The Expressiveness Power of LLM-Based Agent under the Constrain of Finite Context Length

Anonymous ACL submission

Abstract

Large Language Model (LLM)-based agents extend standard Transformers and Chain-of-Thought (CoT) by incorporating tools, memory, and interaction mechanisms, enabling dynamic reasoning and adaptive decision making. Despite their empirical success, their theoretical understanding remains lacking. In this work, we provide a formal characterization of the expressiveness of LLM-based agents. We prove that a finite capacity LLM-based agent with finite context length can achieve Turing completeness, allowing it to simulate arbitrary computational processes using bounded resources. Building on this result, we also analyze the trade-off between context length and environment interaction frequency, showing that extending context length can reduce costly environmental access. Our findings offer new theoretical insights into the expressiveness power and efficiency advantages of LLM-based agents in complex environments.

1 Introduction

Recently, LLM-based agents (Huang et al., 2025; Team, 2025; Guo et al., 2024; Tran et al., 2025) offer a powerful framework for extending the capacities of large language models beyond traditional static text generation. By equipping an LLM with tools, memory and reasoning routines, an agent can interact dynamically with its environment, decompose complex tasks, and adapt its behavior with feedback. This framework allows the system to plan, act and reflect in a structured manner. Moreover, LLM-based agents can integrate external knowledge from environment and perform iterative reasoning. Comparing with standard Transformer and Chain-of-Thought (CoT) (Wei et al., 2022), it significantly improves the problem-solving efficiency and reliability (Varambally et al., 2025; Qiu et al., 2025). These properties make LLM-based agents a promising direction for building general-purpose intelligent systems capable of autonomous decision-making and continual learning. However, despite their empirical success, the theoretical foundations explaining their effectiveness remain limited.

On another side, a series of recent theoretical results has shed light on the expressive power of Transformers and Chain-of-Thought (CoT) reasoning. Several works (Pérez et al., 2021; Liu et al., 2022; Nichani et al., 2024; Sander et al., 2024; Wu et al., 2025; Hao et al., 2025) focus on demonstrating the expressiveness of standard Transformers on sequential reasoning tasks. Another line of research (Feng et al., 2023; Merrill and Sabharwal, 2023; Li et al., 2024; Zhu et al., 2025) highlights the benefits of CoT by allowing Transformers to generate intermediate reasoning steps before producing the final answer. However, both directions largely overlook the role of **context length** in model performance, which is crucial for both efficiency and accuracy. In complex reasoning tasks involving many intermediate steps, a long context length can lead to substantial memory and computation overheads, reducing training and inference efficiency, and may also increase the risk of overfitting to irrelevant long-range dependencies (Du et al., 2025). In contrast, an agent can dynamically manage its context length through interaction with the environment, as well as through context management and data filtering mechanisms (Huang et al., 2025; Team, 2025; Kang et al., 2025b). This ability enables improved computational efficiency and accuracy, particularly in complex reasoning scenarios.

In this paper, we characterize the benefits of LLM-based agents by demonstrating that a finite

capacity Transformer with a finite context length can be Turing complete. This implies that, even when a Turing machine requires a large number of computational steps, its execution can be simulated by an LLM-based agent using only a finite context and finite model capacity. This stands in contrast to prior works (Pérez et al., 2021; Merrill and Sabharwal, 2023), which establish Turing completeness under the assumption of infinite context length. Our contributions can be summarized as:

1. **Expressiveness of LLM-based agent**: We establish the expressive power of LLM-based agents under constraints on both context length and model capacity. In particular, we show that an agent equipped with a finite capacity Transformer and a finite context length can achieve Turing completeness. The proof is constructed via a direct simulation of a Turing machine.

2. **Technical contributions**: On the technical side, we provide explicit constructions for both the reasoning Transformer model and the retrieval-augmented generation (RAG) procedure, clarifying how these components enable effective environment interaction and memory utilization.

3. **Trade-off between context length and access environment time**: Building on the expressiveness result above, we analyze the trade-off between context length and the frequency of the agent's interactions with the external environment. Since each interaction typically incurs a significant computational or communication cost, it is crucial to minimize the number of such accesses. Our analysis shows that this can be achieved by extending the context length, allowing the agent to internally store and utilize sufficient task-relevant information without frequent environmental queries.

2 Related Works

Expressiveness Power of Transformers and Chain-of-Thought The expressive power of Transformers has been extensively studied from multiple perspectives. For instance, Akyürek et al. (2022); Von Oswald et al. (2023); Mahankali et al. (2023); Dai et al. (2022) show that a single attention layer is sufficient to perform one step of gradient descent. Garg et al. (2022); Bai et al. (2024a); Guo et al. (2023) demonstrate that Transformers can implement a broad class of machine learning algorithms in context, while Xie et al. (2021); Wang et al. (2023); Jiang (2023) establish their ability to approximate Bayesian-optimal inference. Other works investigate additional capabilities: Liu et al. (2022) show that Transformers can emulate automata transitions, Lin et al. (2023) demonstrate their ability to implement reinforcement learning algorithms, and Nichani et al. (2024) prove that they can learn Markov causal structures under a fixed transition matrix, ans Hao et al. (2025) extend the result to hidden Markov models. Sander et al. (2024); Wu et al. (2025) further analyze their expressiveness in learning autoregressive models. A recent line of research has explored how CoT reasoning enhances Transformer expressiveness. Feng et al. (2023) show that constant-depth Transformers equipped with CoT can solve certain P-complete problems, while Merrill and Sabharwal (2023) prove that logarithmic CoT steps in input length expand the expressivity upper bound from TC^0 to L. Li et al. (2024) show that, with sufficiently large embedding dimensions, constant-depth Transformers can solve any problem representable by Boolean circuits. Meanwhile, Hao et al. (2024); Zhu et al. (2025) study continuous CoT and its theoretical implications. Despite these advances, the role of *context length* that influences the expressiveness and efficiency of models remains underexplored, which our work is focused on.

Benefits of Agent Though pretrained transformer based LLMs already provide strong foundation abilities for general tasks (Yang et al., 2025; Guo et al., 2025; Zeng et al., 2025; Comanici et al., 2025), they fail to handle complicated scenarios due to restricted capabilities to deal with environment changes. Agents mitigate this shortcoming by equipping models with tools, environment feedback and other systematic abilities. One of the biggest distinctions between agents and ordinary LLMs is that agents can interact with the environment by calling external tools and observing the feedback they receive, thereby forming an interaction loop and evolving. Deep research agents (Huang et al., 2025; Team, 2025) could automatically search the Internet, retrieve relevant information and summarize. Multi-agent (Guo et al., 2024; Tran et al., 2025) system serves as another

way to integrate different expertise from different backbone models together, throught role-playing (Qian et al., 2023). Along with the rapid development of agents itself, accessories like agent memory system (Chhikara et al., 2025; Yang et al., 2024; Kang et al., 2025a) and agentic training framework (Sheng et al., 2024; Luo et al., 2025) appear during a short period. Ever since the release of DeepSeek-R1 (Guo et al., 2025), reasoning agents (Shang et al., 2025; Zhang et al., 2025; Yuan et al., 2025; Gao et al., 2025; Yao et al., 2025) constitute a large portion of all agent categories. Agentic workflows benefit from the ability to conduct tool calling, memory management, reasoning, self-reflection, etc. Therefore, agent-based frameworks usually surpass normal pure transformer models themselves (Varambally et al., 2025; Qiu et al., 2025), which provides a plausible justification for us to concentrate on the analysis of agentic transformer models.

3 Inspired from the Empirical Benefits

Here we conduct experiments and summarize existing results from relevant literature that demonstrate the expressive power of an agent system with retrieval and the external memory database.

Setting and Task To demonstrate the advantages of utilizing an agent system over standalone transformer-based LLMs, we conducted experiments on question-answering (QA) tasks. Our objective is to show that external memory and context management can significantly enhance model performance. Given that QA tasks typically demand a comprehensive understanding of context and precise information retrieval from extensive corpora, we hypothesize that offloading information into external storage—and retrieving only the most pertinent details—will improve final outcomes. The following sections detail the models, datasets, and evaluation frameworks employed in this study.

Models, Datasets and Framework Table 1 and Table 2 directly summarize the results from MemOS (Li et al., 2025) and LongBench (Bai et al., 2024b) respectively. In Table 3, we use HotpotQA (Yang et al., 2018), Triviaqa (Joshi et al., 2017) and NQ (Kwiatkowski et al., 2019) as the evaluation datasets. Due to efficiency issues, we randomly sample 500 examples for each dataset. We use FlashRAG (Jin et al., 2025) for experiment implementation. The retrieval corpus is wiki18_100w from `RUC-NLPIR/FlashRAG_datasets`[1]. We use `intfloat/e5-base-v2` (Wang et al., 2022) to build the indexes.

Results Analysis Table 1 demonstrates the benefits of effective context management, while Table 2 further highlights the advantages of retrieval-based methods. As shown in Table 3, for QA tasks that require precise access to specific pieces of information, LLMs equipped with external context management mechanisms, such as an external memory database or RAG, significantly outperform direct generation approaches. This performance gap underscores the importance of controlled access to the external environment. Moreover, by comparing full-context pure in-context generation with retrieval-based methods and direct generation, we observe that, without retrieval, models struggle to effectively utilize information embedded in very long contexts.

In real-world settings, LLMs are inherently constrained by finite context windows and limited model capacity, which fundamentally restrict their problem-solving capabilities. The above experiments demonstrate that effective context management, combined with access to external environment, can substantially mitigate these limitations and markedly enhance model performance. These empirical findings naturally motivate a theoretical investigation into how and why these strategies expand the expressive capacity of LLM-based agents.

4 Preliminary

Some previous work has shown that Transformers can be Turing complete under the assumption of infinite context length (Pérez et al., 2021; Merrill and Sabharwal, 2023). However, this assumption is unrealistic in real world settings, where both context length and model capacity are inherently finite, thereby fundamentally limiting the computational capabilities of such models. In this section, we show that an LLM-based agent augmented with RAG can achieve Turing completeness even under finite context length and finite model capacity, by leveraging interaction with external memory.

[1] https://huggingface.co/datasets/RUC-NLPIR/FlashRAG_datasets

Method	LOCOMO	LongMemEval	PrefEval-10	PersonaMem
GPT-4o-mini	52.75	55.40	2.80	43.46
MemOS	75.80	77.80	71.90	61.17

Table 1: Performance comparison between vanilla LLM (GPT-4o-mini (Menick et al., 2024)) and memory augmented system (MemOS (Li et al., 2025)). We directly report the results from MemOS.

Model	Setting	Single-Doc Avg	Multi-Doc Avg	Overall Avg
GPT-3.5-Turbo-16k	w/o retrieval	45.10	36.25	40.70
	w/ retrieval	41.60	38.15	39.90
Llama2-7B-chat-4k	w/o retrieval	21.65	18.20	19.90
	w/ retrieval	24.98	23.05	24.00
ChatGLM2-6B-32k	w/o retrieval	37.63	34.65	36.10
	w/ retrieval	35.48	33.40	34.40

Table 2: Comparison of w/o retrieval vs best retrieval method on LongBench (Bai et al., 2024b). We report average scores over Single-Doc QA (1-1 to 1-4) and Multi-Doc QA (2-1 to 2-4) from the original paper.

4.1 Workflow of Agent

The LLM-based agent operates through the following sequential stages:

1. **Perception**: The agent collects contextual information from the environment, including task instructions, historical interactions, and external observations. This information is processed and formatted as input text or structured tokens to the Transformer.

2. **Inference**: Based on the input context, the Transformer functions as the reasoning core (the "brain" of agent), simulate one-step inference in the specific task.

3. **Action**: Based on the reasoning output, the agent selects and executes an action. Actions may include producing textual outputs, invoking external tools or APIs, querying databases, or interacting with an environment interface.

4. **Feedback and Memory Update**: The agent receives feedback from the environment or tool execution, which is incorporated into its memory or context buffer. This update allows the agent to refine its understanding, track progress, and adjust its strategy in subsequent iterations.

5. **Iteration**: The cycle on 1-4 repeats until a termination condition is met, such as task completion or convergence to a final output.

4.2 Turing Machine

We start from a standard reasoning setting, i.e, the Turing machine, which serves as a fundamental model for characterizing the computational capacity required to implement arbitrary algorithms. Here we formalize the *Turing Machine* M as follows. Let Σ denote the finite set of tape symbols (including the blank symbol $\emptyset$). The machine M is characterized by the following components:

1. $q_i \in Q$: state of the machine at step i;

2. $s_i \in \Sigma$: symbol read from the tape at step i;

3. $v_i \in \Sigma$: symbol written on the tape at step i;

4. $m_i \in \{-1, +1\}$: head movement direction at step i, where -1 indicates a left move and $+1$ indicates a right move;

5. $\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, +1\}$: transition function, i.e, at step i, we have $\delta(q_i, s_i) = (q_{i+1}, v_i, m_i)$.

Starting from an initial tape configuration $s_1 s_2 \cdots s_n$ occupying the first n positions on the left side of the tape and an initial state q^0, the machine M evolves by iteratively applying the transition rule δ to update its internal state, write new symbols, and move the head along the tape. A system is said to be Turing complete if it can simulate this process until reaching a final halting state. In our setting, the final state is denoted as either 0 or

Model	Method	HotpotQA (F1)	TriviaQA (F1)	NQ (F1)
Llama-3-8B-Instruct	Direct	26.95	60.67	29.19
	Full context	7.26	12.28	8.40
	RAG	33.32	65.97	35.24
Qwen2.5-7B-Instruct	Direct	28.98	49.09	22.54
	Full context	26.00	56.99	27.39
	RAG	44.15	71.11	49.17
glm-4-9b-chat-hf	Direct	27.06	46.87	27.85
	Full context	0.67	1.15	0.49
	RAG	34.22	66.86	38.26

Table 3: Results (%) of RAG on three benchmarks. "Direct" stands for prompting LLMs to answer directly without any given context. We simulate injecting full context as input by doing RAG with a high top-k (100 in our experiments), and simulate context management (denoted as RAG) by doing retrieval with a smaller top-k (10 for Llama-3-8B-Instruct, and 20 for the other two).

1: if the machine halts in state 1, the final string on the tape is accepted; otherwise, if it halts in state 0, the string is rejected.

Turing Task. We define the *Turing Task* as the problem of simulating the execution of the Turing machine M, making justification at the final step and output the final string written on the tape. In particular, our goal is to construct a finite capacity Transformer model to make inference for the agent. To be specific, at current step t, given a fixed-length context encoding the past several tokens within the selected computation history $\mathcal{C}_t$, i.e, $\{(q_j, s_j, m_j, \cdots)\}_{j\in\mathcal{C}_t}$, the agent aim to correctly predict the next transition $(q_{t+1}, v_t, m_t) = \delta(q_t, s_t)$, then repeating the computational process of M. This formulation captures the essential components of the Turing machine within a finite-context learning framework, enabling us to investigate the expressive power of agent in emulating algorithmic reasoning.

4.3 Transformer Architecture

Attention head. We first recall the definition of the (self-)Attention head $\mathcal{A}ttn(\cdot, Q, K, V)$. With any input matrix M,

$$\mathcal{A}ttn\left(M, Q, K, V\right) = \sigma\left(MQK^TM^T\right)MV,$$

where $\{Q, K, V\}$ refer to the Query, Key and Value matrix respectively. The activation function $\sigma(\cdot)$ can be row-wise softmax function[2].

[2]Given a vector input v, the i-th element of $\mathrm{Softmax}(v)$ is given by $\exp(v_i)/\sum_j \exp(v_j)$.

Transformer. Based on the architecture of Attention head, with the input matrix M, the definition of multi-head multi-layer Transformer $\mathrm{TF}(\cdot)$ is give by

$$H^{(0)} = \mathcal{S}_i,$$

$$H^{(l)} = H^{(l-1)} + \sum_{m=1}^{M_l} \mathcal{A}ttn\left(H^{(l-1)}, Q_m, K_m, V_m\right),$$

for any $l \in [N]$, where N refers to the number of Transformer layers, and M_l is the number of Attention heads on the l-th layer.

ReLU Neural Network. We consider a feedforward neural network parameterized by weight matrices and bias vectors, where each hidden layer applies the ReLU activation function. Formally, for an input vector $x \in \mathbb{R}^d$, a ReLU network with L layers computes

$$f(x) = W_L\,\sigma(W_{L-1}\,\sigma(\cdots\sigma(W_1x+b_1)\cdots)+b_{L-1})+b_L,$$

where $\sigma(z) = \max(0, z)$ is applied elementwise.

Token Embedding. As the process of Turing Machine need to perform step-by-step, here we update the input context for each step. To be more specific, for step-i, we define the input matrix as:

$$S_i := [\bar{y}, \widetilde{y}, \widetilde{y}_c, y_{i-1}, y_i, y_{i+1}]^T \in \mathbb{R}^{6\times d},$$

where $d \geq 5(|Q| + |\Sigma| + 10)$. The last column $y_{i+1} \in \mathbb{R}^d$ is

$$[0^T_{|Q|+|\Sigma|+3}, \cos(i{+}1/T), \sin(i{+}1/T), 1, 0^T_{d-|Q|-|\Sigma|-6}]^T,$$

which only contains some $\{0, 1\}$ constants and position information. We use it to store the information on next step-$(i+1)$. $y^i \in \mathbb{R}^d$ embeds the information at the i-th step as:

$$[q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, 0_{d-|Q|-|\Sigma|-6}]^T.$$

$q_i \in \mathbb{R}^{|Q|}$ is the one-hot vector embedding of the state at step-i, which takes values within $\{e_1, \ldots, e_{|Q|}\}$, $s_i \in \mathbb{R}^{|\Sigma|}$ is the one-hot vector embedding of the read symbol at step-i, $m_{i-1} \in \{-1, +1\}$ refers to the movement at step-$(i-1)$, $c_i \in \mathbb{R}^2$ is the embedding of current position on the tape at step-i, which could be expressed as (the task start with the 0-th position on the tape):

$$c_i = \left[\cos\left(\sum_{j=1}^{i-1} m_j/T\right), \sin\left(\sum_{j=1}^{i-1} m_j/T\right)\right], \tag{1}$$

the following two dimension $[\cos(i/T), \sin(i/T)]$ is the position embedding for step index, and the last $(d - |Q| - |\Sigma| - 5)$-dim vector is the fixed embedding, with elements of ones and zeros. And for all steps $j \leq i-1$, the embedding of y_j has the format as:

$$[q_j^T, s_j^T, m_{j-1}, c_j^T, \cos(j/T), \sin(j/T), 1, q_{j+1}^T, v_j^T, m_j, 0].$$

The other column in $\mathcal{S}_i$ is selected from the database: $\bar{y}$ is defined as:

$$\bar{y} := \begin{cases} y_{i+1}, & i \leq n-1, \\ y_n, & i > n-1, \end{cases} \tag{2}$$

$\widetilde{y}$ has the same state $\{\widetilde{q}, \widetilde{s}\}$ comparing with $\{q_i, s_i\}$ in y_i, and $\widetilde{y}_c$ is the latest token which has the same tape position information comparing with c_{i+1}[3].

5 The Expressiveness Power of Agent with Finite Context Length

In this section, we provide our main result:

Theorem 1. *A Large Language Model (LLM)-based agent is Turing complete with a* 4*-layer, single-head Transformer architecture equipped with ReLU activations and a maximum context length of* 6*. Specifically, it can perform Turing Task using a chain-of-thought reasoning process, while maintaining both finite model capacity and bounded context length.*

Theorem 1 shows that LLM-based agent can be Turing complete, it could perform any length intermediate steps of Turing Machine with finite length context as well as finite capacity Transformer. We now show the complete proof of Theorem 1. For readability, the detailed construction of the Transformer is relegated to Appendix B.

[3]If there is $\emptyset$ previously on c_{i+1}, we choose $\widetilde{y}_c$ as y_i

Following the sequential stages mentioned above, the workflow of the corresponding agent is as follows:

1. **Perception**: Embeds the collected contextual information as input of Transformer.
2. **Inference**: Based on the input, for each time step-i, use the Transformer to generate y_{i+1} based on the inputs.
3. **Action**: Based on the state of the output, firstly, the agent will follow Algorithm 1, update the information on $\mathcal{TP}$ in the external environment. After that, it will take actions case by case:
   (a) If q_{i+1} is an halting state and $q_{i+1} = 0$, the agent will terminate the whole process and output state 0.
   (b) If q_{i+1} is an halting state and $q_{i+1} = 1$, the agent will terminate the whole process and use tools to query $\mathcal{TP}$ in the environment, and print the corresponding symbols to the target space.
   (c) If q_{i+1} is not a terminal state, the agent will use tools to retrieval tokens as the input of next iteration (The detailed process can be seen in Section 5.1).
4. **Feedback and Memory Update**: From the result in previous steps, update the context information.
5. **Iteration**: Repeat the above process until reaching the final halting state.

External Environment. In our setting, we do not impose any constraints on the capacity of the external environment. The environment is assumed to include a database, the complete history of interactions at each step, and an infinitely long tape for storing tape strings. All of this information is represented using the same token embedding scheme described in Section 4.3.

Remark 1 (Context management)**.** *In the finite length prompts, we use tools to retrieval data on the external environment, and design a slide window to capture the short-term history.*

Remark 2 (Compare with previous models)**.** *Comparing with chain-of-thought and in-context learning, LLM-based agent can decide whether stop*

Algorithm 1 Information updating on $\mathcal{TP}$

Input: New token y_i, current $\mathcal{TP}$ in the external environment.
for All token y' on $\mathcal{TP}$ **do**
  Calculate the cosine similarity t' of tape position between y_i and y'.
  Record the tokens $\mathcal{S}_i$ with the maximum value of t'.
end for
if $|\mathcal{S}_i| = 1$ and the maximum cosine similarity equals to 1 **then**
  Replace the token within $\mathcal{S}_i$ by y_i
else if $|\mathcal{S}_i| = 1$ and its element is on the left side **then**
  Put y_i on the left side of the element in $\mathcal{S}_i$.
else if $|\mathcal{S}_i| = 1$ and its element is on the right side **then**
  Put y_i on the right side of the element in $\mathcal{S}_i$.
else
  Put y_i between the elements of $\mathcal{S}_i$.
end if
Output: $\mathcal{TP}$.

the process or not automatically. From the context length aspect, it does not need infinite context length. With tools and interaction with environment, it can be Turing complete while using a 4-layer Transformer with context length at most 6*. Consider efficiency, with RAG on external environment, agent can make the inference more efficient.*

5.1 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a well-established technique designed to enhance the reasoning and generation capabilities of LLMs by integrating an external knowledge retrieval mechanism. This external database can include both general-purpose and domain-specific knowledge sources (Lewis et al., 2020), as well as historical interactions between the agent and its environment over time. Recent surveys such as Gao et al. (2023) have summarized various RAG architectures and their implementation paradigms.

In our framework, we adopt the fundamental principle of RAG: storing knowledge and interaction histories in an external memory and retrieving only the most relevant information when necessary. This design enables efficient context management and mitigates the limitations imposed by finite context length in Transformer-based agents.

Knowledge Storage and Representation At each step in the simulation of a Turing machine, the agent records the tuple of current step into the external database. Since the Transformer's input embeddings at each step depend on two key components – $\widetilde{y}$, determined by the current state and tape symbol $\widetilde{q}, \widetilde{s}$, and $\widetilde{y}_c$, determined by the tape position c_{i+1} – it suffices to store the embedding tokens in the database. This compact representation captures all the essential information required for reconstructing subsequent steps of the Turing process without excessive redundancy.

Retrieval and Similarity Matching When the agent requires information from memory, it formulates a query embedding $\mathbf{e}_{\text{query}}$ that encodes the current reasoning context. The retrieval module then performs a similarity search between $\mathbf{e}$query and the embeddings of all stored entries $\mathbf{e}^i_{\text{data}}$ in the database. A common choice for similarity measurement is the cosine similarity $\text{sim}(\mathbf{e}_{\text{query}}, \mathbf{e}^i_{\text{data}}) = \langle \mathbf{e}_{\text{query}}, \mathbf{e}^i_{\text{data}} \rangle$. Alternative metrics, such as learned neural similarity functions, may also be used depending on the embedding model. The retrieval system then returns the top-k entries with the highest similarity scores, ensuring that only the most contextually relevant knowledge is utilized in subsequent computations.

Reconstruction and Context Integration Given well-trained embeddings, this retrieval mechanism allows the Transformer agent to effectively reconstruct the necessary context for each simulation step. Specifically, it can recover $\widetilde{y}$ by matching entries with the same state–symbol pair $\widetilde{q}, \widetilde{s}$ as the stored q_i, s_i, and reconstruct $\widetilde{y}_c$ by identifying entries corresponding to the same tape position c_{i+1}. Through this process, the agent dynamically integrates retrieved information into its working context, enabling coherent continuation of the Turing machine simulation even under the constraint of finite context length.

This RAG-based mechanism complements the theoretical framework of our Transformer agent by serving as an externalized memory system, which extends the model's effective context window. It allows the agent to selectively access historical or knowledge-based embeddings when needed, rather

than maintaining them within the Transformer's limited context buffer. Consequently, RAG not only reduces the computational overhead associated with long-context processing but also provides a scalable pathway to simulate Turing-complete behavior using bounded model capacity. This retrieval-enhanced setup forms a key component in demonstrating that even a finite-context Transformer agent can achieve full Turing completeness while maintaining efficiency and modularity.

5.2 Overview of Transformer Construction

The key point to prove Theorem 1 is the construction of the Transformer, which acts as the "brain" of the agent.

Layer 1: Lookahead Attention. The first layer handles the special initialization phase ($i \leq n$). Using positional attention, it encodes a lookahead symbol α_i and its positional embedding β_i, which distinguish whether the next symbol y_{i+1} belongs to the initial tape prefix.

Layer 2: Transition Simulation. The second layer implements the transition function δ via a single-head attention lookup. By assigning arbitrarily large attention weights to exact context matches, the model retrieves the correct next state q_{i+1}, write symbol v_i, and movement indicator m_i from previous occurrences.

Intermediate FFN (Tape Movement). A two-layer ReLU network updates the tape position embedding c_{i+1} from (c_i, m_i), effectively implementing a controlled rotation on the unit circle. This construction relies on Lemma 1.

Layer 3: Tape Reading Attention. The third attention layer retrieves the most recent symbol written at the current tape position. Here we define

$$l(i) := \begin{cases} \arg\max_{j<i} \{c_j = c_i\}, & \{j < i, c_j = c_i\} \neq \emptyset, \\ i-1, & \{j < i, c_j = c_i\} \neq \emptyset. \end{cases}$$

By combining step-position and tape-position similarity in the attention score, it uniquely selects the latest index $l(i+1)$ with matching tape coordinates and extracts the stored symbol $v_{l(i+1)}$.

Intermediate FFN (Symbol Resolution). A second ReLU network determines the next read symbol s_{i+1}. For $i+1 \leq n$, it copies the initial symbol α_i; otherwise, it outputs the symbol retrieved from the tape or $\emptyset$ if no prior write exists.

Layer 4: State Update Attention. The final layer copies the newly computed tuple $(q_{i+1}, s_{i+1}, m_i, c_{i+1})$ to y_{i+1} via position-based attention, completing one simulation step.

After this layer, y_{i+1} contains all information required for the next iteration, establishing the inductive step of the simulation.

6 Trade-Off Between Context Length and Environment Access Frequency

In practical scenarios, interacting with the external environment often incurs significant computational or communication costs. Therefore, it is crucial to minimize the number of such interactions while maintaining the model's reasoning capability. In this section, we show that this can be achieved by extending the context length. Specifically, we have the following corollary. (The proof is in Appendix C.)

Corollary 1. *Consider the same model as in Theorem 1. If we extend the context length to $4l + |Q| \times |\mathcal{A}| + 2$, the model can perform the Turing task while accessing the environment only once every l rounds of inference.*

7 Conclusion

In this paper, we provided a theoretical characterization of LLM-based agents, bridging the gap between their empirical success and formal understanding. We showed that, even with finite model capacity and finite context length, such agents can be Turing complete, enabling the simulation of arbitrary computational processes with bounded resources. We further analyzed the trade-off between context length and the frequency of environment interactions, revealing how effective context management can reduce communication costs while preserving reasoning performance. Together, these results offer a theoretical explanation for the expressive power and efficiency of LLM-based agents.

Limitations

Our results are established in the setting of Turing machine simulation. We do not consider more complex tasks or scenarios that more closely resemble real-world applications, which we leave for future work.

References

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. 2024a. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *Advances in neural information processing systems*, 36.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024b. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers. *arXiv preprint arXiv:2212.10559*.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. 2025. Context length alone hurts llm performance despite perfect retrieval. *arXiv preprint arXiv:2510.05381*.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. 2023. Towards revealing the mystery behind chain of thought: a theoretical perspective. *Advances in Neural Information Processing Systems*, 36:70757–70798.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, and 1 others. 2025. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. 2022. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*.

Tianyu Guo, Wei Hu, Song Mei, Huan Wang, Caiming Xiong, Silvio Savarese, and Yu Bai. 2023. How do transformers learn in-context beyond simple functions? a case study on learning with representations. *arXiv preprint arXiv:2310.10616*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Yifan Hao, Chenlu Ye, Chi Han, and Tong Zhang. 2025. Transformers as multi-task learners: Decoupling features in hidden markov models. *arXiv preprint arXiv:2506.01919*.

Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, and 1 others. 2025. Deep research agents: A systematic examination and roadmap. *arXiv preprint arXiv:2506.18096*.

Hui Jiang. 2023. A latent space theory for emergent abilities in large language models. *arXiv preprint arXiv:2304.09960*.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 737–740. ACM.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025a. Memory os of ai agent. *Preprint*, arXiv:2506.06326.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. 2025b. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1.

Licong Lin, Yu Bai, and Song Mei. 2023. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. *arXiv preprint arXiv:2310.08566*.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. 2022. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K Qiu, and Yuqing Yang. 2025. Agent lightning: Train any ai agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. 2023. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*.

Jacob Menick, Kevin Lu, Shengjia Zhao, E Wallace, H Ren, H Hu, N Stathas, and F Petroski Such. 2024. Gpt-4o mini: advancing cost-efficient intelligence. *Open AI: San Francisco, CA, USA*.

William Merrill and Ashish Sabharwal. 2023. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*.

Eshaan Nichani, Alex Damian, and Jason D Lee. 2024. How transformers learn causal structure with gradient descent. *arXiv preprint arXiv:2402.14735*.

Jorge Pérez, Pablo Barceló, and Javier Marinkovic. 2021. Attention is turing-complete. *Journal of Machine Learning Research*, 22(75):1–35.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Jiahao Qiu, Jingzhe Shi, Xinzhe Juan, Zelin Zhao, Jiayi Geng, Shilong Liu, Hongru Wang, Sanfeng Wu, and Mengdi Wang. 2025. Physics supernova: Ai agent matches elite gold medalists at ipho 2025. *arXiv preprint arXiv:2509.01659*.

Michael E Sander, Raja Giryes, Taiji Suzuki, Mathieu Blondel, and Gabriel Peyré. 2024. How do transformers perform in-context autoregressive learning? *arXiv preprint arXiv:2402.05787*.

Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, and 1 others. 2025. rstar2-agent: Agentic reasoning technical report. *arXiv preprint arXiv:2508.20722*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Tongyi DeepResearch Team. 2025. Tongyi deepresearch: A new era of open-source ai researchers. https://github.com/Alibaba-NLP/DeepResearch.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. 2025. Hilbert: Recursively building formal proofs with informal reasoning. *arXiv preprint arXiv:2509.22819*.

Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2023. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Xinyi Wang, Wanrong Zhu, and William Yang Wang. 2023. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, page 3.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Dennis Wu, Yihan He, Yuan Cao, Jianqing Fan, and Han Liu. 2025. Transformers and their roles as time series foundation models. *arXiv preprint arXiv:2502.03383*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Hongkang Yang, Lin Zehao, Wang Wenjin, Hao Wu, Li Zhiyu, Bo Tang, Wei Wenqiang, Jinbo Wang, Tang Zeyun, Shichao Song, Chenyang Xi, Yu Yu, Chen Kai, Feiyu Xiong, Linpeng Tang, and E Weinan. 2024. Memory^3: Language modeling with explicit memory. *Journal of Machine Learning*, 3(3):300–346.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jiarui Yao, Ruida Wang, and Tong Zhang. 2025. Fans–formal answer selection for natural language math reasoning using lean4. *arXiv preprint arXiv:2503.03238*.

Siyu Yuan, Zehui Chen, Zhiheng Xi, Junjie Ye, Zhengyin Du, and Jiecao Chen. 2025. Agent-r: Training language model agents to reflect via iterative self-training. *arXiv preprint arXiv:2501.11425*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. 2025. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*.

A Technique Lemmas

Lemma 1. *Let $y, z \in [0,1]^n$ be positive vectors, and let $b \in \{0,1\}$. There exists a two-layer feed-forward network $f : \mathbb{R}^{2n+1} \to \mathbb{R}^n$ such that*

$$f([y,z,b]) = \begin{cases} y, & b = 0, \\ z, & b = 1. \end{cases}$$

Proof. First, there is a linear mapping :

$$f_1([y,z,b]) = \begin{pmatrix} I_n & 0 & -1_n \\ 0 & I_n & 1_n \end{pmatrix} \begin{pmatrix} y \\ z \\ b \end{pmatrix} + \begin{pmatrix} 0 \\ -1_n \end{pmatrix} = \begin{pmatrix} y - b1_n \\ z + b1_n - 1_n \end{pmatrix}.$$

As $\{y, z\}$ are values within $[0,1]$ on each dimension, with ReLU activation function, we have

$$\sigma(f_1([y,z,b])) = \begin{cases} [y,0]^T, & b = 0, \\ [0,z]^T, & b = 1. \end{cases}$$

Then we consider the linear mapping on the second layer:

$$f_2 \circ \sigma(f_1([y,z,b])) = \begin{pmatrix} I_n & I_n \end{pmatrix} \cdot \sigma(f_1([y,z,b])) = \begin{cases} y, & b = 0, \\ z, & b = 1. \end{cases}$$

Then we can finish the proof. □

B Proofs of Theorem 1

B.1 Overview of Transformer Construction

The key point to prove Theorem 1 is the construction of 4-layer Transformer. which acts as the "brain" of the agent. Here we introduce the layer-wise constructions as follows:

1-st Attention Layer: On the first layer, we design a one-head Attention with query-key matrix as:

$$QK^{(1)} \in \mathbb{R}^{d \times d} := \begin{bmatrix} 0_{(|Q|+|\Sigma|+3)\times(|Q|+|\Sigma|+3)} & 0 & 0 \\ 0 & \gamma_1 I_2 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and the value matrix $V^{(1)} \in \mathbb{R}^{d \times d}$ as Figure 1.

Consider using Mask function on the elements except $[\bar{y}_1, \bar{y}_2]$, and sending $\gamma_1 \to +\infty$, we will obtain the output as

$$\left[\mathcal{A}ttn(S_i, Q^{(1)}, K^{(1)}, V^{(1)})\right]_{(t,\cdot)} = [0^T_{2|Q|+2|\Sigma|+7}, \alpha_i^T, \beta_i^T, 0^T_{d-2|Q|-3|\Sigma|-9}] \in \mathbb{R}^d,$$

for any $t = 1, \cdots, 6$, where we denote

$$\alpha_i = \begin{cases} s_{i+1}, & i \le n-1, \\ s_n, & i \ge n. \end{cases}$$

$$\beta_i = \begin{cases} [\cos(i+1/T), \sin(i+1/T)], & i \le n-1, \\ [\cos(n/T), \sin(n/T)], & i \ge n. \end{cases}$$

So the 5-th row of output for the first layer should be

$$\begin{aligned}[H^{(1)}]_{(5,\cdot)} &:= [S_i + \mathcal{A}ttn(S_i, Q^{(1)}, K^{(1)}, V^{(1)})]_{(5,\cdot)} \\ &= [q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, \\ &\quad 0^T_{|Q|+|\Sigma|+1}, \alpha_i^T, \beta_i^T, 0^T_{d-2|Q|-3|\Sigma|-9}].\end{aligned}$$

2-nd Attention Layer: On the second layer, we consider a one-head Attention architecture to simulate the transition function $\delta(\cdot)$ in-context. To be more specific, the QK matrix is designed as

$$QK^{(2)} := \begin{bmatrix} \gamma_2 I_{|Q|+|\Sigma|} & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{d \times d},$$

and the value matrix is designed as

$$V^{(2)} \in \mathbb{R}^{d \times d} := \begin{bmatrix} 0_{(|Q|+|\Sigma|+6)\times(|Q|+|\Sigma|+6)} & 0 & 0 \\ 0 & I_{|Q|+|\Sigma|+1} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Sending $\gamma_2 \to +\infty$, on the 6-th row, there will be a non-zero score only on the element $\widetilde{y}$, i.e, we use $QK^{(2)}$ to find the elements with the same feature as $[q_i, s_i]$. Use $\widetilde{y}$ to take in-context learning, the 5-th row of the output for the second layer should be

$$\begin{aligned}&[H^{(2)}]_{(5,\cdot)} \\ &:= [H^{(1)} + \mathcal{A}ttn(H^{(1)}, Q^{(2)}, K^{(2)}, V^{(2)})]_{(5,\cdot)} \\ &= [q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, q_{i+1}^T, \\ &\quad v_i^T, m_i, \alpha_i^T, \beta_i^T, 0^T_{d-2|Q|-3|\Sigma|-9}].\end{aligned}$$

$$\begin{bmatrix} 0_{(2|Q|+2|\Sigma|+7)\times|Q|} & 0_{(2|Q|+2|\Sigma|+7)\times|\Sigma|} & 0_{(2|Q|+2|\Sigma|+7)\times 3} & 0_{(2|Q|+2|\Sigma|+7)\times 2} & 0_{(2|Q|+2|\Sigma|+7)\times(d-|Q|-|\Sigma|-5)} \\ 0_{|\Sigma|\times|Q|} & I_{|\Sigma|} & 0_{|\Sigma|\times 3} & 0_{|\Sigma|\times 2} & 0_{|\Sigma|\times(d-|Q|-|\Sigma|-5)} \\ 0_{2\times|Q|} & 0_{2\times|\Sigma|} & 0_{2\times 3} & I_2 & 0_{2\times(d-|Q|-|\Sigma|-5)} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1: Value matrix $V^{(1)}$.

ReLU neural network: Following the two Attention layers above, there will be a two-layer ReLU neural network to calculate c_{i+1} with the information $[c_i, m_i]$. With Equation (1), we can obtain

$$\begin{aligned} &c_{i+1}^T = \\ &[\cos(1/T)\cdot[c_i]_1 - m_i\cdot\sin(1/T)\cdot[c_i]_2, \\ &\cos(1/T)\cdot[c_i]_2 + m_i\cdot\sin(1/T)\cdot[c_i]_1]\,. \end{aligned}$$

The linear mapping on the first layer can be expressed as

$$\begin{aligned} &f^{(1)}(x) := \\ &\begin{bmatrix} 0_{4\times(|Q|+|\Sigma|+1)} & A^{(1)} & 0_{4\times(d-|Q|-|\Sigma|-3)} \\ 0_{1\times(|Q|+|\Sigma|+1)} & 0_{1\times 2} & B^{(1)} \end{bmatrix} x \\ &+ \begin{bmatrix} 0_{4\times 1} \\ 1 \end{bmatrix}, \quad \forall x\in\mathbb{R}^d, \end{aligned}$$

where $A^{(1)} \in \mathbb{R}^{4\times 2}$ and $B^{(1)} \in \mathbb{R}^{1\times(d-|Q|-|\Sigma|-3)}$ are defined as

$$A^{(1)} := \begin{bmatrix} \cos(1/T) & -\sin(1/T) \\ \sin(1/T) & \cos(1/T) \\ \cos(1/T) & \sin(1/T) \\ -\sin(1/T) & \cos(1/T) \end{bmatrix},$$

$$B^{(1)} := \begin{bmatrix} 0_{1\times(|Q|+|\Sigma|+1)} & \frac{1}{2} & 0_{1\times(d-2|Q|-2|\Sigma|-5)} \end{bmatrix}.$$

Taking y_i as the input of $f^{(1)}$, we will obtain that

$$f^{(1)}(y_i) = \begin{pmatrix} \cos(\frac{1}{T})\cdot[c_i]_1 - \sin(\frac{1}{T})\cdot[c_i]_2 \\ \cos(\frac{1}{T})\cdot[c_i]_2 + \sin(\frac{1}{T})\cdot[c_i]_1 \\ \cos(\frac{1}{T})\cdot[c_i]_1 + \sin(\frac{1}{T})\cdot[c_i]_2 \\ \cos(\frac{1}{T})\cdot[c_i]_2 - \sin(\frac{1}{T})\cdot[c_i]_1 \\ \frac{1}{2}m_i + 1 \end{pmatrix}.$$

So the first four dimensions of $f^{(1)}(y_i)$ are values within $[0,1]$, and the last dimension is $\{0,1\}$. So we can use Lemma 1 to design $f^{(2)}$, which is a two-layer ReLU neural network. It will induce the final value of c_{i+1}, i.e,

$$c_{i+1} = \begin{cases} \begin{pmatrix} \cos(\frac{1}{T})\cdot[c_i]_1 - \sin(\frac{1}{T})\cdot[c_i]_2 \\ \cos(\frac{1}{T})\cdot[c_i]_2 + \sin(\frac{1}{T})\cdot[c_i]_1 \end{pmatrix}, & m_i = 1, \\ \begin{pmatrix} \cos(\frac{1}{T})\cdot[c_i]_1 + \sin(\frac{1}{T})\cdot[c_i]_2 \\ \cos(\frac{1}{T})\cdot[c_i]_2 - \sin(\frac{1}{T})\cdot[c_i]_1 \end{pmatrix}, & m_i = -1. \end{cases}$$

Then we construct a linear mapping as:

$$\begin{aligned} &f^{(3)}\circ f^{(2)}\circ f^{(1)}(y_i) \\ &:= \begin{bmatrix} 0_{(2|Q|+3|\Sigma|+9)\times 2} \\ I_2 \\ 0_{(d-2|Q|-3|\Sigma|-11)\times 2} \end{bmatrix} c_{i+1} \\ &= \begin{bmatrix} 0_{2|Q|+3|\Sigma|+9} \\ c_{i+1} \\ 0_{d-2|Q|-3|\Sigma|-11} \end{bmatrix}. \end{aligned}$$

Then the 5-th row of output on this ReLU neural network layer should be

$$\begin{aligned} &[H^{(3)}]_{(5,\cdot)} := [H^{(2)} + f^{(3)}\circ f^{(2)}\circ f^{(1)}(H^{(2)})]_{(5,\cdot)} \\ &= [q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, \\ &\quad q_{i+1}^T, v_i^T, m_i, \alpha_i^T, \beta_i^T, c_{i+1}, 0_{d-2|Q|-3|\Sigma|-11}^T]. \end{aligned}$$

3-rd Attention Layer: This Attention layer is used for obtaining the read symbol on the same position of c_{i+1}. Before construction, we define a notation as:

$$l(i) := \begin{cases} \arg\max_{j<i}\{c_j = c_i\}, & \{j<i, c_j = c_i\} \neq \emptyset, \\ i-1, & \{j<i, c_j = c_i\} \neq \emptyset. \end{cases}$$

To be specific, we need to obtain the information of $[l(i+1), v_{l(i+1)}]$, which would be used to infer the information of s_{i+1}. The QK matrix $QK^{(3)}$ is designed as Figure 2.

For any token y_j, we have

$$\begin{aligned} &y_i^T QK^{(3)} y_j \\ &= \gamma_3 \left(\underbrace{\lambda\cos\left(\frac{i-j}{T}\right)}_{\text{step position}} + \underbrace{[c_j]_1[c_{i+1}]_1 + [c_j]_2[c_{i+1}]_2}_{\text{tape position}} \right). \end{aligned}$$

$$\begin{bmatrix} 0_{(|Q|+|\Sigma|+1)\times(|Q|+|\Sigma|+1)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0_{2\times 2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda\gamma_3 I_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0_{(|Q|+2|\Sigma|+4)\times(|Q|+|\Sigma|+4)} & 0 & 0 \\ 0 & \gamma_3 I_2 & 0 & 0 & 0_{2\times 2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0_{(d-2|Q|-3|\Sigma|-11)\times(d-2|Q|-3|\Sigma|-11)} \end{bmatrix}$$

Figure 2: QK matrix $QK^{(3)}$.

Here we consider $0 < \lambda < 1 - \cos(1/T)$, and send $\gamma_3 \to \infty$, the Attention will only give non-zero score on the token with tape position $l(i+1)$. Then with value matrix $V^{(3)} \in \mathbb{R}^{d\times d}$ as Figure 3, the 5-th row of the output should be

$$\begin{aligned} &[H^{(4)}]_{(5,\cdot)} \\ &:= [H^{(3)} + \mathcal{A}ttn(H^{(3)}, Q^{(3)}, K^{(3)}, V^{(3)})]_{(5,\cdot)} \\ &= [q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, q_{i+1}^T, v_i^T, \\ &\quad m_i, \alpha_i^T, \beta_i^T, c_{i+1}^T, l(i+1)^T, v_{l(i+1)}^T, 0_{d-2|Q|-4|\Sigma|-13}^T]. \end{aligned}$$

ReLU neural network: After obtaining the information on $[l(i+1), v_{l(i+1)}]$, we need to calculate the value of s_{i+1}, which follows that:

$$s_{i+1} := \begin{cases} \alpha_i, & i+1 \le n, \\ v_{l(i+1)}, & i+1 > n, \quad l(i+1) \le i-1, \\ \emptyset, & i+1 > n, \quad l(i+1) = i. \end{cases}$$

In the detailed construction, first we use a linear mapping to obtain the corresponding information:

$$f^{(4)}(y_i) := A^{(2)} y_i + \begin{bmatrix} 0_{|\Sigma|-1} \\ 1 \\ 0_{2|\Sigma|+4}, \end{bmatrix}$$

where $A^2 \in \mathbb{R}^{(3|\Sigma|+6)\times d}$ is expressed as Figure 4.

So we have

$$f^{(4)}(y_i) = \begin{pmatrix} \cos(i/T) \\ \sin(i/T) \\ 1(\emptyset) \\ \alpha_i \\ \beta_i \\ l(i+1) \\ v_{l(i+1)} \end{pmatrix}.$$

Then we aim to generate two indicators $\{b_1, b_2\}$:

$$b_1 := \begin{cases} 1, & i+1 \le n, \\ 0, & i+1 > n, \end{cases}$$

$$b_2 := \begin{cases} 1, & l(i+1) \le i-1, \\ 0, & l(i+1) = i, \end{cases}$$

For b_1, we can use the normalized ReLU activation function as:

$$b_1 = \mathcal{N}orm\left(ReLU\left([\beta_i]_1 - \cos(\frac{n+1}{T})\right)\right),$$

and for b_2, we can consider

$$b_2 = \mathcal{N}orm\left(ReLU\left([l(i+1)]_1 - \cos(i/T)\right)\right).$$

So with a two-head two-layer normalized ReLU activation function and a linear mapping, we have

$$f^{(5)} \circ f^{(4)}(y_i) = \begin{pmatrix} b_1 \\ b_2 \\ 1(\emptyset) \\ \alpha_i \\ v_{l(i+1)} \end{pmatrix}.$$

Then as all dimensions of $f^{(5)} \circ f^{(4)}(y_i)$ are $\{0,1\}$ value, we can consider use Lemma 1 twice to obtain s_{i+1}. To be specific, with the first two-layer RELU neural network, we have

$$f^{(6)} \circ f^{(5)} \circ f^{(4)}(y_i) = \begin{cases} [b_1, \alpha_i^T, v_{l(i+1)}^T]^T, & b_2 = 1, \\ [b_1, \alpha_i^T, 1(\emptyset)^T]^T, & b_2 = 0. \end{cases}$$

Then we consider to use Lemma 1 again, then there will be

$$f^{(7)} \circ f^{(6)} \circ f^{(5)} \circ f^{(4)}(y_i) = \begin{cases} \alpha_i, & b_1 = 1, \\ v_{l(i+1)}, & b_1 = 0, \quad b_2 = 1, \\ 1(\emptyset)^, & b_1 = 0, \quad b_2 = 0. \end{cases}$$

Finally, consider a linear mapping $f^{(8)}$, we will have

$$\begin{aligned} &f^{(8)} \circ f^{(7)} \circ f^{(6)} \circ f^{(5)} \circ f^{(4)}(y_i) \\ &= [0_{2|Q|+4|\Sigma|+13}^T, s_{i+1}^T, 0_{d-2|Q|-5|\Sigma|-13}^T]^T \in \mathbb{R}^d. \end{aligned}$$

So the 5-th row of output on this ReLU neural

$$\begin{bmatrix} 0_{(|Q|+|\Sigma|+1)\times(|Q|+|\Sigma|+1)} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0_{2\times 2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0_{(|Q|+3)\times(|Q|+3)} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0_{|\Sigma|\times|\Sigma|} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0_{(|\Sigma|+5)\times(|\Sigma|+5)} & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{|\Sigma|} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 3: Value matrix $V^{(3)}$.

$$\begin{bmatrix} 0_{2\times(|Q|+|\Sigma|+3)} & I_2 & 0_{2\times(|Q|+|\Sigma|+2)} & 0_{2\times(2+|\Sigma|)} & 0_{2\times 2} & 0_{2\times(2+|\Sigma|)} & 0_{2\times(d-2|Q|-4|\Sigma|-13)} \\ 0_{|\Sigma|\times(|Q|+|\Sigma|+3)} & 0_{|\Sigma|\times 2} & 0_{|\Sigma|\times(|Q|+|\Sigma|+2)} & 0_{|\Sigma|\times(2+|\Sigma|)} & 0_{|\Sigma|\times 2} & 0_{|\Sigma|\times(2+|\Sigma|)} & 0_{|\Sigma|\times(d-2|Q|-4|\Sigma|-13)} \\ 0_{(2+|\Sigma|)\times(|Q|+|\Sigma|+3)} & 0 & 0 & I_{2+|\Sigma|} & 0 & 0 & 0 \\ 0_{(2+|\Sigma|)\times(|Q|+|\Sigma|+3)} & 0 & 0 & 0 & 0 & I_{2+|\Sigma|} & 0 \end{bmatrix}$$

Figure 4: Matrix $A^{(2)}$.

network layer should be

$$\begin{aligned} &[H^{(5)}]_{(5,\cdot)} \\ &:= [H^{(4)} + f^{(8)} \circ f^{(7)} \circ f^{(6)} \circ f^{(5)} \circ f^{(4)}(H^{(4)})]_{(5,\cdot)} \\ &= [q_i^T, s_i^T, m_{i-1}, c_i^T, \cos(i/T), \sin(i/T), 1, q_{i+1}^T, v_i^T, \\ &\quad m_i, \alpha_i^T, \beta_i^T, c_{i+1}^T, l(i+1)^T, v_{l(i+1)}^T, \\ &\quad s_{i+1}, 0_{d-2|Q|-5|\Sigma|-13}^T]. \end{aligned}$$

4-th Attention Layer: On this layer, we will update y_{i+1} based on the information on y_i. Here we consider the QK matrix $QK^{(4)} \in \mathbb{R}^{d\times d}$ as Figure 5, and use Mask function such that for any $j_1 \le j_2$, we have

$$\mathcal{M}ask\left([H^{(5)}QK^{(4)}(H^{(5)})^T]_{(j_1,j_2)}\right) = -\infty.$$

And the value matrix $V^{(4)}$ is as Figure 6.

Sending $\gamma_4 \to +\infty$, the output on the last row should be

$$\begin{aligned} &[H^{(6)}]_{(6,\cdot)} \\ &:= [H^{(5)} + \mathcal{A}ttn(H^{(4)}, Q^{(4)}, K^{(4)}, V^{(4)})]_{(6,\cdot)} \\ &= [q_{i+1}^T, s_{i+1}^T, m_i, c_{i+1}^T, \cos(i+1/T), \\ &\qquad \sin(i+1/T), 1, *, \cdots, *], \end{aligned}$$

where $*$ can be some unimportant values. We can read the information of y_{i+1} from the last row, then use it for next-step inference.

C Proofs of Corollary 1

The key idea in the construction is to filter and encode all information required for l inference rounds within a single access to the environment. The extended context must include the following components:

1. **Transition tokens:** Tokens containing all $|Q|\times|\mathcal{A}|$ possible inputs to the transition function $\delta(\cdot)$, ensuring that the model can infer the transition behavior in-context.

2. **Future-step tokens:** A set of tokens $\bar{y}_{i,1}, \ldots, \bar{y}_{i,l}$, defined similarly to Equation (2):

$$\bar{y}_{i,m} := \begin{cases} y_{i+m}, & i \le n-m, \\ y_n, & i > n-m, \end{cases} \quad \forall m \in [l],$$

   which allows the model to determine whether the $(i+m)$-th step remains within the first n time steps.

3. **Local history tokens:** The most recent history tokens corresponding to tape positions within $[c_i - l, c_i + l]$, enabling accurate updates of the reading symbols on the tape.

With these history tokens incorporated, the remaining procedures follow the same structure as in the proof of Theorem 1.

$$\begin{bmatrix} 0_{(|Q|+|\Sigma|+3)\times(|Q|+|\Sigma|+3)} & 0 & 0 \\ 0 & \gamma_4 I_2 & 0 \\ 0 & 0 & 0_{(d-|Q|-|\Sigma|-5)\times(d-|Q|-|\Sigma|-5)} \end{bmatrix}$$

Figure 5: QK matrix $QK^{(4)}$.

$$\begin{bmatrix}
0_{|Q|\times|Q|} & 0 & 0 & 0 & 0 & I_{|Q|} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0_{|\Sigma|\times|\Sigma|} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_{|\Sigma|} & 0 \\
0 & 0 & 0_{1\times1} & 0 & 0 & 0 & 0 & I_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0_{2\times2} & 0 & 0 & 0 & 0 & 0 & I_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0_{3\times3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0_{|Q|\times|Q|} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0_{|\Sigma|\times|\Sigma|} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0_{1\times1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0_{(|\Sigma|+2)\times(|\Sigma|+2)} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0_{2\times2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0_{(|\Sigma|+2)\times(|\Sigma|+2)} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0_{|\Sigma|\times|\Sigma|} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

Figure 6: value matrix $V^{(4)}$.