From Nondeterministic to Multi-Head Deterministic Finite-State Transducers

Martin Raszyk
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
martin.raszyk@inf.ethz.ch

David Basin
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
basin@inf.ethz.ch

Dmitriy Traytel
Department of Computer Science, ETH Zürich, Universitätstrasse 6, 8092, Switzerland
traytel@inf.ethz.ch

Abstract

Every nondeterministic finite-state automaton is equivalent to a deterministic finite-state automaton. This result does not extend to finite-state transducers – finite-state automata equipped with a one-way output tape. There is a strict hierarchy of functions accepted by one-way deterministic finite-state transducers (1DFTs), one-way nondeterministic finite-state transducers (1NFTs), and two-way nondeterministic finite-state transducers (2NFTs), whereas the two-way deterministic finite-state transducers (2DFTs) accept the same family of functions as their nondeterministic counterparts (2NFTs).

We define *multi-head* one-way deterministic finite-state transducers (MH-1DFTs) as a natural extension of 1DFTs. These transducers have multiple one-way reading heads that move asynchronously over the input word. Our main result is that MH-1DFTs can deterministically express any function defined by a one-way nondeterministic finite-state transducer. Of independent interest, we formulate the all-suffix regular matching problem, which is the problem of deciding for each suffix of an input word whether it belongs to a regular language. As part of our proof, we show that an MH-1DFT can solve all-suffix regular matching, which has applications, e.g., in runtime verification.

2012 ACM Subject Classification Theory of computation → Transducers

Keywords and phrases Formal languages, Nondeterminism, Multi-head automata, Finite transducers

Digital Object Identifier 10.4230/LIPIcs.ICALP.2019.127

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding This research is supported by the Swiss National Science Foundation grant "Big Data Monitoring" (167162).

1 Introduction

Finite-state automata (FAs) are a fundamental model of computation. In its simplest form, a finite-state automaton reads an input word once, from left to right, while updating its state deterministically. FAs accept the regular languages. It is well-known that neither allowing the reading head to move in both directions on the input word nor updating the state nondeterministically extends their expressiveness beyond the regular languages.

A generalization of the finite-state automata are *finite-state transducers* (FTs). FTs extend a finite-state automaton with an output tape and each transition also outputs a (possibly empty) sequence of symbols from an output alphabet. The language accepted by a finite-state transducer is a relation (transduction) between input and output words. A finite-state transducer is *functional* (f-FT) if the relation represents a function. Any deterministic finite-state transducer is functional (hence, we just write DFT instead of


$$\mathcal{L}(\text{1DFA}) = \mathcal{L}(\text{1NFA}) \subsetneq \mathcal{L}(\text{MH-1DFA}) \qquad \mathcal{L}(\text{1DFT}) \subsetneq \mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(\text{MH-1DFT})$$
$$\| \qquad\qquad\qquad\qquad \not\supseteq$$
$$\mathcal{L}(\text{2DFA}) = \mathcal{L}(\text{2NFA}) \qquad \mathcal{L}(\text{2DFT}) = \mathcal{L}(f\text{-2NFT})$$

Figure 1 The language hierarchy accepted by models of automata (left) and transducers (right).

f-DFT). For transducers, nondeterminism makes a difference: adding nondeterminism extends the expressiveness of a finite-state transducer to nonfunctional relations. One can also classify finite-state transducers as one-way or two-way (1FTs or 2FTs) depending on whether the reading head can only move forwards or in both directions on the input word.

Another generalization of finite-state automata adds multiple reading heads that move asynchronously over the input word (see, e.g., [6] for a survey). This results in an expressive computational model with problems like emptiness, finiteness, and equivalence not being semi-decidable already for two reading heads [6]. Multi-head finite-state automata induce a strict hierarchy of languages when increasing the number of reading heads [10]. The problem of simulating two-head one-way nondeterministic finite-state automata by multi-head two-way deterministic finite-state automata is equivalent to the $\mathsf{L} \stackrel{?}{=} \mathsf{NL}$ problem [9].

We combine the previous two generalizations to the notion of multi-head finite-state transducers (Section 2). We show that multi-head one-way deterministic finite-state transducers (MH-1DFTs) can simulate any functional one-way nondeterministic finite-state transducer (f-1NFT), and thereby establish inclusion between the classes of languages accepted by these models. Central to our proof is the ability of an MH-1DFT to decide for each suffix of an input word whether it belongs to a regular language (Section 3); we call this transduction *all-suffix regular matching*. Computing this transduction allows us to deterministically find an accepting computation of the nondeterministic transducer, whenever it exists (Section 4).

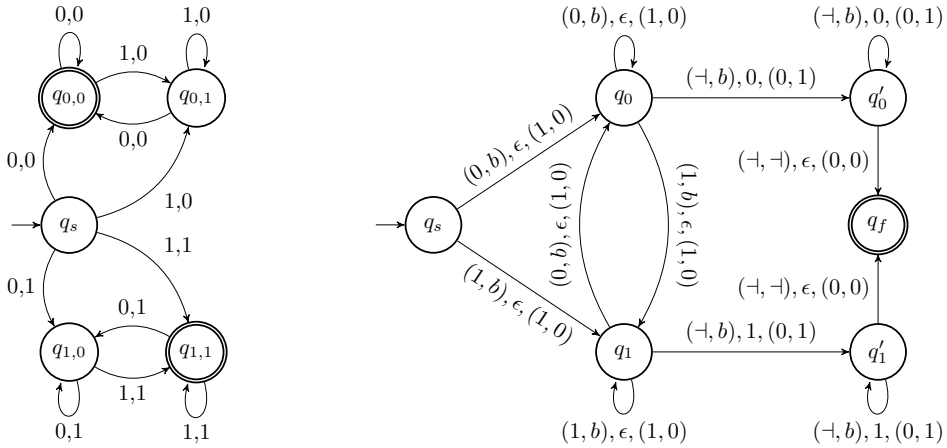
Figure 1 shows how our contributions, the transducer model and the proper inclusion of language classes, highlighted in gray, fit into the landscape of other well-studied language classes. We discuss the remaining inclusions in Section 5.

Preliminaries. Let $\mathbb{I}$ be the set of all finite intervals over the positive integers. We denote an interval $I \in \mathbb{I}$ by $[a..b] = \{x \mid a \leq x \leq b\}$. We define $[a..b) := \emptyset$, if $a \geq b$, and $[a..b) := [a..(b-1)]$, if $a < b$. Moreover, we write $[k]$ for $[1..k]$. Given a finite alphabet Σ, we denote the set of all finite words over Σ by Σ^*, the empty word by ϵ, and the length of a word $w \in \Sigma^*$ by $|w|$. Given a tuple of positions $ps \in [|w|]^{|ps|}$ of length $|ps|$, $w[ps]$ denotes $w[ps_1]w[ps_2]\dots w[ps_{|ps|}]$, i.e., the word consisting of symbols from w at the positions ps.

A *one-way deterministic finite-state automaton* (1DFA) is a tuple $A = (\Sigma, Q, q_s, Q_F, \delta)$, where Σ is the input alphabet, Q is a finite set of states, $q_s \in Q$ is the initial state, $Q_F \subseteq Q$ is the set of accepting states, and $\delta : Q \times \Sigma \to Q$ is the transition function. We extend the function δ to $\delta^* : Q \times \Sigma^* \to Q$ in the natural way. A *one-way nondeterministic finite-state automaton* (1NFA) is obtained by replacing the transition function in the definition of a 1DFA by a transition *relation* $\delta \subseteq Q \times \Sigma \times Q$.

A *one-way nondeterministic finite-state transducer* (1NFT) is derived from an underlying 1NFA by extending its transition relation with output words over an output alphabet Γ, i.e., the transition relation becomes $\delta \subseteq Q \times \Sigma \times Q \times \Gamma^*$. We extend the relation δ to $\delta^* \subseteq Q \times \Sigma^* \times Q \times \Gamma^*$ in the natural way. A 1NFT is functional if the transduction it defines is a function.

▶ **Example 1.** Figure 2a shows a f-1NFT computing the function f that maps a non-empty binary word w to $0^{|w|}$, if w ends with a zero, and $1^{|w|}$, otherwise. In the first step, the transducer guesses the last symbol and moves to a state $q_{i,j}$, where i is the guess and j is



(a) The f-1NFT from Example 1. A transition labeled by i, j represents a transition when reading the symbol i and producing the output j.

(b) The MH-1DFT from Example 3. A transition labeled by $(s_1, s_2), o, (m_1, m_2)$ represents a transition when reading the symbols (s_1, s_2), producing the output o, and advancing the reading heads by the offsets (m_1, m_2). Here, $b \in \{0, 1\}$.

Figure 2 The transducers from Examples 1 and 3.

the current (i.e., first) symbol. In the subsequent steps, the transducer updates the current state based on the current symbol. In each transition, the transducer outputs its original guess. Finally, the transducer accepts if the last symbol equals its guess.

2 Multi-Head One-Way Deterministic Finite-State Transducer

We define multi-head one-way deterministic finite-state transducers (MH-1DFTs) by adapting the definition of multi-head two-way finite-state automata [6] to multi-head one-way deterministic finite-state automata and extending the transition function with output symbols.

The following definition of an MH-1DFT formalizes a transition on a non-final state that reads a κ-tuple of symbols, enters a new state, produces some output, and advances some of the reading heads. We use a special symbol $\dashv$ to mark the end of the input tape, on the right-hand side. We further impose two conditions on the transition function. First, no reading head moves out of the input tape. Second, some reading head advances at each transition except when the new state is accepting. Without loss of generality, the transition relation δ forbids ϵ-transitions to non-final states. Hence, an MH-1DFT can only reject an input word by permitting no further transition. If δ is total, then no input word is rejected and the transduction is a total function.

▶ **Definition 2.** *A multi-head one-way deterministic finite-state transducer (MH-1DFT) is a tuple $A = (\Sigma, \dashv, \Gamma, \kappa, Q, q_s, Q_F, \delta)$, where Σ is an input alphabet, $\dashv \notin \Sigma$ is the right endmarker, Γ is an output alphabet, κ is the number of reading heads, Q is a finite set of states, $q_s \in Q$ is the initial state, $Q_F \subseteq Q$ is a set of accepting states, and $\delta : (Q \setminus Q_F) \times (\Sigma \cup \{\dashv\})^\kappa \to Q \times \Gamma^* \times \{0, 1\}^\kappa$ is the partial transition function such that:*

- *$\delta(q, es) = (q', \tilde{o}, ms)$ and $es_i = \dashv$, for any $i \in [\kappa]$, implies that $ms_i = 0$, and*
- *$\delta(q, es) = (q', \tilde{o}, \vec{0})$ implies that $q' \in Q_F$.*

A *configuration* of an MH-1DFT $A = (\Sigma, \dashv, \Gamma, \kappa, Q, q_s, Q_F, \delta)$ is a tuple $c = (w, o, q, ps) \in \mathcal{C}^A$, where $w \in \Sigma^*$ is the input word, $o \in \Gamma^*$ is the output word produced by A so far, $q \in Q$ is the current state, and $ps \in [|w| + 1]^\kappa$ are the positions of the κ reading heads on the input tape. A configuration is accepting if $q \in Q_F$. A *step* s^A is a relation on

$\mathcal{C}^A$ such that $((w, o, q, ps), (w', o', q', ps')) \in s^A$ if and only if $w = w'$, $o' = o \cdot \tilde{o}$, for some $\tilde{o} \in \Gamma^*$, and $\delta(q, w[ps]) = (q', \tilde{o}, ps' - ps)$. A *computation* of an MH-1DFT A on a word w is a maximal finite sequence of configurations $c_1, c_2, \ldots, c_l$ that starts with the initial configuration $c_1 = (w, \epsilon, q_s, \vec{1})$ and in which all pairs of consecutive configurations are contained in the step relation s^A. (The finiteness is given because $ps' - ps \neq \vec{0}$ except when $q' \in Q_F$ by Definition 2.) A computation is accepting if its last configuration c_l is accepting. Otherwise, it is rejecting.

We say that an MH-1DFT A accepts a language $L \subseteq \Sigma^* \times \Gamma^*$ if the computation on an input word w producing some output o satisfies $(w, o) \in L$ if and only if it is accepting. One can also view a language $L \subseteq \Sigma^* \times \Gamma^*$ accepted by an MH-1DFT A as a function $f : X \to \Gamma^*$, where X is the set of all input words w such that $(w, o) \in L$ for some (unique) output o and $f(w) = o$.

▶ **Example 3.** Figure 2b shows a two-head MH-1DFT computing the function f from Example 1. Initially, the first reading head reads the entire input word to determine the last symbol. Subsequently, the second reading head outputs the last symbol for each input symbol it reads. Once the second reading head has arrived at the right endmarker $\dashv$, the MH-1DFT accepts without advancing any reading head. The empty word ϵ is not mapped to any output as there is no transition from the initial state q_s when reading $(\dashv, \dashv)$.

3 All Suffix Regular Matching

All-suffix regular matching is the problem of deciding for each suffix of an input word whether it belongs to a regular language. More formally, for a regular language L, we define a function $t_L : \Sigma^* \to \{0, 1\}^*$ that maps a word w to a binary word $t_L(w)$ of length $|w| + 1$ such that, for any $1 \leq i \leq |w| + 1$, $t_L(w)[i] = 1$ if and only if the suffix $w[i..|w|]$ is in the language L. The last symbol in $t_L(w)$ denotes whether the empty word ϵ is in the language L.

We show that, for any regular language L, all-suffix regular matching can be solved by an MH-1DFT, i.e., there exists an MH-1DFT computing the function t_L. This construction is subsequently used in our simulation of any f-1NFT by an MH-1DFT (Section 4).

3.1 Informal Account

Let A be a 1DFA with $|Q|$ states. A naive approach to all-suffix regular matching is to run A on each suffix of the input word, i.e., starting from every position. The MH-1DFT solving this problem must output the decisions sequentially starting from the leftmost suffix (the entire word). Suppose we already know the decision for some *past* suffixes and want to compute the decision for a *current* suffix. To reuse the decisions for the past suffixes, we can first run the automaton A again on these suffixes until we reach the current position. Now we continue to run the automaton A on the past suffixes and also run A from its starting state on the current suffix. If the state of A in the run on the current suffix equals at some point the state of A in a run on a past suffix, then we know that the decision for the current suffix is the same as the decision for that past suffix and we can output it without the need to run A further.

Our MH-1DFT keeps a list of decisions and states for past suffixes such that running A on them until the current position yields different states. The length of this list is at most $|Q|$. To compute the decision for the current suffix, our MH-1DFT uses a reading head h positioned at the current position to run A from all the stored states for past suffixes as well as from the initial state for the current suffix. The reading head h stops once the state for the current suffix equals the state for a past suffix (or the end of the input word is reached).

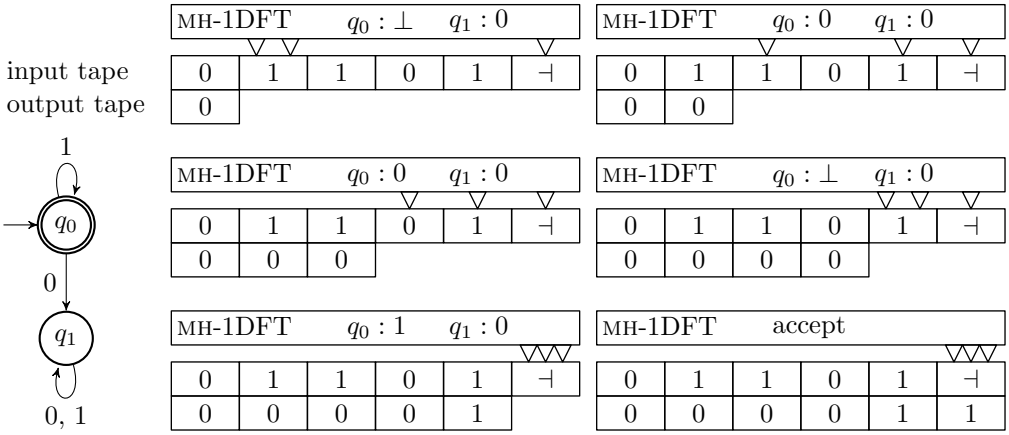


Figure 3 The automaton and configurations of the MH-1DFT from Example 4.

It remains to be shown that a finite number of reading heads suffice, independently of the input word's length. To this end, observe that whenever the reading head h moves on, the list of decisions and different states for past suffixes at that position expands (otherwise, the decision for the current suffix would have been known and h would have stopped). As the length of the list at any position is at most $|Q|$, it suffices to use $|Q| + 1$ reading heads in total. The extra reading head updates the stored states after a decision for the current suffix is computed.

▶ **Example 4.** Consider the regular language L consisting of all binary words without zero. A two-state deterministic automaton A accepting L is depicted in Figure 3. We describe a computation of our MH-1DFT with three reading heads on the input word $w = 01101$.

To compute the decision for the first suffix, one reading head reads the entire input. The remaining two reading heads advance and the decision 0 is stored (with the updated initial state $\delta(q_0, 0) = q_1$) and output (to the output tape, which is below the input tape in Figure 3).

To compute the decision for the second suffix, another reading head positioned at the second symbol advances to the fifth symbol until the two states (for the past and current suffix) become a single state q_1. The only remaining reading head advances and the decision 0 is stored (with the updated initial state $\delta(q_0, 1) = q_0$) and output.

Since the initial state q_0 is now stored with a decision, the decision for the third suffix can be immediately output, the last reading head advanced, and the stored states updated. The decision for the fourth suffix can again be immediately output, the last reading head advanced, and the stored states updated (to a single state q_1 since $\delta(q_0, 0) = \delta(q_1, 0) = q_1$). For the last suffix, the initial state q_0 is no longer stored with a decision, so a reading head reads the last symbol to compute and output the decision 1. Then the last reading head advances and the decision is stored (with an updated initial state $\delta(q_0, 1) = q_0$). Finally, as the last reading head has arrived at the right endmarker, our MH-1DFT outputs the Boolean decision 1 for the empty suffix (the initial state q_0 is accepting) and accepts.

3.2 The Multi-Head Transducer

We now formally define an MH-1DFT that solves the all-suffix regular matching problem for a regular language L. Let A be a 1DFA accepting L. Suppose A's set of states is $Q = \{q_1, q_2, \ldots, q_n\}$ and that the current suffix of an input word w starts at the position

1 $\tilde{\delta}((qbs, q), es)$:
2 $D := \{q_j \mid qbs_j \neq \bot\}; D' := \{q_{j'} \mid \exists j.\ qbs_j = (q_{j'}, \_)\}$
3 $k := |D| + 1; k' := |D'| + 1$
4 if $es_{k'} = \dashv$ then $b := q \in Q_F$
5 else if $\exists j, q_{j'}, \beta_j.\ qbs_j = (q_{j'}, \beta_j) \wedge q = q_{j'}$ then $b := \beta_j$
6 else $b := \bot$ fi
7 if $b \in \{0, 1\}$ then
8 if $es_{n+1} = \dashv$ then output b and return $\tilde{q}_f$ fi
9 $qbs' := \bot^n$
10 $\forall q_j \in D$. let $q_{j'} := \delta(q_j, es_{n+1}), (\_, \beta_j) := qbs_j$ in $qbs'_{j'} := (q_{j'}, \beta_j)$
11 let $q_{j'} := \delta(q_s, es_{n+1})$ in $qbs'_{j'} := (q_{j'}, b)$
12 output b
13 let $\tilde{k} := k + (q_s \notin D)$ in advance reading heads $\tilde{k}, \ldots, n + 1$
14 return (qbs', q_s)
15 else
16 $qbs' := qbs$
17 $\forall q_j \in D$. let $(q_{j'}, \beta_j) := qbs_j$ in $qbs'_j := (\delta(q_{j'}, es_{k'}), \beta_j)$
18 advance reading head k' fi
19 return $(qbs', \delta(q, es_{k'}))$

Figure 4 The transition function $\tilde{\delta}$ of the MH-1DFT $\tilde{A}_L$.

i. Let $\tilde{Q}_i = \{\delta^*(q_s, w[j..i)) \mid j \in [1..i)\}$ be the set of stored states for the past suffixes. For each $i' \in [i, |w| + 1]$, let $\tilde{Q}_{i,i'} = \{\delta^*(q_j, w[i..i')) \mid q_j \in \tilde{Q}_i\}$ be the states obtained by running the stored states for the past suffixes from the current position i up to the position i'. Let $\beta_{w,i}(q)$ equal one if and only if A accepts the word $w[i..|w|]$ when run from the state q.

We define our MH-1DFT $\tilde{A}_L = (\Sigma, \dashv, \Gamma, \kappa, \tilde{Q}, \tilde{q}_s, \tilde{Q}_F, \tilde{\delta})$ as follows. We set the output alphabet to $\Gamma = \{0, 1\}$ and the number of reading heads to $\kappa = |Q| + 1 = n + 1$. The set of states is $\tilde{Q} = (((Q \times \{0, 1\}) \cup \bot)^{|Q|} \times Q) \cup \{\tilde{q}_f\}$, where $\tilde{q}_f$ is a designated accepting state. With the last reading head at the position i, i.e., $ps_{n+1} = i$, and the reading head for the current suffix at the position i' (the current suffix starts at the position i), a state $(qbs, q) \in \tilde{Q}$ consists of an n-tuple qbs whose j-th component $qbs_j = (q_{j'}, \beta_{w,i}(q_j))$ stores the decision $\beta_{w,i}(q_j)$ for a past suffix corresponding to the stored state $q_j \in \tilde{Q}_i$ and the state $q_{j'}$ obtained by running A from the state q_j on $w[i..i')$. If the state q_j is not among the stored states for the current position i, then $qbs_j = \bot$. Furthermore, the state q from the state $(qbs, q) \in \tilde{Q}$ of the MH-1DFT $\tilde{A}_L$ is the state obtained by running A on the current suffix up to the position i', i.e., $q = \delta^*(q_s, w[i..i'))$. We point out that the positions i and i' themselves are not explicitly stored in the state (but $i = ps_{n+1}$ and $i' = ps_{k'}$, with k' as in Figure 4). The initial state is $\tilde{q}_s = (\bot^n, q_s)$. The set of accepting states is $\tilde{Q}_F = \{\tilde{q}_f\}$. The transition function $\tilde{\delta} : (\tilde{Q} \setminus \tilde{Q}_F) \times (\Sigma \cup \{\dashv\})^{n+1} \to \tilde{Q} \times \Gamma^* \times \{0, 1\}^{n+1}$ is defined using pseudocode in Figure 4. The transition function of the MH-1DFT from Example 4 is shown in Figure 5. We use the notation let $q_{j'} := \delta(q_s, es_{n+1})$ in $qbs'_{j'} := (q_{j'}, b)$ for *pattern-matching*, i.e., j' denotes the index of the state $\delta(q_s, es_{n+1})$ in the expression $qbs'_{j'} := (q_{j'}, b)$.

The last reading head is at the current position $i = ps_{n+1}$ and all reading heads j and j' with $j > j'$ satisfy $ps_j \leq ps_{j'}$ (the reading heads do not overtake each other). The transducer maintains the invariant that the number of reading heads beyond the position i' of the reading head for the current suffix equals $|\tilde{Q}_{i,i'}|$. The sets D and D' (Line 2) equal the set of

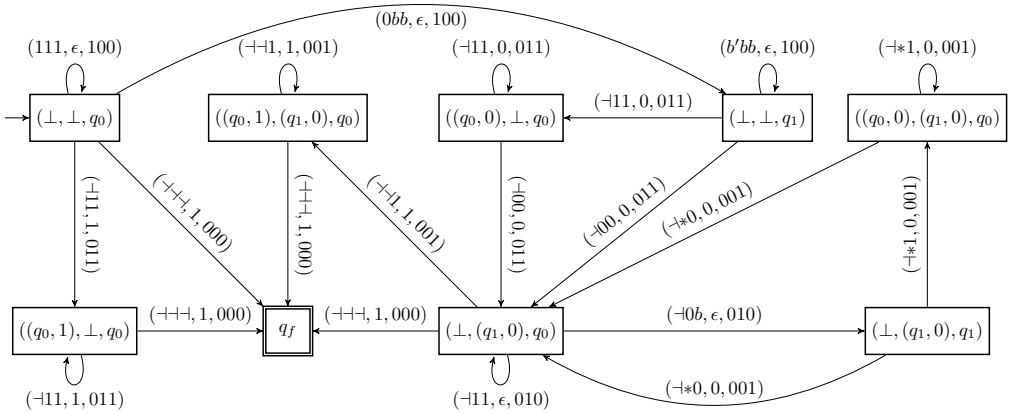


Figure 5 The transition function of the MH-1DFT from Example 4. A transition labeled by $(s_1 s_2 s_3, o, m_1 m_2 m_3)$ represents a transition when reading the symbols (s_1, s_2, s_3), producing the output o, and advancing the reading heads by the offsets (m_1, m_2, m_3). Here, $b, b' \in \{0, 1\}$ denote arbitrary symbols from the input alphabet, whereas $* \in \{0, 1, \dashv\}$ denotes an arbitrary input symbol. (Only states and transitions reachable in a computation on an input word are shown.)

stored states $\tilde{Q}_i$ and the set of states $\tilde{Q}_{i,i'}$, obtained by running A from the stored states $\tilde{Q}_i$ on $w[i..i')$. The invariant implies that $k' = |D'| + 1$ (Line 3) is the reading head for the current suffix.

The transducer $\tilde{A}_L$ first tries to determine the decision for the current suffix (Lines 4–6). If this can be determined, then $\tilde{A}_L$ updates the stored states (Lines 9–11), outputs the decision, advances all reading heads at the current position i, and sets the state for the current suffix to the initial state of A (Line 14). If A's initial state is not among the stored states $D = \tilde{Q}_i$, the reading head for the current suffix has already advanced, i.e., $\tilde{A}_L$ only needs to advance the remaining reading heads $k + 1 = \tilde{k}, \dots, n + 1$ at the current position.

If the decision for i has not been determined, then the states D' and q are updated (Lines 16–17 and 19) and the reading head for the current suffix advances (Line 18).

3.3 Proof of Correctness

To prove that the MH-1DFT $\tilde{A}_L$ computes t_L, we formulate an invariant on a configuration of $\tilde{A}_L$ that is satisfied by each configuration from a computation on an input. For the accepting state $\tilde{q}_f$, the invariant states that the output is correct: $\mathcal{I}(w, o, \tilde{q}_f, ps) \equiv (o = t_L(w))$ (C0).

For a state $(qbs, q) \in \tilde{Q}$, the invariant captures the properties of a state and the reading heads' positions mentioned previously. In addition, it states that correct output has been produced for all positions preceding the current position $i = ps_{n+1}$. In the following, we use the definitions from Lines 2–3 in Figure 4. We further define $i' = ps_{k'}$ to be the position of the reading head for the current suffix. To capture the expansion of the set of stored states by running A on the current suffix, we define $\tilde{Q}'_j = \tilde{Q}_{i,j} \cup \{\delta^*(q_s, w[i..j))\}$, for all $j \in [i..i')$, and $\tilde{Q}'_j = \tilde{Q}_{i,j}$, for all $j \in [i'..(|w| + 1)]$. Then the invariant is as follows:

$$\mathcal{I}(w, o, (qbs, q), ps) \equiv (|o| = ps_{n+1} - 1 \wedge \forall j \in [|o|].\ o_j = t_L(w)[j]) \wedge \quad \text{(C1)}$$

$$\forall j \in [n].\ \forall q_{j'}, \beta_j.\ (qbs_j = (q_{j'}, \beta_j) \implies q_{j'} = \delta^*(q_j, w[i..i')) \wedge \beta_j = \beta_{w,i}(q_j)) \wedge \quad \text{(C2)}$$

$$q = \delta^*(q_s, w[i..i')) \wedge \quad \text{(C3)}$$

$$D = \tilde{Q}_i \wedge \quad \text{(C4)}$$

$$D' = \tilde{Q}_{i,i'} \wedge \tag{C5}$$
$$\forall j \in [i..i'). \, \tilde{Q}_{i,j} \subsetneq \tilde{Q}'_j \wedge \tag{C6}$$
$$(ps_{n+1} \leq \cdots \leq ps_1) \wedge \tag{C7}$$
$$\forall j \in [i..|w|]. \, |\{h \mid ps_h > j\}| = |\tilde{Q}'_j|. \tag{C8}$$

▶ **Lemma 5.** *For any input word w, the initial configuration of $\tilde{A}_L$ satisfies the invariant, i.e., $\mathcal{I}(w, \epsilon, (\bot^n, q_s), 1^{n+1})$ holds.*

Proof. Observe that $i = i' = 1$ and $D = D' = \tilde{Q}_i = \tilde{Q}_{i,j} = \emptyset$, for all $j \in [i..|w|]$. ◀

▶ **Lemma 6.** *Let $c_1 = (w, o, (qbs, q), ps)$ and $c_2 = (w, o', q', ps')$ be two configurations of $\tilde{A}_L$ such that $\mathcal{I}(c_1)$ holds and $(c_1, c_2) \in s^{\tilde{A}_L}$. Then $\mathcal{I}(c_2)$ holds.*

Proof. We refer to Line l in Figure 4 as Ll. Furthermore, we refer to the i-th conjunct in $\mathcal{I}(c_1)$ and $\mathcal{I}(c_2)$ as Ci and Ci', respectively and to the i-th fact labeled in the proof as Fi. Let us denote by i_j, i'_j, D_j, D'_j, k_j, k'_j, ${}^j\tilde{Q}'_{j'}$, ${}^j ps$, $j \in \{1, 2\}$, the respective definitions for the configuration c_j. To derive that $\mathcal{I}(c_2)$ holds, we analyze the transition function $\tilde{\delta}$ in Figure 4.

First we show that $b \neq \bot \implies b = t_L(w)[i_1]$ (F1). If $es_{k'_1} = \dashv$, then $i'_1 = |w| + 1$, C3, and L4 imply that $b = t_L(w)[i_1]$. If L5's condition holds, then C2 and C3 imply that $b = t_L(w)[i_1]$.

Consider the case $b \in \{0, 1\}$ (L8–14). If $es_{n+1} = \dashv$, then $i_1 = i'_1 = |w| + 1$ and $b \neq \bot$ (due to $i'_1 = |w| + 1$ and L4) which with F1 implies $b = t_L(w)[i_1]$. C1 and $b = t_L(w)[i_1]$ imply C0, i.e., $\mathcal{I}(c_2)$ holds since c_2 is accepting and thus $\mathcal{I}(c_2) \equiv$ C0. Otherwise (if $es_{n+1} \neq \dashv$), we have $i_1 \leq |w|$. Because $\tilde{k} = |D_1| + 1 + (q_s \notin D_1) \leq n + 1$, the last reading head advances, i.e., $i_2 = i_1 + 1$ (F2). Now, F1, F2, and C1 imply C1'. Next we show that $|\{h \mid {}^1ps_h > i_1\}| = \tilde{k} - 1$ (F3). From C4, we have $D_1 = \tilde{Q}_{i_1}$. It follows that $|{}^1\tilde{Q}'_{i_1}| = |D_1| + (q_s \notin D_1)$. C8 then implies $|\{h \mid {}^1ps_h > i_1\}| = \tilde{k} - 1$, i.e., that precisely those readings heads at the position i_1 advanced (L13). In particular, this implies C7'. L9–11 and C4 imply $D_2 = D'_2 = \tilde{Q}_{i_1+1} \overset{\text{F2}}{=} \tilde{Q}_{i_2}$ (F4). This immediately implies C4'. Next we show that $i_2 = i'_2$ (F5). If $i_2 = |w| + 1$, then $i_2 = i'_2$ follows from C7'. Suppose that $i_2 \leq |w|$. It follows that $|\{h \mid {}^2ps_h > i_2\}| \overset{\text{F2–3,L13}}{=} |\{h \mid {}^1ps_h > i_1 + 1| \overset{\text{C8}}{=} |{}^1\tilde{Q}'_{i_1+1}|$. If $i'_1 > i_1 + 1$, we derive $|{}^1\tilde{Q}'_{i_1+1}| \overset{i'_1 > i_1+1}{=} |\tilde{Q}_{i_1+1}|$. If $i'_1 \leq i_1 + 1$, we derive $|{}^1\tilde{Q}'_{i_1+1}| \overset{\text{L5}}{=} |\tilde{Q}_{i_1+1}|$. Hence, $|\{h \mid {}^2ps_h > i_2\}| = |\tilde{Q}_{i_1+1}| \overset{\text{F2}}{=} |\tilde{Q}_{i_2}| \overset{\text{F4}}{=} D'_2$, which implies that $i_2 = i'_2$. F5 implies C6' and F5 together with L14 imply C3'. F4–5 imply C5'. L9–11, C2, F1, and F5 further imply C2'. F3 and L13 imply that those reading heads at i_1 advanced. Hence for C8', it suffices to show $|{}^2\tilde{Q}'_j| = |{}^1\tilde{Q}'_j|$, for all $j \in [(i_1 + 1)..|w|]$. We derive $|{}^2\tilde{Q}'_j| \overset{\text{F2,F5}}{=} |\tilde{Q}_{i_1+1,j}|$. If $i'_1 \leq j$, then $|\tilde{Q}_{i_1+1,j}| \overset{\text{L5},i'_1 \leq j}{=} |\tilde{Q}_{i_1,j}| \overset{i'_1 \leq j}{=} |{}^1\tilde{Q}'_j|$. If $i'_1 > j$, then $|\tilde{Q}_{i_1+1,j}| \overset{i'_1 > j}{=} |{}^1\tilde{Q}'_j|$. This completes the proof of C8'. Thus, $\mathcal{I}(c_2)$ holds in the case $b \in \{0, 1\}$.

We continue with the case $b = \bot$ (L17–19). Together with L5, $b = \bot$ implies $|D'_1| < n$. We obtain $k'_1 \leq n$ and that $i_2 = {}^2ps_{n+1} = {}^1ps_{n+1} = i_1$ (F6). Moreover, L17 implies $D_2 \overset{\text{L17}}{=} D_1 \overset{\text{C4}}{=} \tilde{Q}_{i_1} \overset{\text{F6}}{=} \tilde{Q}_{i_2}$ which yields C4'. Since no output is produced, F6 immediately yields C1'. Together with L4, $b = \bot$ implies $i'_1 \leq |w|$. Next we show that $i'_2 = i'_1 + 1$ (F7). We derive $|\{h \mid {}^1ps_h > i'_1\}| \overset{\text{C8},i'_1 \leq |w|}{=} |{}^1\tilde{Q}'_{i'_1}| \overset{\text{C5}}{=} |D'_1| \overset{\text{L3}}{=} k'_1 - 1$ (F8). The fact F8 implies that the single advancing reading head k'_1 is the first reading head at the position i'_1, which further implies C7'. If $i'_1 + 1 = |w| + 1$, then $i'_2 = i'_1 + 1$ follows from $|D'_2| \overset{\text{L17}}{\leq} |D'_1|$. If $i'_1 + 1 \leq |w|$, then $i'_2 = i'_1 + 1$ follows from $|D'_2| \overset{\text{L17}}{=} |{}^1\tilde{Q}'_{i'_1+1}|$ and $|\{h \mid {}^1ps_h > i'_1 + 1\}| \overset{\text{C8},i'_1+1 \leq |w|}{=} |{}^1\tilde{Q}'_{i'_1+1}|$. Then, L17, the facts F6–7 and C2, C4–5 imply C2' and C5'. Furthermore, L19, F6–7 and C3 imply C3'. Together with L5 and F6–7, $b = \bot$ implies that $\tilde{Q}_{i_1,i'_1} \subsetneq {}^2\tilde{Q}'_{i'_1}$ (F9).

F6–7 and F9 together with C6 yield C6'. It suffices to show C8' for $j = i'_1$; otherwise ${}^1\tilde{Q}'_j = {}^2\tilde{Q}'_j$ and the number of heads at positions $> j$ did not change. For $j = i'_1$, we derive $|\{h \mid {}^2ps_h > i'_1\}| \overset{\text{L18}}{=} |\{h \mid {}^1ps_h > i'_1\}| + 1 \overset{\text{C8}}{=} |{}^1\tilde{Q}'_{i'_1}| + 1 \overset{b=\bot,\text{L5}}{=} |{}^2\tilde{Q}'_{i'_1}|$. This completes the proof of C8'. Thus, $\mathcal{I}(c_2)$ holds in the case $b = \bot$. ◀

▶ **Theorem 7.** *For any input word w, the output produced by $\tilde{A}_L$ is $t_L(w)$.*

Proof. Let $c_1, c_2, \ldots, c_l$ be $\tilde{A}_L$'s computation on w. Lemmas 5 and 6 imply that $\mathcal{I}(c_i)$ holds for all configurations c_i, $i \in [l]$. Since $\tilde{A}_L$'s transition function $\tilde{\delta}$ is total (the transducer cannot get stuck), Definition 2 implies that c_l is an accepting configuration. The invariant $\mathcal{I}(w, o, \tilde{q}_f, ps)$ for the last configuration $c_l = (w, o, \tilde{q}_f, ps)$ then implies that $o = t_L(w)$. ◀

4 Simulation

Let A be a f-1NFT and let $L \subseteq \Sigma^* \times \Gamma^*$ be the language accepted by A. We show that there exists an MH-1DFT accepting the same language L. This establishes inclusion between the classes of languages accepted by these models. Because two-head finite-state automata strictly extend the expressiveness of one-head finite-state automata [10], the inclusion is proper.

4.1 Informal Account

To simulate the f-1NFT A with $|Q|$ states on an input w, we first check if w is accepted by A's underlying automaton. This can be done using a single head that reads the entire input. If the automaton rejects, then clearly no $(w, o) \in L$ and our MH-1DFT simulating A may also immediately reject. Otherwise there is exactly one $(w, o) \in L$, since A is functional. Hence, it suffices to follow an accepting computation of the underlying automaton and concatenate the outputs from A's transition relation to produce the output o. A problem with this straightforward approach is the nondeterminism of A that may have multiple transitions from a given state of A on the same symbol. Nonetheless, if we are able to determine a transition to a state from which A's underlying automaton accepts the rest of the input word, then we can follow it, since this transition is part of an accepting computation. So now our problem is reduced to checking from which states a 1NFA accepts a suffix of an input word. But since the language accepted by a 1NFT run from a particular state is regular, this is precisely an instance of all-suffix regular matching, for which we have already constructed an MH-1DFT in the previous section.

Our MH-1DFT simulating A follows the described approach. It tries to find an accepting computation of the underlying automaton of A starting from its initial state. To this end, our MH-1DFT runs $|Q|$ instances of the (distinct) MH-1DFTs for the regular languages accepted by the underlying finite-state automaton of A when run from one of its states. If at some point, no transition can be made from the current state to a new state from which the remainder of the input word is accepted, then our MH-1DFT rejects the input word. Otherwise, it follows one such transition (an arbitrary one if there are multiple transitions) and outputs the output word from the transition of the transducer A. Upon reaching the right endmarker of the input word, our MH-1DFT accepts if the current simulated state is accepting with respect to A.

▶ **Example 8.** We revisit the f-1NFT A from Example 1. In Example 3, we proposed an ad-hoc MH-1DFT that simulates A. Here, we show how to obtain an MH-1DFT that simulates A by following the general approach. First we construct a 1DFA for each regular language accepted by A's underlying automaton when run from one of its states. For instance, the

regular language L_s for the initial state of A contains all non-empty binary words (note that this is the set of all input words w such that $(w, o) \in L$ for some o) and can be accepted by a simple two-state 1DFA. For the states $q_{0,0}$ and $q_{0,1}$, the regular languages $L_{0,0}$ and $L_{0,1}$ contain all non-empty binary words that end with a zero. Analogously, for the states $q_{1,0}$ and $q_{1,1}$, the regular languages $L_{1,0}$ and $L_{1,1}$ contain all non-empty binary words that end with a one. Each of these four regular languages $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$ can be accepted by a simple three-state 1DFA.

Now, for each of the five regular languages (L_s, $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$), we construct an MH-1DFT solving all-suffix-pattern matching. The MH-1DFT for L_s has three reading heads and the MH-1DFTs for $L_{0,0}$, $L_{0,1}$, $L_{1,0}$, and $L_{1,1}$ have four reading heads each. Hence, the resulting MH-1DFT $\tilde{A}$ simulating A has $1 + 3 + 4 \cdot 4 = 20$ reading heads.

Let us analyze $\tilde{A}$'s computation on the input word $w = 01101$. The MH-1DFT for L_s computes that the entire input word is accepted by the underlying automaton of A (because w is non-empty). Hence, there exists an accepting computation of A on w that $\tilde{A}$ will simulate. The MH-1DFTs for $L_{0,0}$ and $L_{0,1}$ compute that neither of the first two suffixes is in either of these two regular languages (because w does not end with a zero). In contrast, the MH-1DFTs for $L_{1,0}$ and $L_{1,1}$ compute that both the first two suffixes are in both these regular languages (because w ends with a one). Now, the MH-1DFT $\tilde{A}$ simulating A must decide between the states $q_{1,0}$ and $q_{1,1}$ based on the actual transition relation of A. Since the first symbol of w is a zero, the next simulated state of A is going to be $q_{1,0}$ and the first output symbol is 1 (which is the correct guess of A about the last symbol of the input word). The rest of A's computation on w is in fact deterministic and we omit it.

4.2 The Multi-Head Transducer

We define an MH-1DFT simulating a f-1NFT $A = (\Sigma, \Gamma, Q, q_s, Q_F, \delta)$. Suppose the set of states of A is $Q = \{q_1, q_2, \ldots, q_n\}$. For each $i \in [n]$, let L_i be the regular language accepted by the 1NFA $A_i = (\Sigma, Q, q_i, Q_F, \delta')$, where $\delta' \subseteq Q \times \Sigma \times Q$ is the transition relation obtained from δ by ignoring the output word Γ^*. In particular, note that A_s is the underlying automaton of A and A_i is obtained from A_s by changing the initial state to q_i.

By Theorem 7, there exists an MH-1DFT $\tilde{A}_{L_i} = (\Sigma, \dashv, \{0,1\}, \kappa_i, \tilde{Q}^i, \tilde{q}^i_s, \tilde{Q}^i_F, \tilde{\delta}_i)$ computing the function t_{L_i}, for each $i \in [n]$. We define our MH-1DFT $\tilde{A} = (\Sigma, \dashv, \Gamma, \kappa, \tilde{Q}, \tilde{q}_s, \tilde{Q}_F, \tilde{\delta})$ simulating A as follows. We set the number of reading heads to $\kappa = 1 + \sum_{k=1}^{n} \kappa_i$ (the extra reading head is needed to simulate the actual step of A using its transition relation δ, in particular, to obtain the output word $\tilde{o} \in \Gamma^*$). The set of states is $\tilde{Q} = (Q \times (\tilde{Q}^1 \times \{\epsilon, 0, 1\}^2) \times \cdots \times (\tilde{Q}^n \times \{\epsilon, 0, 1\}^2)) \cup \{\tilde{q}_f\}$, where $\tilde{q}_f$ is a designated accepting state. The first component of a state $\tilde{q} \in \tilde{Q}$ stores the current simulated state $q = q_{i_j} \in Q$ of A. The $(i+1)$-th component of a state $\tilde{q} \in \tilde{Q}$ stores a tuple $(\tilde{q}^i, t_{i,j}, t_{i,j+1})$, where $\tilde{q}^i \in \tilde{Q}_i$ is a state of the MH-1DFT $\tilde{A}_{L_i}$, and $t_{i,j}$ and $t_{i,j+1}$ are the decision for the current and next suffix (or ϵ if they have not been computed yet). The initial state is $\tilde{q}_s = (q_s, (\tilde{q}^1_s, \epsilon, \epsilon), \ldots, (\tilde{q}^n_s, \epsilon, \epsilon))$. The set of accepting states is $\tilde{Q}_F = \{\tilde{q}_f\}$. The transition function $\tilde{\delta} : (\tilde{Q} \setminus \tilde{Q}_F) \times (\Sigma \cup \{\dashv\})^\kappa \to \tilde{Q} \times \Gamma^* \times \{0,1\}^\kappa$ is defined using pseudocode in Figure 6. We point out that the next simulated state $q_{j'}$ of A (Line 14 in Figure 6) needs not be unique and the MH-1DFT $\tilde{A}$ chooses (deterministically) an arbitrary state if it is not unique.

Recall that the only way for an MH-1DFT to reject an input is by getting stuck. In Figure 6, this is represented by the command "reject" (Lines 4 and 19). If the control flow reaches "reject", then the transition function is not defined for the respective input arguments.

```
1  δ̃((q, (q̃^1, t_{1,1}, t_{1,2}), ..., (q̃^n, t_{n,1}, t_{n,2})), es) :
2    if es_1 = ⊣ then
3      if q ∈ Q_F then return q̃_f
4      else reject fi fi
5    let i be the smallest index such that (t_{i,1}, t_{i,2}) ∉ {0,1}^2
6    let es^i be the symbols belonging to the reading heads of Ã_{L_i}
7    (q̃^i, t, ms^i) := δ̃_i(q̃^i, es^i)
8    advance reading heads belonging to Ã_{L_i} according to the offsets ms^i
9    if t ∈ {0,1} then
10     if t_{i,1} = ϵ then t_{i,1} := t
11     else t_{i,2} := t fi fi
12   if ∀i ∈ [n]. (t_{i,1}, t_{i,2}) ∈ {0,1}^2 then
13     let q_j := q
14     if ∃j' ∈ [n]. t_{j,1} = 1 ∧ t_{j',2} = 1 ∧ (q_j, es_1, q_{j'}) ∈ δ' then
15       let õ be the output of a transition δ(q_j, es_1, q_{j'}, õ, 1) in output õ
16       advance reading head 1
17       return (q_{j'}, (q̃^1, t_{1,2}, ϵ), ..., (q̃^n, t_{n,2}, ϵ))
18     else
19       reject fi fi
20   return (q, (q̃^1, t_{1,1}, t_{1,2}), ..., (q̃^n, t_{n,1}, t_{n,2}))
```

Figure 6 The transition function $\tilde{\delta}$ of the MH-1DFT $\tilde{A}$.

The first reading head simulates the only reading head of the f-1NFT A. If it reads the right endmarker $\dashv$, then the computation is accepted if and only if the current simulated state q is accepting (Lines 2–4). Otherwise, the MH-1DFT $\tilde{A}$ performs a transition of an MH-1DFT $\tilde{A}_{L_i}$ for which either $t_{i,1}$ or $t_{i,2}$ have not been computed yet (Lines 5–8). The MH-1DFT $\tilde{A}$ maintains the invariant that one such unknown decision exists. Then it updates $t_{i,1}$, $t_{i,2}$ accordingly (Lines 9–11). The definition of the transition function $\tilde{\delta}_i$ (Figure 4) implies that at most a single Boolean decision is produced in each transition.

Once all the decisions $t_{i,1}$ and $t_{i,2}$ have been computed, a step of the f-1NFT A can be simulated (Lines 13–19). If there exists a transition from the current state $q = q_j$ to a new state $q_{j'}$ from which the next suffix is accepted by A, then it is taken (Lines 15–17). The decisions for the next suffix become the decisions for the current suffix and the decisions for the next suffix become unknown (Line 17). Otherwise, the input word is rejected (Line 19).

4.3 Proof of Correctness

To prove that the MH-1DFT $\tilde{A}$ simulates the f-1NFT A, we formulate an invariant on a configuration of $\tilde{A}$ that is satisfied by each configuration from a computation on an input word. For the accepting state $\tilde{q}_f$, the invariant merely states that the input word is accepted by A and the output is correct: $\mathcal{J}(w, o, \tilde{q}_f, ps) \equiv (w, o) \in L$ (D0).

For a state $(q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})) \in \tilde{Q}$, the invariant captures the properties of a state mentioned previously. To express them, it uses the invariant $\mathcal{I}$ on the configurations of the MH-1DFT $\tilde{A}_{L_i}$. In addition, it guarantess the existence of the index i from Line 5 in Figure 6 and that the simulation does not get stuck after it successfully performs its first

step. We denote the positions of the reading heads belonging to $\tilde{A}_{L_i}$, $i \in [n]$, by ps^i.

$$\mathcal{J}(w, o, (q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})), ps) \equiv (q_s, w[1..ps_1), q, o) \in \delta^* \wedge \tag{D1}$$

$$\forall i \in [n].\ \exists ts.\ (|ts| = ps_1 - 1 \wedge \mathcal{I}(w, ts \cdot t_{i_1} \cdot t_{i,2}, \tilde{q}^i, ps^i)) \wedge \tag{D2}$$

$$\exists i \in [n].\ (t_{i,1}, t_{i,2}) \notin \{0,1\}^2 \wedge \tag{D3}$$

$$(ps_1 > 1 \Longrightarrow \exists q_f \in Q_F.\ \exists o'.\ (q, w[ps_1..|w|], q_f, o') \in \delta^*) \tag{D4}$$

▶ **Lemma 9.** *For any input word w, the initial configuration of $\tilde{A}$ satisfies the invariant, i.e., $\mathcal{I}(w, \epsilon, (q_s, (\tilde{q}_s^1, \epsilon, \epsilon), \ldots, (\tilde{q}_s^n, \epsilon, \epsilon)), 1^\kappa)$ holds.*

Proof. The invariant follows directly from Lemma 5. ◀

▶ **Lemma 10.** *Let $c_1 = (w, o, (q, (\tilde{q}^1, t_{1,1}, t_{1,2}), \ldots, (\tilde{q}^n, t_{n,1}, t_{n,2})), ps)$ and $c_2 = (w, o', q', ps')$ be two configurations of $\tilde{A}$ such that $\mathcal{J}(c_1)$ holds and $(c_1, c_2) \in s^{\tilde{A}}$. Then $\mathcal{J}(c_2)$ holds.*

Proof. We refer to Line l in Figure 6 as Ll (e.g., L7 denotes Line 7). Furthermore, we refer to the i-th conjunct from $\mathcal{J}(c_1)$ and $\mathcal{J}(c_2)$ as Di and Di', respectively and to the i-th fact labeled in the proof as Fi. To derive that $\mathcal{J}(c_2)$ holds, we analyze the transition function $\tilde{\delta}$ in Figure 6.

If $es_1 = \dashv$, then $ps_1 = |w| + 1$ and D1 implies $(q_s, w[1..|w|], q, o) \in \delta^*$ (F1). The fact that a step from c_1 to c_2 was taken implies that $q \in Q_F$ (otherwise, the transition from c_1 would be undefined, due to L3–4). The fact F1 together with $q \in Q_F$ imply D0, i.e., $\mathcal{J}(c_2)$ holds as c_2 is accepting and thus $\mathcal{J}(c_2) \equiv (w, o) \in L \equiv$ D0.

Suppose that $es_1 \neq \dashv$. Conjunct D3 implies that the index from L5 is well-defined. Lines L5–11, Conjunct D2, and Lemma 6 imply D2' after L11. Furthermore, L16–17 imply D2' also if the branch L15–17 is reached. If the branch L13–19 is not reached, then D1, D4 immediately imply D1', D4', and L12 implies D3'. Otherwise, the branch L15–17 must be reached (otherwise, the transition from c_1 would be undefined, due to L19). Then Conjunct D1 and Lines L14–17 imply D1'. Furthermore, Lines L12 and L17 directly imply D3' (note that reaching L19 would make the transition from c_1 to c_2 undefined, which is a contradiction). Finally, Lines L13–14 and conjuncts D1–2 together with the definition of the invariant $\mathcal{I}$ imply D4'. ◀

▶ **Theorem 11.** *For any f-1NFT A, there exists an* equivalent MH-*1DFT $\tilde{A}$, i.e., both A and $\tilde{A}$ accept the same language.*

Proof. We again refer to Line l in Figure 6 as Ll. Let $L_{\tilde{A}}$ denote the language accepted by $\tilde{A}$. Let $c_1, c_2, \ldots, c_l$ be the computation of $\tilde{A}$ on an input word w. We refer to the i-th conjunct from $\mathcal{J}(c_l)$ as Di. Let o be the output of the computation of $\tilde{A}$ on w. Lemmas 9 and 10 imply that $\mathcal{J}(c_i)$ holds for all configurations c_i, $i \in [1..l]$.

If the computation is accepting (i.e., $(w, o) \in L_{\tilde{A}}$), then $\mathcal{J}(c_l)$ implies that $(w, o) \in L$. Moreover, since A is functional and $\tilde{A}$ deterministic, there exists no $o' \neq o$ such that $(w, o') \in L$ or $(w, o') \in L_{\tilde{A}}$. If the computation is rejecting, then the transition from c_l is undefined due to L4 or L19. If L4 is reached, then $ps_1 = 1$ (otherwise, D4 would imply $q \in Q_F$, which is a contradiction). With D1 and L2, this implies that $q = q_s$ and $w = \epsilon$. The facts that $q = q_s$, $w = \epsilon$, and $q \notin Q_F$ (due to L3–4) imply that the input word w is rejected by A (i.e., no $(w, o) \in L$). Moreover, no $(w, o) \in L_{\tilde{A}}$, since the deterministic computation of $\tilde{A}$ on w is rejecting.

If L19 is reached, then again $ps_1 = 1$ (otherwise, the branch L15–17 would have been taken by D4 and D2). Then L14 and D2 imply that the input word is rejected by A (i.e., no $(w, o) \in L$). Moreover, no $(w, o) \in L_{\tilde{A}}$, since the deterministic computation of $\tilde{A}$ on w is rejecting. ◀

5 Discussion and Related Work

We review results on the expressiveness of transducer models (as depicted in Figure 1) and connections to a practical application.

Expressiveness of Related Formalisms. It is well-known that neither nondeterminism nor a two-way reading head extends the expresiveness of a one-head finite-state automaton beyond the regular languages [7]. Formally, $\mathcal{L}(\text{1DFA}) = \mathcal{L}(\text{1NFA}) = \mathcal{L}(\text{2DFA}) = \mathcal{L}(\text{2NFA})$. But adding reading heads does make a difference: in fact, there is a strict hierarchy of languages accepted by finite-state automata when increasing the number of reading heads [10]. Formally, $\mathcal{L}(\text{2NFA}) \subsetneq \mathcal{L}(\text{MH-1DFA})$. By viewing a finite-state automaton as a functional finite-state transducer that does not produce any output, this further implies that $\mathcal{L}(\text{MH-1DFT}) \not\subseteq \mathcal{L}(f\text{-2NFT})$. Theorem 11 implies that $\mathcal{L}(f\text{-1NFT}) \subseteq \mathcal{L}(\text{MH-1DFT})$. This yields the proper inclusion $\mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(\text{MH-1DFT})$.

Let us consider the function f that maps a non-empty binary word w to $0^{|w|}$, if w ends with a zero, and $1^{|w|}$, otherwise. The function f can be computed by the f-1NFT from Example 1. Nevertheless, f cannot be computed by a 1DFT. Intuitively, a 1DFT cannot start producing any output before seeing the last symbol of the input word; but it cannot remember the input word's length needed to produce the output. We conclude that the expressiveness of f-1NFTs strictly extends that of 1DFTs, i.e., $\mathcal{L}(\text{1DFT}) \subsetneq \mathcal{L}(f\text{-1NFT})$.

Now consider the function $w \mapsto w^R$ that maps a binary word to its reverse. It can be computed by a f-2NFT that first moves its reading head to the end of the input word and then reads the word backwards while outputting its symbols in the reversed order (in fact, this transducer behaves deterministically). Nevertheless, $w \mapsto w^R$ cannot be computed by a f-1NFT [5]. We conclude that the expressiveness of f-2NFTs strictly extends the expressiveness of f-1NFTs, i.e., $\mathcal{L}(f\text{-1NFT}) \subsetneq \mathcal{L}(f\text{-2NFT})$. Surprisingly, adding nondeterminism to functional two-way finite-state transducers does not extend their expressiveness [4], i.e., $\mathcal{L}(\text{2DFT}) = \mathcal{L}(f\text{-2NFT})$. We further conjecture that the function $w \mapsto w^R$ cannot be computed by a MH-1DFT either, and thus the languages accepted by MH-1DFT and 2DFT are incomparable.

We also conjecture that MH-1DFTs are closed under composition, i.e., whenever $f : X \to Y$ and $g : Y \to Z$ are computed by some MH-1DFTs, then there exists an MH-1DFT computing $g \circ f : X \to Z$. Such a composition result could give rise to a more modular construction of MH-1DFTs from f-1NFTs, which would recast our MH-1DFT from Figure 6 as a composition.

Monitoring. The use of MH-1DFT and all-suffix regular matching has applications to *monitoring* (also called *runtime verification*), which is the problem of checking the compliance of an event stream, at each position in the stream, to a policy formalized in a specification language. Our recent work [8] provides evidence that exploiting multiple one-way reading heads significantly improves the efficiency of monitoring policies formalized in metric temporal logic (MTL). Since MTL is less expressive than (timed) regular expressions [3], the monitor from [8] does not implement the proposed solution to all-suffix regular matching. Moreover, its underlying transducer's space complexity depends logarithmically on the input length.

6 Conclusion

We proposed multi-head finite-state transducers as a combination of multi-head finite-state automata and finite-state transducers. We showed that multiple one-way reading heads can replace nondeterminism on functions computable by finite-state transducers. The key insight is that multiple one-way reading heads suffice to solve the all-suffix regular matching problem.

As future work, we plan to use the MH-1DFT construction for all-suffix regular matching to implement an efficient monitor for timed regular expressions [1], which are strictly more expressive than metric temporal logic supported by our multi-head monitor [8]. The problem of all-suffix regular matching has already inspired another monitor of ours [2]. In that monitor, as soon as two past suffixes yield the same state of the automaton, the equivalence between them is output (namely, the monitor outputs that the positions associated with the suffixes will have the same verdict); outputting the actual Boolean value for the equivalent positions is postponed, potentially until the input's end. We envision designing an MH-1DFT that outputs an explicit Boolean value for each position in the input word in the order of their appearance.

References

1 Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, 2002.

2 David Basin, Bhargav Bhatt, Srđan Krstić, and Dmitriy Traytel. Almost Event-Rate Independent Monitoring. *Form. Meth. Sys. Des.*, 2019 (published online February 2019).

3 Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. *Inf. Comput.*, 208(2):97–116, 2010.

4 Joost Engelfriet and Hendrik Jan Hoogeboom. Two-Way Finite State Transducers and Monadic Second-Order Logic. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 1999*, volume 1644 of *LNCS*, pages 311–320. Springer, 1999.

5 Emmanuel Filiot, Olivier Gauwin, Pierre-Alain Reynier, and Frédéric Servais. From Two-Way to One-Way Finite State Transducers. In *LICS 2013*, pages 468–477. IEEE Computer Society, 2013.

6 Markus Holzer, Martin Kutrib, and Andreas Malcher. Complexity of multi-head finite automata: Origins and directions. *Theor. Comput. Sci.*, 412(1-2):83–96, 2011.

7 Michael O. Rabin and Dana S. Scott. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

8 Martin Raszyk, David Basin, Srđan Krstić, and Dmitriy Traytel. Multi-head monitoring of metric temporal logic. In Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza, editors, *ATVA 2019*, LNCS. Springer, 2019. To appear. `http://people.inf.ethz.ch/trayteld/papers/atva19-hydra/hydra.pdf`.

9 Ivan Hal Sudborough. On Tape-Bounded Complexity Classes and Multihead Finite Automata. *J. Comput. Syst. Sci.*, 10(1):62–76, 1975.

10 Andrew Chi-Chih Yao and Ronald L. Rivest. k+1 Heads Are Better than k. *J. ACM*, 25(2):337–340, 1978.