpine: Unveiling the planning capability of autoregressive learning in language models. *Advances in neural information processing systems*, 37:119662–119688, 2024.

Siwei Wang, Yifei Shen, Haoran Sun, Shi Feng, Shang-Hua Teng, Li Dong, Yaru Hao, and Wei Chen. Benefits and pitfalls of reinforcement learning for language model planning: a theoretical perspective. In *International Conference on Learning Representations*, volume 2026, pp. 677–701, 2026.

Xiao-Wen Yang, Xuan-Yi Zhu, Ding-Chu Zhang, Wen-Da Wei, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for symbolic reasoning and planning in language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27657–27665, 2026.

Qimin Zhong, Hao Liao, Siwei Wang, Mingyang Zhou, Xiaoqun Wu, Rui Mao, and Wei Chen. Understanding and enhancing the planning capability of language models via multi-token prediction. *arXiv preprint arXiv:2509.23186*, 2025.

APPENDIX

A THEORETICAL ANALYSIS

Appendix A.1 proves Theorem 2 and Appendix A.2 proves Theorem 3. We also checked auxiliary steps in Lean 4 using the Mathlib library: the ReLU identity (2) and its sum over nodes, the bounds on the errors caused by finite attention scores, the bounds on softmax probabilities, and the counting inequality used in Theorem 3. These checks cover individual lemmas rather than the complete Transformer construction; the attention-score calculation in §A.1 and the cited VC-dimension bound of Bartlett et al. (2019) are not formalised. A numerical check builds the Transformer of Appendix A.1 from its weights at the constants of the proof and verifies both clauses, the dead-end case and the first step on random DAGs.

A.1 CONSTRUCTION AND PROOF OF THEOREM 2

We use the architecture of §3.1 and the notation of §4. The construction below has embedding size $d = 4|V| + 8$ and feed-forward width $r = 2|V|$, of which $|V|$ units form the detector.

Theorem 2. *Given a DAG G with adjacency matrix A^{true} and reachability matrix R^{true}, for every $\varepsilon > 0$ there is a 1-layer, 2-head Transformer with embedding size $O(|V|)$ and feed-forward width $O(|V|)$ that, for every input sequence "$s\ t\ s\ u_1 \dots u_k$" for which $(u_0, \dots, u_{k-1})$ is a valid path prefix, where $u_0 = s$, $k \geq 1$, and $u_k \neq t$, generates with probability at least $1 - \varepsilon$ (i) the token $\mathtt{x}$, if $A^{\mathrm{true}}_{(u_{k-1},\, u_k)} = 0$, and (ii) otherwise a node u_{k+1} that is both an out-neighbour of u_k and able to reach t, whenever such a node exists.*

Proof. Throughout the proofs, e_v denotes the standard basis vector for coordinate v, with dimension determined by the containing space. Layer normalisations are omitted, as in Wang et al. (2024). Write $p = u_{k-1}$, with $u_0 = s$, for the token preceding u_k, so clause (i) asks the model to detect $A^{\mathrm{true}}_{(p,u_k)} = 0$. We first consider exact attention, with each head assigning unit weight to its intended position, then bound the effect of finite attention scores.

Embedding. Take $M = |V| + 2$ tokens (the nodes, "$\backslash\mathtt{n}$", $\mathtt{x}$), per-head size $d_k = 2|V| + 4$ and $d = 2d_k = 4|V|+8$. Coordinates $1, \dots, M$ carry the one-hot token ($\mathcal{V}$ denotes the node coordinates), coordinate $M + 1$ a flag set to a fixed $c_0 > 0$ on row 2 only, coordinates $M + 2$ and $M + 3$ the position ν and ν^2, and two disjoint blocks $\mathcal{R}, \mathcal{P}$ of $|V|$ coordinates in the second half hold the two head outputs. All other embedding coordinates are zero. Positions are numbered starting at 1, so the target occupies position 2. Thus row ν of H_0 is e_{τ_ν} on the token coordinates, with τ_ν the token at position ν, plus $(c_0\mathbf{1}[\nu = 2],\, \nu,\, \nu^2)$. Writing $\Pi = \sum_{v \leq M} e_v$, every row has $\langle (H_0)_{\nu,:}, \Pi \rangle = 1$, which is how queries independent of ν are formed. The DAG path $(u_0, \dots, u_{k-1})$ has distinct nodes. The final node u_k need not extend that path, so $k \leq |V|$ and the length $N = k + 3$ satisfies $N \leq N_{\max} := |V| + 3$. All constants below depend only on $N_{\max}$ and ε.

Heads. For $\gamma > 0$, Head 1 has query $\mathbf{1}_{d_k}$ at every row (from Π) and key $\gamma c_0 \mathbf{1}[m = 2]\mathbf{1}_{d_k}$ at row m, so after scaling its score at column m is $\beta_1 \mathbf{1}[m = 2]$ with $\beta_1 := \sqrt{d_k}\, \gamma c_0$. Its value projection is R^{true} from the node coordinates into $\mathcal{R}$, so at exact attention it writes $R^{\mathrm{true}}_{(t,:)}$ into $\mathcal{R}$. For $\lambda > 0$, Head 2 uses key $\lambda\sqrt{d_k}(m^2, m)$ at row m and query $(-1, 2(\nu - 1))$ at row ν, with all remaining query and key coordinates zero. These are linear projections of the embeddings. After the standard $1/\sqrt{d_k}$ scaling, the score is $\lambda(-m^2 + 2m(\nu - 1)) = \lambda\big((\nu - 1)^2 - (m - (\nu - 1))^2\big)$. At the final position $\nu = N$, the first term is constant in m, so under the causal mask the score is maximised at $m = N - 1$ and every other column is lower by at least λ. Its value projection is A^{true} from the node coordinates into $\mathcal{P}$, so at exact attention it writes the row $A^{\mathrm{true}}_{(p,:)}$ into $\mathcal{P}$. W^A places the two outputs in $\mathcal{R}$ and $\mathcal{P}$ and nothing in $\mathcal{V}$. Both value projections are zero on non-node token coordinates and on positional coordinates; all unspecified projection entries are zero. Thus, after the residual connection $\tilde{H}_{N,\mathcal{V}} = e_{u_k}$ exactly.

Both heads have one column leading all others by a margin, β_1 and λ respectively, among at most $N_{\max}$ columns; the softmax weight off that column is therefore at most $(N_{\max} - 1)e^{-\beta_1}$ and $(N_{\max} - 1)e^{-\lambda}$ by the softmax bound used below.

Feed-forward network. Write $i = u_k$ and $h = \tilde{H}_{N,:}$. At exact attention, the three blocks read by the FFN are

$$h_{\mathcal{R}} \oplus h_{\mathcal{P}} \oplus h_{\mathcal{V}} = R^{\text{true}}_{(t,:)} \oplus A^{\text{true}}_{(p,:)} \oplus e_i,$$

where $\oplus$ denotes concatenation. The first linear map W_1 extracts the current node's adjacency row from e_i: its vth entry is $\sum_{j \in V} e_i(j) A^{\text{true}}_{(j,v)} = A^{\text{true}}_{(i,v)}$. We give W_1 two groups of $|V|$ columns. The entries of b_1 are 0 on the first group and -1 on the second, producing two groups of inputs to the ReLU activation:

$$z^{\text{plan}}_v = h_{\mathcal{R},v} + \sum_{j \in V} h_{\mathcal{V},j} A^{\text{true}}_{(j,v)} = R^{\text{true}}_{(t,v)} + A^{\text{true}}_{(i,v)},$$

$$z^{\text{det}}_v = h_{\mathcal{V},v} + h_{\mathcal{P},v} - 1 = e_i(v) + A^{\text{true}}_{(p,v)} - 1.$$

All entries of W_1 not specified by these expressions are zero. Applying ReLU to both groups in parallel gives

$$q_v = \text{ReLU}(z^{\text{plan}}_v) = R^{\text{true}}_{(t,v)} + A^{\text{true}}_{(i,v)} \in \{0, 1, 2\},$$

$$d_v = \text{ReLU}(z^{\text{det}}_v) = \mathbf{1}[i = v] A^{\text{true}}_{(p,v)}.$$

Thus $q_v = 2$ exactly for the out-neighbours of i that can reach t, $q_v = 1$ for nodes with one of the two properties, and $q_v = 0$ otherwise, as in the adjacency–reachability account of Wang et al. (2024); the planning inputs are non-negative, so their ReLU is the identity. The detection units implement (2): only d_i can be nonzero, and $\sum_v d_v = A^{\text{true}}_{(p,i)}$. The current node's adjacency row is thus read linearly, and the same ReLU layer implements the gate.

Choose $c \geq 4$. In W_2, the output row for q_v is ce_v, while the row for d_v is $-6ce_{\texttt{x}}$; set $b_2 = 3ce_{\texttt{x}}$. The planning units write only to node coordinates, and the detection units write only to the abort coordinate. The abort logit is $3c - 6c \sum_v d_v$, hence $3c$ on a non-edge and $-3c$ on an edge, independent of reachability. The feed-forward width is $r = 2|V|$.

Output at exact attention. Choose W^U to read the token coordinates. The vocabulary logits at row N of $\text{FFN}(\tilde{H}) + \tilde{H}$ are then

$$\begin{aligned} z_v &= c\left(R^{\text{true}}_{(t,v)} + A^{\text{true}}_{(u_k,v)}\right) + \mathbf{1}[v = u_k] \quad (v \in V), \\ z_{\backslash \texttt{n}} &= 0, \\ z_{\texttt{x}} &= 3c - 6c\, A^{\text{true}}_{(p,u_k)}. \end{aligned} \tag{4}$$

A DAG has $A^{\text{true}}_{(u_k,u_k)} = 0$, so a node logit is $2c$ exactly when the node is an out-neighbour of u_k that reaches t, at most $c + 1$ when it has one of the two properties, and at most 1 otherwise. Writing B for the set of desired output tokens, in clause (i) $B = \{\texttt{x}\}$ at $3c$ against at most $2c$ for every other token. In clause (ii), B is the nonempty set of out-neighbours that can reach t, each with logit $2c$, against at most $c + 1$ for other nodes, 0 for $\backslash\texttt{n}$, and $-3c$ for $\texttt{x}$. In both cases

$$\min_{v \in B} z_v - \max_{v \notin B} z_v \;\geq\; g := c/2 \tag{5}$$

because the respective gaps are at least c and $c - 1$. If the most recent transition is valid but no out-neighbour can reach t, the abort logit is still $-3c$, below every non-abort logit. Taking B to be all non-abort tokens then gives a gap of at least $3c$, so (5) also holds in this case. Thus even at a dead end, a valid preceding transition suppresses abort; the next-node guarantee applies only when an out-neighbour that can reach t exists, as stated in clause (ii). At such a dead end the nodes with $q_v = 1$, namely the out-neighbours of u_k and the nodes that can reach t (a set that contains t), have logit at least c, and every other node has logit at most 1. Taking B to be this set gives a gap of at least $c - 1 \geq g$, so the model samples from this narrower set.

Finite attention scores. Two FFN groups must be considered. The planning group reads $\mathcal{R}$, a convex combination of rows of R^{true} that puts weight at most $(N_{\max} - 1)e^{-\beta_1}$ off the row t. The adjacency input is exact, and ReLU cannot increase the difference between two scalar inputs, so the node logits change by at most $c(N_{\max} - 1)e^{-\beta_1}$. This group has no effect on the abort logit. The detection group reads $\mathcal{V}$ exactly and $\mathcal{P}$ as a convex combination of rows of A^{true}, so unit v sees some $\tilde{a}_v \in [0, 1]$ in place of $A^{\text{true}}_{(p,v)}$, with $|\tilde{a}_v - A^{\text{true}}_{(p,v)}| \leq (N_{\max} - 1)e^{-\lambda}$. For $v \neq u_k$ the argument $\tilde{a}_v - 1$ is non-positive and the unit stays at 0; for $v = u_k$ the unit equals $\tilde{a}_{u_k}$. Through its output row this

moves only the x logit, by at most $6c(N_{\max}-1)e^{-\lambda}$. Together, these deviations bound every logit change by

$$\eta := c(N_{\max}-1)e^{-\beta_1} + 6c(N_{\max}-1)e^{-\lambda}.$$

Taking

$$\beta_1 \geq \log(16N_{\max}), \qquad \lambda \geq \log(96N_{\max}),$$

gives $\eta \leq c/16 + c/16 = g/4$, so every logit is within $g/4$ of (4) and the gap is still at least $g - 2\eta \geq g/2$. The softmax probability assigned outside the desired set B is at most $(M-1)e^{-g/2} = (M-1)e^{-c/4}$, which is at most ε once $c \geq 4\log((M-1)/\varepsilon)$. So $c = \max\{4,\ 4\log((M-1)/\varepsilon)\}$ and the two attention-score bounds above give the theorem, with $d = 4|V| + 8$ and $r = 2|V|$. The same bound with B equal to all non-abort tokens shows $\Pr(\mathrm{x}) \leq \varepsilon$ after a valid transition into a dead end. At finite attention the abort logit is exactly $3c - 6c\tilde{a}_{u_k}$ and still has no reachability input. Its softmax probability shares a denominator with the node logits, but whether abort is the highest-logit token is determined by transition validity under the bounds above.

Boundary cases. The theorem concerns positions after at least one generated transition and before the current node equals the target. It does not specify the initial prediction of s from "$s\ t$" or the end-of-line prediction after reaching t. At the first step, $k = 0$ and the input is "$s\ t\ s$". Head 2 then attends to position 2, so the detector reads the pair (t, s), which is not a transition of the path, and $A^{\mathrm{true}}_{(t,s)} = 0$ because $s \rightsquigarrow t$ in a DAG; as constructed, the model would emit x. One further value map covers this step without changing d or r: head 2 also copies the flag coordinate into the abort coordinate with weight $-6c/c_0$. At $\nu = 3$, head 2 places weight at least $1 - (N_{\max}-1)e^{-\lambda}$ on position 2, so the abort logit is at most $-3c + 6c(N_{\max}-1)e^{-\lambda}$, while the planning units select an out-neighbour of s that reaches t, as in clause (ii). At later positions the map lowers the abort logit by at most $6c(N_{\max}-1)e^{-\lambda}$; taking $\lambda \geq \log(192N_{\max})$ in place of $\log(96N_{\max})$ keeps the total change of the abort logit at most $c/16$, so the bounds above are unchanged.

The case analysis uses only that the reachability matrix has entries in $\{0, 1\}$. It therefore also holds with R^{obs} in place of R^{true}, the setting of Remark 1. Detection is unchanged. After a valid transition, or at the first step, if no out-neighbour of the current node is observed to reach t, the model outputs with probability at least $1 - \varepsilon$ an out-neighbour of the current node or a node observed to reach t, provided this set is nonempty. □

A.2 Lower Bound on FFN Width

We prove Theorem 3 in the setting of §4.3: the FFN input is $x_{p,i,t} = e_p \oplus e_i \oplus e_t \in \mathbb{R}^{3n}$ with $n = |V|$, and the FFN's contribution to the abort logit is (3).

Scalar form and fixed target. With $W_1^{(p)}, W_1^{(i)}, W_1^{(t)}$ the row blocks of W_1 from §4.3, let $w_h = \big(W_2\, W^U[:, \mathrm{x}]\big)[h]$ be the weight with which unit h enters the abort logit and $\beta = b_2\, W^U[:, \mathrm{x}]$. Because each block of $x_{p,i,t}$ is one-hot, (3) is the scalar ReLU network

$$z^{\mathrm{FFN}}_{\mathrm{x}}(p, i, t) = \beta + \sum_{h=1}^{r} w_h\, \mathrm{ReLU}\big(W_1^{(p)}[p, h] + W_1^{(i)}[i, h] + W_1^{(t)}[t, h] + b_1[h]\big). \tag{6}$$

We take $z^{\mathrm{FFN}}_{\mathrm{x}} > 0$ to mean "invalid"; the argument is identical under the opposite convention. Fix a target t_0 and absorb $W_1^{(t)}[t_0, h]$ into the bias, $b'_h = b_1[h] + W_1^{(t)}[t_0, h]$, giving

$$F_{t_0}(p, i) := z^{\mathrm{FFN}}_{\mathrm{x}}(p, i, t_0) = \beta + \sum_{h=1}^{r} w_h\, \mathrm{ReLU}\big(W_1^{(p)}[p, h] + W_1^{(i)}[i, h] + b'_h\big). \tag{7}$$

Let $\mathcal{G}_r$ be the class of Boolean functions $\mathbf{1}[F_{t_0}(p, i) > 0]$ on $V \times V$, with $W_1^{(p)}, W_1^{(i)} \in \mathbb{R}^{n \times r}$, $b', w \in \mathbb{R}^r$ and $\beta \in \mathbb{R}$ arbitrary. An FFN that is correct for every target must also be correct for this fixed target. Its decision rule belongs to $\mathcal{G}_r$, so a lower bound for this class also applies when the FFN receives the target as an input.

Theorem 3. *Suppose that a width-r one-hidden-layer ReLU FFN receives the previous node p, current node i, and target t as the input $e_p \oplus e_i \oplus e_t$. If, for every DAG on $|V|$ nodes, there exist FFN parameters for which the sign of $z^{\mathrm{FFN}}_{\mathrm{x}}(p, i, t)$ correctly determines whether $(p, i) \in E$ for every $p, i, t \in V$, then $r = \Omega\big(|V|/\log^2 |V|\big)$.*

Proof. Fix t_0 and let $\mathcal{G}_r$ be the class defined above. The proof compares how many labellings of $V \times V$ the class $\mathcal{G}_r$ can induce with how many it must induce.

Step 1: number of parameters. A member of $\mathcal{G}_r$ is specified by the $2nr$ entries of $W_1^{(p)}$ and $W_1^{(i)}$, the r entries each of b' and w, and β. Hence

$$P = r(2n+2) + 1 \le 4rn \qquad (n \ge 2,\ r \ge 1),$$

since $r(2n+2)+1 \le 4rn$ is equivalent to $1 \le r(2n-2)$.

Step 2: VC dimension. $\mathcal{G}_r$ contains the binary decisions obtained by thresholding a one-hidden-layer ReLU network with P real parameters, evaluated on the finite input set $\{e_p \oplus e_i\} \subset \mathbb{R}^{2n}$. The Vapnik–Chervonenkis (VC) dimension is the largest number of inputs on which a function class can realise every binary labelling. The VC dimension of the sign of a piecewise-linear network with P parameters and a constant number of layers is at most $C\,P\log P$ for an absolute constant C (Bartlett et al., 2019); restricting the domain cannot increase it. By Step 1 and monotonicity of $P\log P$,

$$D := \mathrm{VCdim}(\mathcal{G}_r) \le C\,P\log P \le 4C\,rn\log(4rn). \tag{8}$$

Step 3: how many labellings $\mathcal{G}_r$ can induce. The domain $V \times V$ has $m = n^2$ points. By the Sauer–Shelah lemma, a class of VC dimension D induces at most $\sum_{k=0}^{D}\binom{m}{k} \le (m+1)^D$ distinct labellings of an m-point set. Thus the number induced by $\mathcal{G}_r$ on $V \times V$ satisfies

$$\left|\mathcal{G}_r\right| \le (n^2+1)^D. \tag{9}$$

Step 4: how many labellings it must induce. Fix the node ordering $1,\dots,n$ and consider the graphs whose edges all go from a smaller index to a larger index. Every subset of the $\binom{n}{2}$ pairs $\{(p,i) : p < i\}$ is such a graph, and each is a DAG because its edges follow a fixed topological order. Distinct subsets have distinct adjacency matrices, so there are $2^{\binom{n}{2}} = 2^{n(n-1)/2}$ distinct adjacency relations. By hypothesis each of these DAGs is detected by some FFN, and by the fixed-target reduction the restriction of that FFN's decision rule to $t = t_0$ is a member of $\mathcal{G}_r$ whose labelling of $V \times V$ is the indicator of the non-edges of that DAG. Distinct DAGs give distinct labellings, so

$$\left|\mathcal{G}_r\right| \ge 2^{n(n-1)/2}. \tag{10}$$

Conclusion. Combining (9) and (10) and taking base-two logarithms,

$$D \ge \frac{n(n-1)}{2\log_2(n^2+1)} = \Omega\!\left(\frac{n^2}{\log n}\right).$$

With (8) this gives $rn\log(4rn) = \Omega(n^2/\log n)$, that is,

$$r\log(4rn) = \Omega\!\left(\frac{n}{\log n}\right).$$

If $r > n$, the claimed asymptotic lower bound already holds. Otherwise $r \le n$, so $\log(4rn) \le \log(4n^2) = O(\log n)$, and dividing the display by $\log(4rn)$ gives $r = \Omega(n/\log^2 n)$. In either case $r = \Omega\big(|V|/\log^2|V|\big)$. □

Matching upper bound. The detector of Theorem 2 lies in this class with $r = |V|$. Setting $W_1^{(p)}[p,v] = A^{\mathrm{true}}_{(p,v)}$, $W_1^{(i)}[i,v] = \mathbf{1}[i=v]$, $W_1^{(t)} = 0$, $b_1[v] = -1$, $w_v = -6c$ and $\beta = 3c$, unit v computes (2) and $z_{\mathrm{x}}^{\mathrm{FFN}}(p,i,t) = 3c - 6c\,A^{\mathrm{true}}_{(p,i)}$, which is positive exactly on non-edges. In the Transformer of Appendix A.1 the map $e_p \mapsto A^{\mathrm{true}}_{(p,:)}$ is supplied by head 2; here it is absorbed into $W_1^{(p)}$. Width $|V|$ therefore suffices for every DAG, and Theorem 3 shows that this is optimal up to a factor $O(\log^2|V|)$.

B Experimental Details and Extended Results

B.1 Setup and Evaluation Details

Training objective. The trained models apply layer normalisation before attention, before the FFN and before unembedding. Let $\mathcal{D}$ be the training set. For each sequence $u = (u_1, \ldots, u_N)$, the supervision mask $\mu \in \{0,1\}^N$ marks which tokens receive a loss:

$$\mathcal{L}(\mathcal{D}) \;=\; -\sum_{u\in\mathcal{D}} \sum_{q=1}^{N-1} \mu_{q+1}\, \log \Pr[u_{q+1} \mid u_{1:q}]\,. \tag{11}$$

Correct trajectories supervise every non-padding next token after the first; an error trajectory $(s,\ t,\ s,\ u_1, \ldots, u_k,\ w,\ \texttt{x})$, with $u_0 = s$ when $k = 0$, supervises only the final x. Minibatch losses are averaged over supervised labels.

Graph data. The distinct non-edges used in the error trajectories cover 82–93% of all non-edges, more at larger n. The +x data sets range from 67,000 sequences (2.1 million padded token slots) at $n = 100$ to 2.49 million sequences (79.6 million slots) at $n = 500$. On the $n = 300$ data, $37.4 \pm 0.3\%$ of the positions supervised to emit x also end at a node from which the target is unreachable, so the abort label is not a dead-end label. A correct trajectory carries 6.1 supervised labels on average at $n = 100$ and 8.1 at $n = 500$, whereas an error trajectory carries one; with half of each +x batch made of error trajectories, the +x model sees $0.50\times$ the baseline's correct trajectories per update in expectation (measured at $n = 100$, 300 and 500). This is a ratio of trajectory counts, not a ratio of loss weights. Appendix B.3 doubles +x updates to match correct-trajectory exposure.

Output classification. Prompted with s t, the model emits tokens until end-of-line or, for a +x model, the abort token, subject to the generation-length limit. An output is abort if it contains x, valid if it is a path from s to t containing only graph edges, and invalid otherwise. An invalid output $v_0, \ldots, v_L$ is classified in the evaluator's order as syntax (too few node tokens), wrong-end ($v_0 \neq s$ or $v_L \neq t$) or non-edge (correct endpoints but some $(v_{q-1}, v_q) \notin E$); a wrong-end output may also contain a non-edge.

Evaluation, abort precision and selective accuracy. Harness results use 3,000 instances drawn with replacement from the held-out set, with $K = 100$, in all three domains of §6. Graph single-shot validity is evaluated on every held-out instance, the Countdown single-shot results use the same 3,000 instances as the harness, and the maze results average two independent sampling runs per maze. An abort emitted after a prefix $y_{1:k}$ is

$$\text{justified} \iff \text{no completion of } y_{1:k} \text{ is a valid output for the instance}, \tag{12}$$

and abort precision is the fraction of emitted aborts that are justified. In the graph domain a prefix admits a valid completion exactly when it starts at s, follows edges and ends at a node that can reach t, which we check exhaustively on G. For the fitted models of Appendix B.2, we instead report the fraction of total abort probability assigned to prefixes that admit a valid completion.

We also compare aborting with rejecting outputs that have low mean token log-probability. Coverage is the fraction of instances whose outputs are retained; selective accuracy is the fraction of those retained outputs that are valid. For +x, we retain every non-aborting output. For the baseline, we discard the lowest-scoring outputs until the same fraction is retained. These measurements use one draw per instance, without retry.

Compute. Graph models are trained in bfloat16 on one NVIDIA H200 GPU. The two d=120 architectures have the same 0.2 million parameters, because increasing the head count divides the same embedding size among more heads; the $d = 384$ model has about 1.9 million.

B.2 Additional Mechanistic Results

Interventions on previous-node attention and the FFN. The interventions of §5.1 use the $1/2/384$ +x models of six $n = 300$ graphs: the five evaluation seeds and the graph of Figure 6. For each graph,

we take 2,000 held-out valid prefixes for which a previous-node swap exists, append a non-successor node and measure $P(\mathrm{x})$ at the next position. Removing attention to a position sets its weight at the final query to zero and renormalises the remaining weights; the main head is the head with more previous-node attention on invalid prefixes. Removing the FFN output sets it to zero before the residual addition. Means $\pm$ s.e. are over graphs.

Table 2 adds controls and the full sample to Table 1. On target-reachable prefixes, removing target or current-node attention leaves $P(\mathrm{x})$ at 0.911 or 0.997, and removing previous-node attention in the other head leaves it at 0.861, so the effect is specific to previous-node attention, mostly in the main head. On the other $81.7 \pm 0.3\%$ of prefixes the appended node cannot reach the target, so an abort remains justified after the swap; the model still aborts there with mean probability 0.72 ± 0.01, which keeps the all-prefix column near 0.6. On valid prefixes, removing previous-node attention in both heads makes $P(\mathrm{x}) > 0.5$ on only $2.4 \pm 0.7\%$ of prefixes.

As a consistency check, we also copy the complete attention output at the final position between the invalid and swapped prefixes. Because the model has only one layer, the final token w and its positional embedding are identical in the two prefixes, and the attention output is the only component at that position that depends on the earlier tokens before the FFN. The patched prefix therefore gives the same abort decision as the corresponding swapped input. On target-reachable prefixes, copying the swapped prefix's attention output into the invalid prefix gives $P(\mathrm{x}) = 0.001$, as does the previous-node swap itself (Table 2). This checks the intervention implementation rather than providing an independent test of the mechanism.

Table 2: Interventions after an injected invalid transition, six $n = 300$ graphs: mean $P(\mathrm{x}) \pm$ s.e. over all prefixes and over those whose target remains reachable. Removing target or current-node attention, and the valid prefix, are controls.

condition	all prefixes	target reachable
invalid prefix, no intervention	0.997 ± 0.000	0.987 ± 0.003
remove previous-node attention, main head	0.594 ± 0.037	0.089 ± 0.086
remove previous-node attention, other head	0.949 ± 0.037	0.861 ± 0.097
remove previous-node attention, both heads	0.604 ± 0.007	0.008 ± 0.002
remove target attention, both heads	0.729 ± 0.081	0.911 ± 0.046
remove current-node attention, both heads	0.998 ± 0.001	0.997 ± 0.001
drop the FFN output	0.000 ± 0.000	0.000 ± 0.000
swap the previous node (last transition valid)	0.590 ± 0.009	0.001 ± 0.000
patch attention output from the valid swap	0.590 ± 0.009	0.001 ± 0.000
valid prefix	0.000 ± 0.000	0.000 ± 0.000

Direct logit attribution. At the final position, write the residual stream before the final layer normalisation as $H = Y + M$, where $Y = X + \mathrm{MHA}(\mathrm{LN}_1(X))$ is the stream entering the FFN and M is the FFN output, as in §5.2. With w_{x} the abort column of W^U and g the scale of the final layer normalisation, which has no bias, the abort logit is $w_{\mathrm{x}}^\top \big[g \odot (H - \mathrm{mean}(H))/\,\mathrm{std}(H)\big]$, where $\mathrm{std}(H)$ includes the numerical stabiliser. Centring is linear and $\mathrm{std}(H)$ is a single scalar for the forward pass, so the logit splits exactly into $z_{\mathrm{x}}^Y + z_{\mathrm{x}}^M$ with

$$z_{\mathrm{x}}^C = w_{\mathrm{x}}^\top \Big[g \odot \frac{C - \mathrm{mean}(C)}{\mathrm{std}(H)} \Big], \qquad C \in \{Y, M\}.$$

For the abort margin, replace w_{x} by $w_{\mathrm{x}} - w_{v^*}$, where v^* is the node with the highest logit on that prefix. The trained FFN has no output bias, so $M = \sum_h \alpha_h W_2[h, :]$ with α_h the activation of unit h, and the same formula gives each unit's contribution. Table 3 reports these quantities on the prefixes of Table 2.

The pre-FFN contribution stays near -1 in all four conditions, including the valid continuation and the target-unreachable invalid prefixes. On valid prefixes the FFN's contribution to the abort logit is positive on 90% of prefixes, while its margin contribution is -8.1: it raises the highest node logit more than the abort logit. This polarity is the reverse of the construction in Appendix A.1, where a constant abort bias is cancelled by units that fire after a valid transition. In the trained models the ten units with the most negative contributions on valid prefixes sum to only -1.1 ± 0.1 there, and to $+8.4 \pm 1.7$ on invalid prefixes. A detection unit $\mathrm{ReLU}(\mathbf{1}[i = v] - A^{\mathrm{true}}_{(p,v)}) = \mathbf{1}[i = v](1 - A^{\mathrm{true}}_{(p,v)})$,

Table 3: Direct logit attribution at the abort position, means over six $n = 300$ graphs (s.e. at most 0.4). Contributions of the pre-FFN stream $X + \mathrm{MHA}(\mathrm{LN}_1(X))$ and FFN output M to the abort logit and margin (abort logit minus highest node logit); "swap-valid" swaps the previous node to make the last transition valid.

		to abort logit		to margin	
Prefix	Abort logit	pre-FFN	FFN	pre-FFN	FFN
valid continuation	3.0	-0.95	$+4.0$	-1.6	-8.1
swap-valid, target reachable	0.3	-1.15	$+1.5$	-1.7	-10.2
invalid, target reachable	29.0	-0.68	$+29.6$	-1.0	$+22.4$
invalid, target unreachable	28.8	-0.46	$+29.2$	-0.7	$+22.0$

which fires after an invalid transition, implements the same check with the same width when its output weight is $+6c$ and the abort bias is $-3c$. Most of the FFN contribution on invalid prefixes comes from a small set of units: 16 units exceed 0.25 when the target remains reachable (22 when it does not), the ten largest carry 63–87% of the FFN contribution, and 16 (31) units reach 90% of it.

Additive fits. Could the supervised abort decision be reproduced by separate additive contributions of the previous node p, current node i and target t? We fit, to the supervised training labels of the $n = 300$ graph of Figure 6 rather than to the trained model, the additive-logit model

$$z(\ell) \;=\; W^{M}_{(i,\ell)} + W^{V}_{(t,\ell)} + W^{P}_{(p,\ell)}, \tag{13}$$

where ℓ is a candidate output token and $W^M, W^V, W^P \in \mathbb{R}^{|V|\times M}$ are free matrices: a linear readout of three one-hot inputs with no term depending jointly on p and i. This is the form computed by a fixed-attention Transformer in which two heads copy the target and previous-node representations, the residual supplies the current node, and the remaining maps are linear. With $W^P = 0$ it is the additive model of Wang et al. (2024), which gives every prefix with the same (i, t) the same next-token distribution. The abort probability of (13) is a softmax over all candidate logits rather than a thresholded additive score, so Proposition 1 does not apply directly; we test the fit empirically.

The fit uses every supervised position whose prefix ends at $u_0 = s$ or at a later node, 2,784,440 positions grouped by (p, i, t): 425,002 follow an invalid transition, and the rest come from correct trajectories and all admit a valid completion. The logits are linear in the fitted matrices, so the objective is convex; Adam followed by L-BFGS reaches $\|\nabla\overline{\mathcal{L}}\|_\infty \le 6.3 \times 10^{-7}$ for the loss averaged over these positions. Let $\mathcal{S}$ be the multiset of these prefixes and $\mathcal{C} \subseteq \mathcal{S}$ those admitting a valid completion (source-only prefixes, for which the fit sets $p = i = s$, belong to $\mathcal{C}$). The *unjustified-abort mass fraction* $\sum_{y\in\mathcal{C}} P(\mathtt{x} \mid y)/\sum_{y\in\mathcal{S}} P(\mathtt{x} \mid y)$ is the share of total abort probability placed on prefixes that could still be completed correctly; we also report the mean $P(\mathtt{x} \mid y)$ over $\mathcal{I} = \mathcal{S} \setminus \mathcal{C}$. For the fit these two percentages sum to 100%, because at a stationary point of the objective the total abort probability equals the number of abort labels; this need not hold for the trained Transformer.

Table 4: The previous-node additive fit against the trained $1/2/384$ Transformer on the same 2.78 million supervised positions of one $n = 300$ graph.

Model	Mean $P(\mathtt{x})$ after an invalid transition	Unjustified-abort mass fraction
Additive (13)	56.6%	43.4%
Trained, same positions	99.99%	0.03%

The fit also assigns about 8% abort probability after a valid transition, against 0.01% for the trained model (Table 4). Separate additive contributions therefore do not reproduce the learned rejection quality even with the previous node available.

Planning representations and attention heads. We examine planning information with two probes adapted from Wang et al. (2024). Let $\overline{W}_V := [W_V^{(1)} \mid \cdots \mid W_V^{(H)}]W^A$ be the value map of the attention layer when both heads place all their attention on the target position, and let e_v be the one-hot vector of node v. The probes are

$$W^{M'}_{(i,\ell)} = \big[(\mathrm{FFN}(e_i^\top W_t) + e_i^\top W_t)W^U\big]_\ell, \qquad W^{V'}_{(t,\ell)} = \big[(o_t + \mathrm{FFN}(o_t))W^U\big]_\ell,$$

with $o_t = e_t^\top W_t \overline{W}_V$. The first probe passes the current-node embedding through the FFN with its residual connection and probes adjacency information associated with the current node; the second does the same for the target embedding after the value map and probes reachability information associated with the target. Here FFN is the trained GELU map, and positional embeddings and layer normalisations are omitted, following Wang et al. (2024). Because the FFN is nonlinear, the probes are diagnostics of the learned weights, not decompositions of the actual logits; primes distinguish them from the fitted matrices of (13).

At the end of training, the $W^{M'}$ edge/non-edge gap of the $+\mathrm{x}$ models is slightly above the baselines' at both sizes, 0.119 against 0.110 at d=120 and 0.220 against 0.202 at d=384 (Figure 5a). The gap compares edges and non-edges among forward pairs $i < v$ in the graph's node ordering. This is consistent with error trajectories providing additional adjacency supervision.

For reachability, a pair (t, u) is observed reachable if $R^{\mathrm{obs}}_{(t,u)} = 1$, with R^{obs} as in Remark 1, unobserved reachable if $R^{\mathrm{true}}_{(t,u)} = 1$ but $R^{\mathrm{obs}}_{(t,u)} = 0$, and unreachable if $R^{\mathrm{true}}_{(t,u)} = 0$. Figure 5b shows mean $W^{V'}$ values over $u < t$ for these groups in the two-head baseline models. The observed-reachable minus unobserved-reachable gap is positive across all three architectures and five graph sizes, and the unobserved-reachable minus unreachable gap is larger in 13 of these 15 configurations, so $W^{V'}$ carries some information about true reachability beyond the observed pairs, with additional separation for observed pairs.

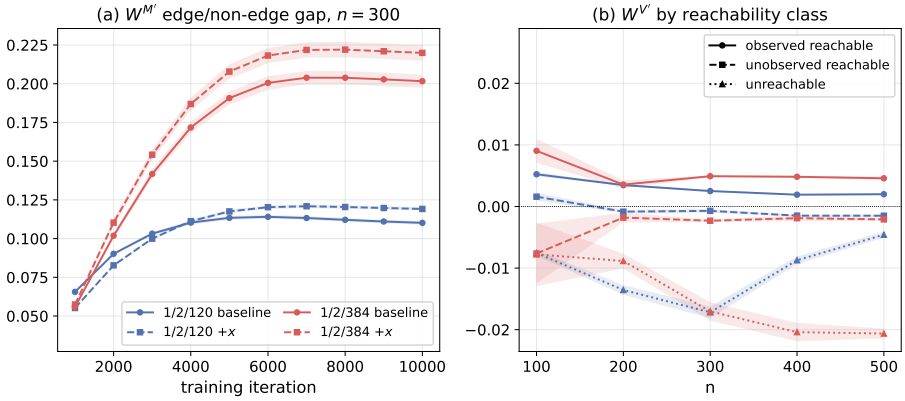


Figure 5: Planning representation probes over five seeds; shaded bands are s.e. (a) Edge–non-edge gap in $W^{M'}$ at $n = 300$. (b) $W^{V'}$ by reachability group in baseline models. Architecture colours match (a).

At $1/2/384$ one head attends to the target and the other to the current position in all five seeds, for both baseline and $+\mathrm{x}$ models (Figure 6); which head takes which role varies between runs.

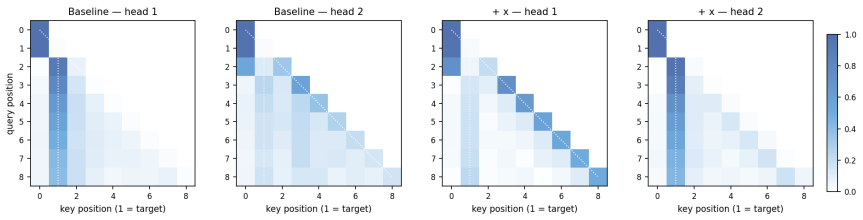


Figure 6: Average per-head attention for baseline and $+\mathrm{x}$ models, $1/2/384$ at $n = 300$. One representative seed is shown because head identities can swap across runs.

B.3 Additional Graph Planning Results

Single-shot validity and model size. Abort supervision leaves single-shot validity unchanged at $n = 100$ for all three architectures. It lowers it for the d=120 models by 2.0 ± 0.2 and 2.5 ± 0.3

percentage points at $n = 200$ (one and two heads) and by 11.6 ± 0.3 and 11.3 ± 0.4 points at $n = 500$; at d=384 the change is within the error bars at $n = 200$ and between -0.9 ± 0.1 and -2.8 ± 0.0 points from $n = 300$ on (Figure 2a). The cost is larger for the d=120 models, whose abort precision is also lower (68–94%, against 94–99% at d=384). The difference is associated with embedding size rather than head count: $1/2/120$ tracks $1/1/120$ within about one point at the same parameter count, which is why §5 uses the two-head models.

Matched correct-trajectory supervision. Because half of the trajectories in a +x batch carry a single label, the +x model receives half the baseline's correct-trajectory supervision per update (§B.1). We therefore also train both $1/2/384$ models for 20,000 updates on the $n = 300$ graph of Figure 6, after which the +x model has seen the correct-trajectory supervision the baseline receives in 10,000. Harness validity in percent on that graph:

updates	baseline + retry	+x + retry	gain
10,000 (the setting used throughout)	97.6	99.6	+2.0
20,000	96.9	99.6	+2.7

At matched correct-trajectory exposure, the 20,000-update +x model exceeds the 10,000-update baseline by 2.0 percentage points. The longer-trained baseline is slightly worse, its invalid-transition rate rising from 2.4% to 3.1%.

B.4 Additional Tasks and Method Comparisons

A pretrained model on Blocksworld. The three domains above use one-layer models trained from scratch. We also evaluate abort supervision on a pretrained language model, Qwen3-0.6B-Base, on Blocksworld with 8 or 9 blocks. A move takes an uncovered block and places it on the table or on another uncovered block. The prompt gives the initial and goal configurations as stacks; the model writes one move per line (`move A onto B`, `move A onto table`) and then `done`, and never restates the state. Each instance is an initial–goal pair whose shortest solution, found by breadth-first search from the goal, has 4–15 moves. Of 300 goals (150 per block count), 255 are training goals with 40 initial states each, giving 10,200 plans; 45 are held-out goals with 60 initial states each, and evaluation uses 2,000 of these 2,700 instances. Error trajectories take a valid prefix of a training plan, append one invalid move (moving a buried block, moving onto a buried block, or a no-op) and a new token `<|abort|>` playing the role of x, with the loss on that token only; $\rho = 8$. Both models are fine-tuned for 3 epochs (learning rate 2×10^{-5}, batch size 64) on one NVIDIA A100 GPU; because the +x training set is nine times as large, it receives nine times as many updates, and a second baseline trained for 27 epochs matches its update count. The harness draws at most $K = 10$ plans per instance at temperature 1 and accepts the first that does not abort and ends with `done`; it does not check moves or the final state, and validity is scored by simulation.

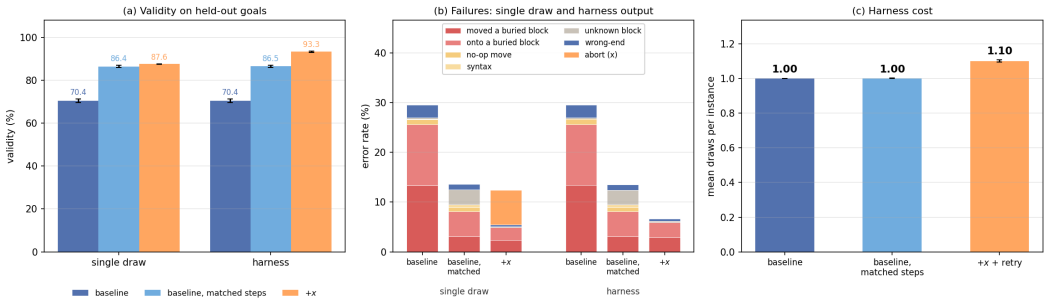


Figure 7: Blocksworld, Qwen3-0.6B-Base; mean ± s.e. over three seeds. (a) Validity for a single draw and under the harness. (b) Failures for a single draw and for the harness output. (c) Mean draws per instance.

Retrying on the abort raises harness validity from $70.4 \pm 0.9\%$ to $93.3 \pm 0.3\%$, a gain of $+22.9 \pm 1.0$ percentage points at 1.10 draws per instance, and of $+6.8 \pm 0.6$ points over the baseline trained for the same number of updates. Illegal moves account for most failures in every arm: they appear in

26.7% of the outputs returned by the baseline harness, 8.9% for the matched-steps baseline and 5.9% for the +x harness (Figure 7b). At the coverage of the +x model, 93.1% of instances, its selective accuracy is 94.1%, against 75.1% for a mean-log-probability threshold on the baseline and 88.4% on the matched-steps baseline.

Self-Backtracking on Countdown. We adapt Self-Backtracking (Yang et al., 2026) to the same error trajectories as +x. Its loss supervises the correct prefix and a backtrack token after the invalid step, masking only that step. Training otherwise matches the Countdown runs of §6: the same 1/2/384 model, split, error trajectories at $\rho = 1$, 10,000 updates, optimiser and five seeds, and the backtrack token uses the same vocabulary entry as the abort token. The baseline and +x models are the $\rho = 1$ models of §6, evaluated on the same 3,000 test instances with fresh samples, so their single-shot validity differs slightly from the values there.

At inference, Self-Backtracking first samples N trajectories and keeps completed outputs without a backtrack token as candidates. For outputs that emit the token, it removes the last equation and selects up to $\lfloor \sqrt{N} \rfloor$ distinct shortened prefixes. It then samples N continuations from each selected prefix. This is one round of backtracking, denoted $b = 1$. The likelihood selection rule returns the candidate with the highest mean token log-probability. We also evaluate the harness rule of §6, which selects the first candidate whose last written value is the target.

For each instance, the output-token count used by Self-Backtracking at a given N sets a hard budget for baseline and +x restart. A final restart draw that would exceed this budget is truncated and discarded. We also exchange the inference procedures to separate their effect from the training loss: the Self-Backtracking model uses restart under the same token budget, and the +x model uses backtracking with the same N. The latter has similar, but not identical, token counts. All combinations are evaluated under both selection rules; the resource comparison concerns generated output tokens, not wall-clock time.

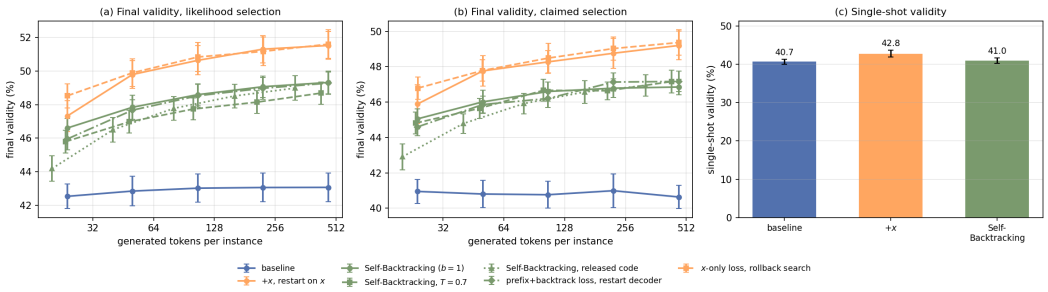


Figure 8: Countdown, five seeds, mean ± s.e. (a), (b) Final validity against output tokens per instance under the likelihood and harness selection rules; budgets are set by Self-Backtracking at $N \in \{2, 4, 8, 16, 32\}$. Dashed orange and dash-dotted green exchange the inference procedures. (c) Single-shot validity.

Figure 8 shows the results. Single-shot validity rises by $+2.1 \pm 0.5$ percentage points for +x and by $+0.3 \pm 0.7$ for Self-Backtracking. Under the likelihood rule at $N = 32$ (about 470 tokens per instance) the baseline reaches $43.1 \pm 0.8\%$, Self-Backtracking $49.3 \pm 0.7\%$ and +x $51.5 \pm 0.8\%$. The paired difference between +x and Self-Backtracking is +2.0 to +2.2 points at every budget from $N = 4$ on and positive on every seed, and +1.7 to +2.3 points under the harness rule ($+0.7 \pm 0.7$ and $+0.8 \pm 0.9$ at $N = 2$). Exchanging inference procedures suggests that the loss accounts for most of the difference in this setting: at $N = 32$, the +x model with backtracking reaches $51.6 \pm 0.9\%$, and the Self-Backtracking model with restart reaches $49.3 \pm 0.7\%$, close to their results with the original procedures. One round of backtracking adds 0.3 to 1.6 points over selecting from the same initial samples without backtracking ($b = 0$). We also evaluate the released-code variant, which expands only one shortened prefix and does not expand it if no equations remain, and a temperature-0.7 variant. Both are shown in the figure and perform at or below the $\lfloor \sqrt{N} \rfloor$ rule at temperature 1 under likelihood selection.