Deterministic moles cannot solve liveness

Christos Kapoutsis
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
`cak@mit.edu`

Abstract

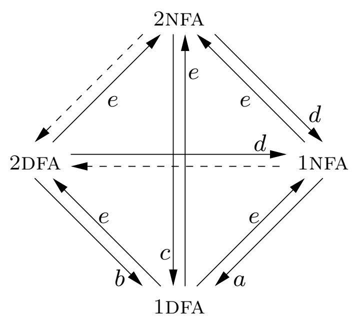
We examine the conjecture that one-way nondeterministic finite automata (1NFAs) can be exponentially more succinct than two-way deterministic ones (2DFAs). Equivalently, that no polynomial-size family of 2DFAs can recognize B, where B is *liveness*: a family of languages which is hardest among those recognizable by polynomial-size families of 1NFAs. We focus on *moles*, a restricted class of automata that includes the members of the polynomial-size family of 1NFAs solving B. We show that, in contrast, no family of 2DFA moles can solve B, irrespective of size.

1 Introduction

It has been known for a long time [13, 14] that the power of one-way deterministic finite automata (1DFAs) does not increase when they are enhanced with nondeterminism and/or bidirectionality: be they one-way nondeterministic (1NFAs), two-way deterministic (2DFAs), or even two-way nondeterministic (2NFAs), finite automata still fail against non-regular problems. However, this describes the situation only from the point of view of computability. From the complexity perspective, the extra capabilities do increase the power of 1DFAs, in the sense that against the same problems the enhanced automata occasionally manage to stay exponentially smaller [1, 11, 12]. This observation has initiated a more general and systematic investigation: *when we convert a machine of some type into an equivalent machine of another type, how much 'larger' need the new machine be?* Even the small world of regular languages hosts some intriguing instances of this question.

The automata mentioned above define 12 conversions (Fig. 1). For the famous one, we know every n-state 1NFA can be simulated by a 1DFA that has at most $2^n - 1$ states [14]; we also know that, for every n, some n-state 1NFAs have no equivalent 1DFAs with fewer than $2^n - 1$ states [11]; so, we say that the *trade-off* from 1NFAs to 1DFAs is *exactly* $2^n - 1$. Similarly, the exact value for each of the remaining trade-offs is known [3, 9]. Except two.

Nondeterminism in two-way finite automata. It is surprising how little we know about the 2NFA-to-2DFA conversion. Not only do we not know the exact value of the trade-off, but we cannot even tell if it is polynomial: the best known upper bound is exponential (by the conversion down to 1DFAs, Fig. 1c) and the best known lower bound is quadratic (by the conversion from just unary 1NFAs [5]). It is widely believed that the trade-off is exponential. In fact, when he first posed the question, Seiferas [16] conjectured the trade-off is at least $2^n - 1$ even when the starting 2NFA is only a 1NFA. Put another way, this



$a = 2^n - 1$ [14, 11]

$b = n\big(n^n - (n-1)^n\big)$ [9]

$c = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \binom{n}{i}\binom{n}{j}\big(2^i - 1\big)^j$ [9]

$d = \binom{2n}{n+1}$ [9]

$e = n$ [3]

Figure 1: The 12 conversions defined by nondeterminism and bidirectionality, and the known exact trade-offs. Dashed arrows mark the two open problems.

conjecture says that the best general method for going from 1NFAs to 2DFAs is via 1DFAs, a rather impressive claim. It is this problem that we study. To sum up, the trade-off from 1NFAs to 2DFAs is conjectured to be $2^n - 1$, equal to its best known upper bound, and much greater than the best known lower bound —the quadratic one mentioned above.

Already in [16], Seiferas proved the conjecture under the restriction that the 2DFAs are *single-pass* (i.e., they halt as soon as they reach an end-marker). Later, Sipser [17] did the same under the restriction that the 2DFAs are *sweeping*, (i.e., they can switch direction only on end-markers) —Leung [10] repeated this on a binary alphabet, as opposed to the exponentially large one of [17]. Lately, Hromkovic and Schnitger [7] showed the conjecture for 2DFAs that are *oblivious* (i.e., they move identically on all inputs of the same length) —for a sublinear (in the input length) number of distinct head trajectories, they showed an exponential lower bound, too. None of these theorems resolves the general conjecture, as full 2DFAs can be exponentially more succinct than each of these restricted variants [16, 17].

Beyond limited bidirectionality, Chrobak [5] disproved the conjecture for *unary* automata, showing the trade-off is at most $O(n^2)$ —and at least $\Omega(n^2)$, as above. For the more general conjecture, on the unary 2NFA-to-2DFA conversion, Geffert, Mereghetti and Pighizzini [6] have recently shown the trade-off is at most $2^{\Theta(\lg^2 n)}$.

Finally, variations of the problem have appeared. If we demand that the 2DFA *can* correctly simulate the 2NFA *no matter what state and input position the latter is started at* (a condition conceptually stronger than ordinary simulation, but always satisfiable [4]) then the trade-off is at least $2^{\lg^k n}$, for any k [8]. If we demand that the 2DFA simulates the 2NFA correctly *only on all polynomially long inputs* (a condition conceptually weaker than ordinary simulation) then an exponential lower bound would imply that nondeterminism is essential in logarithmic-space Turing machines (L$\neq$NL) [2]. Last, if we allow the starting 2NFA to be a *Hennie machine* (a more powerful device, but still regular), then converting to a 2DFA costs exponentially, but only because converting to a 2NFA already does [3].

Liveness. Soon after Seiferas posed the problem, Sakoda and Sipser [15] suggested a family of languages $B = (B_n)_{n\geq 1}$ which is *complete* for 1NFA-to-2DFA conversions:

- *small* 1NFA*s can solve* B: there exists a family $N = (N_n)_{n\geq 1}$ of 1NFAs such that, for all n, the automaton N_n has n states and recognizes B_n; and
- *if small* 2DFA*s can solve* B*, then the trade-off grows slowly*: if there is a polynomial p and a family $D = (D_n)_{n\geq 1}$ of 2DFAs such that, for all n, D_n has $\leq p(n)$ states and

recognizes B_n, then every n-state 1NFA has an equivalent 2DFA with $\leq p(n)$ states. Therefore, *the trade-off is super-polynomial* iff *small* 2DFA*s cannot solve* B.

The nth member B_n of B is defined over the alphabet Σ_n of all directed 2-column graphs with n nodes per column and only rightward arrows (Fig. 2a). An m-long string over Σ_n is naturally viewed as a *directed* $(m+1)$-column graph (Fig. 2b) in which, *for simplicity*, the direction of arrows is omitted (Fig. 2c). The string is in B_n iff the graph contains a path from its 0th to its mth column. We then say the string is *live*, so B_n is the property of *liveness*. A *dead* string is one that is not live.

Moles. How does N_n work? At every step, each thread 'remembers' one of the 'live' (i.e., reachable from the far left) nodes of the current column; on reading the next symbol, the thread finds what arrows depart from that node, chooses one nondeterministically, and follows it. Note that, although the entire next symbol is read, only part of its information is used: the thread 'sees' only the arrows leaving the node it is *focused* on and ignores the rest of the symbol. Put another way, in the 'network of tunnels' defined by the input, N_n behaves like a nondeterministic robot that reads only the index of its current node and the tunnels departing from it. Intuitively, N_n is an n-state one-way nondeterministic *mole*.

Formally, a 2NFA over Σ_n is any function δ that, for some sets Q of states and $\{\vdash,\dashv\}$ of end-markers, totally maps $Q \times (\Sigma_n \cup \{\vdash,\dashv\})$ to the powerset of $Q \times \{\mathtt{l},\mathtt{r}\}$. A *focus* is any pair of *index* and *side* $\phi = (i,s) \in [n] \times \{\mathtt{l},\mathtt{r}\}$; we write $\overline{s}$ for the side opposite to s and $\overline{\phi}$ for the focus $(i,\overline{s})$. The *ϕth node* of a string $x \in \Sigma_n^*$ is the ith node of its leftmost (resp., rightmost) column, if $s = \mathtt{l}$ (if $s = \mathtt{r}$). The *ϕth component* of x is the connected component of its ϕth node in the implied undirected graph. By $x \restriction \phi$ we denote the unique string that has the length of x, all edges of the ϕth component of x, and no other edges.

Definition 1. An *assignment of foci* for δ is any mapping $\varphi : Q \to [n] \times \{\mathtt{l},\mathtt{r}\}$ such that, for every two states $p,q \in Q$, symbol $a \in \Sigma_n$, and side $s \in \{\mathtt{l},\mathtt{r}\}$:
i. $\delta(p,a) = \delta(p, a \restriction \varphi(p))$, and
ii. $\delta(p,a) \ni (q,s) \implies (\exists i \in [n])\big(\varphi(q) = (i,\overline{s})\ \&\ a \restriction \varphi(p) = a \restriction (i,s)\big)$.
We say $\varphi(p)$ is the *focus* of p. If an assignment of foci for δ exists, then δ is a *mole*.

So, δ is a mole if there is a way φ to assign a focus to each state so that, when at state p with $\varphi(p) = (i_p, s_p)$, δ behaves like a mole located at the $\varphi(p)$th node of the current symbol, facing $\overline{s_p}$: (i) it 'sees' only the component of that node, and (ii) it 'moves' only to nodes in the same component.[1] Clearly, N_n satisfies this definition.

Our approach. What happens to the size of N_n if we replace nondeterminism with bidirectionality? Can small two-way deterministic moles solve liveness? *Is it possible that, after all, small* 2DFA*s* can *solve liveness and the algorithm that achieves this is nothing more than a clever exploration of the graph implied by the input?*

We give a strong negative answer: for all $n \geq 5$, no two-way deterministic mole can recognize B_n —no matter how many states it has. Hence, the following holds:

[1] For example, when $s = \mathtt{r}$ condition (ii) says: If p on a moves *right* into q, then in the new position q must focus on the *left* column $[\varphi(q) = (\cdot,\mathtt{l})]$ (the column shared with the previous position). Moreover, if in this column q focuses on the ith node $[\varphi(q) = (i,\mathtt{l})]$, then in the previous position this same node (now the ith node of the *right* column) must belong to the same component as the node focused on by p $[a \restriction \varphi(p) = a \restriction (i,\mathtt{r})]$. Note that this implies that every state can only be reached from the side of its focus.

THEOREM. *No family of deterministic moles can solve liveness.*

More generally, we view this study as a first step in a qualitatively new direction: instead of studying 2DFAs of *restricted bidirectionality* (single-pass, sweeping, oblivious) but *unrestricted information* (the automata use all current symbol information), we examine 2DFAs of *unrestricted bidirectionality* but of *restricted information.*

2 Preliminaries

By $[n]$ we denote the set $\{1, 2, \ldots, n\}$. If α is a pair of objects, then α_0 and α_1 are its first and second components. For A and B sets, $|A|$ denotes size and $A \ominus B$ denotes symmetric difference. To refer to a symbol from Σ_n we list its arrows in brackets: e.g., the rightmost symbol in Fig. 2(a) is `[12,14,25,44]` and `[]` is the *empty* symbol. For $x \in \Sigma_n^*$, we write $|x|$ for the length of x and $V_x := \{(i,j) \mid i \in [n] \;\;\&\;\; 0 \leq j \leq |x|\}$ for its set of nodes. The *left-degree* of $v = (i,j) \in V_x$ is the number of its neighbors on column $j-1$, or 0 if $j = 0$; similarly for *right-degree.* If $y \in \Sigma_n^*$, too, then $x \cup y$ is the $\max(|x|, |y|)$-long string that has all edges of x, all edges of y, and no other.

We **fix** n:=5 and a deterministic mole $M : Q \times (\Sigma_5 \cup \{\vdash, \dashv\}) \to \mathcal{P}(Q \times \{\mathtt{l}, \mathtt{r}\})$. So, M is a 2NFA that takes values of size ≤ 1 and respects Definition 1 via some assignment of foci, say φ. We write $\text{COMP}_q(x)$ for the computation we see when M is started at state q on the leftmost symbol of x, if $\varphi(q)_1 = \mathtt{l}$; or the rightmost one, if $\varphi(q)_1 = \mathtt{r}$. If $0 \leq j \leq |x|$, then $\text{COMP}_{q,j}(x)$ is what we see when M is started 'on the jth column of x': i.e., on the $j+1$st symbol, if $\varphi(q)_1 = \mathtt{l}$; or the jth symbol, if $\varphi(q)_1 = \mathtt{r}$. Set-theoretically, a computation is a sequence of pairs (q_t, i_t), where q_t is the state and i_t the index of the current input symbol after t steps, for $t \geq 0$.

Our goal is to construct two strings over Σ_5, a live and a dead one, on which M decides identically. This will prove M fails against B_5 —no matter how large it is.

Dilemmas. Suppose $T \subseteq \Sigma_5^*$ is non-empty and 'infinitely extensible to the right', namely $(\forall x \in T)(\exists y)(|y| \neq 0 \;\;\&\;\; xy \in T)$. For example, Σ_5^* and B_5 are such sets.

Then, for $x \in T$, we consider the following experiment: For each $q \in Q$ with $\varphi(q)_1 = \mathtt{l}$, we check whether $\text{COMP}_q(x)$ *hits right* (i.e., falls off the right end of x) or not (i.e., *hangs*, *loops*, or *hits left*). In the first case we set $a_{x,q} = 1$, otherwise we set $a_{x,q} = 0$. The outcome of the experiment is the vector $a_x = (a_{x,q})_q$.

Now, when we right-extend x into $xy \in T$, how do the vectors a_x, a_{xy} compare? Clearly, all 0s in a_x stay 0s in a_{xy}; but some 1s may become 0s; overall, $a_x \geq a_{xy}$ where "$\geq$" denotes the natural component-wise order. Then, what if we further right-extend x into $xyy' \in T$? And then into $xyy'y'' \in T$? As x is infinitely right-extensible in T but the outcome can decrease only finitely often, we know that from some point on the outcome will stop changing. This motivates the following.

A `lr`-*dilemma* over T is any $x \in T$ such that, for all $xy \in T$ and all $q \in Q$ with $\varphi(q)_1 = \mathtt{l}$: $\text{COMP}_q(xy)$ hits right iff $\text{COMP}_q(x)$ does. The argument above shows that `lr`-dilemmas exist over any infinitely right-extensible set. Similarly, `rl`-*dilemmas* can be defined and exist for every set which is infinitely left-extensible.

In [16], dilemmas are called "blocking strings". Both names reveal the usage: Assume x is an `lr`-dilemma over T and the 2DFA computes on wxy, where $xy \in T$. While inside

x, the automaton faces a 'dilemma': either it will stay forever inside wx, never crossing the wx-y boundary; or it will cross it but then never visit w again. So, x 'blocks' the computation from returning to w after having seen y.

Paths. A *path* is any $x \in \Sigma_5^*$ with exactly $|x|$ edges that form 1 live path. If this connects the $(i, \mathtt{l})$th node to the $(i', \mathtt{r})$th one and visits only nodes with indices from $I \subseteq [5]$, for some $I \ni i, i'$, then we say x is an i-I-i' path (Fig. 2d).

The *behavior* of M on an i-I-i path x is the mapping $f_x : Q \to Q_\perp$ encoding all possible entry-exit pairs of states defined by the computations of M on x:

$$f_x(q) := \begin{cases} p & \text{if } \varphi(q)_0 = i \text{ and } \textsc{comp}_q(x) \text{ hits left or right into } p, \\ \perp & \text{if } \varphi(q)_0 = i \text{ and } \textsc{comp}_q(x) \text{ hangs or loops, or } \varphi(q)_0 \neq i. \end{cases}$$

So, if $f_x(q) \neq \perp$, the computation $\textsc{comp}_q(x)$ exits the input, off its left or right end and into state $p = f_x(q)$; and if then $\varphi(p) = (i, s)$ is p's focus, the exit side is $\overline{s}$.

A generic path. Pick an i-I-i path x. Collecting the states into which all $\mathtt{lr}$-crossings of x can end, and then repeating for all $\mathtt{rl}$-crossings, we get the two sets of states:

$$Q_{\mathrm{LR}}(x) := \{f_x(q) \mid q \in Q \;\;\&\;\; \varphi(q) = \varphi(f_x(q)) = (i, \mathtt{l})\},$$
$$Q_{\mathrm{RL}}(x) := \{f_x(q) \mid q \in Q \;\;\&\;\; \varphi(q) = \varphi(f_x(q)) = (i, \mathtt{r})\}.$$

If y is also an i-I-i path, extending x to xy cannot increase the number of distinct states producible by $\mathtt{lr}$-crossings; also, every state producible by an $\mathtt{lr}$-crossing of xy is producible by an $\mathtt{lr}$-crossing of y. Similar facts hold for $\mathtt{rl}$-crossings:

$$|Q_{\mathrm{LR}}(x)| \geq |Q_{\mathrm{LR}}(xy)| \qquad \text{and} \qquad |Q_{\mathrm{RL}}(xy)| \leq |Q_{\mathrm{RL}}(y)| \tag{1}$$
$$Q_{\mathrm{LR}}(y) \supseteq Q_{\mathrm{LR}}(xy) \qquad\qquad Q_{\mathrm{RL}}(xy) \subseteq Q_{\mathrm{RL}}(x) \tag{2}$$

Using (1), (2) and reasoning as in the previous section, we conclude some i-I-i path ϑ admits no extension that exhibits sets of smaller size, i.e., for all i-I-i paths x, y:

$$|Q_{\mathrm{LR}}(\vartheta)| = |Q_{\mathrm{LR}}(\vartheta y)| \qquad \text{and} \qquad |Q_{\mathrm{RL}}(x\vartheta)| = |Q_{\mathrm{RL}}(\vartheta)|. \tag{3}$$

For the remainder, we **fix** $i := 2$, $I := \{1, 2\}$ and such a ϑ for these i, I; we then let

$$\kappa := |\vartheta|, \quad L := Q_{\mathrm{RL}}(\vartheta), \quad R := Q_{\mathrm{LR}}(\vartheta), \quad \text{and} \quad \mu := \max(|L|!, |R|!)$$

denote its length, the sets of states producible by crossing it, and the least common multiple of the sizes of the corresponding permutation groups. It is not hard to verify (by (3)) that ϑ is both an $\mathtt{lr}$- and an $\mathtt{rl}$-dilemma over the set of i-I-i paths.

Behavior in the context of ϑ. Pick any i-I-i path x and consider the i-I-i path $\vartheta x \vartheta$. Any $q \in R$ is produced by some $\mathtt{lr}$-crossing of ϑ; this crossing can be viewed as the prefix of a computation on $\vartheta x \vartheta$ that starts on the 0th column and first crosses the ϑ-$x\vartheta$ boundary into q; since ϑ is an $\mathtt{lr}$-dilemma, this computation hits right, into some state; by (2), this state belongs to R. Overall, the function $g_x : R \to R$ with

$$g_x(q) = \text{the state which } \textsc{comp}_{q,|\vartheta|}(\vartheta x \vartheta) \text{ hits right into},$$

is well-defined, and (by (3)) permutes R. Similarly, the function $h_x : L \to L$

$$h_x(q) = \text{the state which } \textsc{comp}_{q,|\vartheta x|}(\vartheta x \vartheta) \text{ hits left into,}$$

is a well-defined permutation of L. The pair (g_x, h_x) fully describes the behavior of M 'in the context of ϑ': for x, y two i-I-i paths, $(g_x, h_x) = (g_y, h_y) \Rightarrow f_{\vartheta x \vartheta} = f_{\vartheta y \vartheta}$. Moreover, if x contains a copy of ϑ, namely $x = y_1 \vartheta y_2$ for i-I-i paths y_1 and y_2, then its pair is a composition of the pairs for y_1, y_2: $(g_{y_1 \vartheta y_2}, h_{y_1 \vartheta y_2}) = (g_{y_1} \circ g_{y_2}, h_{y_2} \circ h_{y_1})$. This generalizes to any number of copies of ϑ. In particular, for the string $x = y^{(k)} = y(\vartheta y)^{k-1}$ of k ϑ-separated copies of y, we have $(g_{y^{(k)}}, h_{y^{(k)}}) = \big((g_y)^k, (h_y)^k\big)$.

Fooling the mole. Now, for every length $l \geq 1$, we can collect into a set $\mathit{\Omega}_l$ all behaviors that M can exhibit in the context of ϑ on l-long i-I-i paths:

$$\mathit{\Omega}_l = \{(g_x, h_x) \mid x \text{ is an } i\text{-}I\text{-}i \text{ path and } |x| = l\}.$$

Then every behavior that is exhibited on some l-long path is also exhibited on some path of length $l + 2\mu(l + \kappa)$, namely $\mathit{\Omega}_l \subseteq \mathit{\Omega}_{l+2\mu(l+\kappa)}$. Because, for every l-long i-I-i path x, the string $x^{(2\mu+1)} = x^{(\mu)} \vartheta x \vartheta x^{(\mu)}$ is also an i-I-i path, has length $(2\mu+1)l + 2\mu\kappa = l + 2\mu(l+\kappa)$ and, by the selection of μ, causes behavior $(g_x^{2\mu+1}, h_x^{2\mu+1}) = (g_x, h_x)$. For the remainder, we **fix** λ to be any l at which $\mathit{\Omega}_l$ achieves *maximum size* and also **fix** $\mathit{\Lambda} := 2\mu(\lambda + \kappa)$. Then $\mathit{\Omega}_\lambda \subseteq \mathit{\Omega}_{\lambda+\mathit{\Lambda}}$ and also $\mathit{\Omega}_{\lambda+\mathit{\Lambda}}$ can get no larger, so

$$\mathit{\Omega}_\lambda = \mathit{\Omega}_{\lambda+\mathit{\Lambda}}.$$

Hence, when *between two copies of ϑ, any i-I-i path of either length (λ or $\lambda + \mathit{\Lambda}$) can be replaced by some path of the other length* without M noticing the trick.

Three special paths. For $d \geq 0$, we say that y is *d-disjoint* on x if in $x \cup (\text{[]}^d y)$ the edges of x and y meet at *no* vertex (Fig. 3a). We will construct three i-I-i paths, π, ϱ, σ, that look indistinguishable to M and satisfy certain disjointness relationships.

We start with ϱ. We **set** $\varrho := \vartheta \hat{\varrho} \vartheta$, where $\hat{\varrho}$ is a $(\lambda + \mathit{\Lambda})$-long i-I-i path that guarantees ϱ is $\mathit{\Lambda}$-disjoint on itself. Constructing it is easy: we pick $i' := 1$ and paths

$\eta :=$ any i-I-i' path of length λ,
$\vartheta' :=$ the i'-I-i' path of length κ that is 0-disjoint on ϑ,
$\iota :=$ any i'-I-i' path of length $\mathit{\Lambda} - (2\kappa + \lambda)$, and
$\eta' :=$ the i'-I-i path of length λ that is 0-disjoint on η.

Then, setting $\hat{\varrho} := \eta \vartheta' \iota \vartheta' \eta'$ we see this is indeed an i-I-i path of length $\lambda + \mathit{\Lambda}$; and shifting $\varrho = \vartheta \hat{\varrho} \vartheta = \vartheta \eta \vartheta' \iota \vartheta' \eta' \vartheta$ on a copy of itself by $\mathit{\Lambda} = |\vartheta \eta \vartheta' \iota|$ causes only its prefix $\vartheta \eta \vartheta'$ to overlap with the mirroring suffix $\vartheta' \eta' \vartheta$, so that no vertex is shared.

We continue with π. Since $\hat{\varrho}$ is of length $\lambda + \mathit{\Lambda}$, the behavior $(g, h) := (g_{\hat{\varrho}}, h_{\hat{\varrho}})$ of M on $\hat{\varrho}$ in the context of ϑ is in $\mathit{\Omega}_{\lambda+\mathit{\Lambda}}$, and therefore in $\mathit{\Omega}_\lambda$. Hence, there exists a λ-long i-I-i path $\hat{\pi}$ on which M exhibits the same behavior, $(g_{\hat{\pi}}, h_{\hat{\pi}}) = (g, h)$. We **set** $\pi := \vartheta \hat{\pi} \vartheta$. Clearly, $f_\pi = f_{\vartheta \hat{\pi} \vartheta} = f_{\vartheta \hat{\varrho} \vartheta} = f_\varrho$ and M cannot distinguish π from ϱ.

We conclude with σ. First, we pick an i-I-i path ϱ' that is as long as $\hat{\varrho}$ and shares no vertex with π when the two paths are 'centered'[2] on top of each other: namely, $|\varrho'| = \lambda + \mathit{\Lambda}$

[2]By 'centering' the two paths on top of each other we mean shifting one on top of the other so that their 'centers of mass' coincide. Since ϱ' is longer than π by $|\varrho'| - |\pi| = |\hat{\varrho}| - |\vartheta \hat{\pi} \vartheta| = (\lambda + \mathit{\Lambda}) - (\lambda + 2\kappa) = \mathit{\Lambda} - 2\kappa$, this in turn means shifting π on top of ϱ' by $d = (|\varrho'| - |\pi|)/2 = \frac{\mathit{\Lambda}}{2} - \kappa$.

and π is $(\frac{\Lambda}{2}-\kappa)$-disjoint on ϱ'.[3] The behavior $(g',h') := (g_{\varrho'}, h_{\varrho'})$ of M on ϱ' is in $\Omega_{\lambda+\Lambda}$, and hence in Ω_λ. So, some λ-long i-I-i path π' forces the same behavior, $(g_{\pi'}, h_{\pi'}) = (g',h')$. Now consider the i-I-i path $\hat\sigma := (\pi')^{(\mu)}\vartheta\varrho'\vartheta(\pi')^{(\mu-1)}\vartheta\hat\pi$, containing $2\mu+1$ ϑ-separated i-I-i paths, all copies of π' except the middle and rightmost ones, which copy ϱ' and $\hat\pi$. This forces the behavior

$$\left(\ (g_{\pi'})^\mu g_{\varrho'}(g_{\pi'})^{\mu-1}g_{\hat\pi},\ \ h_{\hat\pi}(h_{\pi'})^{\mu-1}h_{\varrho'}(h_{\pi'})^\mu\ \right) = \left(\ (g')^{2\mu}g,\ \ h(h')^{2\mu}\ \right) = (g,h),$$

where we used the selection of μ. Overall, $\hat\sigma$ is $(\lambda+2\Lambda)$-long and forces the same behavior as $\hat\varrho$, $\hat\pi$. We **set** $\sigma := \vartheta\hat\sigma\vartheta$ and verify that $f_\sigma = f_\varrho = f_\pi$, so M cannot distinguish among σ, ϱ, π. Also, σ symmetrically extends ϱ' in both directions by

$$|\vartheta(\pi')^{(\mu)}\vartheta| = |\vartheta(\pi')^{(\mu-1)}\vartheta\hat\pi\vartheta| = \kappa + \big((\mu-1)\kappa+\mu\lambda\big) + \kappa = \tfrac{\Lambda}{2}+\kappa,$$

so 'centering' π on σ causes no common nodes: π is Λ-disjoint on σ.

Mazes. A path is a special case of a more general object: a maze. Intuitively, this is any string on which some nodes have been selected as entry/exit points (Fig. 3b). More formally, let $V_x^0 \subseteq V_x$ consist of every node of the string x that has exactly one of its two degrees equal to 0 (and hence is eligible to serve as a *gate* for 'entering' or 'exiting' the graph of x). Then a *maze* on x is any pair (x,X) where $X \subseteq V_x^0$. For example, in these terms, any i-I-i' path x is the maze $\big(x, \{(i,0),(i',|x|)\}\big)$.

Computing on a maze generalizes computing on a path. For $\chi = (x,X)$ a maze, $u = (i,j) \in X$ a gate with 0-degree side s, and q a state with $\varphi(q) = (i,s)$, the computation $\textsc{comp}_{q,u}(\chi)$ of M on χ from q and u is[4] what we get when M is started at q and the jth column and run until (if ever) it 'exits the maze'. Here, 'exits the maze' means it hits left or right, or 'exits a gate'; in turn, the mole 'exits' gate (i,j) with 0-degree side s iff it is at a state q with $\varphi(q) = (i,\overline{s})$ and on the j-th column.

Composition. To compose two mazes means to draw their strings on top of each other and then discard all coinciding gates (Fig. 3c). More carefully, $\chi = (x,X)$ and $\psi = (y,Y)$ are *composable* iff $|x| = |y|$ (so that $V_x = V_y = V$) and their graphs 'touch' each other only at gates and only appropriately: every $v \in V$

- *either* has both its degrees equal to 0 in at least one of x, y,
- *or* is a gate in both mazes with a different 0-degree side in each of them.

If χ, ψ are composable, then their *composition* is the pair $\chi\circ\psi = (x\cup y, X\ominus Y)$.

Clearly, the composition is a maze. Moreover, in each symbol of $x\cup y$, every non-empty connected component comes entirely from exactly one of x, y. Hence, when M reads a symbol, its next step depends on exactly one of x, y: the one that contains the component currently holding M's focus. More generally, for $w \in X\ominus Y$, every computation $c = \textsc{comp}_{q,w}(\chi\circ\psi)$ can be 'broken' into a list of 'subcomputations' $c_1, c_2, \dots$, called *fragments* of c, such that: every odd-indexed fragment is a computation on χ (wlog, say $w \in X \setminus Y$); every even-indexed fragment is a computation on ψ; *either* all fragments are finite, and then the list is infinite iff c is, *or* not, in which case the list is finite and the

[3] Finding ϱ' is trivial: simply take the unique i'-I-i' path that is as long as π and 0-disjoint on it, then extend by $\frac{\Lambda}{2}-\kappa$ in both directions into any i-I-i path.

[4] Note that $\textsc{comp}$ applied on mazes has different argument types from $\textsc{comp}$ applied on strings.

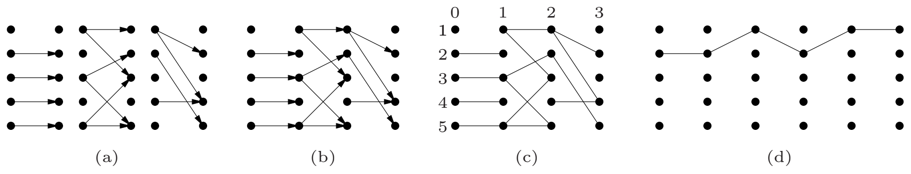


Figure 2: (a) Three symbols in Σ_5. (b) The string they define. (c) The same string, simplified and indexed. (d) A 5-long 1-$\{1,2\}$-2 path.

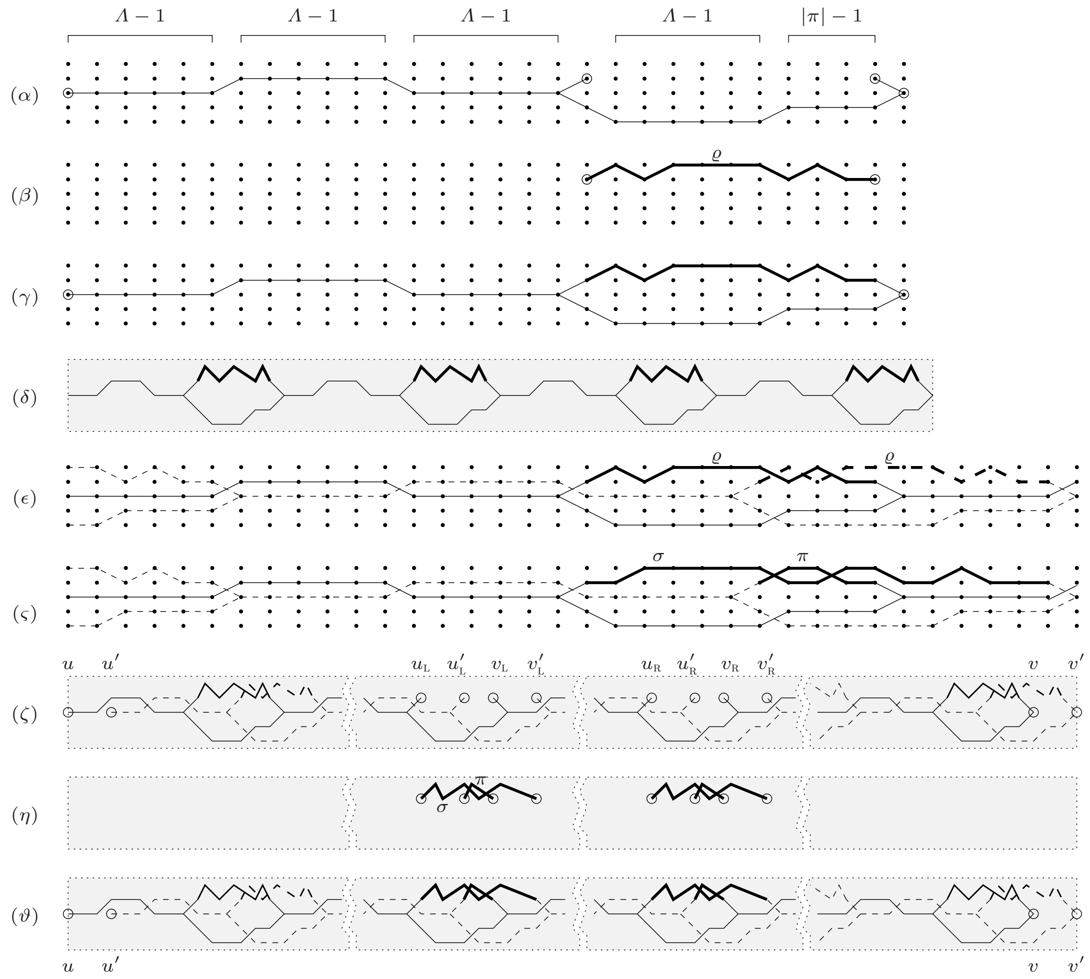


Figure 3: (a) in each of α, β, γ: a 29-long string which is 6-disjoint on itself (see ϵ). (b) in each of α, β, γ: a maze; its gates marked with circles. (c) in γ: the composition of the mazes of α and β. (d) in α, β, γ: examples of τ_2, τ_1, τ, respectively, for a schematic case $\varLambda = 6$, $|\pi| = 4$, and a schematic ϱ. (e) in δ: a schematic of τ^4; in ϵ: a $(|\tau| + \varLambda)$-long snippet of the union of a τ^i with a $\varLambda$-shifted copy of itself. (f) in ζ: a schematic of χ', focusing on the snippets around the leftmost, i_1th distinguished, i_2th distinguished, and rightmost (pairs of) copies of τ. (g) in η: a schematic of ψ', for the same snippets. (h) in ϑ: a schematic of the composition $\omega' = \chi' \circ \psi'$, for the same snippets; in ϵ, ς: a more detailed view of how σ and π connect the two disjoint graphs of x' when they replace two copies of ϱ.

only infinite fragment is the last one; c_1 starts at q and w, every other fragment starts at the state and gate where the previous one finishes, and the last fragment (if it exists and is finite) finishes identically to c.

3 The argument

A maze of questions. We start (Fig. 3d) with the two strings $\tau_1 = \texttt{[]}^{3\mathit{\Lambda}}\varrho\texttt{[]}$ and

$$\tau_2 = \texttt{[33]}^{\mathit{\Lambda}-1}\texttt{[32]}\,\texttt{[22]}^{\mathit{\Lambda}-1}\texttt{[23]}\,\texttt{[33]}^{\mathit{\Lambda}-1}\texttt{[32,34]}\,\texttt{[45]}\,\texttt{[55]}^{\mathit{\Lambda}-1}\texttt{[54]}\,\texttt{[44]}^{|\pi|-1}\texttt{[23,43]},$$

which are equally long and each is $\mathit{\Lambda}$-disjoint on itself (recall the selection of ϱ). Also, in $\tau := \tau_1 \cup \tau_2$ their graphs 'touch' only at the endpoints of ϱ, so τ is $\mathit{\Lambda}$-disjoint on itself, too. Which implies τ^i is also $\mathit{\Lambda}$-disjoint on itself, for all $i \geq 1$ (Fig. 3e).

Let $T := \{\tau^i \mid i \geq 1\}$ be the set of all powers of τ, select τ_{L} and τ_{R} as $\texttt{lr}$- and a $\texttt{rl}$-dilemmas over T, respectively, and fix $m = 2^{|Q|} + 1$. The live string $z = \tau_{\mathrm{L}}\tau^m\tau_{\mathrm{R}}$ is also a power of τ and in it we think of the m 'middle' copies of τ as *distinguished*.

Now consider the maze $\omega = (z, Z) = (z, \{u, v\})$, where $u = (3, 0)$ and $v = (3, |z|)$ are the two natural gates to the graph of z, and the $|Q|$ computations $\mathrm{COMP}_{q,w}(\omega)$ that we get as we vary $q \in Q$ and pick $w = u$ when $\varphi(q)_1 = \texttt{l}$, and $w = v$ otherwise. Disregard every computation that is infinite (i.e., loops) or is finite but non-crossing (i.e., hangs; or starts and ends on the same gate), keep only those that are *finite* and *crossing* (i.e., start and end in different gates), and let $k \leq |Q|$ be their number.

Fix d to be one of these k computations and (wlog) assume it starts at u and ends at v (do a similar analysis for $\texttt{rl}$-crossings). Then d visits all m distinguished copies of τ and we want to describe its behavior on each of them. So, fix an $1 \leq i \leq m$.

If we 'rip off' ϱ from the ith distinguished copy of τ and then add the two endpoints u_i, v_i of the path as new gates, we construct a new maze,

$$\chi_i = \big((\tau_{\mathrm{L}}\tau^{i-1})\;\tau_2\;(\tau^{m-i}\tau_{\mathrm{R}}),\;\;\{u, v, u_i, v_i\}\big).$$

By the 'complementary' operation, where we rip off everything *except* the particular copy of ϱ, we can construct the 'complementary' maze,

$$\psi_i = \big((\texttt{[]}^{|\tau_{\mathrm{L}}\tau^{i-1}|})\;\texttt{[]}^{3\mathit{\Lambda}}\varrho\texttt{[]}\;(\texttt{[]}^{|\tau^{m-i}\tau_{\mathrm{R}}|}),\;\;\{u_i, v_i\}\big).$$

Sure enough, $\omega = \chi_i \circ \psi_i$, and d is a finite computation on this composition. Hence, we can break it into its finitely many, finite fragments $d_1, d_2, \ldots, d_\nu$. We know every even(-indexed) fragment is a computation on ψ_i; we call it *crossing* if its starting and ending gates differ. We care about the parity of the number of such fragments:

$$b_{i,d} = 0 \iff d \text{ exhibits an even number of crossing even fragments},$$

i.e., the bit $b_{i,d}$ is 0 iff throughout d the copy of ϱ in the ith distinguished copy of τ is 'fully crossed' an even number of times.

Intuitively, as the mole develops d, each distinguished copy of τ poses the question: "odd or even?" The mole answers with the parity of the number of times that it fully crosses ϱ in that copy in either direction. The bits $b_{i,d}$ record these answers.

Organizing these $m \times k$ bits into m k-long vectors $b_i = (b_{i,d})_d$, for $i = 1, \ldots, m$, we see that there are more vectors than values for them: $2^k \leq 2^{|Q|} < 2^{|Q|} + 1 = m$. Hence, $b_{i_1} = b_{i_2}$ for some $1 \leq i_1 < i_2 \leq m$. Which means that, in each crossing finite computation, the answer to the i_1th question equals the answer to the i_2th one.

A labyrinth. We return to $\omega = (z, \{u, v\})$, simultaneously rip off ϱ from the i_1th and i_2th distinguished copies of τ (as above), and name the four natural new gates $u_{\mathrm{L}}, v_{\mathrm{L}}$ (endpoints of ϱ in i_1th copy) and $u_{\mathrm{R}}, v_{\mathrm{R}}$ (endpoints of ϱ in i_2th copy) to get

$$\chi = (x, X) = \big((\tau_{\mathrm{L}}\tau^{i_1-1})\ \tau_2\ (\tau^{i_2-i_1-1})\ \tau_2\ (\tau^{m-i_2}\tau_{\mathrm{R}}),\ \ \{u, v, u_{\mathrm{L}}, v_{\mathrm{L}}, u_{\mathrm{R}}, v_{\mathrm{R}}\}\big).$$

As above, the 'complementary' maze (rip off everything *except* the two ϱ's) is

$$\psi = (y, Y) = \big((\cdots)\ \texttt{[]}^{3\Lambda}\varrho\texttt{[]}\ (\cdots)\ \texttt{[]}^{3\Lambda}\varrho\texttt{[]}\ (\cdots),\ \ \{u_{\mathrm{L}}, v_{\mathrm{L}}, u_{\mathrm{R}}, v_{\mathrm{R}}\}\big),$$

where ellipses stand for appropriately many $\texttt{[]}$s. We obviously have $\omega = \chi \circ \psi$.

Note that x is Λ-disjoint on itself (because z is). So, in the union $x' = x \cup (\texttt{[]}^{\Lambda}x)$ of x with a Λ-shifted copy of itself, the two graphs do not 'touch' each other. (Fig. 3f.) So, letting $\chi' = (x', X')$ where $X' = X \cup \{u', v', u'_{\mathrm{L}}, v'_{\mathrm{L}}, u'_{\mathrm{R}}, v'_{\mathrm{R}}\}$ contains all gates of χ plus their counterparts in the shifted copy, we know every computation on χ' visits and depends on exactly one of the two disjoint graphs.

Similarly, y is Λ-disjoint on itself (because ϱ is), the union $y \cup (\texttt{[]}^{\Lambda}y)$ contains two pairs of disjoint copies of ϱ, and $Y' = Y \cup \{u'_{\mathrm{L}}, v'_{\mathrm{L}}, u'_{\mathrm{R}}, v'_{\mathrm{R}}\}$ contains their endpoints. Now view each pair of copies of ϱ as a copy of $\varrho \cup (\texttt{[]}^{\Lambda}\varrho)$, replace it with a copy of $\tilde{\varrho} = \sigma \cup (\texttt{[]}^{\Lambda}\pi)$, call the new string y', and set $\psi' = (y', Y')$. (Fig. 3g.) Crucially, the substitution (i) *preserves the lengths of strings*: $|y'| = |y \cup (\texttt{[]}^{\Lambda}y)|$, because

$$|\tilde{\varrho}| = |\sigma \cup (\texttt{[]}^{\Lambda}\pi)| = |\sigma| = 2k + l + 2\Lambda = |\varrho| + \Lambda = |\varrho \cup (\texttt{[]}^{\Lambda}\varrho)|;$$

(ii) *preserves the number and disjointness of paths*: since π is Λ-disjoint on σ, we know $\tilde{\varrho}$ also contains two disjoint paths; and (iii) *preserves the set of endpoints of paths*: e.g., on the copy of $\tilde{\varrho}$ on the left, σ and π have endpoints u_{L}, v'_{L} and u'_{L}, v_{L}. Note that every computation on ψ' visits and depends on exactly one of the paths.

Clearly, the graphs of x' and y' 'touch' each other only at the gates in Y', so χ', ψ' compose, into $\omega' = (z', Z') = \chi' \circ \psi' = (x' \cup y', \{u, v, u', v'\})$. (Fig. 3h.) Note that u, u' are on the far left; v, v' are on the far right; and the paths of y' connect the two graphs of x': the mole can switch graphs only if it fully crosses one of the paths.

The hidden exit. Consider the computation c' on the dead input $z'\texttt{[]}$. This is what we see when M is started on the string $\vdash z'\texttt{[]} \dashv$ at its start state and on $\vdash$. Since the first symbol of z' is $\texttt{[33]}$, any attempt of the mole to depart from $\vdash$ in a state of index other than 3 is followed by a step back to $\vdash$. Disregarding these attempts and noting that the mole can never move past $\texttt{[]}$, we see c' consists of one or more computations of the form $\textsc{comp}_{q,0}(z'\texttt{[]})$ with $\varphi(q) = (3, \texttt{1})$. Assume (wlog) that c' consists of exactly one such computation (for more, do a similar analysis).

So, $c' = \textsc{comp}_{q,0}(z'\texttt{[]})$ where $\varphi(q) = (3, \texttt{1})$. As a mole, every time M visits the 0-degree side of the nodes u', v, v', it changes direction to return into the graph of z'. Call every such visit a *turn* and break c' into *segments* $c'_1, c'_2, \ldots$ so that successive segments are joined at a turn: the later segment starts at the state and node following the last state and node of the earlier one. Clearly, each segment is a computation on ω'; $c'_1 = \textsc{comp}_{q,u}(\omega')$ but later segments start at a gate in $\{v, u', v'\}$; and either all segments are finite, in which case their list is finite iff c' is, or not, in which case the list is finite and only the last segment is infinite.

Our goal is to prove that c' never visits the 0-degree side of v'; intuitively, that *the labyrinth implied by z' hides the gate v' from the mole.* This will immediately imply the failure of M: on the *live* input z'`[33]` the mole will compute exactly as on the *dead* input z`[]`, as it will never visit the 0-degree side of v' to note the difference.

To reach our goal, it is sufficient to show that no segment ends in v'. This, in turn, is a corollary of the following: (i) c'_1 starts at u, (ii) a finite segment that starts at u and does not hang necessarily ends on either u or v, and (iii) a finite segment that starts at v and does not hang necessarily ends on either u or v. We show (ii); (iii) is similar and (i) is known.

The final argument. Let d' be a non-hanging finite segment of c' that starts at u. As a finite computation on $\omega' = \chi' \circ \psi'$, it can be broken into finitely many, finite fragments $d'_1, d'_2, \dots, d'_\nu$, where odd(-indexed) fragments compute on χ' and even(-indexed) ones compute on ψ'. By previous remarks, every odd fragment visits and depends on exactly one of the two graphs (unshifted and shifted) inside x'; and every even fragment visits and depends on exactly one of the four paths in y'. Calling an even fragment *crossing* if its start and final gates differ, we see that two successive odd fragments visit different graphs in x' iff the even fragment between them is crossing. Generalizing, and since d' starts on u, *each odd fragment visits the* shifted *graph in x' iff the number of crossing even fragments that precede it is* odd.

Towards a contradiction, assume d' does not end in u or v. Then it ends in either u' or v'. Hence, d'_ν is an odd segment that visits the shifted graph in x'. Which implies *the total number of crossing even fragments* (before d'_ν, and so throughout d') *is odd.* In particular, even fragments exist and d'_1 *necessarily ends in a gate in Y.*

Now let $\zeta : X' \to X$ be the function that maps every gate in X' to its unprimed version in X; e.g., $\zeta(u_{\text{L}}) = \zeta(u'_{\text{L}}) = u_{\text{L}}$. We extend ζ to the fragments $d'_1, d'_2, \dots, d'_\nu$.

∘ If d'_i is an odd fragment (hence, a computation on exactly one of the two graphs in χ') starting at state p on gate w and ending at state $\dot{p}$ on gate $\dot{w}$, we let $d_i = \zeta(d'_i)$ be the computation on (the unique graph of) χ that starts at p on $\zeta(w)$ and ends at $\dot{p}$ on $\zeta(\dot{w})$. Clearly, d_i exists. In particular, $d_1 = \zeta(d'_1)$ starts at $\zeta(u) = u$ and (by the remark above) ends in a gate in $\zeta(Y) = Y$.

∘ If d'_i is an even fragment (hence, a computation on exactly one of the four paths in ψ') starting at p on w and ending at $\dot{p}$ on $\dot{w}$, we let $d_i = \zeta(d'_i)$ be the computation on (one of the two copies of ϱ in) ψ that starts at p on $\zeta(w)$ and ends at $\dot{p}$ on $\zeta(\dot{w})$. This time d_i exists because ϱ is indistinguishable from each of σ and π.

We review the list $d_1, d_2, \dots, d_\nu$: d_1 starts on $\zeta(u) = u$; for $1 \leq i < \nu$, d_i ends at the state and gate where d_{i+1} starts; d_ν ends on $\zeta(u') = u$ or $\zeta(v') = v$; and every even fragment d_i is crossing (on the path of ψ that it visits) iff d'_i is (on the path of ψ' that it visits). Hence, concatenation builds a computation d on $\chi \circ \psi = \omega$, that starts on u, ends on u or v, and contains an odd number of crossing even fragments.

But is this possible?

If d ends in u, then it never moves beyond τ_{L} (if it did, it would traverse the `lr`-dilemma and get 'blocked' away from u). In particular, d_1 never reaches a gate in Y. But (by previous remark) this is where it is supposed to end. Contradiction.

If d ends in v, then it is a crossing finite computation on ω. As ω equals each of the compositions $\chi \circ \psi$, $\chi_{i_1} \circ \psi_{i_1}$, and $\chi_{i_2} \circ \psi_{i_2}$, we know d can be fragmented in three different ways. Clearly, every *even* fragment with respect to either $\chi_{i_1} \circ \psi_{i_1}$ or $\chi_{i_2} \circ \psi_{i_2}$ is also an *even* fragment with respect to $\chi \circ \psi$, and vice versa; and is *crossing* or not (on the copy of ϱ that it visits) irrespective of which composition we look at it through. So, letting ξ, ξ_1, ξ_2 be the numbers of *crossing even* fragments with respect to the three compositions, we know $\xi = \xi_1 + \xi_2$ and (as established above) ξ is odd. Yet, by the selection of i_1 and i_2, the parities of ξ_1, ξ_2 are respectively $b_{i_1,d}$, $b_{i_2,d}$ and hence equal (as $b_{i_1} = b_{i_2}$), so that ξ should be even. Contradiction.

Acknowledgement. Many thanks to Michael Sipser for the hints and the encouragement throughout our countless discussions about this intriguing but so beautiful problem.

References

[1] Bruce H. Barnes. A two-way automaton with fewer states than any equivalent one-way automaton. *IEEE Transactions on Computers*, C-20(4):474–475, 1971.

[2] Piotr Berman and Andrzej Lingas. On complexity of regular languages in terms of finite automata. Report 304, Institute of Computer Science, Polish Academy of Sciences, Warsaw, 1977.

[3] Jean-Camille Birget. Two-way automata and length-preserving homomorphisms. Report 109, Department of Computer Science, University of Nebraska, 1990.

[4] Jean-Camille Birget. Positional simulation of two-way automata: proof of a conjecture of R. Kannan and generalizations. *Journal of Computer and System Sciences*, 45:154–179, 1992.

[5] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47:149–158, 1986.

[6] Viliam Geffert, Carlo Mereghetti, and Giovanni Pighizzini. Converting two-way nondeterministic unary automata into simpler automata. *Theoretical Computer Science*, 295:189–203, 2003.

[7] Juraj Hromkovič and Georg Schnitger. Nondeterminism versus determinism for two-way finite automata: generalizations of Sipser's separation. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, pages 439–451, 2003.

[8] Ravi Kannan. Alternation and the power of nondeterminism. In *Proceedings of the Symposium on the Theory of Computing*, pages 344–346, 1983.

[9] Christos Kapoutsis. Removing bidirectionality from finite automata. In preparation.

[10] Hing Leung. Tight lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 63(3):384–393, 2001.

[11] Albert R. Meyer and Michael J. Fischer. Economy of description by automata, grammars, and formal systems. In *Proceedings of the Symposium on Switching and Automata Theory*, pages 188–191, 1971.

[12] Frank R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transactions on Computers*, 20(10):1211–1214, 1971.

[13] Michael O. Rabin. Two-way finite automata. In *Proceedings of the Summer Institute of Symbolic Logic*, pages 366–369, Cornell, 1957.

[14] Michael O. Rabin and Dana Scott. Remarks on finite automata. In *Proceedings of the Summer Institute of Symbolic Logic*, pages 106–112, Cornell, 1957.

[15] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two-way finite automata. In *Proceedings of the Symposium on the Theory of Computing*, pages 275–286, 1978.

[16] Joel I. Seiferas. Manuscript communicated to M. Sipser. October 1973.

[17] Michael Sipser. Lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 21(2):195–202, 1980.