



# Lower Bounds for Graph-Walking Automata

Olga Martynova  

Department of Mathematics and Computer Science, St. Petersburg State University, Russia

Alexander Okhotin  

Department of Mathematics and Computer Science, St. Petersburg State University, Russia

---

## Abstract

Graph-walking automata (GWA) traverse graphs by moving between the nodes following the edges, using a finite-state control to decide where to go next. It is known that every GWA can be transformed to a GWA that halts on every input, to a GWA returning to the initial node in order to accept, as well as to a reversible GWA. This paper establishes lower bounds on the state blow-up of these transformations: it is shown that making an  $n$ -state GWA traversing  $k$ -ary graphs return to the initial node requires at least  $2(n-1)(k-3)$  states in the worst case; the same lower bound holds for the transformation to halting automata. Automata satisfying both properties at once must have at least  $4(n-1)(k-3)$  states. A reversible automaton must have at least  $4(n-1)(k-3) - 1$  states. These bounds are asymptotically tight to the upper bounds proved using the methods from the literature.

**2012 ACM Subject Classification** Theory of computation → Formal languages and automata theory; Theory of computation → Models of computation

**Keywords and phrases** Finite automata, graph-walking automata, halting, reversibility

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2021.52

**Funding** Research supported by the Russian Science Foundation, project 18-11-00100.

## 1 Introduction

Graph-walking automata (GWA) are finite automata that traverse labelled undirected graphs.

On the one hand, this is a model of a robot with limited memory navigating a discrete environment. There is an early result by Budach [2] that for every automaton there is a graph that it cannot fully explore; a short proof of this fact was later given by Fraigniaud et al. [5]. This work has influenced the current research on algorithms for graph traversal using various small-memory models, equipped with a limited number of pebbles, etc. [3, 4].

On the other hand, GWA naturally generalize such important models as tree-walking automata [1] (TWA) and two-way finite automata (2DFA). More generally, a GWA can represent various models of computation, if a graph is regarded as the space of memory configurations, and every edge accordingly represents an operation on the memory. This way, quite a few models in automata theory and in complexity theory, such as multi-head and multi-tape automata and space-bounded complexity classes, can be regarded as GWA. Then, some results on GWA apply to all these models.

Among such results, there are transformations of GWA to several important subclasses: to automata that halt on every input graph; to automata that return to the initial node in order to accept; to reversible automata. Such transformations have earlier been established for various automaton models, using a general method discovered by Sipser [15], who constructed a halting 2DFA that traverses the tree of computations of a given 2DFA leading to an accepting configuration, in search of an initial configuration. Later, Kondacs and Watrous [7] ensured the reversibility and optimized this construction for the number of states, motivated by the study of quantum automata. Sipser's idea has been adapted to proving that reversible space equals deterministic space [11], to making tree-walking automata halt [13],



© Olga Martynova and Alexander Okhotin;

licensed under Creative Commons License CC-BY 4.0

38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021).

Editors: Markus Bläser and Benjamin Monmege; Article No. 52; pp. 52:1–52:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



to complementing 2DFA [6], to making multi-head automata reversible [12], etc. Each transformation leads to a certain blow-up in the number of states, usually between linear and quadratic. No lower bounds on the transformation to halting have been established yet. For the transformation to reversible, a lower bound exists for the case of 2DFA [8], but it is quite far from the known upper bound.

For the general case of GWA, constructions of halting, returning and reversible automata were given by Kunc and Okhotin [9], who showed that an  $n$ -state GWA operating on graphs with  $k$  edge labels can be transformed to a returning GWA with  $3nk$  states and to a reversible GWA with  $6nk + 1$  states, which is always halting. Applied to special cases of GWA, such as TWA or multi-head automata, these generic constructions produce fewer states than the earlier specialized constructions.

The goal of this paper is to obtain lower bounds on the complexity of these transformations. To begin with, the constructions by Kunc and Okhotin [9] are revisited in Section 3, and it turns out that the elements they are built of can be recombined more efficiently, resulting in improved upper bounds based on the existing methods. This way, the transformation to a returning GWA is improved to use  $2nk + n$  states, the transformation to halting can use  $2nk + 1$  states, and constructing a reversible GWA (which is both returning and halting) requires at most  $4nk + 1$  states. The main result of the paper is that, with these improvements, each of these constructions is asymptotically optimal.

The lower bounds are proved according to the following plan. For each  $n$  and  $k$ , one should construct an  $n$ -state automaton operating on graphs with  $k$  direction labels, so that any returning, halting or reversible automaton recognizing the same language would require many states. The  $n$ -state automaton follows a particular path in an input graph in search for a special node. The node is always on that path, so that the automaton naturally encounters it if it exists. On the other hand, the graph is constructed, so that getting back is more challenging.

The graph is made of elements called *diodes*, which are easy to traverse in one direction and hard to traverse backwards. Diodes are defined in Section 4, where it is shown that a GWA needs to employ extra states to traverse a diode backwards.

The graph used in all lower bound arguments, constructed in Section 5, has a main path made of diodes leading to a special node, which makes returning more complicated, so that a returning automaton needs at least  $2(n-1)(k-3)$  states. A variant of this graph containing a cycle made of diodes, presented in Section 6, poses a challenge to a halting automaton, which needs at least  $2(n-1)(k-3)$  states. Section 7 combines the two arguments to establish a lower bound of  $4(n-1)(k-3)$  on the number of states of an automaton that is returning and halting at the same time. This bound is adapted to reversible automata in Section 8: at least  $4(n-1)(k-3) - 1$  states are required.

Overall, each transformation requires ca.  $C \cdot nk$  states in the worst case, for a constant  $C$ . Each transformation has its own constant  $C$ , and these constants are determined precisely.

## 2 Graph-walking automata and their subclasses

This section provides a succinct introduction to graph-walking automata and to their halting and reversible subclasses. For more details, an interested reader is directed to the paper by Kunc and Okhotin [9], whereas some general explanations can be found in a recent survey [14].

The definition of graph-walking automata (GWA) is an intuitive extension of two-way finite automata (2DFA) and tree-walking automata (TWA). However, formalizing it requires extensive notation. First, there is a notion of a *signature*, which is a generalization of an alphabet to the case of graphs.

► **Definition 1** (Kunc and Okhotin [9]). A signature  $S$  consists of

- A finite set  $D$  of directions, that is, labels attached to edge end-points;
- A bijection  $-: D \rightarrow D$  providing an opposite direction, with  $-(-d) = d$  for all  $d \in D$ ;
- A finite set  $\Sigma$  of node labels;
- A non-empty subset  $\Sigma_0 \subseteq \Sigma$  of possible labels of the initial node;
- A set of directions  $D_a \subseteq D$  for every label  $a \in \Sigma$ . Every node labelled with  $a$  must be of degree  $|D_a|$ , with the incident edges corresponding to the elements of  $D_a$ .

Graphs are defined over a signature, like strings over an alphabet.

► **Definition 2.** A graph over a signature  $S = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  is a quadruple  $(V, v_0, +, \lambda)$ , where

- $V$  is a finite set of nodes;
- $v_0 \in V$  is the initial node;
- $+: V \times D \rightarrow V$  is a partial function, such that if  $v + d$  is defined, then  $(v + d) + (-d)$  is defined and equals  $v$ ;
- a total mapping  $\lambda: V \rightarrow \Sigma$ , such that  $v + d$  is defined if and only if  $d \in D_{\lambda(v)}$ , and  $\lambda(v) \in \Sigma_0$  if and only if  $v = v_0$ .

Once graphs are formally defined, a graph-walking automaton is defined similarly to a 2DFA.

► **Definition 3.** A (deterministic) graph-walking automaton (GWA) over a signature  $S = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  is a quadruple  $A = (Q, q_0, F, \delta)$ , where

- $Q$  is a finite set of states;
- $q_0 \in Q$  is the initial state;
- $F \subseteq Q \times \Sigma$  is a set of acceptance conditions;
- $\delta: (Q \times \Sigma) \setminus F \rightarrow Q \times D$  is a partial transition function, with  $\delta(q, a) \in Q \times D_a$  for all  $a$  and  $q$  where  $\delta$  is defined.

A computation of a GWA on a graph  $(V, v_0, +, \lambda)$  is a uniquely defined sequence of configurations  $(q, v)$ , with  $q \in Q$  and  $v \in V$ . It begins with  $(q_0, v_0)$  and proceeds from  $(q, v)$  to  $(q', v + d)$ , where  $\delta(q, \lambda(v)) = (q', d)$ . The automaton accepts by reaching  $(q, v)$  with  $(q, \lambda(v)) \in F$ .

On each input graph, a GWA can accept, reject or loop. There is a natural subclass of GWA that never loop.

► **Definition 4.** A graph-walking automaton is said to be halting, if its computation on every input graph is finite.

Another property is getting back to the initial node before acceptance: if a GWA is regarded as a robot, it returns to its hangar, and for a generic model of computation, this property means cleaning up the memory.

► **Definition 5.** A graph-walking automaton  $A = (Q, q_0, F, \delta)$  over a signature  $S = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  is called returning, if  $F \subseteq Q \times \Sigma_0$ , which means that it can accept only in the initial node.

A returning automaton is free to reject in any node, and it may also loop, that is, it need not be halting.

The next, more sophisticated property is *reversibility*, meaning that, for every configuration, the configuration at the previous step can be uniquely reconstructed. This property is essential in quantum computing, whereas irreversibility in classical computers causes energy dissipation, which is known as *Landauer's principle* [10].

The definition of reversibility begins with the property that every state is reachable from only one direction.

► **Definition 6.** A graph-walking automaton  $A = (Q, q_0, F, \delta)$  over a signature  $S = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  is called *direction-determinate*, if there is a function  $d: Q \rightarrow D$ , such that, for all  $p \in Q$  and  $a \in \Sigma$ , if  $\delta(p, a)$  is defined, then  $\delta(p, a) = (q, d(q))$  for some  $q \in Q$ .

► **Definition 7.** A graph-walking automaton  $A = (Q, q_0, F, \delta)$  over a signature  $S = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  is called *reversible*, if

- $A$  is direction-determinate;
- for all  $a \in \Sigma$  and  $q \in Q$ , there is at most one state  $p$ , such that  $\delta(p, a) = (q, d(q))$ ; in other words, knowing a state and a previous label, one can determine the previous state;
- The automaton is returning, and for each  $a_0 \in \Sigma_0$  there exists at most one such state  $q$ , that  $(q, a_0) \in F$ .

In theory, a reversible automaton may loop, but only through the initial configuration. In this case, it can be made halting by introducing an extra initial state.

Every GWA can be transformed to each of the above subclasses [9]. In the next section, the known transformations will be explained and slightly improved.

### 3 Upper bounds revisited

Before establishing the lower bounds on all transformations, the existing constructions of Kunc and Okhotin [9] will be somewhat improved by using fewer states. This is achieved by recombining the elements of the original construction, and, with these improvements, the constructions shall be proved asymptotically optimal.

All transformations are based on the following lemma.

► **Lemma 8** (Kunc and Okhotin [9, Lemma 4]). *For every direction-determinate GWA  $A$ , one can construct a reversible GWA  $B$  with twice as many states, so that for every accepting configuration  $(q, v)$  of  $A$  on a graph  $G$ , if  $B$  starts on  $G$  in  $(q, v - d(q))$ , then  $B$  reversibly traverses the tree of all computations of  $A$  that lead to the configuration  $(q, v)$ . If  $B$  ever finds the initial configuration, it accepts, and otherwise it rejects in a copy of the accepting configuration of  $A$ .*

Note that the computation of  $A$  starting from the initial configuration can reach at most one accepting configuration  $(q, v)$ , whereas for any other accepting configuration  $(q, v)$ , the automaton  $B$  will not find the initial configuration and will reject as stated in the lemma.

To transform a given  $n$ -state GWA  $\hat{A}$  over a signature with  $k$  directions to a returning automaton, Kunc and Okhotin [9] first transform it to a direction-determinate automaton  $A$  with  $nk$  states; let  $B$  be the  $2nk$ -state automaton obtained from  $A$  by Lemma 8. Then they construct an automaton that first operates as  $A$ , and then, after reaching an accepting configuration, works as  $B$  to return to the initial node. This results in a returning direction-determinate automaton with  $3nk$  states.

If the goal is just to return, and remembering the direction is not necessary, then  $2nk + n$  states are actually enough.

► **Theorem 9.** *For every  $n$ -state GWA over a signature with  $k$  directions, there exists a returning automaton with  $2nk + n$  states recognizing the same set of graphs.*

Indeed, the original automaton  $\widehat{A}$  can be first simulated as it is, and once it reaches an accepting configuration, one can use the same automaton  $B$  as in the original construction to return to the initial node. There is a small complication in the transition from  $\widehat{A}$  to  $B$ , because in the accepting configuration, the direction last used is unknown. This is handled by cycling through all possible previous configurations of  $A$  at this last step, and executing  $B$  from each of them. If the direction is guessed correctly, then  $B$  finds the initial configuration and accepts. Otherwise, if the direction is wrongly chosen,  $B$  returns back, and then, instead of rejecting, it is executed again starting from the next direction. One of these directions leads it back to the initial node.

Kunc and Okhotin [9] did not consider halting automata separately. Instead, they first transform an  $n$ -state GWA to a  $3nk$ -state returning direction-determinate automaton, then use Lemma 8 to obtain a  $6nk$ -state reversible automaton, and add an extra initial state to start it. The resulting  $(6nk + 1)$ -state automaton is always halting.

If only the halting property is needed, then the number of states can be reduced.

► **Theorem 10.** *For every  $n$ -state direction-determinate automaton, there exists a  $(2n + 1)$ -state halting and direction-determinate automaton that recognizes the same set of graphs.*

First, an  $n$ -state automaton  $\widehat{A}$  is transformed to a direction-determinate  $nk$ -state automaton  $A$ , and Lemma 8 is used to construct a  $2nk$ -state automaton  $B$ . Then, the automaton  $B$  is *reversed* by the method of Kunc and Okhotin [9], resulting in an automaton  $B^R$  with  $2nk + 1$  states that carries out the computation of  $B$  backwards. The automaton  $B^R$  is a halting automaton that recognizes the same set of graphs as  $\widehat{A}$ : it starts in the initial configuration, and if  $B$  accepts from an accepting configuration of  $A$ , then  $B^R$  finds this configuration and accepts; otherwise,  $B^R$  halts and rejects.

The construction of a reversible automaton with  $6nk + 1$  states can be improved to  $4nk + 1$  by merging the automata  $B$  and  $B^R$ . The new automaton first works as  $B^R$  to find the accepting configuration of  $A$ . If it finds it, then it continues as  $B$  to return to the initial node. In addition, this automaton halts on every input.

► **Theorem 11.** *For every  $n$ -state direction-determinate automaton there exists a  $(4n + 1)$ -state reversible and halting automaton recognizing the same set of graphs.*

With the upper bounds improved, it is time to establish asymptotically matching lower bounds.

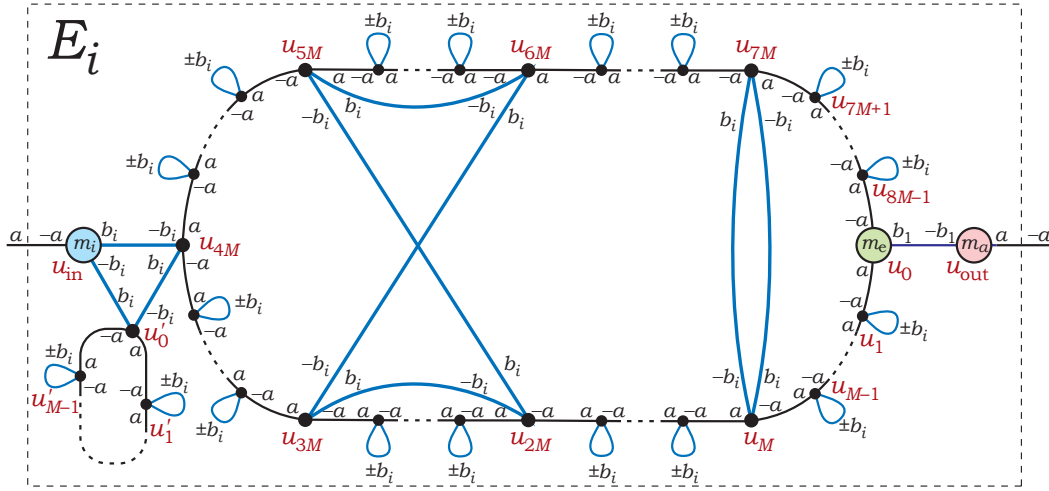
## 4 Construction of a “diode”

Lower bounds on the size of GWA obtained in this paper rely on quite involved constructions of graphs that are easy to traverse from the initial node to the moment of acceptance, whereas traversing the same path backwards is hard. An essential element of this construction is a subgraph called a *diode*; graphs in the lower bound proofs are made of such elements.

A diode is designed to replace an  $(a, -a)$ -edge. An automaton can traverse it in the direction  $a$  without changing its state. However, traversing it in the direction  $-a$  requires at least  $2(|D| - 3)$  states, where  $D$  is the set of directions in the diode’s signature.

If an automaton never moves in the direction  $-a$ , then it can be transformed to an automaton with the same number of states, operating on graphs in which every  $(a, -a)$ -edge is replaced with a diode.

Lower bound proofs for automata with  $n$  states over a signature with  $k$  directions use a diode designed for these particular values of  $n$  and  $k$ . This diode is denoted by  $\Delta_{n,k}$ .



■ **Figure 1** Element  $E_i$ . Filled circles are nodes labelled with  $m$ , each with  $r - 1$  loops in directions  $\pm b_s$ , with  $s \neq i$ .

For  $n \geq 2$  and  $k \geq 4$ , let  $M = (4nk)!$ , and let  $r = \lfloor \frac{k-2}{2} \rfloor$ . A diode  $\Delta_{n,k}$  is defined over a signature  $S_k$  that does not depend on  $n$ .

► **Definition 12.** A signature  $S_k = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  consists of:

- the set of directions  $D = \{a, -a\} \cup \{b_1, b_{-1}, \dots, b_r, b_{-r}\}$ ;
- opposite directions  $-(a) = (-a)$ ,  $-b_i = b_{-i}$ , for  $1 \leq i \leq r$ ;
- the set of node labels  $\Sigma = \{m_1, \dots, m_r\} \cup \{m_{-1}, \dots, m_{-r}\} \cup \{m, m_e, m_a\}$ , with no initial labels defined ( $\Sigma_0 = \emptyset$ ) since the diode is inserted into graphs;
- sets of directions allowed at labels:  $D_m = D$ ,  $D_{m_i} = D_{m_{-i}} = \{-a, b_i, -b_i\}$ , for  $i = 1, \dots, r$ ,  $D_{m_e} = \{b_1, a, -a\}$ ,  $D_{m_a} = \{-b_1, a\}$ .

A diode is comprised of  $2r$  elements  $E_i, E_{-i}$ , for  $i \in \{1, \dots, r\}$ . Each element  $E_i$  and  $E_{-i}$  is a graph over the signature  $S_k$ , with two external edges, one with label  $a$ , the other with  $-a$ . By these edges, the elements are connected in a chain.

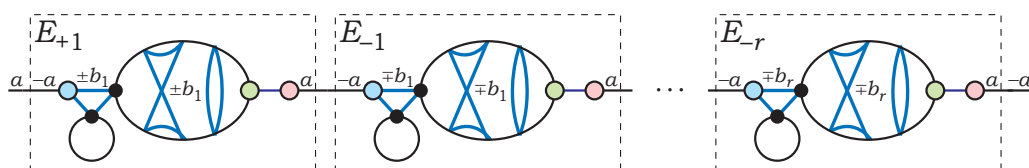
The form of an element  $E_i$  is illustrated in Figure 1. Its main part is a cycle of length  $8M$  in directions  $a, -a$ ; these are nodes  $u_0, \dots, u_{8M-1}$ , where the arithmetic in the node numbers is modulo  $8M$ , e.g.,  $u_{-1} = u_{8M-1}$ . The node numbers are incremented in direction  $a$ . Besides the main cycle, there are two extra nodes: the entry point  $u_{in}$  and the exit  $u_{out}$ , as well as a small circle of length  $M$  in directions  $a, -a$  with the nodes  $u'_0, \dots, u'_{M-1}$ . All nodes are labelled with  $m$ , except three:  $u_{in}$  with label  $m_i$  matching the index of the element,  $u_{out}$  with label  $m_a$ , and  $u_0$  has label  $m_e$ .

An element  $E_i$  has specially defined edges in directions  $b_i$  and  $-b_i$ . Each node  $u_j$  with  $j \not\equiv 0 \pmod{M}$  has a  $(b_i, -b_i)$ -loop. The nodes  $u_j$  with  $j \in \{M, 2M, 3M, 5M, 6M, 7M\}$  are interconnected with edges, as shown in Figure 1; these edges serve as traps for an automaton traversing the element backwards. The node  $u_{4M}$  has a different kind of trap in the form of a cycle  $u'_0, \dots, u'_{M-1}$ . For all  $s \neq i$ , each node labelled with  $m$  has a  $(b_s, -b_s)$ -loop.

The element  $E_{-i}$  is the same as  $E_i$ , with the directions  $b_i$  and  $-b_i$  swapped.

The diode  $\Delta_{n,k}$  is a chain of such elements, as illustrated in Figure 2. Each element can be traversed from the entrance to the exit without changing the state: at first, the automaton sees the label  $m_i$ , and accordingly moves in the direction  $b_i$ ; then, on labels  $m$ , it proceeds in the direction  $a$  until it reaches  $u_0$ , labelled with  $m_e$ . Then the automaton leaves the element by following directions  $b_1$  and  $a$ .





■ **Figure 2** Diode  $\Delta_{n,k}$ : a chain of elements  $E_1, E_{-1}, E_2, E_{-2}, \dots, E_r, E_{-r}$ .

The diode is hard to traverse backwards, because the node  $u_{4M}$  is not specifically labelled, and in order to locate it, the automaton needs to move in directions  $\pm b_i$  from many nodes, and is accordingly prone to falling into traps.

The diode is used as a subgraph connecting two nodes of a graph as if an  $(a, -a)$ -edge. For a graph  $G$  over some signature  $\tilde{S}$ , let  $G'$  be a graph obtained by replacing every  $(a, -a)$ -edge in  $G$  with the diode  $\Delta_{n,k}$ . Denote this graph operation by  $h_{n,k}: G \mapsto G'$ .

The following lemma states that if an automaton never traverses an  $(a, -a)$ -edge backwards, then its computations can be replicated on graphs with these edges substituted by diodes, with no extra states needed.

► **Lemma 13.** *Let  $\tilde{S}$  be any signature containing directions  $a, -a$ , which has no node labels from the signature  $S_k$ . Let  $A = (Q, q_0, F, \delta)$  be a GWA over the signature  $\tilde{S}$ , which never moves in the direction  $-a$ .*

*Then, there exists a GWA  $A' = (Q', q'_0, F', \delta')$  over a joint signature  $\tilde{S} \cup S_k$ , with  $|Q'| = |Q|$ , so that  $A$  accepts a graph  $G$  if and only if  $A'$  accepts the graph  $G' = h_{n,k}(G)$ .*

Lemma 13 shows that, under some conditions, a substitution of diodes can be implemented on GWA without increasing the number of states. The next lemma presents an *inverse substitution of diodes*: the set of pre-images under  $h_{n,k}$  of graphs accepted by a GWA can be recognized by another GWA with the same number of states.

► **Lemma 14.** *Let  $k \geq 4$  and  $n \geq 2$ , denote  $h(G) = h_{n,k}(G)$  for brevity. Let  $\tilde{S}$  be a signature containing the directions  $a, -a$  and no node labels from the diode's signature  $S_k$ . Let  $B$  be a GWA over the signature  $\tilde{S} \cup S_k$ . Then there exists an automaton  $C$  over the signature  $\tilde{S}$ , using the same set of states, with the following properties.*

- *For every graph  $G$  over  $\tilde{S}$ , the automaton  $C$  accepts  $G$  if and only if  $B$  accepts  $h(G)$ .*
- *If  $C$  can enter a state  $q$  by a transition in direction  $-a$ , then  $B$  can enter the state  $q$  after traversing the diode backwards.*
- *If  $B$  is returning, then so is  $C$ .*
- *If  $B$  is halting, then  $C$  is halting as well.*

The automaton  $C$  is constructed by simulating  $B$  on small graphs, and using the outcomes of these computations to define the transition function and the set of acceptance conditions of  $C$ . Note that the signatures  $\tilde{S}$  and  $S_k$  may contain any further common directions besides  $a, -a$ : this does not cause any problems with the proof, because the node labels are disjoint, and thus  $B$  always knows whether it is inside or outside a diode.

► **Lemma 15.** *Let  $A = (Q, q_0, F, \delta)$  be a GWA over a signature that includes the diode's signature  $S_k$ , with  $|Q| \leq 4nk$ . Assume that  $A$ , after traversing the diode  $\Delta_{n,k}$  backwards, can leave the diode in any of  $h$  distinct states. Then  $A$  has at least  $2h(k-3)$  states.*

**Sketch of a proof.** While moving through an element  $E_i$  backwards, the automaton sees labels  $m$  most of the time, and soon begins repeating a periodic sequence of states. Without loss of generality, assume that this periodic sequence contains more transitions in the direction

$a$  than in  $-a$ . Then the automaton reaches the node  $u_M$ , and at this point it may teleport between  $u_M$  and  $u_{-M}$  several times. Let  $w \in \{b_i, -b_i\}^*$  be the sequence of these teleportation moves, and let  $x$  be the corresponding sequence of states. Depending on the sequence  $w$ , the automaton may eventually exit the cycle to the node  $u_{in}$ , or fall into one of the traps and get back to  $u_0$ . It is proved that for the automaton to reach  $u_{in}$ , the string  $w$  must be non-empty and of even length; furthermore, if  $|w| = 2$ , then  $w = (-b_i)b_i$ .

Now consider the  $h$  backward traversals of the diode ending in some states  $p_1, \dots, p_h$ . When the traversal ending in  $p_j$  proceeds through the element  $E_i$ , the strings  $w_{i,j} \in \{b_i, -b_i\}^*$  and  $x_{i,j}$  are defined as above. Then, as the last step of the argument, it is proved that whenever  $|w_{i,j}| = 2$ , the states in  $x_{i,j}$  cannot occur in any other string  $x_{i',j'}$ . For  $w_{i,j}$  of length 4 or more, the states in  $x_{i,j}$  can repeat in other strings  $x_{i',j'}$ , but only once. It follows that there are at least  $2h(k-3)$  distinct states in these strings. ◀

## 5 Lower bound on the size of returning automata

By the construction of Kunc and Okhotin [9], as improved in Section 3, an  $n$ -state GWA over a signature with  $k$  directions can be transformed to a returning GWA with  $2nk + n$  states. A closely matching lower bound will now be proved by constructing an automaton with  $n$  states over a signature with  $k$  directions, such that every returning automaton that recognizes the same set of graphs must have at least  $2(n-1)(k-3)$  states.

The first step is a construction of a simple automaton over a signature  $\tilde{S}$  with four directions  $a, -a, b, -b$  and two graphs over this signature, so that the automaton accepts one of them and rejects the other. The automaton will have  $n$  states, it will never move in the direction  $-a$ , and every returning automaton recognizing the same set of graphs can enter  $n-1$  distinct states after transitions in the direction  $-a$ . Then, Lemma 15 shall assert that every returning automaton recognizing the same graphs with diodes substituted must have the claimed number of states.

► **Definition 16.** *The signature  $\tilde{S} = (D, -, \Sigma, \Sigma_0, (D_a)_{a \in \Sigma})$  uses the set of directions  $D = \{a, -a, b, -b\}$ , with  $-(a) = (-a)$ ,  $-(b) = (-b)$ . The set of node labels is  $\Sigma = \{c_0, c, c_l, c_r, c_{acc}\}$ , with initial labels  $\Sigma_0 = \{c_0\}$ . The allowed directions are  $D_c = D$ ,  $D_{c_0} = D_{c_l} = \{a\}$ , and  $D_{c_r} = D_{c_{acc}} = \{-a\}$ .*

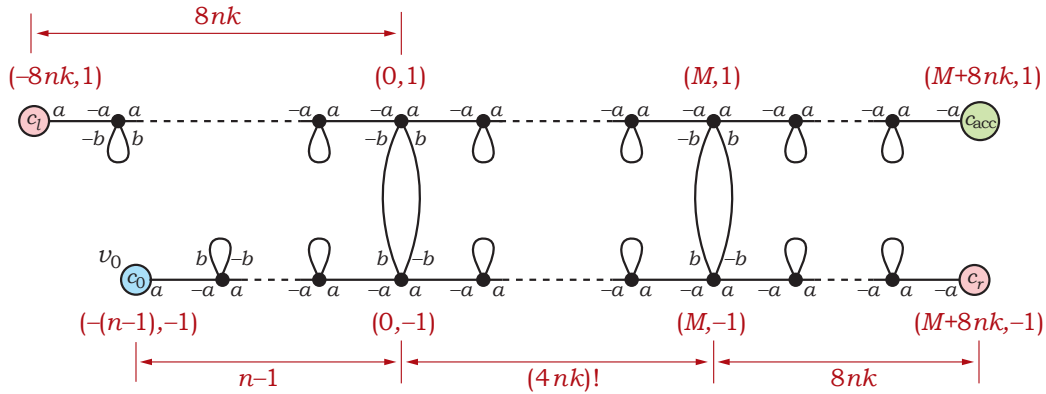
For  $n \geq 2$  and  $k \geq 4$ , let  $M = (4nk)!$  be as in the definition of the diode  $\Delta_{n,k}$ . Let  $G_{n,k}^{accept}$  and  $G_{n,k}^{reject}$  be two graphs over the signature  $\tilde{S}$ , defined as follows. The graph  $G_{n,k}^{accept}$  is illustrated in Figure 3; the other graph  $G_{n,k}^{reject}$  is almost identical, but the node that determines acceptance is differently labelled.

Both graphs consist of two horizontal chains of nodes, connected by bridges at two places. Nodes are pairs  $(x, y)$ , where  $y \in \{-1, 1\}$  is the number of the chain, and  $x$  is the horizontal coordinate, with  $-(n-1) \leq x \leq M + 8nk$  for the lower chain ( $y = -1$ ) and  $-8nk \leq x \leq M + 8nk$  for the upper chain ( $y = 1$ ).

All nodes except the ends of chains have labels  $c$ . The node  $-(n-1), -1$  is the initial node, with label  $c_0$ . The other left end  $(-8nk, 1)$  is labelled with  $c_l$ . The node  $(M + 8nk, -1)$  has label  $c_r$ . The node  $(M + 8nk, 1)$  is labelled with  $c_{acc}$  in  $G_{n,k}^{accept}$  and with  $c_r$  in  $G_{n,k}^{reject}$ ; this is the only difference between the two graphs.

The horizontal chains are formed of  $(a, -a)$ -edges, with  $a$  incrementing  $x$  and  $-a$  decrementing it. Edges with labels  $(b, -b)$  are loops at all nodes except for  $(0, 1)$ ,  $(0, -1)$ ,  $(M, 1)$  and  $(M, -1)$ . The latter four nodes form two pairs connected with bridges in directions  $(b, -b)$ .





■ **Figure 3** The graph  $G_{n,k}^{accept}$ .

An  $n$ -state automaton  $A$ , that accepts the graph  $G_{n,k}^{accept}$ , does not accept any graphs without labels  $c_{acc}$  and never moves in the direction  $-a$ , is defined as follows. In the beginning, it moves in the direction  $a$  in the same state  $q_0$ , then makes  $n - 2$  further steps in the direction  $a$ , incrementing the number of state. Next, it crosses the bridge in the direction  $b$  and enters the last,  $n$ -th state, in which it moves in the direction  $a$  until it sees the label  $c_{acc}$ .

► **Lemma 17.** *Every returning automaton that accepts the same set of graphs as  $A$ , and has at most  $4nk$  states, may enter at least  $n - 1$  distinct states after transitions in the direction  $-a$ .*

**Sketch of a proof.** On the graph  $G_{n,k}^{accept}$ , a returning automaton, after seeing the node  $(M + 8nk, 1)$ , must find its way back to the initial node. At some point, it leaves one of the ends of the upper chain,  $(-8nk, 1)$  or  $(M + 8nk, 1)$ , and then arrives at one of the ends of the lower chain,  $(-(n - 1), -1) = v_0$  or  $(M + 8nk, -1)$ . On the way, it passes through nodes labelled with  $c$ , and eventually starts behaving periodically. It is claimed that its periodic sequence of directions contains at least  $n - 1$  moves in the direction  $-a$ .

First assume that the automaton leaves the node  $(M + 8nk, 1)$ . Let  $s$  be the difference between the number of  $-a$  and  $a$  in the periodic sequence. Then  $s > 0$ , and each period the automaton shifts by  $s$  edges to the left. As the automaton passes through the nodes  $(M, \pm 1)$ , it may move to the lower chain without noticing that; but if it does so, then later at the nodes  $(0, \pm 1)$ , the sequence of transitions will move it back to the upper chain. If it stays on the same chain at  $(M, \pm 1)$ , then it will stay on it at the second time as well. If there are too few directions  $-a$ , then, on the way through  $(0, \pm 1)$ , it would not reach the node  $(-(n - 1), -1) = v_0$ , and will end up at the node  $(-8nk, 1)$ . This contradicts that assumption that the automaton has left both ends of the upper chain for good.

If the automaton leaves the node  $(-8nk, 1)$ , the argument is similar. The number  $s$  is defined, and now it must be negative. As the automaton passes through  $(0, \pm 1)$ , if it has too few directions  $-a$ , then it cannot reach  $v_0$ , and it eventually proceeds to  $(M + 8nk, 1)$ , which is again a contradiction. ◀

It remains to combine this lemma with the properties of the diode to obtain the desired theorem.

► **Theorem 18.** *For every  $k \geq 4$ , there exists a signature with  $k$  directions, such that, for every  $n \geq 2$ , there is an  $n$ -state graph-walking automaton, such that every returning automaton recognizing the same set of graphs must have at least  $2(n - 1)(k - 3)$  states.*

## 52:10 Lower Bounds for Graph-Walking Automata

**Proof.** The proof uses the automaton  $A$  defined above. By Lemma 13, the  $n$ -state automaton  $A$  over the signature  $\tilde{S}$ , is transformed to  $n$ -state automaton  $A'$  over the signature  $\tilde{S} \cup S_k$ . The directions  $\pm a$  are the same for  $\tilde{S}$  and  $S_k$ , and  $\pm b$  in  $\tilde{S}$  are merged with  $\pm b_1$  in  $S_k$ , so there are  $k$  directions in total.

For every graph  $G$ , the automaton  $A'$  accepts a graph  $h_{n,k}(G)$  with  $(a, -a)$ -edges replaced by diodes, if and only if  $A$  accepts  $G$ . The automaton  $A'$  is the desired example: it is claimed that every returning automaton  $B$  recognizing the same set of graphs as  $A'$  has at least  $2(n-1)(k-3)$  states.

Let  $B$  be any returning automaton with at most  $4nk$  states recognizing these graphs. By Lemma 14, there is an automaton  $C$  over the signature  $\tilde{S}$  and with the same number of states, which accepts a graph  $G$  if and only if  $B$  accepts  $h(G)$ . This is equivalent to  $A$  accepting  $G$ , and so  $C$  and  $A$  accept the same set of graphs. Since  $B$  is returning, by Lemma 14,  $C$  is returning too. Then, Lemma 17 asserts that the automaton  $C$  may enter  $n-1$  distinct states after moving in the direction  $-a$ .

Then, according to Lemma 14, the automaton  $B$  enters at least  $n-1$  distinct states after traversing the diode backwards. Therefore, by Lemma 15, this automaton should have at least  $2(k-3)(n-1)$  states. ◀

### 6 Lower bound on the size of halting automata

Every  $n$ -state GWA with  $k$  directions can be transformed to a halting GWA with  $2nk+1$  states, as shown in Section 3. In this section, the following lower bound for this construction is established.

► **Theorem 19.** *For every  $k \geq 4$ , there is a signature with  $k$  directions, such that for every  $n \geq 2$  there is an  $n$ -state GWA, such that every halting automaton accepting the same set of graphs has at least  $2(n-1)(k-3)$  states.*

The argument shares some ideas with the earlier proof for the case of returning automata: the signature  $\tilde{S}$ , the graphs  $G_{n,k}^{accept}$  and  $G_{n,k}^{reject}$ , and the automaton  $A$  are the same as constructed in Section 5. The proof of Theorem 19 uses the following lemma, stated similarly to Lemma 17 for returning automata.

► **Lemma 20.** *Every halting automaton  $A'$ , accepting the same set of graphs as  $A$  and using at most  $4nk$  states, must be able to enter at least  $n-1$  distinct states after transitions in the direction  $-a$ .*

**Sketch of a proof.** Consider the computation of  $A'$  on the graph  $G$ , defined by merging the nodes  $(M+8nk, 1)$  and  $(-8nk, 1)$  in  $G_{n,k}^{reject}$ , into a single node  $v_{joint}$ , with label  $c$ . The automaton  $A'$  must visit this node, because it is the only difference between  $G$  and  $G_{n,k}^{accept}$ . By the time the automaton reaches  $v_{joint}$ , it already behaves periodically, and in order to stop, it needs to visit any label other than  $c$ , that is, return to one of the end-points of the lower chain. As in Lemma 17, the automaton can reach either end-point only if the periodic sequence contains at least  $n-1$  moves in the direction  $-a$ . ◀

The proof of Theorem 19 is completed via Lemmata 13, 14 and 15, in the same way as for returning automata, only using Lemma 20 instead of Lemma 17.

## 7 Lower bound on the size of returning and halting automata

An  $n$ -state GWA over a signature with  $k$  directions can be transformed to an automaton that *both* halts on every input *and* accepts only in the initial node: a reversible automaton with  $4nk + 1$  states, described in Section 3, will do.

This section establishes a close lower bound on this transformation. The witness  $n$ -state automaton is the same as in Sections 5–6, for which Theorem 18 asserts that a returning automaton needs at least  $2(n-1)(k-3)$  states, whereas Theorem 19 proves that a halting automaton needs at least  $2(n-1)(k-3)$  states. The goal is to prove that these two sets of states must be disjoint, leading to the following lower bound.

► **Theorem 21.** *For every  $k \geq 4$ , there exists a signature with  $k$  directions, such that for every  $n \geq 2$ , there is an  $n$ -state graph-walking automaton, such that every returning and halting automaton recognizing the same set of graphs must have at least  $4(n-1)(k-3)$  states.*

As before, the automaton is obtained from  $A$  by Lemma 13. For the argument to proceed, the following property needs to be established.

► **Lemma 22** (cf. Lemma 17). *Every returning and halting automaton that recognizes the same set of graphs as  $A$ , and has at most  $4nk$  states, enters at least  $2(n-1)$  distinct states after transitions in the direction  $-a$ .*

**Sketch of a proof.** Consider any such returning and halting automaton. Since it is returning, as shown in Lemma 17, on the graph  $G_{n,k}^{accept}$ , the automaton uses a periodic sequence of states to return from  $(M + 8nk, 1)$  to  $v_0$ . Since it is at the same time halting, Lemma 20 asserts that on the graph  $G$  it uses another periodic sequence of states to escape the cycle after visiting  $v_{joint}$ . Each of these two sequences makes transitions in the direction  $-a$  in at least  $n-1$  distinct states. It remains to prove that these sequences are disjoint.

Suppose the sequences have a common element, then they coincide up to a cyclic shift. Then it is possible to modify  $G$  so that the computation coming to  $v_{joint}$  later continued as the computation on  $G_{n,k}^{accept}$ , and led to acceptance. ◀

The proof of the theorem is inferred from Lemmata 13, 14, 15 and 22, as in the earlier arguments.

## 8 Lower bound on the size of reversible automata

For the transformation of a GWA with  $n$  states and  $k$  directions to a reversible automaton,  $4nk + 1$  states are sufficient. A close lower bound shall now be established.

► **Theorem 23.** *For every  $k \geq 4$ , there exists a signature with  $k$  directions, such that for every  $n \geq 2$ , there is an  $n$ -state GWA, such that every reversible GWA recognizing the same set of graphs has at least  $4(n-1)(k-3) - 1$  states.*

**Proof.** By Theorem 21, there is such an  $n$ -state automaton  $A'$  that every returning and halting automaton recognizing the same set of graphs has at least  $4(n-1)(k-3)$  states. Suppose that there is a reversible automaton with fewer than  $4(n-1)(k-3) - 1$  states that accepts the same graphs as  $A'$ . Let  $m$  be the number of states in it. Then, by the construction of reversing a reversible automaton given by Kunc and Okhotin [9], there is a returning and halting automaton with  $m + 1$  states, that is, with fewer than  $4(n-1)(k-3)$  states. This contradicts Theorem 21. ◀

## 9 Conclusion

The new bounds on the complexity of transforming graph-walking automata to automata with returning, halting and reversibility properties are fairly tight. However, for their important special cases, such as two-way finite automata (2DFA) and tree-walking automata (TWA), the gaps between lower bounds and upper bounds are still substantial.

For an  $n$ -state 2DFA, the upper bound for making it halting is  $4n + \text{const}$  states [6]. No lower bound is known, and any lower bound would be interesting to obtain. A 2DFA can be made reversible using  $4n + 3$  states [9], with a lower bound of  $2n - 2$  states [8]; it would be interesting to improve these bounds.

The same question applies to tree-walking automata: they can be made halting [13], and, for  $k$ -ary trees, it is sufficient to use  $4kn + 2k + 1$  states to obtain a reversible automaton [9]. No lower bounds are known, and this subject is suggested for further research.

Furthermore, it would be interesting to try to apply the lower bound methods for GWA to limited memory algorithms for navigation in graphs.

---

## References

- 1 Mikolaj Bojanczyk and Thomas Colcombet. Tree-walking automata cannot be determinized. *Theor. Comput. Sci.*, 350(2-3):164–173, 2006. doi:10.1016/j.tcs.2005.10.031.
- 2 Lothar Budach. Automata and labyrinths. *Mathematische Nachrichten*, 86(1):195–282, 1978. doi:10.1002/mana.19780860120.
- 3 Yann Disser, Jan Hackfeld, and Max Klimm. Undirected graph exploration with  $\Theta(\log \log n)$  pebbles. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 25–39. SIAM, 2016. doi:10.1137/1.9781611974331.ch3.
- 4 Amr Elmasry, Torben Hagerup, and Frank Kammer. Space-efficient basic graph algorithms. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPICs*, pages 288–301. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICs.STACS.2015.288.
- 5 Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph exploration by a finite automaton. *Theor. Comput. Sci.*, 345(2-3):331–344, 2005. doi:10.1016/j.tcs.2005.07.014.
- 6 Viliam Geffert, Carlo Mereghetti, and Giovanni Pighizzini. Complementing two-way finite automata. *Inf. Comput.*, 205(8):1173–1187, 2007. doi:10.1016/j.ic.2007.01.008.
- 7 Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 66–75. IEEE Computer Society, 1997. doi:10.1109/SFCS.1997.646094.
- 8 Michal Kunc and Alexander Okhotin. Reversible two-way finite automata over a unary alphabet. Technical Report 1024, Turku Centre for Computer Science, 2011.
- 9 Michal Kunc and Alexander Okhotin. Reversibility of computations in graph-walking automata. *Inf. Comput.*, 275:104631, 2020. doi:10.1016/j.ic.2020.104631.
- 10 Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Dev.*, 5(3):183–191, 1961. doi:10.1147/rd.53.0183.
- 11 Klaus-Jörn Lange, Pierre McKenzie, and Alain Tapp. Reversible space equals deterministic space. *J. Comput. Syst. Sci.*, 60(2):354–367, 2000. doi:10.1006/jcss.1999.1672.
- 12 Kenichi Morita. A deterministic two-way multi-head finite automaton can be converted into a reversible one with the same number of heads. In Robert Glück and Tetsuo Yokoyama, editors, *Reversible Computation, 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012. Revised Papers*, volume 7581 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2012. doi:10.1007/978-3-642-36315-3\_3.

- 13 Anca Muscholl, Mathias Samuelides, and Luc Segoufin. Complementing deterministic tree-walking automata. *Inf. Process. Lett.*, 99(1):33–39, 2006. doi:10.1016/j.ip1.2005.09.017.
- 14 Alexander Okhotin. Graph-walking automata: From whence they come, and whither they are bound. In Michal Hospodár and Galina Jirásková, editors, *Implementation and Application of Automata - 24th International Conference, CIAA 2019, Košice, Slovakia, July 22-25, 2019, Proceedings*, volume 11601 of *Lecture Notes in Computer Science*, pages 10–29. Springer, 2019. doi:10.1007/978-3-030-23679-3\_2.
- 15 Michael Sipser. Lower bounds on the size of sweeping automata. *J. Comput. Syst. Sci.*, 21(2):195–202, 1980. doi:10.1016/0022-0000(80)90034-3.