Weight-Reducing Hennie Machines and Their Descriptional Complexity*

Daniel Průša

Czech Technical University, Faculty of Electrical Engineering,
Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
prusapa1@cmp.felk.cvut.cz

Abstract. We present a constructive variant of the Hennie machine. It is demonstrated how it can facilitate the design of finite-state machines. We focus on the deterministic version of the model and study its descriptional complexity. The model's succinctness is compared with common devices that include the nondeterministic finite automaton, two-way finite automaton and pebble automaton.

Keywords: Finite automata, two-way automata, Hennie machine, descriptional complexity.

1 Introduction

Regular languages are naturally defined via finite automata. Various extensions of this basic model preserve its recognition power (nondeterminism, two-way movement, use of a pebble). However, measured in the number of states or transitions, they provide a more economic mean of a language description.

Great attention has been paid to the cost of transformations among the models [15,8,13,4]. From our point of view, the related studies usually do not cover automata which, in some restricted way, can rewrite the content of the tape. A two-way finite automaton with write-once tracks presented by Durak [2] is one of the few.

A very general representative of rewriting devices we have in mind is a Hennie machine. It is a bounded, single-tape Turing machine performing constantly many transitions over each tape field, independently on the input's length. Hennie proved that the machine recognizes only regular languages and also generalized this result on any Turing machine working in linear time [6]. Hartmanis later showed that even time $\mathcal{O}(n \log n)$ still leads to recognition of regular languages [5]. Only going beyond this time complexity allows to recognize a non-regular language.

Generality of the model causes some unpleasant properties. It is undecidable whether a given Turing machine is a Hennie machine. Moreover, there is no computable function bounding the blowup in states when transforming to a finite

* The author was supported by the Grant Agency of the Czech Republic under the project P103/10/0783.

A.-H. Dediu et al. (Eds.): LATA 2014, LNCS 8370, pp. 553–564, 2014.

automaton. The aim of this paper is to define a reasonable constructive subclass of deterministic Hennie machines and study its descriptional complexity. To achieve this, a weight-reducing property is utilized – each transition is required to lower a weight of the scanned symbol.

A two-dimensional variant of the weight-reducing Hennie machine has already been introduced in [10]. Relation to other two-dimensional models with respect to the recognition power was investigated there.

The paper is structured as follows. In the next section we demonstrate that the possibility of rewriting can greatly facilitate the design of a finite automaton. Then we define the weight-reducing Hennie machine and compare it with the original one. Section 4 focuses on the descriptional complexity. The relation to other automata is studied. The paper closes with a short summary and some open problems in Section 5.

2 Motivation

Consider a system of objects aligned in a row. The leftmost one is a transmitter that sends a signal and the rightmost one is a receiver that waits for it. The signal can be in two states, it is either *normal* or *amplified*. The objects between the transmitter and receiver are of three types: *silencer*, *reflecting silencer* and *amplifier*. The silencer changes the first amplified signal it receives to normal. After that it becomes passive and, from that time, it does not influence incoming signals at all. The reflecting silencer behaves like the silencer, but in addition, it reflects the first received amplified signal back. Finally, the amplifier changes a normal signal to amplified, however, a signal (normal or amplified) can pass trough it only k times. If it passes there $k + 1$-st time, the amplifier is burned and the signal is lost.

Now, given such a system, the question is whether a normal signal sent by the transmitter will get to the receiver. Examples are shown in Figure 1.

Fig. 1. Two systems composed of amplifiers (a), silencers (s) and reflecting silencers (r). On the left, the signal passes to the receiver provided that $k \geq 3$, on the right, it returns back to the transmitter. The signal is normal in the dashed parts of the trajectory and amplified in the solid parts.

Each system is encoded by a string over $\Sigma = \{a, r, s\}$. It is clear that systems complying with the condition can be recognized by a finite automaton. However, if we try to design such an automaton, even for $k = 3$, we find that it is not entirely easy task, despite the fact there is a deterministic solution with 8 states, depicted in Figure 2 (one "dead" state is hidden there).

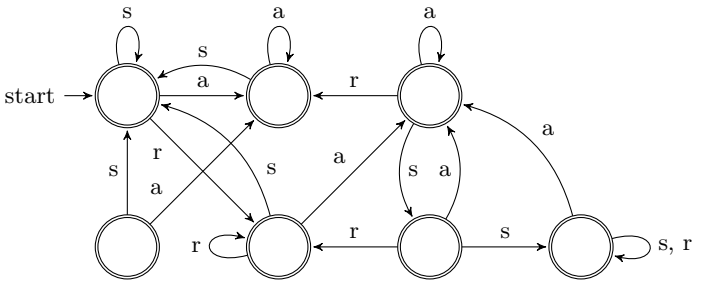

Fig. 2. A deterministic finite automaton accepting codes of systems in which the signal reaches the receiver for $k = 3$. All the displayed states are accepting. Missing transitions are heading to a hidden dead state which is rejecting.

One the other hand, it is not difficult to construct a Turing machine that tracks the signal, marks objects that become inactive and counts in amplifiers how many times the signal passed trough them. In fact, the machine will be a Hennie machine. This demonstrates that a usage of tape rewriting could simplify the design of a finite automaton, by the assumption, we have an automatic procedure that converts the rewriting device to it.

The following table lists sizes of minimal deterministic finite automata accepting our systems for greater (odd) values k. It suggests that, in this case, the dependency is bounded by an exponential function.

k	3	5	7	9	11	13	15
states	8	16	30	56	102	188	346

3 Weight-Reducing Hennie Machines

Given an input string w, a *bounded* Turing machine M operates on a tape which initially stores $\vdash w \dashv$, where $\vdash$, $\dashv$ are special end markers, not contained in the working alphabet of M. Whenever the machine reaches $\vdash$ or $\dashv$ it immediately moves the head back and does not rewrite the end marker. If w is the empty string, the head scans $\vdash$ in the initial configuration. The computation ends after performing one transition. A computation is accepting if M finishes in a final state and rejecting when it is not finite or M terminates due to non-applicability of any instruction. The language accepted by M is denoted as $L(M)$.

We say that a bounded Turing machine M a is a *Hennie machine* if there is a constant k limiting the number of transitions performed over any tape field during any computation. Let $\nu(M)$ denote the smallest such k for M.

Proposition 1 ([6]). *If M is a Hennie machine, $L(M)$ is a regular language.*

Theorem 2. *It is undecidable if a Turing machine T is a Hennie machine.*

Proof. The halting problem reduces to the stated question. Assume T has its working tape unbounded in one direction. Let w be an input and $t(w)$ the length

of the computation of T for w. It is possible to construct a bounded Turing machine M fulfilling these two conditions.

- If T halts on w, M is a Hennie machine with $\nu(M) \leq t(w)$.
- If T does not halt on w, for any input v such that $|v| \geq |w|$, M either does not halt or it visits the leftmost tape field at least $|v|$ times.

The idea is to take the tape fields storing an input v as the only space available for M to perform the simulation of T on w. Thus, M memorizes w in states, writes it on the tape and erases all remaining symbols of v (if $|v| < |w|$, it halts). After that it simulates T. If the simulation exceeds the space $|v|$, M visits the leftmost tape field $|v|$ times and halts. □

A *weight-reducing* Hennie machine is a bounded Turing machine equipped by a weight function defined on working symbols. A transition has to lower the weight of the scanned symbol. A formal definition follows.

Definition 3. A weight-reducing Hennie machine *is a tuple*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu) \text{ where}$$

- Q *is a finite set of states,*
- Γ *is a working alphabet,*
- $\Sigma \subseteq \Gamma$ *is an input alphabet,*
- $q_0 \in Q$ *is the initial state,*
- $Q_F \subseteq Q$ *is a set of final states,*
- $\delta : (Q \setminus Q_F) \times (\Gamma \cup \{\vdash, \dashv\}) \to 2^{Q \times (\Gamma \cup \{\vdash, \dashv\}) \times \{\leftarrow, 0, \rightarrow\}}$ *is a transition relation, with the set of the head movements* $\{\leftarrow, 0, \rightarrow\}$*,*
- $\mu : \Gamma \to \mathbb{N}$ *is a weight function.*

Moreover, the following properties are fulfilled:

- $(Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *is a bounded Turing machine,*
- *the transition relation is* weight-reducing*:*

$$\text{for all } q, q' \in Q, d \in \{\leftarrow, 0, \rightarrow\}, a, a' \in \Gamma : \quad (q', a', d) \in \delta(q, a) \Rightarrow \mu(a') < \mu(a).$$

M *is* deterministic *(det-wr Hennie machine) iff* $|\delta(q, a)| \leq 1$ *for all* $q \in Q$ *and* $a \in \Gamma \cup \{\vdash, \dashv\}$*.*

Observe that the weight-reducing property of δ can be easily algorithmically verified and that $\nu(M) \leq |\Gamma|$.

Lemma 4. *Let* $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_F)$ *be a Hennie machine. There is a weight-reducing Hennie machine* A *such that* $L(A) = L(M)$ *and the working alphabet of* A *has no more than* $(\nu(M) + 1)|\Gamma|$ *symbols. Moreover, if* M *is deterministic, then* A *is deterministic as well.*

Proof. Denote $k = \nu(M)$. Define $A = (Q, \Sigma, \Gamma', \delta', q_0, Q_F, \mu)$, where $\Gamma' = \Sigma \cup (\Gamma \times \{1, \ldots, k\})$ and each instruction $(q, a) \to (q', a', d)$ from δ where $a \in \Gamma$ is represented in δ' by the following instruction set:

$$\begin{aligned} (q, a) &\to (q', (a', 1), d), \\ (q, (a, i)) &\to (q', (a', i+1), d) \quad \forall i \in \{1, \ldots, k-1\}. \end{aligned}$$

Finally, define

$$\begin{aligned} \mu(a) &= k+1 \qquad \forall a \in \Sigma, \\ \mu((a, i)) &= k+1-i \quad \forall (a, i) \in \Gamma \times \{1, \ldots, k\}. \end{aligned}$$

It is easy to see that $L(A) = L(M)$ and that every deterministic δ produces deterministic δ'. □

When designing a weight-reducing Hennie machine accepting some language L, it suffices to describe a Hennie machine M accepting L and derive $\nu(M)$. Then M can be transformed by Lemma 4. This will be applied in constructions presented in Section 4.

For a Turing machine T, it is natural to count the number of its transitions to measure the size of its description. It would not make much sense to count solely states, since each Turing machine has an equivalent with only two active states [14]. We apply this measure based on transitions also to Hennie machines.

Let $\tau(T)$ denote the number of transitions of T. Let Q be the set of states of T and Γ be its working alphabet. Note that $\tau(T) = \mathcal{O}(|Q|^2|\Gamma|)$. If T is deterministic, then $\tau(T) = \mathcal{O}(|Q||\Gamma|)$. If a Hennie machine M is transformed by Lemma 4 to a weight-reducing Hennie machine M', then $\tau(M') \leq \nu(M)\,\tau(M)$.

4 Results on Descriptional Complexity

In this section we give results on trade-offs between a det-wr Hennie machine and common models including a deterministic finite automaton (1DFA), nondeterministic finite automaton (1NFA), their two-way generalizations (2DFA, 2NFA), alternating finite automaton (1AFA) and deterministic one-pebble automaton.

Theorem 5. *There is no recursive function bounding the blowup in transitions when transforming a deterministic Hennie machine to a* 1DFA.

Proof. We utilize busy beaver function $\mathrm{S}(n)$, defined as the maximum number of steps performed by a halting 2-state Turing machine with a binary working alphabet when started over a blank tape. It is known that $\mathrm{S}(n)$ is noncomputable and grows asymptotically faster than any computable function [11].

For each $n > 0$, let w_n be the string over $\Sigma = \{a\}$ of the length $\mathrm{S}(n)$. Moreover, define one-string languages $L_n = \{w_n\}$. Each L_n is accepted by a Hennie machine with $\mathcal{O}(n)$ states and $\mathcal{O}(1)$ working tape symbols. The machine works as follows. Simulate an n-state busy beaver. Whenever the beaver performs an i-th step, mark the i-th tape field and return to the original position. Accept if and only if the simulation marks all the input tape fields and does not attempt to mark the right-end marker $\dashv$.

On the other hand, a 1DFA accepting L_n has at least $|w_n| = \mathrm{S}(n)$ states. □

The following theorem is implied by results in [6]. We present a simplified proof for deterministic machines, because the procedure is essential for an automatic conversion to 1DFA.

Theorem 6. *For each n-state, m-working symbol det-wr Hennie machine, there is a $2^{2^{\mathcal{O}(m \log n)}}$-state* 1DFA *accepting the same language.*

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_F, \mu)$ be a det-wr Hennie machine such that $|Q| = n$ and $|\Gamma| = m$. Assume M can never reenter its initial state q_0. Moreover, assume M can reach a final state in Q_F only by that transition which moves the head from the right-end marker $\dashv$ to the preceding input field. Any det-wr Hennie machine can be modified to fulfill these restrictions by adding a constant number of states and working symbols.

Consider a sequence $R = ((r_1, d_1), \ldots, (r_\ell, d_\ell))$ where each $r_i \in Q$ and $d_i \in \{\leftarrow, 0, \rightarrow\}$. Such a sequence records in which states M performs transitions over some tape field, possibly including the last state in which M terminates. A pair (r_i, d_i) says that the i-th transition over the field starts in state r_i. Moreover, d_i indicates, which head movement precedes reaching r_i in the field. We can see R as a variant of the crossing sequence, however, defined over a tape field, not over the border between two neighboring fields. Let $\mathcal{R}$ be the set of all such nonempty sequences of length at most $m + 1$. Since $\nu(M) \leq m$, this covers all those sequences emerging during the computation of M.

Construct a 1NFA A with the set of states $\mathcal{R}$. Let A be in a state $R = ((r_1, d_1), \ldots, (r_\ell, d_\ell))$ and the scanned symbol be a. Define transitions by the following rules.

- R is initial iff $(r_1, d_1) = (q_0, 0)$. In such a state, A checks if R is consistent with the behavior of M over the prefix $\vdash a$.
- If R is not initial, M only checks whether it is consistent with the scanned symbol a. It is also required that $d_1 = \rightarrow$, because the field is reached first time after moving the head there from the left neighboring field.
- The next state $R' \in \mathcal{R}$ is nondeterministically guessed. It has to be consistent with transitions in R that move the A's head to the right.
- R is accepting iff $r_\ell \in Q_F$. A can enter such a state only if R is consistent with the behavior of M over the suffix $a \dashv$.

A has $|\mathcal{R}| = \sum_{i=1}^{m+1} (3n)^i = 2^{\mathcal{O}(m \log n)}$ states. If it is transformed to a minimal 1DFA, the desired automaton is obtained. □

The next step is to prove that the trade-off between a det-wr Hennie machine and a 1DFA is really double exponential. For $n \in \mathbb{N}$, define a language B_n over $\{0, 1, \$\}$ consisting of strings $v_1\$v_2\$\ldots\$v_j$ where $j \in \mathbb{N}$, every $v_i \in \{0,1\}^*$, $|v_j| \leq n$ and there is $\ell < j$ such that $v_\ell = v_j$.

Informally, every string in B_n is a sequence of binary substrings which are separated by the symbol \$. Moreover, the last substring is of length at most n and it is a copy of one of the preceding substrings. For example,

$$v_1\$v_2\$v_3\$v_4\$v_5\$v_6 = 11\$0101110\$011\$0011\$001\$011 \in B_4$$

since $v_3 = v_6$ and $|v_6| \leq 4$.

Lemma 7. *Every B_n is accepted by a det-wr Hennie machine with $\mathcal{O}(1)$ states and $\mathcal{O}(n)$ working symbols.*

Proof. Let $\Sigma = \{0, 1, \$\}$. To accept B_n, we will construct a det-wr Hennie machine A with the working alphabet $\Gamma = \Sigma \cup \{0, 1, \$, x, f\} \times \{1, \ldots, 2n\}$. The numeric part i of each $(a, i) \in \Gamma$ is used to count the number of transitions over a tape field. In the next description, we omit technical details on it and focus rather on the role of elements in $\{0, 1, \$, x, f\}$.

Let $w \in \Sigma^*$ be an input string. Write it as $w = v_1\$v_2\$\ldots\$v_j$ where each $v_j \in \{0,1\}^*$. Let $s = |v_j|$ and $v_j = a_1 \ldots a_s$. A iterates trough symbols $a_s, a_{s-1}, \ldots, a_1$ and for each of them performs a traversal trough the tape to detect in which substrings v_i the symbol appears at the same position from the back. Specifically, the first iteration starts by moving the head to the right end of w. Then, a_s is memorized in the control unit and replaced on the tape by x. After that, the head moves leftwards until it scans the left-end marker. Whenever it enters a new binary substring v_j (i.e., it has passed symbol \$), it checks if its last symbol equals a_s. If so, it is replaced by x, otherwise it is replaced by f (indicating that the check has failed). When the left-end marker is reached, A moves the head to the right end of w and starts the next iteration by locating a_{s-1}, which is the first tape field leftwards not marked by x. The initial tape and the outcome of all iterations are illustrated by the following example.

$$\begin{array}{c} 11\$0101110\$011\$0011\$001\$011 \\ 1x\$010111f\$01x\$001x\$00x\$01x \\ xx\$01011xf\$0xx\$00xx\$0fx\$0xx \\ xx\$0101fxf\$xxx\$0xxx\$xfx\$xxx \end{array}$$

A accepts w during the last iteration iff there is some v_ℓ ($\ell < n$) whose all symbols have been rewritten by x, including one symbol rewritten during the last iteration (this guaranteers that $|v_\ell| = |v_n|$). In the example above, A accepts since it rewrites all three symbols in v_3, each of them in one of the three iterations.

Finally, if $|v_j| > n$, A terminates and rejects, since it cannot reduce weights of symbols during the $n+1$-st iteration. □

Lemma 8. *Every* 1DFA *accepting B_n has at least 2^{2^n} states.*

Proof. Encode each subset of $\{0,1\}^n$ as a sequence of its elements separated by the symbol \$. There are 2^{2^n} such subsets. Let w_1 and w_2 encode two different subsets and let u be a binary substring represented in w_1 but not in w_2 (or vice versa). Then, $w_1\$u \in B_n$ and $w_2\$u \notin B_n$ (or vice versa), hence $\$u$ is a distinguishing extension and, by Myhill-Nerode theorem, each 1DFA accepting B_n has at least 2^{2^n} states. □

We use the following proposition to show $2^{\Omega(\sqrt{n})}$ trade-off in transitions when transforming det-wr Hennie machine to 2NFA.

Proposition 9 ([7]). *Let L be a finite language over an unary alphabet accepted by a* 2NFA *with n states. The longest string in L has length at most $n+2$.*

For $n \in \mathbb{N}$, define $U_n = \{a^{2^n}\}$ – a one-string language over the unary alphabet $\Sigma = \{a\}$. Every 2NFA accepting U_n has $\Omega(2^n)$ states.

Lemma 10. *There is a $\mathcal{O}(n)$-state, $\mathcal{O}(n)$-working symbol det-wr Hennie machine accepting U_n.*

Proof. For $n \in \mathbb{N}$, we first construct a deterministic Hennie machine M accepting U_n. It will have $\mathcal{O}(n)$ states and the working alphabet $\Gamma = \{a, 0, 1\}$. To process a given input $w \in \{a\}^*$, M uses a binary counter of length n. At the beginning, the counter is represented in the first n tape fields and it is initialized by value n. The least significant bit is in its leftmost field, see Figure 3(a). To perform the initialization, M memorizes in states the binary representation of n and fills the counter in each step accordingly. In the subsequent phase, it repeatedly increases the counter by one and simultaneously shifts its representation on the tape by one field to the right (the former leftmost field of the counter is rewritten to a). This guaranteers that the position of the right end of the counter representation always equals the counter's value. Hence, M easily checks if the counter has been increased to value $2^n - 1$ just when there is exactly one tape field between the right end of the counter and the right-end marker $\dashv$, see Figure 3(b). The counter's increment as well as its shift is done by one traversal trough the related tape fields. This means that $\nu(M) = \mathcal{O}(n)$, hence, by Lemma 4, M can be transformed to a det-wr Hennie machine with $\mathcal{O}(n)$ states and $\mathcal{O}(n)$ working symbols. □

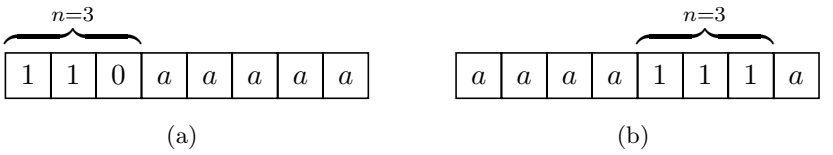

Fig. 3. A binary counter of length $n = 3$ is used by a det-wr Hennie machine to check if the input's length is $2^n = 8$. It store value 3 at the beginning (a) and value 7 at the end (b).

The acceptance of a 1NFA relates to the following problem. Given an undirected graph $G = (V, E)$ and two its vertices s, t, the undirected s-t connectivity problem (USTCON) is to determine if there is a path between s and t. USTCON is solvable by a deterministic logarithmic-space algorithm [12]. We utilize this fact in the proof of the next theorem.

Theorem 11. *Let A be an n-state* 1NFA *working over an input alphabet Σ of size $s = |\Sigma|$. There is a det-wr Hennie machine M accepting $L(A)$, with the number of transitions polynomial in n and s.*

Proof. Let Q be the set of states of A. Without loss of generality, A has one initial (q_I) and one accepting (q_A) state. We distinguish two cases by the length of the input string $w \in \Sigma^*$.

If $|w| \leq n$, M solves the question whether 1NFA A accepts w as an instance of USTCON problem. The related graph is depicted in Figure 4(a). Each column corresponds to one tape field and contains a vertex for each state of A. Vertices are thus pairs (q, p) where $q \in Q$ and $p \in \{1, \ldots, |w|\}$. Two vertices are connected by an edge iff they are in neighboring columns and a transition from the configuration on the left to the configuration on the right is allowed. The input is accepted iff $(q_I, 1)$ and $(q_A, |w|)$ are connected. How many states are needed for M? First, to memorize $\mathcal{O}(sn^2)$ transitions of A. Second, to provide $\mathcal{O}(\log n)$ space required to solve USTCON. This space permits polynomially many (in n) different configurations, hence polynomially many states of M are sufficient.

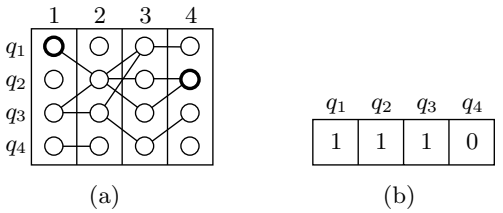

Fig. 4. (a) An undirected graph of configurations and transitions induced by a 1NFA A over an input string of length 4. State q_1 is initial, state q_2 is accepting. (b) A block representing states reachable by M when its head scans the third input's symbol.

If $|w| > n$, M has enough space to record states reachable by A on the tape. The tape is split into blocks of length n, with the exception of the last block whose length is $n + (|w| \mod n)$. The i-th field of a block stores a one-bit flag indicating the reachability of the i-th state of A, see Figure 4(b). Each block is used only when the simulated head of A is inside the block. When the head of A leaves the block, M copies the flags to the next one. This ensures that the number of transitions done by M over a tape field depends only on n (not on $|w|$). More precisely, the simulation inside a block is done in time $\mathcal{O}(n^3)$ since M carries out three nested cycles – trough fields scanned by A in the block, states of A marked as reachable and transitions of A. □

It would be possible to apply a similar approach to the simulation of an n-state 2NFA A by a det-wr Hennie machine, however, two things will change. The question of acceptance of A reduces to the directed s-t connectivity problem (STCON). We know it is solvable by a nondeterministic algorithm in logarithmic space [9], thus, by Sawitch theorem, deterministically in space $\mathcal{O}(\log^2 n)$. However, this leads to $\mathcal{O}(n^{\log n})$ different configurations and this amount of states would be needed by a det-wr Hennie machine. The other thing is that the simulation of a two-way automaton on inputs w such that $|w| > n$ would require a different technique.

As an n-state 2NFA can be simulated by a n^2-state 1AFA [1], we also cannot expect an easy cheap simulation of 1AFA by a det-wr Hennie machine.

Nevertheless, we can show that the blowup in transitions is polynomial when simulating deterministic one-pebble automata.

Theorem 12. *Each deterministic one-pebble n-state automaton can be simulated by a det-wr Hennie machine with the number of transitions polynomial in n.*

Proof. Let A be a deterministic one-pebble automaton with the set of states Q and the input alphabet Σ. We will design a Hennie machine M simulating A. Throughout the proof, assume that A always halts, i.e., it never goes into a loop. It will be clear in the end of the construction that all looping computations of A result in a det-wr Hennie machine which terminates since it scans a symbol with the lowest weight.

Given an input $w \in \Sigma^*$. If $|w| \leq n$, M simply represents the pebble on the tape by a marker and simulates A transition by transition. There are $\big(|Q||w|^2\big)$ different configurations of A, thus M finishes in time $\mathcal{O}(n^3)$, visiting each tape field at most $n|w| = \mathcal{O}(n^2)$ times.

If $|w| > n$, a direct simulation could easily exceed the targeted polynomial number of transitions performed over a tape field. To handle this, M splits the input into blocks[1] of length n (all blocks except the last one) or $n+(|w| \mod n)$ (the last block) and computes what happens whenever the head of A moves outside a block B in a state q while the marker is left inside B. A either returns back in a state from Q or finishes, thus, two mappings $f_L, f_R : Q \to Q \times \{acc, rej\}$ are computed for each block: f_L determines the outcome when the head crosses the left border of B, while f_R relates to the right border. The constants acc and rej represent accepting and rejecting by A. An example is given in Figure 5. Values of both mappings are recorded in B. Its i-th field stores the pair $f_L(q_i)$ and $f_R(q_i)$. There are $(n+2)^2$ different values of these pairs, required to have representatives in the working alphabet of M.

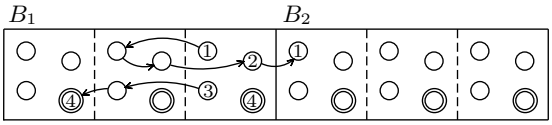

Fig. 5. The simulation of a deterministic one-pebble automaton with the set of states $Q = \{1, 2, 3, 4\}$ where 4 is a final state. All transitions done after leaving the block B_2 without the pebble are displayed for all the states. Mapping f_L for B_2 is as follows: $f_L(1) = 1$, $f_L(2) = 1$, $f_L(3) = acc$, $f_L(4) = acc$.

If all mappings are stored, M performs the simulation of A transition by transition inside blocks as well as in cases when A moves the pebble. When A leaves a block without the pebble, the corresponding mapping is used to decide what happens. This again ensures that time spent by M in a block is polynomial in n.

[1] Splitting points are located by counting to n in states.

The remaining task is to efficiently compute the mappings. We describe the computation of f_L. Let $B_1, B_2, \ldots, B_s$ be all the blocks from left to right. To compute f_L for B_1 is trivial. When the head of A leaves its left border, it moves to the left-end marker. Assume f_L has been already computed and represented for a block B_i. It is used to determine f_L for the next block B_{i+1}. For each state $q \in Q$, M simulates in B_i what happens when the head of A enters the rightmost field of B_i and the control unit is in state q. Whenever A is about to enter B_{i-1}, the mapping f_L of B_i is used to continue the tracking. Note that mappings f_R can be computed analogously, taking them in the reversed order.

Since time spent by M in each block is polynomial in n, it can be converted to a det-wr Hennie machine with polynomially many transitions. □

5 Conclusion

We presented deterministic weight-reducing Hennie machines as a constructive subclass of Hennie machines. We showed that their ability to rewrite symbols can significantly facilitate design of devices accepting regular languages. When we get rid of rewriting by converting the machine to a 1DFA, the blowup in transitions is at most double exponential. This is the same order of blowup as exhibited, e.g., by alternating automata or two-way one-pebble automata [3].

The power of the model is further illustrated by the other proved trade-offs. It can simulate at a low cost a pebble used by a deterministic two-way automaton. It cannot be replaced at a low cost by a 2NFA.

Sakoda and Sipser stated two famous open problems [13]. Is it possible to simulate an n-state 1NFA or 2NFA by a 2DFA with polynomially many states? We studied these questions for a more powerful device and showed that det-wr Hennie machine can do it in the case of 1NFA. This result can be interpreted as a sort of 1NFA's determinization which keeps the size of its description small. An open question remains what is the trade-off in the case of 2NFA. We have suggested that an n-state 2NFA could be simulated by a det-wr Hennie machine with $\mathcal{O}(n^{\log n})$ transitions. Can we achieve a polynomial blowup here as for 1NFA?

References

1. Birget, J.C.: State-complexity of finite-state devices, state compressibility and incompressibility. Mathematical Systems Theory 26(3), 237–269 (1993)
2. Durak, B.: Two-way finite automata with a write-once track. J. Autom. Lang. Comb. 12(1), 97–115 (2007)
3. Globerman, N., Harel, D.: Complexity results for two-way and multi-pebble automata and their logics. Theoretical Computer Science 169, 161–184 (1996)
4. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS 8(2), 193–234 (2002)
5. Hartmanis, J.: Computational complexity of one-tape Turing machine computations. J. ACM 15(2), 325–339 (1968)

6. Hennie, F.: One-tape, off-line Turing machine computations. Information and Control 8(6), 553–578 (1965)
7. Kari, J., Moore, C.: New results on alternating and non-deterministic two-dimensional finite-state automata. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 396–406. Springer, Heidelberg (2001)
8. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: SWAT (FOCS), pp. 188–191. IEEE Computer Society (1971)
9. Papadimitriou, C.M.: Computational complexity. Addison-Wesley, Reading (1994)
10. Průša, D., Mráz, F.: Two-dimensional sgraffito automata. In: Yen, H.-C., Ibarra, O.H. (eds.) DLT 2012. LNCS, vol. 7410, pp. 251–262. Springer, Heidelberg (2012)
11. Radó, T.: On non-computable functions. Bell System Technical Journal 41(3), 877–884 (1962)
12. Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4), 1–17 (2008)
13. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, STOC 1978, New York, NY, USA, pp. 275–286 (1978)
14. Shannon, C.E.: A universal Turing machine with two internal states. Annals of Mathematics Studies 34, 157–165 (1956)
15. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM J. Res. Dev. 3(2), 198–200 (1959)