Proof note for Transformer with CoT Circuit Complexity

Abstract

1 Proof note for [MS23]

1.1 Turing Machine and Automaton

Definition 1.1 (Single-tape Turing machine).

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

where

- Q is the finite set of states
- Σ is the input alphabet
- Γ is the tape alphabet follows $\Sigma \subseteq \Gamma$ and blank symbol $\sqcup \in \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state where $q_{\text{accept}} \neq q_{\text{reject}}$

When a single-tape TM computes, we have one ***infinite tape***, one ***head*** which can read the symbol in the current cell, write the symbol to that cell, and move one step left or right. A configuration of the machine has the entire tape contents (finite nonblank region + blanks), the current state, and the current head position. The computation starts with input string $x := x_1 x_2 \ldots x_n \in \Sigma^*$ written in the first n cells, and other cells contain blank $\sqcup$. The ***head*** starts from x_1 and machine state is q_0.

Suppose the machine is in state q and the head is reading symbol a. Then it will look up the transition function $\delta(q, a) = (q', b, d)$ where q' is the next state, $b \in \Gamma$ is the symbol to write in the current cell, and $d \in \{L, R\}$ is the moving direction of the head. Then, this function indicates the head will overwrite the current cell with b, change the state to q' and move the head one cell left or right. The halting mechanism includes the following:

- If the machine enters state q_{accept}, then it accepts and ***halts***
- If the machine enters state q_{reject}, then it rejects and ***halts***
- if neither, it runs indefinitely on the input.

Definition 1.2 (Multi-tape Turing machine)**.** For $k \in \mathbb{Z}$. a k-tape turing machine is running k single-tape TMs from Definition 1.1 in parallel but with a shared state. The different is the transition function is defined as the following:

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

For each tape $i \in \{1, \dots, k\}$ and a single current state $q \in Q$. First, we write x in the first cells of tape $i = 1$, and rest of the tapes cells are $\sqcup$. Each head starts at the leftmost cell of its tape and initial state is q_0.

We suppose the current state is q, and the symbols under the heads are $(a_1, a_2, \dots, a_k) \in \Gamma^k$. Then we look up the transition function $\delta(q, a_1, a_2, \dots, a_k) = (q', b_1, b_2, \dots, b_k, d_1, d_2, \dots, d_k)$ where q' is the next state, each b_i is the symbol to write on tape i, and each $d_i \in \{L, R, S\}$ shows the movement of head i.

Both Single-tape and Multi-tape TMs compute the algorithms by encoding the current state of the algo, apply a fixed rule δ, and halt. The benefit of multi-tape is that we can utilize multiple tapes as stack, work array, intermidate results, scratch, etc.

Remark 1.1 (Multitape to Single Tape Equivalence, [Sip96, Thm 7.8]). Let $t(n)$ be a function, where $t(n) \geq n$. Then every $t(n)$ time multitape Turing machine has an equivalent $O(t^2(n))$ time single-tape Turing machine.

Conceptually, one multitape step can be simulated by up to $O(t(n))$ single-tape steps. Over $t(n)$ mutitape steps, the time is bounded by $O(t^2(n))$

1.2 $\mathsf{TIME}(t(n)) \subseteq \mathsf{CoT}(t(n))$

Theorem 1.1 (Transformer simulate TM, [MS23, Restatement of Thm 2])**.** *If we have the following conditions:*

- *M is a multitape TM from Definition 1.2*
- *input x length is $1 + n$*
- *at most of $t(n)$ steps*

Then, there exists that

- *a decoder only projected* ***pre-norm*** *transformer with strict causal saturated attention on input x*
- *it takes $t(n)$ decoding steps and $|M(x)|$ more steps to output $M(x)$.*

Proof With Explanations. The goal of this proof is to construct a transformer decoder that uses a single decoding step to simulate each Turning machine step. ***Explain: Instead of storing configs, they chose to store the diffs Δ and recover the contents using recursion.***

We want to show that for input tokens $x := x_1 x_2 \dots x_n$ and CoT tokens at positions $i \geq n$, we need to show that $x_i = \delta_{i-n}$.

Base Case: we need to show $i = n$ is true. For $i \in \{0, 1, \dots, n-1\}$. the first n input tokens. At position $i = n$, it starts the decoding step, by Definition 1.2, $\delta_0 = \langle q_0, b^{k+1}, 0^{k+2} \rangle$

Inductive Step: □

Corollary 1.2. $\mathsf{TIME}(t(n)) \subseteq \mathsf{CoT}(t(n))$

1.3 $\mathsf{CoT}(t(n)) \subseteq \mathsf{SPACE}(t(n) + \log n)$

Theorem 1.3. $\mathsf{CoT}(t(n)) \subseteq \mathsf{SPACE}(t(n) + \log n)$

1.4 $\mathsf{CoT}(t(n)) \subseteq \widehat{\mathsf{TIME}}(n^2 + t(n^2))$

Theorem 1.4. $\mathsf{CoT}(t(n)) \subseteq \widehat{\mathsf{TIME}}(n^2 + t(n^2))$

2 Proof Note for [LLZM24, Section C]

Problem 2.1 (DFA and Language)**.** How is language defined using DFA?

We define a 5-tuple DFA as $\mathcal{A} := (\Sigma, Q, \delta, q_0, F)$ as the machine. We let $\mathcal{L}(\mathcal{A})$ denote the set of strings the machine accepts, which we call it language.

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \widehat{\delta}(q_0, w) \in F\}$$

where $w \in \Sigma^*$ is a string, $\widehat{\delta} : Q \times \Sigma^* \to Q$, F is the accepting states.

Example 2.1. We present some simple examples as follows.

- We define $\mathcal{A}(\{a\}, \{s\}, \delta(s, a) = s, s, \emptyset)$, then the language it can represent is $\mathcal{L}(\mathcal{A}) = \varnothing$ which means it rejects everything.
- We define $\mathcal{A}(\{a\}, \{s\}, \delta(s, a) = s, s, s)$, then the language it can represent is $\mathcal{L}(\mathcal{A}) = \{a\}^* = \{\epsilon, a, aa, aaa, \ldots\}$, which means it accepts all 'a' strings.

Now, we introduce non-deterministic finite automaton.

Definition 2.1 (ϵ-NFA)**.** We define an ϵ-NFA as follows.

$$\mathcal{N} = (Q, \Sigma, \overline{\delta}, q_0, F),$$

where $\overline{\delta} : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ returns a set of next states. Note that ϵ-closure of $S \subseteq Q$ is defined as $\mathsf{EClose}(S)$ is the smallest $T \supseteq S$ such that if $q \in T$ and $p \in \overline{\delta}(q, \epsilon)$, then $p \in T$.

We provide an example to ϵ-NFA.

Example 2.2 (ϵ-NFA Example)**.** We give an example of NFA that recognize language $\{ab\} \cup \{ba\}$ as the following. We define the set $Q = \{q_0, q_1, q_2, q_3, q_4, f\}$, $\Sigma = \{a, b\}$, $F = \{f\}$. Then, we define the tranistion function $\overline{\delta}$ as follows.

$$\begin{aligned}
&\overline{\delta}(q_0, \epsilon) = \{q_1, q_3\} \\
&\overline{\delta}(q_1, a) = \{q_2\}, \quad \overline{\delta}(q_2, b) = \{f\} \\
&\overline{\delta}(q_3, b) = \{q_4\}, \quad \overline{\delta}(q_4, a) = \{f\}
\end{aligned}$$

References

[LLZM24] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.

[MS23] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.