A $k\text{AExp}_{pol}$ problem for deterministic machines

Alessio Mansutti
Department of Computer Science, University of Oxford, Oxford, UK

Abstract

In these notes we give a direct proof that the $k\text{AExp}_{pol}$-prenex problem for multi-tapes deterministic Turing machines is complete for the class $k\text{AExp}_{pol}$, i.e. the class of all problems solvable by an alternating Turing machine running in k exponential time and performing a polynomial amount of alternations, with respect to the input size.

2012 ACM Subject Classification Theory of computation → Abstract machines

Keywords and phrases Elementary hierarchy, deterministic and alternating Turing machines

1 Alternation problems for deterministic Turing machines

In this section, we introduce the $k\text{AExp}_{pol}$*-prenex problem*, a decision problem for deterministic multi-tape Turing machines, and prove that this problem is $k\text{AExp}_{pol}$-complete (see below for the definition of this complexity class). The $k\text{AExp}_{pol}$-prenex problem is a straightforward generalisation of the *TM alternation problem* shown AExp_{pol}-complete in [2, page 292]. In fact, given the AExp_{pol}-completeness of the *TM alternation problem*, one can show that the $k\text{AExp}_{pol}$-prenex problem is $k\text{AExp}_{pol}$-complete by relying on standard padding arguments. However, here we prefer giving a direct proof of this result, which does not rely on prior knowledge on alternating Turing machines.

Notation for regular languages. Let A and B be two regular languages. As usual, we write $A \cup B$ and $A \setminus B$ for the set theoretical union and difference of languges, respectively. With $A \cdot B$ we denote the language obtained by concatenating words in A with words in B. Then, $A^0 = \{\epsilon\}$ and for all $n \in \mathbb{N}$, $A^{n+1} = A^n \cdot A$. Lastly, $A^* = \bigcup_{n\in\mathbb{N}} A^n$ and $A^+ \stackrel{\text{def}}{=} A^* \setminus A^0$. Given a finite *alphabet* Σ, the set Σ^n is the set of words over Σ of length $n \in \mathbb{N}$ and, given a finite word w, we write $|w|$ for its length.

Complexity classes. We recall some standard complexity classes that appear throughout these notes. Below and across the whole paper, given natural numbers $k, n \geq 1$, we write $\mathfrak{t}$ for the tetration function inductively defined as $\mathfrak{t}(0, n) \stackrel{\text{def}}{=} n$ and $\mathfrak{t}(k, n) = 2^{\mathfrak{t}(k-1,n)}$. Intuitively, $\mathfrak{t}(k, n)$ defines a tower of exponentials of height k.

- $k\text{NExp}$ is the class of all problems decidable with a non-deterministic Turing machine running in time $\mathfrak{t}(k, f(n))$ for some polynomial $f : \mathbb{N} \to \mathbb{N}$, on each input of length n.
- $\Sigma_j^{k\text{Exp}}$ is the class of all problems decidable with an alternating Turing machine (ATM, [1]) in time $\mathfrak{t}(k, f(n))$, starting on an existential state and performing at most $j-1$ alternations, for some polynomial $f : \mathbb{N} \to \mathbb{N}$, on each input of length n. By definition, $\Sigma_1^{k\text{Exp}} = k\text{NExp}$.
- $k\text{AExp}_{pol}$ is the class of all problems decidable with an ATM running in time $\mathfrak{t}(k, f(n))$ and performing at most $g(n)$ number of alternations, for some polynomials f, g, on each input of length n. The inclusion $\Sigma_j^{k\text{Exp}} \subseteq k\text{AExp}_{pol}$ holds for every $j \in \mathbb{N}_+$.

Deterministic n-tapes TM. Let $n \in \mathbb{N}_+$. Following the presentation in [2], we define the class of *deterministic n-tapes Turing machines* (nTM). A nTM is a deterministic machine $\mathcal{T} = (n, \Sigma, S, s_0, s_\top, s_\bot, \delta)$, where Σ is a finite alphabet containing a *blank symbol* $\#$, and S is a finite set of *states* including the initial state s_0, the accepting state $s_\top$ and the rejecting state $s_\bot$. The nTM operates on n distinct *tapes* numbered from 1 to n, and it has one

read/write head shared by all tapes. Every *cell* of each tape contains a symbol from Σ and is indexed with a number from $\mathbb{N}$ (i.e. the tapes are infinite only in one direction). Lastly, the deterministic transition function $\delta\colon S \times \Sigma \to (S \times \Sigma \times \{-1,+1\}) \cup (S \times [1,n])$ is such that for every $a \in \Sigma$, $\delta(s_\top, a) \stackrel{\text{def}}{=} (s_\top, 1)$ and $\delta(s_\bot, a) \stackrel{\text{def}}{=} (s_\bot, 1)$.

A configuration of $\mathcal{T}$ is given by the content of the n tapes together with a *positional state* $(s,j,k) \in S \times [1,n] \times \mathbb{N}$. The content of the tape is represented by an n-uple of finite words $(w_1,\dots,w_n) \in (\Sigma^*)^n$, where each w_i specifies the content of the first $|w_i|$ cells of the i-th tape, after which the tape only contains the blank symbols #. The positional state (s,j,k) describes the current state s of the machine together with the position (j,k) of the read/write head, placed on the k-th cell of the j-th tape. At each step, given the positional state (s,j,k) of $\mathcal{T}$ and the symbol $a \in \Sigma$ read by the read/write head in position (j,k), one of the following occurs:

case $\delta(s,a) = (s',b,i)$*, where* $s \in S$*,* $b \in \Sigma$ *and* $i \in \{-1,1\}$ *(ordinary moves)*: If $k+i \in \mathbb{N}$, then $\mathcal{T}$ overwrites the symbol a in position (j,k) with the symbol b. Afterwards $\mathcal{T}$ changes its positional state to $(s',j,k+i)$. Otherwise ($k+i \notin \mathbb{N}$, i.e. $k=0$ and $i=-1$), $\mathcal{T}$ does not modify the tapes and changes its positional state to $(s_\bot,1,0)$.

case $\delta(s,a) = (s',j')$*, where* $s' \in S$ *and* $j' \in [1,n]$ *(jump moves)*: The read/write head moves to the k-th cell of the j'-th tape, i.e. $\mathcal{T}$ changes its positional state to (s',j',k). Jump moves do not modify the content of the n tapes.

Initially, each tape contains a word in Σ^* written from left to right starting from the position 0 of the tape. Hence, an input of $\mathcal{T}$ can be described as a tuple $(w_1,\dots,w_n) \in (\Sigma^*)^n$ where, for all $j \in [1,n]$, w_j is the input word for the j-th tape. We write $\mathcal{T}(w_1,\dots,w_n)$ for the run of the nTM on input $(w_1,\dots,w_n) \in (\Sigma^*)^n$, starting from the positional state $(s_0,1,0)$. The run $\mathcal{T}(w_1,\dots,w_n)$ *accepts* in time $t \in \mathbb{N}$ if $\mathcal{T}$ reaches the accepting state $s_\top$ in at most t steps.

▶ **Definition 1** (kAExp$_{pol}$-prenex TM problem). *Fix* $k \in \mathbb{N}_+$. **Input:** $(n, \mathbf{Q}, \mathcal{T})$*, where* $n \in \mathbb{N}_+$ *is written in unary,* $\mathbf{Q} = (Q_1,\dots,Q_n) \in \{\exists,\forall\}^n$ *with* $Q_1 = \exists$*, and a* $\mathcal{T}$ *is a* nTM *working on alphabet* Σ. **Question:** *is it true that*

$$Q_1 w_1 \in \Sigma^{\mathfrak{t}(k,n)}, \dots, Q_n w_n \in \Sigma^{\mathfrak{t}(k,n)} : \mathcal{T}(w_1,\dots,w_n) \text{ accepts in time } \mathfrak{t}(k,n) \ ?$$

We analyse the complexity of the kAExp$_{pol}$-prenex TM problem depending on the type of *quantifier prefix* $\mathbf{Q}$. For arbitrary quantifier prefixes, the problem is kAExp$_{pol}$-complete. The membership in kAExp$_{pol}$ is straightforward, whereas the hardness follows by reduction from the acceptance problem for alternating Turing machines running in k-exponential time and performing a polynomial number of alternations, as we show in Sec. 2.

▶ **Theorem 2.** *The* kAExp$_{pol}$*-prenex* TM *problem is* kAExp$_{pol}$*-complete.*

For bounded alternation, let $\mathrm{alt}(\mathbf{Q})$ be the number of alternations between existential and universal quantifiers in $\mathbf{Q} \in \{\exists,\forall\}^n$, plus one. That is, $\mathrm{alt}(\mathbf{Q}) \stackrel{\text{def}}{=} 1 + |\{i \in [2,n] : Q_i \neq Q_{i-1}\}|$. For $j \in \mathbb{N}_+$, the $\Sigma_j^{k\text{Exp}}$-prenex TM problem is the kAExp$_{pol}$-prenex TM problem restricted to instances $(n, \mathbf{Q}, \mathcal{T})$ with $\mathrm{alt}(\mathbf{Q}) = j$. Refining Thm. 2, this problem is $\Sigma_j^{k\text{Exp}}$-complete.

▶ **Theorem 3.** *The* $\Sigma_j^{k\text{Exp}}$*-prenex* TM *problem is* $\Sigma_j^{k\text{Exp}}$*-complete.*

Thm. 3 implies that the problem is kNExp-complete on instances where $\mathbf{Q} \in \{\exists\}^n$.

Some properties of $\mathfrak{t}(k,n)$. Before moving to the proofs of Thm. 2 and Thm. 3, we discuss some trivial properties of the tetration function $\mathfrak{t}$, which we later need. In Lemmas 4–6 below, let $k, n \in \mathbb{N}_+$.

▶ **Lemma 4.** *$p(\mathfrak{t}(k,n)) \leq \mathfrak{t}(k, p(n))$, for every polynomial $p(\mathtt{x}) = \alpha\mathtt{x}^d + \beta$ with $\alpha, d, \beta \in \mathbb{N}_+$.*

Proof. Straightforward induction on $k \in \mathbb{N}_+$:
base case: $k = 1$. Trivially, $\alpha(2^n)^d + \beta \leq 2^{\alpha n^d + \beta}$.
induction step: $k > 1$. By induction hypothesis, $\alpha\mathfrak{t}(k-1,n)^d + \beta \leq \mathfrak{t}(k-1, \alpha n^d + \beta)$, so,
$\alpha\mathfrak{t}(k,n)^d + \beta = \alpha(2^{\mathfrak{t}(k-1,n)})^d + \beta \leq 2^{\alpha\mathfrak{t}(k-1,n)^d + \beta} \leq 2^{\mathfrak{t}(k-1,\alpha n^d+\beta)} = \mathfrak{t}(k, \alpha n^d + \beta)$. ◀

▶ **Lemma 5.** *$\mathfrak{t}(k,n)^2 \leq \mathfrak{t}(k, 2n)$.*

Proof. For $k = 1$, we have $\mathfrak{t}(1,n) = (2^n)^2 = 2^{2n} = \mathfrak{t}(1,2n)$. For $k > 1$, by Lemma 4 $2\mathfrak{t}(k-1,n) \leq \mathfrak{t}(k-1,2n)$. So, $\mathfrak{t}(k,n)^2 = (2^{\mathfrak{t}(k-1,n)})^2 = 2^{2\mathfrak{t}(k-1,n)} \leq 2^{\mathfrak{t}(k-1,2n)} = \mathfrak{t}(k,2n)$. ◀

We inductively define the function $\overline{\log_2}^k : \mathbb{N} \to \mathbb{N}$ as follows:

$$\overline{\log_2}^1(\mathtt{x}) \stackrel{\text{def}}{=} \lceil \log_2(\mathtt{x}) \rceil \quad \text{and} \quad \overline{\log_2}^{k+1}(\mathtt{x}) \stackrel{\text{def}}{=} \overline{\log_2}^k(\lceil \log_2(\mathtt{x}) \rceil).$$

▶ **Lemma 6.** *For every $m \in \mathbb{N}$, $m > \mathfrak{t}(k,n)$ if and only if $\overline{\log_2}^k(m) > n$.*

Proof. Straightforward induction on $k \in \mathbb{N}_+$:
base case: $k = 1$. ($\Rightarrow$): Suppose $m > 2^n$. Then, $\overline{\log_2}^1(m) = \lceil \log_2(m) \rceil \geq \log_2(m) > n$.
($\Leftarrow$): Conversely, suppose $m \leq 2^n$. Then $\log_2(m) \leq n$. As $n \in \mathbb{N}$, $\lceil \log_2(m) \rceil \leq n$.
induction step: $k > 1$. ($\Rightarrow$): Suppose $m > \mathfrak{t}(k,n)$. So, $\lceil \log_2(m) \rceil \geq \log_2(m) > \mathfrak{t}(k-1,n)$. By induction hypothesis, $\overline{\log_2}^{k-1}(\lceil \log_2(m) \rceil) > n$, i.e. $\overline{\log_2}^k(m) > n$.
($\Leftarrow$): Conversely, suppose $m \leq \mathfrak{t}(k,n)$. Therefore, $\log_2(m) \leq \mathfrak{t}(k-1,n)$. Since $n \in \mathbb{N}$, $\lceil \log_2(m) \rceil \leq \mathfrak{t}(k-1,n)$. By induction hypothesis, $\overline{\log_2}^k(m) = \overline{\log_2}^{k-1}(\lceil \log_2(m) \rceil) \leq n$. ◀

Lemma 6 allows us to check whether $m > \mathfrak{t}(k,n)$ holds in time $\mathcal{O}(k \cdot m + n)$, by first computing $r = \overline{\log_2}^k(m)$ and then testing whether $r > n$. Fundamentally, in this way we avoid computing $\mathfrak{t}(k,n)$ explicitly. This fact is later used in Lemma 9, where we introduce a nTM that shall test whether its inputs $w_1, \ldots, w_n$ are greater than $\mathfrak{t}(k,n)$, while running in polynomial time in $|w_1|, \ldots, |w_n|$, k and n. For similar reasons, this machine also requires to compute the *integer square root*, defined as $\underline{\sqrt{\mathtt{x}}} \stackrel{\text{def}}{=} \lfloor \sqrt{\mathtt{x}} \rfloor$.

▶ **Lemma 7.** *Given $n, m \in \mathbb{N}$, $m \geq n^2$ if and only if $\underline{\sqrt{m}} \geq n$.*

Proof. ($\Rightarrow$): Suppose $m \geq n^2$. Then, $\sqrt{m} \geq n$. Since $n \in \mathbb{N}$, we conclude $\lfloor \sqrt{m} \rfloor \geq n$.
($\Leftarrow$): Suppose $\underline{\sqrt{m}} \geq n$. Trivially, $\sqrt{m} \geq n$ and so $m \geq n^2$. ◀

2 Alternating Turing machines and proof of Thm. 2

This section contains the auxiliary technical tools needed to prove Thm. 2, as well as the proof of this theorem and of Thm. 3. The section does not assume any prior knowledge on alternating Turing machines (ATM), which we define adapting the presentation of [1].

Alternating Turing machines. An ATM $\mathcal{M} = (\Sigma, S_\exists, S_\forall, s_0, s_\top, s_\bot, \delta)$ is a single-tape machine where Σ is a finite *alphabet* containing at least two symbols, one of which is the *blank symbol* #, the sets $S_\exists$ and $S_\forall$ are two disjoint sets of *states*, respectively called *existential* and *universal* states, $s_0 \in S_\exists$ is the *initial state*, and $s_\top$ and $s_\bot$ are two auxiliary states that do not belong to $S_\exists \cup S_\forall$. The state $s_\top$ is the *accepting state* and the state $s_\bot$ is the *rejecting state*. Every *cell* of the tape contains a symbol from Σ and is indexed with a number from $\mathbb{N}$ (i.e. the tapes are infinite only in one direction). Let $S \stackrel{\text{def}}{=} S_\exists \cup S_\forall \cup \{s_\top, s_\bot\}$. The *transition function* is a function of the form $\delta\colon S \times \Sigma \to 2^{(S\times\Sigma\times\{-1,+1\})}$, where for every $a \in \Sigma$, $\delta(s_\top, a) = \{(s_\top, a, +1)\}$, $\delta(s_\bot, a) = \{(s_\bot, a, -1)\}$, and for every $s \in S_\forall \cup S_\exists$, $\delta(s, a) \neq \varnothing$. We write $|\mathcal{M}|$ for the size of $\mathcal{M}$, i.e. the number of symbols needed in order to describe $\mathcal{M}$.

A *configuration* of $\mathcal{M}$ is given by a triple (w, s, k). The finite word $w \in \Sigma^*(\Sigma \setminus \{\#\})$ specifies the content of the first $|w|$ cells fo the tape, after which the tape only contains the blank symbol #. Notice that w does not end with #, which means that distinct words do not describe the same content of the tape. The *positional state* (s, k) describes the current state s of the machine together with the position of the read/write head on the tape, here corresponding to the k-th cell.

Let $c = (w, s, k)$ be a configuration of $\mathcal{M}$ and consider the symbol $a \in \Sigma$ read by the read/write head. We write $\Delta(c)$ for the set of configurations reachable in exactly one step from c. In particular, $(w', s', k') \in \Delta(c)$ if and only if there is $(s'', b, i) \in \delta(s, a)$ such that

- $k + i \in \mathbb{N}$ (i.e. $k \neq 0$ or $i \neq -1$), $k' = k + i$, $s'' = s'$ and w' describes the tape after the read/write head modifies the content of the k-th cell to b, or
- $k + i \notin \mathbb{N}$, $k' = 0$, $w' = w$ and $s' = s_\bot$.

A *computation path* of $\mathcal{M}$ is a sequence $(c_0, \ldots, c_d)$ of configurations such that $c_i \in \Delta(c_{i-1})$ holds for every $i \in [1, d]$. If the states $s_\top$ and $s_\bot$ appear in the last configuration c_d, the path is called *terminating*. Notice that if $s_\top$ (or $s_\bot$) belongs to some c_i ($i \in [1, d]$), then by definition of δ it also belongs to every c_j with $j > i$, and thus the configuration is terminating.

To describe the notion of acceptance for $\mathcal{M}$, we introduce a *partial* labelling function $\gamma_\mathcal{M}\colon \Sigma^* \times S \times \mathbb{N} \to \{\texttt{acc}, \texttt{rej}\}$ (we drop the prefix $\mathcal{M}$ from $\gamma_\mathcal{M}$ when clear from the context). Given $w \in \Sigma^*$ and $k \in \mathbb{N}$, the function γ is defined as follows:

- $\gamma(w, s_\top, k) \stackrel{\text{def}}{=} \texttt{acc}$ and $\gamma(w, s_\bot, k) \stackrel{\text{def}}{=} \texttt{rej}$,
- for every $s \in S_\exists$,
  - $\gamma(w, s, k) = \texttt{acc}$ if $\gamma(w', s', k') = \texttt{acc}$ holds for some $(w', s', k') \in \Delta(w, s, k)$,
  - $\gamma(w, s, k) = \texttt{rej}$ if $\gamma(w', s', k') = \texttt{rej}$ holds for every $(w', s', k') \in \Delta(w, s, k)$,
  - otherwise, $\gamma(w, s, k)$ undefined,
- for every $s \in S_\forall$,
  - $\gamma(w, s, k) = \texttt{acc}$ if $\gamma(w', s', k') = \texttt{acc}$ holds for every $(w', s', k') \in \Delta(w, s, k)$,
  - $\gamma(w, s, k) = \texttt{rej}$ if $\gamma(w', s', k') = \texttt{rej}$ holds for some $(w', s', k') \in \Delta(w, s, k)$,
  - otherwise $\gamma(w, s, k)$ undefined.

The language described by $\mathcal{M}$ is $\mathcal{L}(\mathcal{M}) \stackrel{\text{def}}{=} \{w \in \Sigma^* \mid \gamma(w, s_0, 0) = \texttt{acc}\}$. For our purposes, we are only interested in the notions of time-bounded and alternation-bounded acceptance. We say that $\mathcal{M}$ *accepts* (resp. *rejects*) a word $w \in \Sigma^*$ in time $t \in \mathbb{N}$ whenever examining all the terminating computation paths $(c_0, \ldots, c_d)$, where $c_0 = (w, s_0, 0)$ and $d \in [0, t]$, is sufficient to conclude whether $\gamma(w, s_0, 0) = \texttt{acc}$ (resp. $\gamma(w, s_0, 0) = \texttt{rej}$). The machine $\mathcal{M}$ *halts* on the word w in time t if it either accepts or rejects w in time t.

Consider functions $f, g\colon \mathbb{N} \to \mathbb{N}$. The ATM $\mathcal{M}$ is f-time bounded and g-alternation bounded if and only if for every $w \in \Sigma^*$, $\mathcal{M}$ halts on w in time $f(|w|)$ and, along each terminating computation path $(c_0, \dots, c_d)$, where $c_0 = (w, s_0, 0)$ and $d \in [0, t]$, the positional state of the machine alternate between existential and universal states at most $g(|w|)$ times.

▶ **Proposition 8** ([1]). *Consider $k \in \mathbb{N}_+$, polynomials $f, g\colon \mathbb{N} \to \mathbb{N}$, and $h(\mathtt{x}) \stackrel{\text{def}}{=} \mathfrak{t}(k, f(\mathtt{x}))$. Let $\mathcal{M}$ be an h-time bounded and g-alternation bounded* ATM *on alphabet Σ. Let $w \in \Sigma^*$. The problem of deciding whether $w \in \mathcal{L}(\mathcal{M})$ is $k\mathrm{AExp}_{pol}$-complete.*

From ATM **to** nTM**.** Working towards a proof of Thm. 2, we now aim at defining a nTM that checks if its input words represent computation paths of an alternating Turing machine. Throughout this section, we fix $k \in \mathbb{N}_+$ (encoded in unary), two polynomials $f, g\colon \mathbb{N} \to \mathbb{N}$ and $h(\mathtt{x}) \stackrel{\text{def}}{=} \mathfrak{t}(k, f(\mathtt{x}))$. We consider a h-time bounded and g-alternation bounded ATM $\mathcal{M} = (\Sigma, S_\exists, S_\forall, s_0, s_\top, s_\bot, \delta)$. Let $w \in \Sigma^*$. Since $\mathcal{M}$ is h-time bounded, we can represent the configurations of $\mathcal{M}$ as words of the finite language $C = \{\widehat{w} \in \Sigma^* \cdot S \cdot \Sigma^+ \mid |\widehat{w}| = h(|w|)\}$. More precisely, we say that a word $\widehat{w} \in C$ *encodes the configuration* $(w', s, k) \in \Sigma^* \times S \times \mathbb{N}$ whenever there are words $w'' \in \Sigma^*$, $w''' \in \Sigma^+$ and $w_\# \in \{\#\}^*$. such that $w' \cdot w_\# = w'' \cdot w'''$, $|w''| = k$ and $\widehat{w} = w'' \cdot s \cdot w'''$. For instance, the configuration $(ab, s, 3)$ is encoded by the word $ab\#s\# \dots \#$ of length $h(|w|)$. Each word $\widetilde{w} \in C$ encodes exactly one configuration of $\mathcal{M}$, which we denote by $c(\widetilde{w})$.

We extend our encoding to computation paths of $\mathcal{M}$, with the aim at defining a nTM that checks if its input words represent a computation path of $\mathcal{M}$. To this end, we introduce a new symbol ¢ that does not appear in $\Sigma \cup S$ and that is used to delimit the end of a computation path. Let $\overline{\Sigma} = \Sigma \cup S \cup \{¢\}$ and consider a word $\overline{w} \in \overline{\Sigma}^*$. Again, since $\mathcal{M}$ is h-time bounded, its computation paths can be encoded by a sequence of at most $h(|w|)$ words from C (so, a word of length at most $h(|w|)^2$). Hence, let $\widetilde{w} \in (\Sigma \cup S)^*$ be the only prefix of the word $\overline{w} \cdot ¢$ such that either $\widetilde{w} \cdot ¢$ is a prefix of $\overline{w} \cdot ¢$ or $|\widetilde{w}| = h(|w|)^2$. We say that $\overline{w}$ encodes a computation path of $\mathcal{M}$ if and only if $\widetilde{w} = w_1 \cdot \ldots \cdot w_d$, for some words $w_1, \dots, w_d \in (\Sigma \cup S)^*$ such that

- for every $i \in [1, d]$, $w_i \in C$,
- for every $i \in [1, d-1]$, $c(w_{i+1}) \in \Delta(c(w_i))$.

Each word in $\overline{\Sigma}^*$ encodes at most one computation path of $\mathcal{M}$. Moreover, notice that given two words $\overline{w}_1, \overline{w}_2 \in \overline{\Sigma}^*$, if $\overline{w}_1$ has length at least $h(|w|)^2$, then $\overline{w}_1$ and $\overline{w}_1 \cdot \overline{w}_2$ encode the same computation path (if they encode one).

We remind the reader that given a nTM $\mathcal{T}$ working on an alphabet Σ, we write $\mathcal{T}(w_1, \dots, w_n)$ for the run of the nTM on input $(w_1, \dots, w_n) \in \Sigma^n$. We extend this notation and, given $m \leq n$, write $\mathcal{T}(w_1, \dots, w_m)$ for the run of $\mathcal{T}$ on input $(w_1, \dots, w_m, \epsilon, \dots, \epsilon) \in \Sigma^n$, i.e. an input where the last $n - m$ tapes are empty. Given two inputs $(w_1, \dots, w_n) \in \Sigma^n$ and $(w'_1, \dots, w'_n) \in \Sigma^n$ we say that $\mathcal{T}(w_1, \dots, w_n)$ and $\mathcal{T}(w'_1, \dots, w'_n)$ *perform the same computational steps* if the sequence of states produced during the runs $\mathcal{T}(w_1, \dots, w_n)$ and $\mathcal{T}(w'_1, \dots, w'_n)$ is the same. In particular, given $t \in \mathbb{N}$, $\mathcal{T}(w_1, \dots, w_n)$ accepts (resp. rejects) in time t if and only if $\mathcal{T}(w'_1, \dots, w'_n)$ accepts (resp. rejects) in time t.

One last notion is desirable in order to prove Thm. 2. We say that a computation path $\pi = (c_0, \dots, c_d)$ of $\mathcal{M}$, is an (existential or universal) *hop*, if one of the following holds:

(existential hop): all the states in the configurations $c_0, \dots, c_{d-1}$ belong to $S_\exists$, and the state of c_d belongs to $S_\forall \cup \{s_\top, s_\bot\}$,

(universal hop): all the states in the configurations $c_0, \dots, c_{d-1}$ belong to $S_\forall$, and the state of c_d belongs to $S_\exists \cup \{s_\top, s_\bot\}$,

Intuitively, in a hop $\pi = (c_0, \ldots, c_d)$, no alternation occurs in the computation path $(c_0, \ldots, c_{d-1})$ and either π is terminating or an alternation occurs between c_{d-1} and c_d.

We are now ready to state the essential technical lemma (Lemma 9) that allows us to prove Thm. 2. For simplicity, the lemma is stated referring to k, f, g, h, $\mathcal{M}$, w and $\overline{\Sigma}$ as defined above.

▶ **Lemma 9.** *Let $m = g(|w|)$ and $n \geq m + 3$. One can construct in polynomial time in n, k, $|\mathcal{M}|$ and $|w|$ a nTM $\mathcal{T}$ working on alphabet $\overline{\Sigma}$ with blank symbol ¢, and such that*

I *$\mathcal{T}$ runs in polynomial time on the length of its inputs,*

II *given $i \in [1, m]$ and $w_1, \ldots, w_i, \ldots, w_m, w'_i \in \overline{\Sigma}^*$, such that $|w_i| \geq h(|w|)^2$, $\mathcal{T}(w_1, \ldots, w_m)$ and $\mathcal{T}(w_1, \ldots, w_i \cdot w'_i, \ldots w_m)$ perform the same computational steps,*

III *for all $w_1, \ldots, w_m \in \overline{\Sigma}^*$ and $w_{m+1}, w'_{m+1}, \ldots, w_n, w'_n \in \overline{\Sigma}^*$, $\mathcal{T}(w_1, \ldots, w_m, w_{m+1}, \ldots, w_n)$ and $\mathcal{T}(w_1, \ldots, w_m, w'_{m+1}, \ldots, w'_n)$ perform the same computational steps,*

IV *$w \in \mathcal{L}(\mathcal{M})$ iff $Q_1 w_1 \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2}$: $\mathcal{T}(w_1, \ldots, w_m)$ accepts,*

where for every $i \in [1, m]$, $Q_i = \exists$ if i is odd, and otherwise $Q_i = \forall$.

Proof. Roughly speaking, the machine $\mathcal{T}$ we shall define checks if the input $(w_1, \ldots, w_m)$ encodes an accepting computation path in $\mathcal{M}$, where each w_i represents an hop.

Notice that the lemma imposes $n \geq m + 3$, with $m = g(|w|)$. This is done purely for technical convenience, as it allows us to rely on three additional tapes, i.e. the n-th, $(n-1)$-th and $(n-2)$-th tapes, whose initial content does not affect the run of $\mathcal{T}$ (see property (**III**)). These three tapes are used by $\mathcal{T}$ as auxiliary tapes. To simplify further the presentation of the proof without loss of generality, we assume that $\mathcal{T}$ features *pseudo-oracles* that implement the standard unary functions on natural numbers $(\,.\,+1)$ $\lceil \log(.) \rceil$, $\lfloor \sqrt{.} \rfloor$, as well as the function $(\,.\,> f(|w|))$. As in the case of oracles, the machine $\mathcal{T}$ can call a pseudo-oracle, which read the *initial content* of the current tape of $\mathcal{T}$ (i.e. the word in $(\overline{\Sigma} \setminus \{¢\})^*$ that occurs the beginning of the tape currently scanned by the read/write head, and delimited by the first occurrence of ¢) and produce the result of the function they implement at the beginning of the (auxiliary) n-th tape. However, all the functions implemented by pseudo-oracles can be computed in polynomial time, and one can construct in polynomial time Turing machines that compute them. More precisely, $\mathcal{O}(1)$ states are needed to implement $(\,.\,+1)$, $\lceil \log(.) \rceil$ and $\lfloor \sqrt{.} \rfloor$, and $\mathcal{O}(f(|w|))$ states are needed in order to implement $(\,.\,> f(|w|))$.[1] So, the pseudo-oracles *can be effectively removed* by incorporating their equivalent Turing machine directly inside $\mathcal{T}$, only growing its size polynomially, and resulting in $\mathcal{T}$ being a standard nTM. Fig. 1 formalises the semantics of the pseudo-oracles, given the initial content $c \in (\overline{\Sigma} \setminus \{¢\})^*$ of the current tape. Without loss of generality, we also assume $|w|$ and $f(|w|)$ to be at least 1. Lastly, again in order to simplify the presentation and without loss of generality, we assume that $\mathcal{T}$ is able to reposition the read/write head to the first position of a given tape. This is a standard assumption, which can be removed by simply adding a new symbol to the

[1] Given an input written in unary, the only non-standard function $(\,.\,> f(|w|))$ can be implemented with a Turing machine that runs in linear time on the size of its input and relies on a chain of $f(|w|) + 1$ states, plus the accepting state and rejecting state. At the i-th step of the computation, with $i \leq f(|w|)$, the read/write head is on the i-th symbol c of the input tape and the machine is in its i-th state. If c is non-blank, the machine moves the read/write head to the right and switch to the $(i+1)$-th state. Otherwise, the machine rejects the input. On the last state (reached at step $i = f(|w|) + 1$, if the machine did not previously reject), if the head reads a non-blank symbol, then the machine accepts, otherwise it rejects.

$(\,.+1)$: write $c \cdot a \cdot \mathfrak{c}$ at the beginning of the n-th tape, where $a \in \overline{\Sigma} \setminus \{\mathfrak{c}\}$ is a fixed symbol. So, the length of the initial content on the n-th tape becomes $|c|+1$.

$\lceil \log(.) \rceil$: write $a^{\lceil \log(|c|) \rceil} \cdot \mathfrak{c}$ at the beginning of the n-th tape, where $a \in \overline{\Sigma} \setminus \{\mathfrak{c}\}$ is fixed. So, the length of the initial content of the n-th tape becomes $\lceil \log(|c|) \rceil$.

$\lfloor \sqrt{(.)} \rfloor$: write $a^{\lfloor \sqrt{(|c|)} \rfloor} \cdot \mathfrak{c}$ at the beginning of the n-th tape, where $a \in \overline{\Sigma} \setminus \{\mathfrak{c}\}$ is fixed. So, the length of the initial content of the n-th tape becomes $\lfloor \sqrt{(|c|)} \rfloor$.

$(\,. > f(|w|))$: if $|c| > f(|w|)$, write $a \cdot \mathfrak{c}$ at the beginning of the n-th tape, where $a \in \overline{\Sigma} \setminus \{\mathfrak{c}\}$ is fixed. Else, write $\mathfrak{c}$ at the beginning of the n-th tape. So, the initial content of the n-th tape becomes empy if and only if $|c| \leq f(|w|)$.

Figure 1 Pseudo-oracles; $c \in (\overline{\Sigma} \setminus \{\mathfrak{c}\})^*$ is the initial content of the current tape.

alphabet (historically, ə), which shall precede the inputs of the machine. Then, the machine can retrieve the first (writeable) position on the tape by moving the read/write head to the left until it reads the new symbol ə, to then move right once, without overwriting ə.

Before describing $\mathcal{T}$, we notice that property (**II**) requires to check whether $|w_i| \geq h(|w|)^2$. We rely on Lemmas 6 and 7 in order to perform this test in polynomial time on $|w_1|$, without computing $h(|w|)$ explicitly. We have,

$$
\begin{aligned}
|w_i| \geq h(|w|)^2 \text{ iff } & \underline{\sqrt{|w_i|}} \geq h(|w|), && \text{by Lemma 7} \\
\text{iff } & \underline{\sqrt{|w_i|}} + 1 > \mathfrak{t}(k, f(|w|)), \\
\text{iff } & \overline{\log_2^k}(\underline{\sqrt{|w_i|}} + 1) > f(|w|). && \text{by Lemma 6}
\end{aligned}
$$

To check whether $\overline{\log_2^k}(\underline{\sqrt{|w_i|}}+1) > f(|w|)$ holds, it is sufficient to write w_i on the n-th tape, and then call first the pseudo-oracle for $\lfloor \sqrt{.} \rfloor$, followed by the pseudo-oracle for $(\,.+1)$ and by k calls to the pseudo-oracle for $\lceil \log(.) \rceil$ (recall that k is written in unary), and lastly one call to the pseudo-oracle for $(\,. > f(|w|))$. When taking into account the number of states needed in order to implement these pseudo-oracles, we conclude that the function $\overline{\log_2^k}(\underline{\sqrt{.}}+1) > f(|w|)$ can be implemented by a Turing machine with $\mathcal{O}(k + f(|w|))$ states. Hence, below we assume without loss of generality that $\mathcal{T}$ also features pseudo-oracles for the function $\overline{\log_2^k}(\underline{\sqrt{.}}+1) > f(|w|)$ as well as the (similar) functions $\overline{\log_2^k}(\,.+1) > f(|w|)$ and $\overline{\log_2^k}(\,.\,) > f(|w|)$, both requiring $\mathcal{O}(k + f(|w|))$ states to be implemented. Ultimately, because of the polynomial bounds on the number of states required to implement these functions, $\mathcal{T}$ (without pseudo-oracles) can be constructed in PTime on n, k, $|w|$ and $|\mathcal{M}|$.

We are now ready to construct the nTM $\mathcal{T}$ that works on alphabet $\overline{\Sigma}$. For an input $(w_1, \ldots, w_n) \in (\overline{\Sigma}^*)^n$, $\mathcal{T}$ operates following m "macrosteps" on the first m tapes, disregarding the input of the last $n - m \geq 1$ tapes. At step $i \in [1, m]$, the machine operates as follows:

macrostep $i = 1$. Recall that $Q_1 = \exists$. $\mathcal{T}$ works on the first tape.

1. $\mathcal{T}$ reads the prefix $\widetilde{w}$ of w_1, such that if $|w_1| \leq h(|w|)^2$ then $\widetilde{w} = w_1$, else $|\widetilde{w}| = h(|w|)^2$. (*Note: $\mathcal{T}$ does not consider the part of the input after the $h(|w|)^2$-th symbol*)
2. The nTM $\mathcal{T}$ then checks whether $\widetilde{w}$ corresponds to an *existential hop* in $\mathcal{M}$ starting from the state s_0. It does so by analysing from left to right $\widetilde{w}$ in chunks of length $h(|w|)$ (i.e. the size of a word in the language C of the configurations of $\mathcal{M}$), possibly with the exception of the last chunk, which can be of smaller size. Let d be the number of chunks, so that $\widetilde{w} = \widetilde{w}_1 \cdot \ldots \cdot \widetilde{w}_d$, where $\widetilde{w}_j$ is the j-th chunk analysed by $\mathcal{T}$.
   (a) If $\widetilde{w}_j \notin C$, $\mathcal{T}$ **rejects**, (*Note: else, $\widetilde{w}_j$ encodes the configuration $c(\widetilde{w}_j)$ of $\mathcal{M}$*)

(b) if the symbol s_0 does not occur in the first chunk $\widetilde{w}_1$, then $\mathcal{T}$ **rejects**,

(c) if $\widetilde{w}_j$ is not the last chunk and contains a symbol from $S_\forall$, then $\mathcal{T}$ **rejects**,

(d) if $\widetilde{w}_j$ is not the last chunk and $c(\widetilde{w}_{j+1}) \notin \Delta(c(\widetilde{w}_j))$, then $\mathcal{T}$ **rejects**.

(e) If the length of the last chunk $\widetilde{w}_d$ is not $h(|w|)$ (i.e. the length of $\widetilde{w}$ is not a multiple of $h(|w|)$), then $\mathcal{T}$ **rejects**, (*Note: this case subsumes the case where w_1 is empty*)

(f) if the last chunk $\widetilde{w}_d$ contains a symbol from $S_\exists$, then $\mathcal{T}$ **rejects**.

3. $\mathcal{T}$ analyses the last chunk $\widetilde{w}_d$ of the previous step. If $s_\top$ occurs in $\widetilde{w}_d$, then $\mathcal{T}$ **accepts**. Otherwise, if either $m = i = 1$ or $\widetilde{w}_d$ contains the symbol $s_\bot$, then $\mathcal{T}$ **rejects**. Else, $\mathcal{T}$ writes down $\widetilde{w}_d \cdot ¢$ at the beginning of the first tape, and **moves to macrostep** 2. Let $w^{(1)} = \widetilde{w}_d$, i.e. the word (currently store at the beginning of the first tape) encoding the last configuration of the computation path encoded by $\widetilde{w}$.

From Step (1), we establish the following property of macrostep 1.

▷ Claim 10. Macrostep 1 only reads at most the first $h(|w|)^2$ character of the input of the first tape, and does not depend on the input written on other tapes.

We analyse in details the complexity of macrostep 1.

▷ Claim 11. The macrostep 1 runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and only required polynomially many states to be implemented, with respect to k, $|w|$ and $|\mathcal{M}|$.

Proof. We give a more fine grained description of the various steps, and analyse their complexity. Recall that the tapes n, $n-1$ and $n-2$ are auxiliary. *Step* (1) can be performed in polynomial time in $|w_1|$, k, $|w|$ and $|\overline{\Sigma}| \leq |\mathcal{M}|$, as shown below.

```
1  write ¢^{|w_1|+1} at the beginning the (n−1)-th tape
2  while true
3     //Invariant:  The (n−1)-th tape contains a prefix of w_1
4     //            of length less than min(|w_1|, h(|w|)^2)
5     let i be the length of the initial content of the (n−1)-th tape
6     read the i-th symbol a of the first tape
7     if a = ¢, break
8     write a in the i-th position of the (n−1)-th tape
9     call the pseudo-oracle for log_2^k(√· + 1) > f(|w|) on the (n−1)-th tape
10    if the initial content of the n-th tape is not empty, break
```

At the end of this procedure, the $(n-1)$-th tape contains the word $\widetilde{w}$. Indeed, lines 3, 4 and 6 copy the i-th character of w_1 to the $(n-1)$-th tape. If $|w_1| < h(|w|)^2$, then $\widetilde{w} = w_1$ and after copying w_1 on the $(n-1)$-th tape, the test in line 5 becomes true. Otherwise, after copying the first $h(|w|)^2$ characters of w_1 to the $(n-1)$-th tape, the pseudo-oracle call of line 7 will write a non-blank symbol on the n-th tape, making the test in line 8 true. Time-wise, the complexity of the procedure above is polynomial in $|w_1|$, k (line 9), $|w|$ (line 9) and $|\overline{\Sigma}| \leq |\mathcal{M}|$ (lines 5–8). Space-wise, performing line 1 of the procedure requires a constant number of states. Performing lines 5–8 requires $\mathcal{O}(|\overline{\Sigma}|)$ states, as $\mathcal{T}$ needs to keep track of which symbol was read on the first tape, in order to write in on the tape $(n-1)$. For line 9 instead, we must take into account the number of states required to implement the computation done by the pseudo-oracle directly in $\mathcal{T}$. As stated at the beginning of the proof, these are $\mathcal{O}(k + f(|w|))$ many states. Hence, $\mathcal{T}$ can implement the procedure above with $\mathcal{O}(k + f(|w|) + |\mathcal{M}|)$ many states.

Let us now move to *Step* (2). Recall that, after *Step* (1), $\widetilde{w}$ is stored at the beginning of tape $(n-1)$. The machine $\mathcal{T}$ continues as follows, where the lines marked with symbols of the form ⓘ are later expanded and analysed in details.

```
1  write w̃ · ¢ at the beginning of the first tape
2  if the initial content of the first tape is empty, reject
3  ① if s_0 does not appear in the first h(|w|) characters of w̃, reject
4  while true
5      //Invariant: The 1st tape starts with a non-empty suffix of w̃.
6      let i be the length of the initial content of the first tape
7      write ¢^{i+1} on both the (n−1)-th and (n−2)-th tapes
8      call the pseudo-oracle for ⁻log₂^k(. + 1) > f(|w|) on the first tape
9      if the initial content of the n-th tape is empty, rejects
10     ② copy the first h(|w|) characters of the first tape to the (n−1)-th tape
11     //The tape (n−1) now contains a chunk w̃_j.
12     if the initial content of the (n−1)-th tape and the first tape is equal, break
13     ③ if the initial content of the (n−1)-th tape does not belong to C, reject
14     ④ if a symbol from S_∀ occur in the initial content of the (n−1)-th tape, reject
15     ⑤ shift the initial content of the first tape h(|w|) positions to the left,
           effectively removing the first h(|w|) characters from the tape
16     //The first tape now starts with the chunk w̃_{j+1}.
17     ⑥ copy the first h(|w|) characters of the first tape to the (n−2)-th tape
18     //The tape (n−2) now contains the chunk w̃_{j+1}.
19     ⑦ perform step (2d) //w̃_j and w̃_{j+1} on tapes (n−1) and (n−2), respectively
20 //Post: The tape (n−1) contains the last chunk w̃_d.
21 ⑧ if the initial content of the (n−1)-th tape does not belong to C, reject
22 ⑨ if a symbol from S_∃ occur in the initial content of the (n−1)-th tape, reject
```

Provided that the lines marked with symbols of the form ⓘ can be performed in polynomial time and can be implemented with a polynomial number of states, it is straightforward to see that this procedure also runs in polynomial time and only uses polynomially many states. Indeed, at the j-th iteration of the while loop, the j-th chunk of $\widetilde{w}$ is analysed. Therefore, the body of the while loop is executed at most $d = \lceil \frac{|\widetilde{w}|}{h(|w|)} \rceil$ times, where d is the number of chunks (which is polynomial in $|w_1|$). Moreover, it is easy to see that the lines 1,2, 6–9 and 12 runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and only requires $\mathcal{O}(k + f(|w|) + |\mathcal{M}|)$ many states to be implemented. Indeed, line 1 can be implemented in linear time on $|\widetilde{w}| \leq |w_1|$ and $|\overline{\Sigma}|$, and requires $\mathcal{O}(\overline{\Sigma})$ states to be implemented, in order to track the symbols read on the $(n-1)$-th tape, and copy them on the first tape. A similar analysis can be done for line 12. Line 2 runs in time $\mathcal{O}(1)$ and requires $\mathcal{O}(1)$ states. Line 7 runs in time $\mathcal{O}(|\widetilde{w}|)$ and requires $\mathcal{O}(1)$ states. Line 8 is analogous to line 9 of Step (1), and checks whether the current suffix of $\widetilde{w}$ stored in the first tape contains at least $h(|w|)$ characters. Together with line ⑤, this line assures that $|\widetilde{w}|$ is a multiple of $h(|w|)$, effectively implementing the step (2e).

Let us now focus on the lines marked with ⓘ. Line ①, implements the step (2b). To test whether $h(|w|)$ character have been read, we can rely on the pseudo-oracle for $\overline{\log_2^k}(. + 1) > f(|w|)$, similary to what it is done in line 7. Here is the resulting procedure.

```
1  write ¢^{|w̃|+1} on the (n−1)-th tape
```

```
2  while true
3      //Invariant:  The (n − 1)-th tape contains a prefix of w̃
4      //            of length less than min(|w̃|, h(|w|))
5      let i be the length of the initial content of the (n − 1)-th tape
6      read the i-th symbol a of the first tape
7      if a = s⊤, break
8      if a = ¢, reject
9      write a in the i-th position of the (n − 1)-th tape
10     call the pseudo-oracle for log_2^k(. + 1) > f(|w|) on the (n − 1)-th tape
11     if the initial content of the n-th tape is not empty, reject
12 //Post: s⊤ appears in the first h(|w|) characters of w̃
```

With a similar analysis to what done in Step (1), we conclude that ① runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and requires $\mathcal{O}(k + f(|w|) + |\mathcal{M}|)$ states to be implemented. Line ② is very similar to line ①. Notice that form line 6 of Step (2), the initial content of the tape $(n-1)$ is empty. Moreover, from line 7, the initial content of the first tape has at least $h(|w|)$ characters. Here is the procedure for ②.

```
1  while true
2      //Invariant:  The (n − 1)-th tape contains a prefix of the initial content
3      //            of the first tape, of length less than h(|w|)
4      let i be the length of the initial content of the (n − 1)-th tape
5      read the i-th symbol a of the first tape
6      write a in the i-th position of the (n − 1)-th tape
7      call the pseudo-oracle for log_2^k(. + 1) > f(|w|) on the (n − 1)-th tape
8      if the initial content of the n-th tape is not empty, break
9  //Post: the initial content of the (n − 1)-th tape is a prefix of the
10 //      initial content of the first tape, of length h(|w|)
```

As in the case of line ①, this procedure runs in polynomial time on $|w_1|$, k $|w|$ and $|\mathcal{M}|$, and requires $\mathcal{O}(k + f(|w|) + |\mathcal{M}|)$ states to be implemented. After executing line ①, the initial content of the tape $(n-1)$ is a chunk of $\widetilde{w}$, say $\widetilde{w}_j$.
Lines ③ and ④ implement the steps (2a) and (2c). The machine $\mathcal{T}$ can implement both lines simultaneously, by iterating through the initial content of the $(n-1)$-th tape as shown below.

```
1  move the read/write head to the first position of the (n − 1)-th tape
2  while true
3      let a be the symbol corresponding to the read/write head
4      if a = ¢, reject
5      if a ∈ S∀, reject
6      if a ∈ S \ S∀, break
7      move the read/write head to the right
8  //Post: a symbol from S \ S∀ was found
9  move the read/write head to the right
10 if the the read/write head reads ¢, reject
11 while the read/write head does not read ¢
```

```
12      if the read/write had reads a symbol from S, reject
13      move the read/write head to the right
14  //Post: the initial content of the (n − 1)-th tape belongs to C and does not
15  //      contain symbols from S_∀
```

The correctness of this procedure with respect to the description of lines ③ and ④ is straightforward as we recall that $C = \{\widehat{w} \in \Sigma^* \cdot S \cdot \Sigma^+ \mid |\widehat{w}| = h(|w|)\}$, and by line ① the initial content of the tape $(n-1)$ has length $h(|w|)$. The running time of the procedure above is polynomial in the length of the initial content of the $(n-1)$-th tape, and thus polynomial on $|w_1|$. The number of states required to implement this procedure is in $\mathcal{O}(1)$. Let us move to line ⑤. Essentially, in this lines $\mathcal{T}$ removes from the initial content of the first tape the chunk that is currently analysed and stored in the tape $(n-1)$, so that the initial content of the first tape starts with the next chunk. A possible implementation of this line is given below. Recall that, from line 6 of Step (2), the $(n-2)$-th tape starts with the word $¢^{i+1}$, where i is the length of the initial content of the first tape. Moreover, from line 12 of Step (2), the length of the initial content of the first tape is greater than the length of the initial content of the tape $(n-1)$.

```
1   move the read/write head to the first occurrence of ¢ on the (n − 1)-th tape
2   //The read/write head is now on the position |w̃_j| of the (n − 1)-th tape
3   move the read/write head to the first tape
4   //Read/write head currently in position |w̃_j| of the first tape
5   //From line 12 of Step (2), the head reads a symbol different from ¢
6   while true
7       let a be the symbol corresponding to the read/write head
8       write ¢
9       move the read/write head to the right
10      if the read/write head reads ¢,
11          move the read/write head to the first occurrence of ¢ on the (n − 2)-th tape
12          write a
13          break
14      else
15          move the read/write head to the first occurrence of ¢ on the (n − 2)-th tape
16          write a
17          move the read/write head to the first occurrence of ¢ on the first tape
18          //read/write head in the position where a was previously stored
19          move the read/write head right
20  //Post: The initial content of the tape (n − 2) is the word required by line ⑤
21  let w̃′ be the initial content of the (n − 1)-th tape
22  write w̃′ · ¢ at the beginning of the first tape
```

Again, this procedure runs in polynomial time on $|\widetilde{w}|$ and $|\mathcal{M}|$. Since $\mathcal{T}$ needs to keep track of the symbol read in line 7, as well as implementing lines 21–22, the number of states required to implement the procedure is $\mathcal{O}(\overline{\Sigma})$, and thus polynomial in $|\mathcal{M}|$.
Line ⑥ copies the chunk $\widetilde{w}_{j+1}$ to the tape $(n-2)$, so that $\mathcal{T}$ can then perform the step (2d) (line ⑦). Line ⑥ is implemented analogously to line ②, i.e. the line that copied the chunk $\widetilde{w}_j$ to the $(n-1)$-th tape. Compared to the code for line ②, it is sufficient to

replace the steps done to the tape $(n-1)$ to equivalent steps done on the $(n-2)$ step in order to obtain the code for line ⑥, which therefore runs in polynomial time, and requires $\mathcal{O}(k + f(|w|) + |\mathcal{M}|)$ many sstates in order to be implemented.

Let us now assume that the initial content of the $(n-1)$-th step is the chunk $\widetilde{w}_j$, and that the initial content of tape $(n-2)$ is the chunk $\widetilde{w}_{j+1}$. According to line ⑦, we must check whether $c(\widetilde{w}_{j+1}) \in \Delta(c(\widetilde{w}_j))$ (step (2d)). Notice that we do now know whether $\widetilde{w}_{j+1} \in C$, as this test will be performed at the next iteration of the while loop of Step (2). However, we can still easily implement step (2d) as follows: we find the pair $(s, a) \in S \times \Sigma$ such that $w' \cdot s \cdot a \cdot w'' = \widetilde{w}_j$ for some words w' and w''. This decomposition is guaranteed from $\widetilde{w}_j \in C$, which holds from line ③. The machine then iterates over all elements of $\delta(s, a)$, updating $\widetilde{w}_j$ accordingly, on the n-th tape. After each update, $\mathcal{T}$ checks whether the initial content of the n-th tape corresponds to $\widetilde{w}_{j+1}$. Here is the procedure.

```
1  let w̃_j be the initial content of the (n−1)-th tape
2  let w̃_{j+1} be the initial content of the (n−2)-th tape
3  move the read/write head to the beginning of the tape (n−1)
4  while the read/write had reads a character not from S
5      move the read/write head to the right
6  //Since w̃_j ∈ C, the loop above terminates
7  //Post: the read/write head reads a character from S
8  let s be the symbol corresponding to the read/write head
9  move the read/write head to the right
10 let a be the symbol corresponding to the read/write head
11 //Since w̃_j ∈ C, a belongs to Σ
12 for (s', a', i) ∈ S × Σ × {−1, +1} belonging to δ(s, a)
13     write w̃_j · ¢ at the beginning of the n-th tape
14     move to the read/write tape to the (only) occurrence of s on the n-th tape
15     if i = 1,
16         write a' //overwrites s
17         move the read/write tape to the right
18         write a' //overwrites a
19     else if s occurs at the beginning of the n-th tape,
20         write s_⊥
21     else
22         move the read/write tape left
23         let b be the symbol corresponding to the read/write head
24         write s' //overwrites b
25         move the read/write tape to the right
26         write b //overwrites s
27         move the read/write tape to the right
28         write a' // overwrites a
29     if the initial content of the n-th tape equals w̃_{j+1}, goto line 32
30 //Post : c(w̃_{j+1}) ∉ Δ(c(w̃_j))
31 reject
32 //If this line is reached, c(w̃_{j+1}) ∈ Δ(c(w̃_j))
```

Briefly, if the test in line 14 holds, then $\widetilde{w}_j = w' \cdot s \cdot a \cdot w''$ is copied on the n-th tape and updated to $w' \cdot a' \cdot s' \cdot w''$, according to the semantics of the ATM $\mathcal{M}$ for the case $(s', a', 1) \in \delta(s, a)$. If instead the test in line 19 holds, then $\widetilde{w}_j$ is of the form $s \cdot a \cdot w''$ we are considering a triple $(s', a', -1) \in \delta(s, a)$. According to the semantics of the ATM $\widetilde{w}_j$ must be updated to $s_\perp \cdot a \cdot w''$, as done in line 20. Lastly, if the else branch in line 21 is reached, then $\widetilde{w}_j$ is of the form $w' \cdot b \cdot s \cdot a \cdot w''$, and since we are considering a triple $(s', a', -1) \in \delta(s, a)$, it must be updated to $w' \cdot s' \cdot b \cdot a' \cdot w''$, as done in lines 22–28. This procedure runs in polynomial time on $|\widetilde{w}_j| \leq |w_1|$ and $|\mathcal{M}|$, and because of the *for* loop in line 12 together with the necessity to copy $\widetilde{w}_j$ to the n-th tape and keeping track of the symbols s, a and b, it requires a number of states that is polynomial in $|\mathcal{M}|$.
Lastly, the lines ⑧ and ⑨ are analogous to the lines ③ and ④ (the only difference being that $S_\forall$ is replaced by $S_\exists$ in order to correctly implement ⑨ and so (2f)). We conclude that Step (2) of the procedure runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$. and requires a polynomial number of states with respect to k, $|w|$ and $|\mathcal{M}|$.
At the end of Step (2), the initial content of the tape $(n-1)$ corresponds to the last chunk $\widetilde{w}_d$ of $\widetilde{w}$. Then, the step (3) performed by $\mathcal{T}$ is implemented as follows.

```
1  let w̃_d be the initial content of the (n − 1)-th tape
2  if s_⊤ appears in w̃_d, accept
3  if s_⊥ appears in w̃_d , reject
4  if m = 1, reject
5  write w̃_d · ¢ at the beginning of the first tape //see w^(1)
6  move to macrostep 2
```

Clearly, this step runs in polynomial time on $|\widetilde{w}_d| \leq |w_1|$ and $|\overline{\Sigma}| \leq |\mathcal{M}|$, and requires $\mathcal{O}(|\overline{\Sigma}|)$ many states to be implemented (see line 5). Considering all the steps, we conclude that macrostep 1 runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and requires polynomially many states to be implemented, w.r.t. k, $|w|$ and $|\mathcal{M}|$. ◁

The semantics of $\mathcal{T}$ during macrostep 1 is summarised in the following claim.

▷ Claim 12. Let $\widetilde{w}$ prefix $\widetilde{w}$ of $w_1 \cdot ¢$, s.t. either $\widetilde{w} \cdot ¢$ is a prefix of $w_1 \cdot ¢$ or $|\widetilde{w}| = h(|w|)^2$.

- if $\widetilde{w}$ encodes an existential hop of $\mathcal{M}$ starting on state s_0 and ending in $s_\top$, $\mathcal{T}$ accepts,
- else, if $1 < m$ and $\widetilde{w}$ encodes an existential hop of $\mathcal{M}$, starting on state s_0 and ending in a state from $S_\forall$, then $\mathcal{T}$ writes the last configuration of this path at the beginning of the first tape (i.e. the word $w^{(1)}$) and moves to macrostep 2,
- otherwise, $\mathcal{T}$ rejects.

macrostep: $i > 1$, i odd. Recall that $Q_i = \exists$. $\mathcal{T}$ works on the i-th tape and on the word $w^{(i-1)}$ written in on the $(i-1)$-th tape at the end of step $i-1$. This macrostep resembles the first one.

1. $\mathcal{T}$ reads the prefix $\widetilde{w}$ of w_i, such that if $|w_i| \leq h(|w|)^2$ then $\widetilde{w} = w_i$, else $|\widetilde{w}| = h(|w|)^2$. (*Note: $\mathcal{T}$ does not consider the part of the input after the $h(|w|)^2$-th symbol*)
2. The nTM $\mathcal{T}$ then checks whether $w^{(i-1)} \cdot \widetilde{w}$ encodes an existential hop in $\mathcal{M}$. It does so by analysing (from left to right) $\widetilde{w}$ in chunks of length $h(|w|)$, possibly with the exception of the last chunk, which can be of smaller size. Let d be the number of chunks, and let $\widetilde{w}_j$ be the j-th chunk analysed by $\mathcal{T}$.
   (a) If $\widetilde{w}_j \notin C$, $\mathcal{T}$ **rejects**,
   (b) if the first chunk $\widetilde{w}_1$ is such that $c(\widetilde{w}_1) \notin \Delta(c(w^{(i-1)}))$, then $\mathcal{T}$ **rejects**,

(c) if $\widetilde{w}_j$ is not the last chunk and contains a symbol from $S_\forall$, then $\mathcal{T}$ **rejects**,

(d) if $\widetilde{w}_j$ is not the last chunk and $c(\widetilde{w}_{j+1}) \notin \Delta(c(\widetilde{w}_j))$, then $\mathcal{T}$ **rejects**,

(e) if the length of the last chunk $\widetilde{w}_d$ is not $h(|w|)$ (i.e. the length of $\widetilde{w}$ is not a multiple of $h(|w|)$), then $\mathcal{T}$ **rejects**,

(f) if the last chunk $\widetilde{w}_d$ contains a symbol from $S_\exists$, then $\mathcal{T}$ **rejects**.

3. $\mathcal{T}$ analyses the last chunk $\widetilde{w}_d$ of the previous step. If $s_\top$ occurs in $\widetilde{w}_d$, then $\mathcal{T}$ **accepts**. Otherwise, if $i = m$ or $s_\bot$ occurs in $\widetilde{w}_d$, then $\mathcal{T}$ **rejects**. Else, $\mathcal{T}$ writes down $\widetilde{w}_d \cdot \mathfrak{c}$ at the beginning of the i-th tape, and **moves to macrostep** $(i+1)$. Let $w^{(i)} = \widetilde{w}_d$.

From macrostep $(i-1)$, the word $w^{(i-1)}$ corresponds to the initial content of the $(i-1)$-th tape, and its length is bounded by $|w_{i-1}|$. Moreover, $w^{(i-1)} \in C$. The step (2b) can be implemented as line ⑦ in the proof of Claim 11. The analysis of this macrostep follows essentially the same arguments given for the macrostep 1, allowing us to establish the following claims.

▷ Claim 13. Let $i > 1$ odd. Macrostep i only reads at most the first $h(|w|)^2$ characters of the input of the i-th tape, and at most the first $h(|w|)$ characters of the $(i-1)$-th tape. It does not depend on the content written on the other tapes.

▷ Claim 14. Let $i > 1$ odd. The macrostep i runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and required polynomially many states to be implemented, w.r.t. k, $|w|$ and $|\mathcal{M}|$.

▷ Claim 15. Let $i > 1$ odd, and let $\widetilde{w}$ prefix of $w_i \cdot \mathfrak{c}$, such that either $\widetilde{w} \cdot \mathfrak{c}$ is a prefix of $w_i \cdot \mathfrak{c}$ or $|\widetilde{w}| = h(|w|)^2$. Suppose that $\mathcal{T}$ did not halt on the macrosteps $1, \ldots, i-1$.

- if $w^{(i-1)} \cdot \widetilde{w}$ encodes an existential hop of $\mathcal{M}$ ending in state $s_\top$, then $\mathcal{T}$ accepts,
- else, if $i < m$ and $w^{(i-1)} \cdot \widetilde{w}$ encodes an existential hop of $\mathcal{M}$, ending in a state from $S_\forall$, then $\mathcal{T}$ writes the last configuration of this path (i.e. $w^{(i)}$) at the beginning of the i-th tape and moves to macrostep $(i+1)$,
- otherwise, $\mathcal{T}$ rejects.

macrostep: $i > 1$, i even. Recall that $Q_i = \forall$. $\mathcal{T}$ works on the i-th tape and on the word $w^{(i-1)}$ written in on the $(i-1)$-th tape at the end of step $i-1$.

1. $\mathcal{T}$ reads the prefix $\widetilde{w}$ of w_i, such that if $|w_i| \leq h(|w|)^2$ then $\widetilde{w} = w_i$, else $|\widetilde{w}| = h(|w|)^2$. (*Note: $\mathcal{T}$ does not consider the part of the input after the $h(|w|)^2$-th symbol*)

2. The nTM $\mathcal{T}$ works "dually" with respect to the macrosteps where i is odd. It checks whether $w^{(i-1)} \cdot \widetilde{w}$ corresponds to an universal hop of $\mathcal{M}$. However, differently from the case where i is odd, if $\widetilde{w}$ does not encode such a computation path, then $\mathcal{T}$ accepts (as in this case the input is not "well-formed" and should not be considered for the satisfaction of the quantifier $Q_i = \forall$). As in the other macrosteps, $\mathcal{T}$ perform this step by analysing (from left to right) $\widetilde{w}$ in chunks of length $h(|w|)$, possibly with the exception of the last chunk, which can be of smaller size. Let d be the number of chunks, and let $\widetilde{w}_j$ be the j-th chunk analysed by $\mathcal{T}$.

(a) If $\widetilde{w}_j \notin C$, $\mathcal{T}$ **accepts**,

(b) if the first chunk $\widetilde{w}_1$ is such that $c(\widetilde{w}_1) \notin \Delta(c(w^{(i-1)}))$, then $\mathcal{T}$ **accepts**,

(c) if $\widetilde{w}_j$ is not the last chunk and contains a symbol from $S_\exists$, then $\mathcal{T}$ **accepts**,

(d) if $\widetilde{w}_j$ is not the last chunk and $c(\widetilde{w}_{j+1}) \notin \Delta(c(\widetilde{w}_j))$, then $\mathcal{T}$ **accepts**,

(e) if the length of the last chunk $\widetilde{w}_d$ is not $h(|w|)$ (i.e. the length of $\widetilde{w}$ is not a multiple of $h(|w|)$), then $\mathcal{T}$ **accepts**,

(f) if the last chunk $\widetilde{w}_d$ contains a symbol from $S_\forall$, then $\mathcal{T}$ **accepts**.

3. $\mathcal{T}$ analyses the last chunk $\widetilde{w}_d$ of the previous step. If $s_\top$ occurs in $\widetilde{w}_d$, then $\mathcal{T}$ **accepts**. If $s_\bot$ occurs in $\widetilde{w}_d$ or $i = m$, then $\mathcal{T}$ **rejects**. Else, $\mathcal{T}$ writes down $\widetilde{w}_d \cdot \mathtt{¢}$ at the beginning of the i-th tape, and **moves to macrostep** $(i+1)$. Let $w^{(i)} = \widetilde{w}_d$.

Following the same arguments of the first macrostep, the following claims are established.

▷ Claim 16. Let $i > 1$ even. Macrostep i only reads at most the first $h(|w|)^2$ characters of the input of the i-th tape, and at most the first $h(|w|)$ characters of the $(i-1)$-th tape. It does not depend on the content written on the other tapes.

▷ Claim 17. Let $i > 1$ even. The macrostep i runs in polynomial time on $|w_1|$, k, $|w|$ and $|\mathcal{M}|$, and required polynomially many states to be implemented, w.r.t. k, $|w|$ and $|\mathcal{M}|$.

▷ Claim 18. Let $i > 1$ even, and let $\widetilde{w}$ prefix of $w_i \cdot \mathtt{¢}$, such that either $\widetilde{w} \cdot \mathtt{¢}$ is a prefix of $w_i \cdot \mathtt{¢}$ or $|\widetilde{w}| = h(|w|)^2$. Suppose that $\mathcal{T}$ did not halt on the macrosteps $1, \dots, i-1$.

- if $w^{(i-1)} \cdot \widetilde{w}$ encodes an universal hop of $\mathcal{M}$ ending in state $s_\bot$, then $\mathcal{T}$ rejects,
- else, if $i = m$ and $w^{(i-1)} \cdot \widetilde{w}$ encodes an universal hop of $\mathcal{M}$, ending in a state from $S_\exists$, then $\mathcal{T}$ rejects,
- else, if $i < m$ and $\widetilde{w}$ encodes an universal hop of $\mathcal{M}$, ending in a state from $S_\exists$, then $\mathcal{T}$ writes the last configuration of this path (i.e. $w^{(i)}$) at the beginning of the i-th tape and moves to macrostep $(i+1)$,
- otherwise, $\mathcal{T}$ accepts.

This ends the definition of $\mathcal{T}$. Since $\mathcal{T}$ performs $m = g(|w|)$ macrosteps, where g is a polynomial, from Claim 11, Claim 14 and Claim 17 we conclude that $\mathcal{T}$ can be constructed in polynomial time in $n > m$ (number of tapes), k, $|\mathcal{M}|$ and $|w|$. Moreover, $\mathcal{T}$ runs in polynomial time, as required by property (**I**). By Claim 10, Claim 13 and Claim 16, $\mathcal{T}$ only looks at a prefix of its inputs of length at most $h(|w|)^2$, and its runs are independent on the input of the last $n - m$ tapes. Therefore, both the properties (**II**) and (**III**) are satisfied.

To conclude the proof, it remains to show that $\mathcal{T}$ satisfies property (**IV**), which follows directly from the following claim, proved thanks to Claim 12, Claim 15 and Claim 18.

▷ Claim 19. Let $\ell \in [1, m]$ and $(\pi_1, \dots, \pi_\ell)$ be a ℓ-uple such that,

- $\pi_1 \cdot \ldots \cdot \pi_\ell$ is a computation path of $\mathcal{M}$,
- for every $i \in [1, \ell]$ odd, $\pi_i = (c_0^i, \dots, c_{d_i}^i)$ is an existential hop,
- for every $i \in [1, \ell]$ even, $\pi_i = (c_0^i, \dots, c_{d_i}^i)$ is an universal hop,
- $c_0^1 = (w, s_0, 0)$.

For every $i \in [1, \ell]$, let $w_i \in \overline{\Sigma}^{h(|w|)^2}$ be a word encoding the computation path π_i. Then,

$$\gamma(c_{d_\ell}^\ell) = \mathtt{acc} \text{ iff } Q_{\ell+1} w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2}, \dots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \dots, w_m) \text{ accepts, } (1)$$

where for every $i \in [\ell+1, m]$, $Q_i = \exists$ if i is odd, and otherwise $Q_i = \forall$.

Proof. The proof is by induction on $m - \ell$.

base case: $m = \ell$. Since $\mathcal{M}$ is g-alternation bounded and $m = g(|w|)$, in this case $c_{d_m}^m$ is either $s_\top$ of $s_\bot$. If $\gamma(c_{d_m}^m) = \mathtt{acc}$, then the state of the last configuration $c_{d_m}^m$ is the accepting state $s_\top$, and from Claim 12, Claim 15 and Claim 18, we conclude that $\mathcal{M}(w_1, \dots, w_m)$ accepts. Else, suppose that $\mathcal{T}(w_1, \dots, w_m)$ accepts. Then, since $w_1 \cdot \ldots \cdot w_m$ encodes a computation path of $\mathcal{M}$, by Claim 12, Claim 15 and Claim 18, the state of the last configuration $c_{d_m}^m$ cannot be $s_\bot$. Hence, it is $s_\top$, and $\gamma(c_{d_m}^m) = \mathtt{acc}$.

induction step: $m - \ell > 1$. First of all, if the path $\pi_1 \cdot \ldots \cdot \pi_\ell$ is terminating, then (1) follows exactly as in the base case. Below, let us assume that $\pi_1 \cdot \ldots \cdot \pi_\ell$ is not terminating. Notice that this means that the length of the computation path is less than $h(|w|)$, since the ATM $\mathcal{M}$ is h-time bounded. Below, $\overline{c}$ is short for $c^\ell_{d_\ell}$ and we write $\overline{w}$ for the word from C that encodes $\overline{c}$. We split the proof depending on the parity of ℓ.

case: ℓ even. In this case, $Q_{\ell+1} = \exists$, π_ℓ is an universal hop, and $\overline{c}$ contains a state in $S_\exists$. ($\Rightarrow$): Suppose $\gamma(\overline{c}) = \texttt{acc}$. Since $\overline{c}$ is an existential configuration, this implies that there a computation path $\pi_{\ell+1} = (c_0, \ldots, c_d)$ such that $\overline{c} \cdot \pi_{\ell+1}$ is an existential hop, and $\gamma(c_d) = \texttt{acc}$. Let $w_{\ell+1}$ be a word encoding $\pi_{\ell+1}$, of minimal length. Since $\mathcal{M}$ is h-time bounded, $w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2}$. By induction hypothesis,

$$Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

Hence,

$$\exists w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2} Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

($\Leftarrow$): Conversely, suppose that the right hand side of (1) holds. Since $Q_{\ell+1} = \exists$, from Claim 15 there is a word $w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2}$ that encodes a computation path $\pi_{\ell+1} = (c_0, \ldots, c_d)$ of $\mathcal{M}$, such that $\overline{c} \cdot \pi_{\ell+1}$ is an existential hop and

$$Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

By induction hypothesis, $\gamma(c_d) = \texttt{acc}$, which yields $\gamma(\overline{c}) = \texttt{acc}$, by definition of γ.

case: ℓ odd. In this case, $Q_{\ell+1} = \exists$, π_ℓ is an existential hop, and $\overline{c}$ contains a state in $S_\forall$. ($\Rightarrow$): Suppose $\gamma(\overline{c}) = \texttt{acc}$. Since $\overline{c}$ is an universal configuration, $\gamma(c_d) = \texttt{acc}$ holds for every computation path $\pi = (c_0, \ldots, c_d)$ such that $c^\ell_{d_\ell} \cdot \pi$ is an universal hop. Let $w_{\ell+1}$ be a word in $\overline{\Sigma}^{h(|w|)^2}$. If $\overline{w} \cdot w_{\ell+1}$ does not encode an universal hop, then from Claim 18, $\mathcal{T}(w_1, \ldots, w_{\ell+1}, x_{\ell+2}, \ldots, x_m)$ accepts for every values of $x_{\ell+2}, \ldots, x_m$. If Suppose instead that $\overline{w} \cdot w_{\ell+1}$ encodes the universal hop $\pi_{\ell+1} = (c_0, \ldots, c_d)$. From $\gamma(c_d) = \texttt{acc}$ and by induction hypothesis,

$$Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

We conclude that

$$\forall w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2} Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

($\Leftarrow$): Conversely, suppose that the right hand side of (1) holds. Since $Q_{\ell+1} = \forall$, this means that for every $w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2}$

$$Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts.}$$

Since $\overline{c}$ is an universal configuration, in order to conclude that $\gamma(\overline{c}) = \texttt{acc}$ holds, it is sufficient to check that, for every computation path $\pi_{\ell+1} = (c_0, \ldots, c_d)$ of $\mathcal{M}$, if $\overline{c} \cdot \pi_{\ell+1}$ is an universal hop, then $\gamma(c_d) = \texttt{acc}$. So, consider a computation path $\pi_{\ell+1} = (c_0, \ldots, c_d)$ of $\mathcal{M}$, such that $\overline{c} \cdot \pi_{\ell+1}$ is an universal hop. Since $\mathcal{M}$ is h-time bounded, there is $w_{\ell+1} \in \overline{\Sigma}^{h(|w|)^2}$ that encodes $\pi_{\ell+1}$. We have,

$$Q_{\ell+2}w_{\ell+2} \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts,}$$

and by induction hypothesis we obtain $\gamma(c_d) = \texttt{acc}$, concluding the proof. ◁◀

The proof of Thm. 2 stems directly from Lemma 9.

▶ **Theorem 2.** *The kAExp$_{pol}$-prenex* TM *problem is kAExp$_{pol}$-complete.*

Proof. Fix $k \geq 1$. Since the $k\text{AExp}_{pol}$-prenex problem is solvable in $k\text{AExp}_{pol}$ directly from its definition, we focus on the $k\text{AExp}_{pol}$-hardness. We aim at a reduction from the membership problem described in Prop. 8, relying on Lemma 9. Let $f, g\colon \mathbb{N} \to \mathbb{N}$ be two polynomials, and define $h(\mathtt{x}) \stackrel{\text{def}}{=} \mathfrak{t}(k, f(\mathtt{x}))$. Consider a h-time bounded, g-alternation bounded ATM $\mathcal{M} = (\Sigma, S_\exists, S_\forall, s_0, s_\top, s_\bot, \delta)$, as well as a word $w \in \Sigma^*$. We want to check whether $w \in \mathcal{L}(\mathcal{M})$. To simplify the proof without loss of generality, we assume $|w| \geq 1$ and that $f(\mathtt{x}) \geq \mathtt{x}$ for every $\mathtt{x} \in \mathbb{N}$. Then, $f(|w|) \geq |w| \geq 1$. Let $m = \max(f(|w|), g(|w|)) + 3$, and let $\overline{\Sigma} \stackrel{\text{def}}{=} \Sigma \cup S \cup \{s_\top, s_\bot, \mathfrak{c}\}$. We consider the mDTM $\mathcal{T} = (m, \overline{\Sigma}, S', s'_0, s'_\top, s'_\bot, \delta')$ derived from $\mathcal{M}$ by applying Lemma 9. From the property (**I**) of $\mathcal{T}$, there is a polynomial $p\colon \mathbb{N} \to \mathbb{N}$ such that, for every $(w_1, \dots, w_m) \in (\overline{\Sigma}^*)^m$, $\mathcal{T}(w_1, \dots, w_m)$ halts in time $p(\max(|w_1|, \dots, |w_m|))$. W.l.o.g., we can assume $p(\mathtt{x}) = \alpha \mathtt{x}^d + \beta$ for some $\alpha, d, \beta \in \mathbb{N}_+$. So, for all $\mathtt{x} \in \mathbb{N}$, $p(\mathtt{x}) \geq \mathtt{x}$. We rely on $\mathcal{T}$ to build a polynomial instance of the $k\text{AExp}_{pol}$-prenex problem that is satisfied if and only if $w \in \mathcal{L}(\mathcal{M})$. Let $n \stackrel{\text{def}}{=} p(2 \cdot m)$, and consider the nTM $\mathcal{T}' = (n, \overline{\Sigma}, S', s'_0, s'_\top, s'_\bot, \delta')$. Notice that $\mathcal{T}'$ is defined exactly as $\mathcal{T}$, but has $n \geq m$ tapes, where n is polynomial in $|w|$. However, since the two machines share the same transition function δ', only the first m tapes are effectively used by $\mathcal{T}'$, and for all $(w_1, \dots, w_m, \dots, w_n) \in \overline{\Sigma}^n$, $\mathcal{T}(w_1, \dots, w_m)$ and $\mathcal{T}'(w_1, \dots, w_n)$ perform the same computational steps. Let $\mathbf{Q} = (Q_1, \dots, Q_n) \in \{\exists, \forall\}^n$ such that for all $i \in [1, n]$, $Q_i = \exists$ iff i is odd. We consider the instance of the $k\text{AExp}_{pol}$-prenex problem given by $(n, \mathbf{Q}, \mathcal{T}')$. This instance is polynomial in $|w|$ and $|\mathcal{M}|$, and asks whether

$$Q_1 w_1 \in \overline{\Sigma}^{\mathfrak{t}(k,n)}, \dots, Q_n w_n \in \overline{\Sigma}^{\mathfrak{t}(k,n)} : \mathcal{T}'(w_1, \dots, w_n) \text{ accepts in time } \mathfrak{t}(k, n). \tag{2}$$

To conclude the proof, we show that Eq. (2) holds if and only if

$$Q_1 w_1 \in \overline{\Sigma}^{h(|w|)^2}, \dots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \dots, w_m) \text{ accepts.} \tag{3}$$

Indeed, directly from the property (**IV**) of $\mathcal{T}$, this equivalence implies that Eq. (2) is satisfied if and only if $w \in \mathcal{L}(\mathcal{M})$, leading to the $k\text{AExp}_{pol}$-hardness of the $k\text{AExp}_{pol}$-prenex problem directly by Prop. 8.

By definition of the polynomial $p\colon \mathbb{N} \to \mathbb{N}$, if $\mathcal{T}(w_1, \dots, w_m)$ accepts, then it does so in at most $p(\max(|w_1|, \dots, |w_m|))$ steps. Therefore, Eq. (3) holds if and only if

$$Q_1 w_1 \in \overline{\Sigma}^{h(|w|)^2}, \dots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \dots, w_m) \text{ accepts in time } p(h(|w|)^2). \tag{4}$$

By Lemmas 4 and 5, we have

$$\mathfrak{t}(k, n) \;=\; \mathfrak{t}(k, p(2 \cdot m)) \;\geq\; p(\mathfrak{t}(k, 2 \cdot f(|w|))) \;\geq\; p(\mathfrak{t}(k, f(|w|)^2)) \;\geq\; p(h(|w|)^2). \tag{5}$$

This chain of inequalities implies that Eq. (4) holds if and only if

$$Q_1 w_1 \in \overline{\Sigma}^{h(|w|)^2}, \dots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \dots, w_m) \text{ accepts in time } \mathfrak{t}(k, n), \tag{6}$$

where, again, the right to left direction holds since, if $\mathcal{T}(w_1, \dots, w_m)$ accepts, then it does so in at most $p(h(|w|)^2)$ steps. Now, from the property (**II**) of $\mathcal{T}$, and again by Eq. (5) together with the assumption $p(\mathtt{x}) \geq \mathtt{x}$ (for all $\mathtt{x} \in \mathbb{N}$), we can extend the bounds on the words $w_1, \dots, w_m$ we consider, and conclude that Eq. (6) is equivalent to

$$Q_1 w_1 \in \overline{\Sigma}^{\mathfrak{t}(k,n)}, \dots, Q_m w_m \in \overline{\Sigma}^{\mathfrak{t}(k,n)} : \mathcal{T}(w_1, \dots, w_m) \text{ accepts in time } \mathfrak{t}(k, n). \tag{7}$$

Lastly, since for all $(w_1, \dots, w_m, \dots, w_n) \in \overline{\Sigma}^n$, $\mathcal{T}(w_1, \dots, w_m)$ and $\mathcal{T}'(w_1, \dots, w_n)$ perform the same computational steps, we conclude that Eq. (7) holds iff Eq. (2) holds. ◀

▶ **Theorem 3.** *The $\Sigma_j^{k\text{EXP}}$-prenex* TM *problem is $\Sigma_j^{k\text{EXP}}$-complete.*

Proof. (*sketch*) Let $f\colon \mathbb{N} \to \mathbb{N}$ be a polynomials, and define $h(\mathtt{x}) \stackrel{\text{def}}{=} \mathfrak{t}(k, f(\mathtt{x}))$. Consider a h-time j-alternation bounded ATM $\mathcal{M}$ (i.e. $\mathcal{M}$ alternates between existential and universal states at most $j \in \mathbb{N}_+$ times), as well as a word $w \in \Sigma^*$. We want to check whether $w \in \mathcal{L}(\mathcal{M})$. Let $m = \max(f(|w|), j) + 3$. One can easily revisit Lemma 9 with respect to $\mathcal{M}$ as above, so that the property (**IV**) of the mTM $\mathcal{T}$ is updated as follows.

$$w \in \mathcal{L}(\mathcal{M}) \text{ iff } Q_1 w_1 \in \overline{\Sigma}^{h(|w|)^2}, \ldots, Q_m w_m \in \overline{\Sigma}^{h(|w|)^2} : \mathcal{T}(w_1, \ldots, w_m) \text{ accepts},$$

where $Q_1 = \exists$, for every $i \in [1, j]$, $Q_i \neq Q_{i+1}$, and for every $i \in [j+1, m]$, $Q_i = Q_{j+1}$. In particular, now $\mathbf{Q} = (Q_1, \ldots, Q_m)$ is such that $\text{alt}(\mathbf{Q}) = j$. The proof then carries out analogously to the proof of Thm. 2, the only difference being that for the nTM $\mathcal{T}'$, we consider the quantifier prefix $(Q_1, \ldots, Q_n)$ such that, for every $i \in [m+1, n]$, $Q_i = Q_m$, in order to keep the number of alternations bounded by j. ◀

References

1 A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

2 A. Molinari. *Model checking: the interval way.* PhD thesis, University of Udine, 2019.