Tracking World States with Language Models: State-Based Evaluation Using Chess

Romain Harang [1] **Jason Naradowsky** [1] **Yaswitha Gujju** [1] **Yusuke Miyao** [1]

Abstract

Large Language Models (LLMs) exhibit emergent capabilities in structured domains, suggesting they may implicitly internalize high-fidelity representations of world models. While probing techniques have shown promising signs of this in scientific and game-based settings, they rely on model-specific internal activations, which limit interpretability and generalizability. In this work, we propose a model-agnostic, state-based evaluation framework using chess as a benchmark to assess whether LLMs preserve the semantics of structured environments. Our method analyzes the downstream legal move distributions (state affordances) to estimate semantic fidelity between predicted and actual game states. This approach offers a more meaningful evaluation than conventional string-based metrics by aligning more closely with the strategic and rule-governed nature of chess. Experimental results demonstrate that our metrics capture deficiencies in state-tracking, highlighting limitations of LLMs in maintaining coherent internal models over long sequences. Our framework provides a robust tool for evaluating structured reasoning in LLMs without requiring internal model access, and generalizes to a wide class of symbolic environments.

1. Introduction

Large Language Models (LLMs), consisting of billions of parameters trained on massive text corpora, have demonstrated capabilities far beyond their original next-token prediction task. Recent studies suggest that this enhanced ability arises from their implicit recovery of high-fidelity representations of structured domains embedded within their training data. This implicit modeling aligns closely with the concept of a "world model," defined in (Ha & Schmidhuber, 2018) for Neural Networks in general. In our case, it can be understood as a representation where an environment can be summarized by a finite set of states and rules governing transitions between them—effectively modeled as a deterministic finite automaton (DFA) (Vafa et al., 2024).

Language models have shown promise in recovering such world models purely from sequential data in complex scientific domains like protein design, genetics, and chemistry (Chowdhury et al., 2022; Lin et al., 2023; Benegas et al., 2023; Jablonka et al., 2024). This ability offers a powerful alternative to explicitly constructing detailed representations of complex environments, highlighting the capacity of language models to extract rich domain knowledge solely from sequences. However, these successes rest on a critical assumption: that the language model has genuinely internalized the underlying world model. This raises a fundamental question—how can we reliably determine whether a sequence model has truly learned the domain's structure?

A common strategy for evaluating whether a model has internalized a world model involves probing its internal neural representations to see if they can reconstruct real-world states (Hewitt & Liang, 2019; Li et al., 2021; Abdou et al., 2021; Jin & Rinard, 2024; Li et al., 2023). For example, (Toshniwal et al., 2022) and (Li et al., 2023) evaluate whether sequence models trained on board game transcripts, such as chess and Othello, have internalized the underlying game rules. However, these probing-based methods rely heavily on accessing and interpreting internal model states, which can be model-specific, opaque, and challenging to generalize. Furthermore, evaluating the quality of generated move sequences in chess remains difficult because most existing metrics focus on syntactic-level comparisons, such as exact match, edit distance, Levenshtein distance, or direct board state comparisons. While straightforward to compute, these metrics fail to capture the strategic and semantic richness of chess, where moves vary greatly in their impact—some drastically influence the course of the game, while others are strategically neutral or suboptimal.

Motivated by these limitations, our work proposes a model-agnostic, sequence-based evaluation framework that di-

[1]Department of Computer Science, University of Tokyo, Bunkyo, Japan. Correspondence to: Romain Harang <romain-harang@g.ecc.u-tokyo.ac.jp>.

ICML 2025 Workshop on Assessing World Models.

rectly examines the model's generated outputs to determine whether they preserve the semantic properties of the original game state. We accomplish this by analyzing the space of valid action sequences that can unfold from a given position and evaluating the similarity between predicted and actual states based on the moves they enable. Although computationally more demanding than string-based metrics, our approach provides a richer and more informative signal, capturing whether an LLM's output retains the tactical and strategic affordances inherent in real chess positions. This framework enables robust inference and also applies broadly across different model architectures without requiring probing of internal representations.

2. Background

We base our framework on Finite State Automata (FSA), which offer a natural way to model state tracking in structured environments. By comparing the sets of valid continuations from different action sequences, we capture a semantics-aware notion of similarity.

2.1. State tracking

Finite State Automata We define a Finite State Automaton (FSA) as the tuple $\mathcal{A} = (\mathcal{S}^*, \Sigma, \delta, S_0)$ We define the setting as follows. Let $\mathcal{S}$ denote the finite set of valid states. We augment this set to $\mathcal{S}^* = \mathcal{S} \cup \{0\}$, where 0 represents a special error or sink state that captures invalid transitions. The input alphabet, denoted by Σ, is the finite set of actions that can be applied to states. The process begins from an initial state S_0, which belongs to $\mathcal{S}$. Transitions between states are governed by a function δ, which maps a pair consisting of a current state (from $\mathcal{S}^*$) and an action (from Σ) to a new state in $\mathcal{S}^*$. Each state $S \in \mathcal{S}$ has an associated set of permitted actions, denoted $\Sigma_S \subseteq \Sigma$.

Transition Sequences. Given an action sequence $s = (a_1, \ldots, a_n) \in \Sigma^{\mathbb{N}}$ and the initial state S_0, the resulting sequence of states $(S_1, \ldots, S_n)$ is defined recursively by $S_i = \delta(S_{i-1}, a_i), \quad \text{for } i = 1, \ldots, n$. We can also extend the transition function's definition over a sequence of moves rather than a single move, for example here: $S_n = \delta(S_0, s)$. Thus, an action sequence s uniquely determines the final state S_n, provided the transition function δ and initial state S_0 are known. In this sense, the final state is a function of the sequence.

Evaluating State Tracking. To assess whether a model has internalized a form of world-modeling or state tracking, we examine its performance on structured action sequences. The model only uses action sequences as inputs, without access to the explicit underlying state structure. If generating valid continuations requires an implicit estimate of the current state, we may conclude that the model has tracked the relevant state information. This framework allows us to test a model's state awareness: if it can accurately predict or generate valid next actions under the constraints imposed by δ, then it implicitly represents a state in a manner consistent with a world model.

2.2. Existing approaches

Existing approaches are typically developed with the specific task in mind. In the case of chess, which we consider in our experiments, the two main evaluation metrics are board accuracy and edit distance (Feng et al., 2023). However, these techniques do not take into account the affordances associated with a particular board state. As a result, they can yield "false positives" in certain scenarios. For example, if two states differ only by the removal of a king, the edit distance might be as low as 1 and the board accuracy could exceed 98%, even though the resulting state is nonsensical from an affordance perspective.

While we expect the metrics derived from our approach to be correlated with these baseline metrics, we argue that they more directly capture the LLM's understanding of the task. Specifically, a higher score in our framework consistently corresponds to a state that is semantically closer to the ground truth, unlike traditional metrics. Although it is possible to introduce heuristic weighting into edit distance or accuracy, a key advantage of our method is that it requires no prior knowledge of the task. It can be applied directly in any setting, as long as δ and Σ are defined.

3. Main idea

3.1. State reconstruction task

We define a complementary evaluation objective, the state reconstruction task, which assesses whether a model can explicitly generate the current environment state after observing a sequence of actions. In domains where states admit a textual representation (e.g., algebraic notation in chess), we prompt the model to produce this representation given only the preceding action sequence. While failure to reconstruct the state does not conclusively imply that state tracking has not occurred (e.g., due to formatting or generation noise), successful reconstruction provides strong evidence that the model has internalized a structured world model capable of state estimation.

3.2. Metrics

Given an action sequence $s = (a_1, \ldots, a_n)$ and its corresponding true state $S = \delta(S_0, s)$, we prompt the model to generate a predicted state $\tilde{S}$ using s as an input. We evaluate the model's performance by comparing S and $\tilde{S}$ using a

suite of metrics designed to capture both syntactic accuracy and semantic fidelity.

State based metrics To better capture semantic correctness, we define metrics based on the sets of valid action sequences under a given state. Let $\mathcal{A}_S^m$ denote the set of all valid action sequences of length m starting from state S. We compare the sets $\mathcal{A}_S^m$ and $\mathcal{A}_{\tilde{S}}^m$ using the precision/recall formulation in Appendix A. In practice, computing these sets exactly is infeasible due to their exponential size in m. However, uniform sampling from $\mathcal{A}_S^m$ is itself intractable. Instead, we approximate this via **uniform branch sampling**: at each step i, we sample an action a_i uniformly from the valid set $\Sigma_{S_{i-1}}$ and apply it via the transition function to obtain $S_i = \delta(S_{i-1}, a_i)$. Repeating this m times yields a trajectory $s = (a_1, \ldots, a_m)$. Let $U_b(S)$ denote the distribution over such sequences. We then define approximate precision and recall as:

$$p_m(S, \tilde{S}) = \mathbb{E}_{s \sim U_b(\tilde{S})} \left[\mathbf{1}_{\mathcal{A}_S^m}(s) \right]$$

$$r_m(S, \tilde{S}) = \mathbb{E}_{s \sim U_b(S)} \left[\mathbf{1}_{\mathcal{A}_{\tilde{S}}^m}(s) \right]$$

These quantities reflect how well the predicted state $\tilde{S}$ preserves the behavior of the true state S, in terms of valid action trajectories. While state-based metrics are more faithful to the underlying semantics of state prediction, they are computationally expensive and depend on the trajectory length m. In practice, m can be selected based on task complexity or evaluation constraints. Despite their cost, these metrics provide a much richer and more actionable signal than simpler string-based comparisons.

Expected Values Consider a simplified case where the tree of possible action sequences generated from a state S is *homogeneous*, meaning that at each node, the proportion of child nodes corresponding to valid continuations under the state $\tilde{S}$ is constant. Let p denote this proportion, i.e., the probability that a randomly selected legal action from any state results in a sequence accepted by $\tilde{S}$.

In this setting, since each step in the sequence independently maintains a success probability p, the expected probability that a full sequence of length m is accepted by $\tilde{S}$ is $p_m = p^m$. This analysis reveals that p_m decays exponentially with the sequence length m in the homogeneous case. Moreover, even in non-homogeneous trees, if the local acceptance probability at each step, denoted $p_s = \mathbb{P}(s_i \in \mathcal{A}_{\tilde{S}}^1 \mid s_{<i})$, is uniformly bounded above by some constant $M < 1$, then p_m still decays exponentially as $p_m \leq M^m$. This exponential behavior highlights a fundamental challenge in estimating long-horizon compatibility between predicted and true states, motivating the need for careful design of sampling strategies and smoothing techniques in practice.

4. Sampling algorithm

Sample Complexity Analysis In our estimation procedure, we sample N sequences of length m from a given state. To ensure that the standard error of our estimate is of the same order of magnitude as the mean p_m, we analyze the variance of the binary indicator variable $\mathbf{1}_{\mathcal{A}_{\tilde{S}}^m}(s)$, which follows a Bernoulli distribution with parameter p_m.

The variance is $p_m(1 - p_m)$ while the standard error (SE) of the mean over N independent samples is $\text{SE} = \sqrt{\frac{p_m(1-p_m)}{N}} \approx \sqrt{\frac{p_m}{N}}$, where the approximation assumes $p_m \ll 1$, which holds due to the exponential decay of p_m with m.

To achieve a signal-to-noise ratio (mean divided by standard error) of order one, we require $\frac{p_m}{\text{SE}} \approx \sqrt{N p_m} \approx 1$, implying that $N \approx \frac{1}{p_m}$.

In the homogeneous tree case, we previously showed that the probability of a valid sequence of length m is given by $p_m = e^{-\lambda m}$, for some constant $\lambda = -\log p > 0$, where p is the branching probability. Substituting this into our sample complexity estimate, we obtain $N = O(e^{\lambda m})$. This result implies that the sample complexity of our estimation procedure grows exponentially with the sequence length m.

4.1. Intermediate probability estimator

Because a naive sampler must run for an exponential increasing amount of shots with respect to the depth we look at using an estimator based on conditional probabilities $p(s' \in \mathcal{A}_{\tilde{S}}^m | s \in \mathcal{A}_{\tilde{S}}^m)$ where s' is s with a random action appended. Note that we call this quantity p_s. Now we look for a certain m at $\mathbb{E}_{s \sim U_b(S)}(p_s) = v_m$ (important $p_m \neq v_m$). From this we reconstruct $p_m = \Pi_{i=1}^m v_i$. In the case of an homogeneous tree the expectation of this product is p^m as $v_m = p$. Here the error for each v_i is $(1 - v_i)\sqrt{\frac{v_i}{N}}$. If we maintain $(1 - v_i)\sqrt{\frac{v_i}{N}} << v_i$ the total error can be approximated to $\sum_{i=1}^m (1 - v_i)\sqrt{\frac{v_i}{N}}$. If we fall back to an homogeneous tree then the error is $m(1 - p)\sqrt{\frac{p}{N}}$. From this we get $N = O(m^2(1 - p)^2 p)$. Here the number of necessary shots increases quadratically with m instead of exponentially.

5. Experimental Setup

5.1. Metrics validation

We first validate our proposed metrics by examining their behavior on two fixed chessboard positions derived from random move sequences. We analyze the evolution of p_m with increasing m and assess the variance across multiple runs in figure 1.

With a fixed sample size $N = 500$, the variance grows with

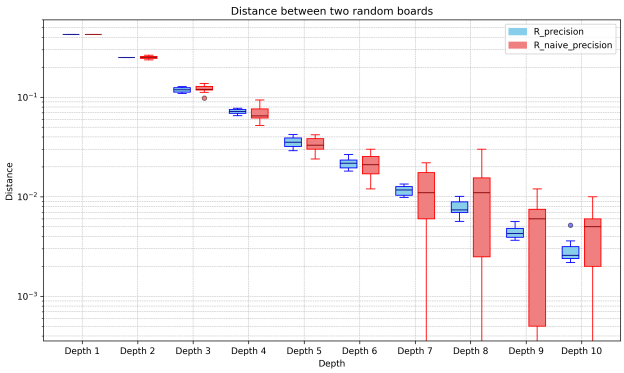

Figure 1. p_m as a function of m for two fixed states.

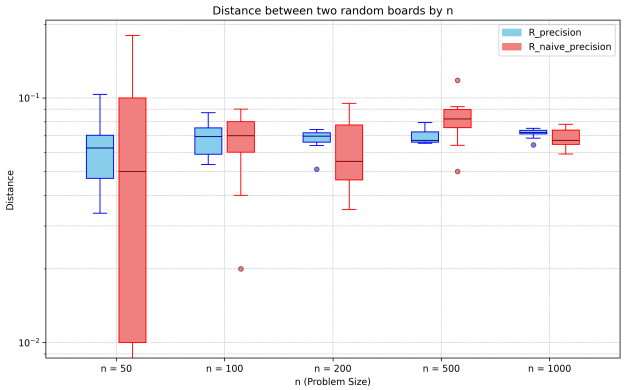

Figure 2. p_m as a function of N for two fixed states at $m = 4$.

m on a log scale. As expected, the naive estimator becomes increasingly noisy as m increases, while our second method remains significantly more stable.

Next, we examine the impact of sample size N at a fixed depth $m = 4$ for the same states in figure 2.

The variance of our second estimator decreases substantially faster than the naive method. This advantage grows with deeper sequences or more divergent states, where v_m approaches zero. For reference, the states differ by an edit distance of 17 and have zero exact matches.

5.2. Comparison to previous metrics

We look at the correlations between our proposed metrics and the edit distance (levenshtein distance). To do so, we select a sample of 10000 real chess games selected from the website Lichess. We then separate in groups of 2000 samples, for each we select a specific game length (from 5 to 50 moves) and we cut the game to the specified length (we make sure beforehand that the total games are longer). Then we give to openai's GPT4o model the pgn interpretation of the game as well as instructions on what it has to do (convert to the "FEN" standard board representation of the game). Then we use our metrics (we take depth $m = 4$) as well as levenshtien distance to evaluate our outputs.

In order to tell how correlated our metrics are to the edit distance we look at the kendall's tau of the two distributions; state precision and $-1\times$ edit distance, first for the entirety of our sample and then by group. The overall kendall's tau is of 0.69, which can be interpreted as a strong correlation. However upon looking at values by groups we discover a different story, see figure 3

We see that the correlation actually decreases with the number of moves of the game. This can be interpreted as when the number of moves increases the scores overall gets lower (see figure **??**) and when they do the metrics we use become almost uncorrelated.

The sharp decline in p_m as k increases highlights GPT-4o's growing difficulty in accurately reconstructing the board state. Notably, the probability of producing a legal next move remains near one, indicating that the model still captures some aspects of valid gameplay. For comparison, the probability of a legal next move on random boards after 5 moves is approximately $(8 \pm 2) \times 10^{-4}$, whereas GPT-4o achieves an average p_m around 0.6. This demonstrates that while the model's states are significantly better than random, performance degrades on longer sequences, reflecting challenges in both long-range state tracking and precise board reconstruction.

6. Discussion

A major limitation of our method is its sensitivity to the prompting strategy used to query the language model. Variations in phrasing, formatting, or the inclusion of intermediate reasoning steps (e.g., chain-of-thought prompting) can substantially affect the generated state representations. While this reflects a realistic deployment scenario, it complicates the interpretation of benchmarking results. Systematic studies on prompt robustness, prompt tuning, or self-verification techniques could help mitigate this. Another limitation is the assumption of access to a reliable and executable action model for the environment (e.g., the rules of chess), which may not be available or tractable in more open-ended or less formalized domains. Our approach also assumes that the ground-truth states are accurately labeled and that the representation space (e.g., FEN for chess) is sufficiently expressive to capture task-relevant differences. This assumption may not hold in domains with latent or ambiguous state representations. Additionally, although we framed the m-step precision/recall computation as a hyperparameter-dependent process, it introduces a trade-off between metric sensitivity and computational cost. Future

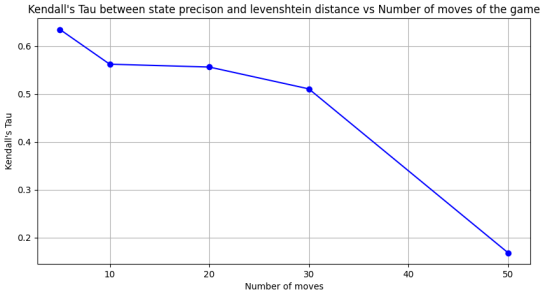

Figure 3. Values of Kendall's tau per group with varying number of moves per group

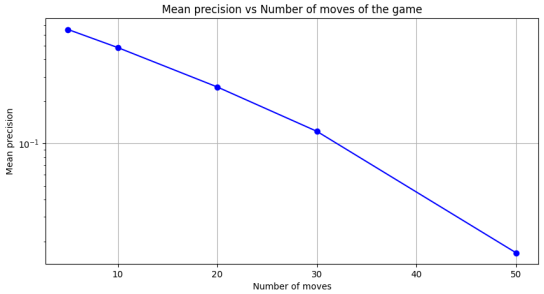

Figure 4. p_4 on average for each group (log scale, values from 0.6 to 0.015)

work could explore strategies to average over multiple values of m, or replace it with an adaptive stopping criterion. We also note the possibility of using importance sampling or guided rollouts based on fitness functions or model confidence to further improve sampling efficiency. Finally, while we focused on the forward action space as the basis for our metric, a more comprehensive evaluation could also incorporate backward reasoning (e.g., whether the state is plausible given earlier context), or model uncertainty.

7. Conclusion

We propose a novel framework to evaluate language models' state-tracking abilities through metrics grounded in downstream task validity, rather than superficial representation similarity. Our key contributions include new evaluation metrics, notably p_m and r_m, and efficient sampling-based estimators capable of handling exponentially large search spaces. Experiments on the chess domain demonstrate that our metrics provide sensitive and reliable assessments, revealing notable limitations in state reconstruction even for powerful models like GPT-4o as sequence lengths increase. Compared to traditional metrics such as exact match or edit distance, our approach better captures the semantic correctness of predicted states by considering their impact on valid subsequent actions. Despite its strengths, our method is sensitive to prompt design and computationally intensive for large depths m. Future work includes addressing these challenges and extending the framework to other structured domains such as program synthesis, dialog tracking, and robotic planning. Overall, this work offers a more principled, task-aware evaluation paradigm that aligns model assessment with downstream utility in structured reasoning tasks.

References

Abdou, M., Kulmizev, A., Hershcovich, D., Frank, S., Pavlick, E., and Søgaard, A. Can language models encode perceptual structure without grounding? a case study in color. *arXiv preprint arXiv:2109.06129*, 2021.

Benegas, G., Batra, S. S., and Song, Y. S. Dna language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences*, 120(44):e2311219120, 2023.

Chowdhury, R., Bouatta, N., Biswas, S., Floristean, C., Kharkar, A., Roy, K., Rochereau, C., Ahdritz, G., Zhang, J., Church, G. M., et al. Single-sequence protein structure prediction using a language model and deep learning. *Nature Biotechnology*, 40(11):1617–1623, 2022.

Feng, X., Luo, Y., Wang, Z., Tang, H., Yang, M., Shao, K., Mguni, D., Du, Y., and Wang, J. Chessgpt: Bridging policy learning and language modeling, 2023. URL https://arxiv.org/abs/2306.09200.

Ha, D. and Schmidhuber, J. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Hewitt, J. and Liang, P. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368*, 2019.

Jablonka, K. M., Schwaller, P., Ortega-Guerrero, A., and Smit, B. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, 6(2):161–169, 2024.

Jin, C. and Rinard, M. Emergent representations of program semantics in language models trained on programs. In *Forty-first International Conference on Machine Learning*, 2024.

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023.

Toshniwal, S., Wiseman, S., Livescu, K., and Gimpel, K. Chess as a testbed for language model state tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11385–11393, 2022.

Vafa, K., Chen, J., Rambachan, A., Kleinberg, J., and Mullainathan, S. Evaluating the world model implicit in a generative model. *Advances in Neural Information Processing Systems*, 37:26941–26975, 2024.

A. Appendix

Finite State Automaton

State based metrics To better capture semantic correctness, we define metrics based on the sets of valid action sequences under a given state. Let $\mathcal{A}_S^m$ denote the set of all valid action sequences of length m starting from state S. We compare the sets $\mathcal{A}_S^m$ and $\mathcal{A}_{\tilde{S}}^m$ using a precision/recall formulation:

$$\text{Precision} = \frac{|\mathcal{A}_S^m \cap \mathcal{A}_{\tilde{S}}^m|}{|\mathcal{A}_{\tilde{S}}^m|}, \quad \text{Recall} = \frac{|\mathcal{A}_S^m \cap \mathcal{A}_{\tilde{S}}^m|}{|\mathcal{A}_S^m|}$$

In practice, computing these sets exactly is infeasible due to their exponential size in m. To address this, we express these quantities as expectations over indicator functions:

$$\text{Precision} = \mathbb{E}_{s \sim \mathcal{U}(\mathcal{A}_{\tilde{S}}^m)} \left[\mathbf{1}_{\mathcal{A}_S^m}(s) \right],$$

$$\text{Recall} = \mathbb{E}_{s \sim \mathcal{U}(\mathcal{A}_S^m)} \left[\mathbf{1}_{\mathcal{A}_{\tilde{S}}^m}(s) \right]$$

However, uniform sampling from $\mathcal{A}_S^m$ is itself intractable. Instead, we approximate this via **uniform branch sampling**: at each step i, we sample an action a_i uniformly from the valid set $\Sigma_{S_{i-1}}$ and apply it via the transition function to obtain $S_i = \delta(S_{i-1}, a_i)$. Repeating this m times yields a trajectory $s = (a_1, \ldots, a_m)$. Let $U_b(S)$ denote the distribution over such sequences. We then define approximate precision and recall as:

$$p_m(S, \tilde{S}) = \mathbb{E}_{s \sim U_b(\tilde{S})} \left[\mathbf{1}_{\mathcal{A}_S^m}(s) \right]$$

$$r_m(S, \tilde{S}) = \mathbb{E}_{s \sim U_b(S)} \left[\mathbf{1}_{\mathcal{A}_{\tilde{S}}^m}(s) \right]$$

These quantities reflect how well the predicted state $\tilde{S}$ preserves the behavior of the true state S, in terms of valid action trajectories. While state-based metrics are more faithful to the underlying semantics of state prediction, they are computationally expensive and depend on the trajectory length m. In practice, m can be selected based on task complexity or evaluation constraints. Despite their cost, these metrics provide a much richer and more actionable signal than simpler string-based comparisons.

- $\mathcal{S}$ is the finite set of valid states.
- $\mathcal{S}^* = \mathcal{S} \cup \{0\}$ is the augmented state space, including a special error (or sink) state 0.
- Σ is the finite input alphabet (i.e., the set of actions).
- $S_0 \in \mathcal{S}$ is the initial state.
- $\delta : \mathcal{S}^* \times \Sigma \to \mathcal{S}^*$ is the transition function.

The transition function is defined as:

$$\delta(S, a) = \begin{cases} S' \in \mathcal{S} & \text{if } a \in \Sigma_S \text{ and } S \neq 0, \\ 0 & \text{if } a \notin \Sigma_S \text{ or } S = 0. \end{cases}$$

This construction ensures that δ is total, i.e., it produces a well-defined output for all pairs $(S, a) \in \mathcal{S}^* \times \Sigma$. Once the automaton transitions to the error state 0, it remains there for all subsequent actions:

$$\delta(0, a) = 0 \quad \forall a \in \Sigma.$$

Algorithm 1 Intermediate Probability Estimation

Require: N: maximum list size, m: trajectory depth, s_1: starting state, s_2: comparison FSA
Ensure: Approximate $\mathbb{P}_{s\sim U_b(s_1)}[s \in \mathcal{A}_{s_2}^m]$ by computing $\prod_{i=1}^{k} v_k$ iteratively

$L \leftarrow [(s_1, 1)]$
for $i = 1$ to m **do**
  $L' \leftarrow [\,]$
  for all $(j, w) \in L$ **do**
    new $\leftarrow$ `Legal_moves_add`(j) ▷ Get legal next states
    $w' \leftarrow w/|\text{new}|$
    for all $m \in$ new **do**
      if s_2.`accepts`(m) **then**
        L'.`append`$((m, w'))$
      end if
    end for
  end for
  if $|L'| > N$ **then**
    $L' \leftarrow$ `sample`(L', N) ▷ During sampling weights are rescaled
  end if
  $L \leftarrow L'$
end for
return $\sum_{(m,w)\in L} w$

Exact Match We measure the probability that the predicted state exactly matches the ground-truth state:

$$\mathbf{p}(\tilde{S} = S)$$

This metric evaluates whether the model can perfectly reconstruct the correct state from the input sequence. It is strict and binary, any deviation from the target is counted as an error. As a result, low scores under this metric do not convey how close the predicted state is to the correct one, limiting its informativeness.

Edit distance To provide a more graded notion of correctness, we use the Levenshtein distance $\text{lev}(S, \tilde{S})$ between the textual representations of the true and predicted states. This measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into the other. Since $\text{lev}(S, \tilde{S}) \in \mathbb{N}$ and can be unbounded, we normalize it into the $[0, 1]$ range using an exponential kernel:

$$\mathbb{E}\left[e^{-\lambda \cdot \text{lev}(S,\tilde{S})}\right]$$

where $\lambda > 0$ is a hyperparameter. When λ is large, the metric behaves similarly to exact match; when small, it becomes insensitive to differences. A drawback of edit distance is that it treats all changes equally, regardless of their semantic impact; i.e., how a change affects the resulting valid action set Σ_S is not considered.

A.1. Naive algorithm

Algorithm 2 Naive Precision/Recall Estimation

Require: N: maximum number of sequences, m: depth, s_1: initial state, s_2: comparison FSA
Ensure: Approximate $\mathbb{P}_{s \sim U_b(s_1)}[s \in \mathcal{A}_{s_2}^m]$

$L \leftarrow [s_1]$
for $i = 1$ to m **do**
  $L' \leftarrow [\,]$
  for all $j \in L$ **do**
    $L' \leftarrow L' \cup \texttt{Legal\_moves\_add}(j)$
  end for
  if $|L'| > N$ **then**
    $L' \leftarrow \texttt{sample}(L', N)$
  end if
  $L \leftarrow L'$
end for
$K \leftarrow |L|$
$A \leftarrow 0$
for all seq $\in L$ **do**
  if s_2.accepts(seq) **then**
    $A \leftarrow A + 1$
  end if
end for
return A/K