Lower Bounds for Symbolic Computation on Graphs: Strongly Connected Components, Liveness, Safety, and Diameter

Krishnendu Chatterjee*   Wolfgang Dvořák†   Monika Henzinger‡

Veronika Loitzenbauer§

Abstract

A model of computation that is widely used in the formal analysis of reactive systems is *symbolic algorithms*. In this model the access to the input graph is restricted to consist of *symbolic operations*, which are expensive in comparison to the standard RAM operations. We give lower bounds on the number of symbolic operations for basic graph problems such as the computation of the strongly connected components and of the approximate diameter as well as for fundamental problems in model checking such as safety, liveness, and co-liveness. Our lower bounds are linear in the number of vertices of the graph, even for constant-diameter graphs. For none of these problems lower bounds on the number of symbolic operations were known before. The lower bounds show an interesting separation of these problems from the reachability problem, which can be solved with $O(D)$ symbolic operations, where D is the diameter of the graph.

Additionally we present an approximation algorithm for the graph diameter which requires $\tilde{O}(n\sqrt{D})$ symbolic steps to achieve a $(1 + \epsilon)$-approximation for any constant $\epsilon > 0$. This compares to $O(n \cdot D)$ symbolic steps for the (naive) exact algorithm and $O(D)$ symbolic steps for a 2-approximation. Finally we also give a refined analysis of the strongly connected components algorithms of [15], showing that it uses an optimal number of symbolic steps that is proportional to the sum of the diameters of the strongly connected components.

1 Introduction

Graph algorithms are central in the formal analysis of reactive systems. A reactive system consists of a set of variables and a state of the system corresponds to a set of valuations, one for each of these variables. This naturally induces a directed graph: Each vertex represents a state of the system and each directed edge represents a state transition that is possible in the system. As the number of vertices is exponential in the number of variables of the system, these graphs are huge and, thus, they are usually not explicitly represented during their analysis. Instead they are *implicitly represented* using e.g., binary-decision diagrams (BDDs) [6, 7]. To avoid considering specifics of the implicit representation and their manipulation, an elegant theoretical model for algorithms that work on this implicit representation has been developed, called *symbolic algorithms* (see e.g. [8, 13, 24, 14, 12, 15, 9]). In this paper we will give novel upper and (unconditional) lower bounds on the number of operations required by a symbolic algorithm for solving classic graph-algorithmic questions, such as computing the strongly connected components and the (approximate) diameter, as well as for graph-algorithmic questions that are important in the analysis of reactive systems, such as safety, liveness, and co-liveness objectives. Our lower bounds are based on new reductions of problems from communication complexity to symbolic algorithms.

Symbolic algorithms. A symbolic algorithm is allowed to use the same mathematical, logical, and memory access operations as a regular RAM algorithm, except for the access to the input graph: It is not given access to the input graph through an adjacency list or adjacency matrix representation but instead *only* through two types of *symbolic operations*:

1. *One-step operations Pre and Post:* Each *predecessor* (*Pre*) (resp., *successor* (*Post*)) operation is given a set X of vertices and returns the set of vertices Y with an edge to (resp., edge from) some vertex of X.

2. *Basic set operations:* Each basic set operation is given one or two sets of vertices and performs a union, intersection, or complement on these sets.

An initial set of vertices is given as part of the input, often consisting of a single vertex.

Symbolic operations are more expensive than the non-symbolic operations and thus one is mainly interested in the number of symbolic operations of such an algorithm (and the exact number of non-symbolic operations is often neglected). Moreover, as the symbolic model is motivated by the compact representation of huge graphs, we aim for symbolic algorithms that only store $O(1)$ or $O(\log n)$ many sets of vertices as otherwise

---
*IST Austria.

†TU Wien, Institute of Information Systems, Vienna, Austria.

‡University of Vienna, Faculty of Computer Science, Vienna, Austria

§University of Vienna, Faculty of Computer Science, Vienna, Austria and Bar-Ilan University, Israel

algorithms become impractical due to the huge space requirements. Additionally, every computable graph-algorithmic question can be solved with $2n$ symbolic one-step operations when storing $O(n)$ many sets (and allowing an unbounded number of non-symbolic operations): For every vertex v perform a *Pre* and a *Post* operation, store the results, which represent the full graph, and then compute the solution on this graph, using only non-symbolic operations. Note, however, that our lower bounds do not depend on this requirement, i.e., they also apply to symbolic algorithms that store an arbitrary number of sets. Furthermore the basic set operations (that only require vertices, i.e., the current state variables) are computationally much less expensive than the one-step operations (that involve both variables of the current and of the next state). Thus, to simplify the analysis of symbolic algorithms, we only analyze the number of one-step operations in the lower bounds that we present. For all upper bounds in prior work and in our work the number of basic-set operations is at most linear in the number of one-step operations.

There is an interesting relationship between the two types of symbolic operations and Boolean matrix-vector operations: Interpreting the edge relationship as a Boolean matrix and a vertex set as a Boolean vector, the one-step operations correspond to (left- and right-sided) matrix-vector multiplication, where the matrix is the adjacency matrix, and basic set operations correspond to basic vector manipulations. Thus, an equivalent way of interpreting symbolic algorithms is by saying that the access to the graph is only allowed by performing a Boolean matrix-vector multiplication, where the vector represents a set of vertices and the matrix is the adjacency matrix.

Note also that there is a similarity to the CONGEST and the LOCAL model in synchronous distributed computation, as in these models each vertex in a synchronous network knows all its neighbors and can communicate with all of them in one round (exchanging $O(\log n)$ bits in the CONGEST model), and the algorithmic complexity is measured in rounds of communication. While in these models all neighbors of every individual vertex, i.e., all *edges* of the graph, can be determined in one round of communication, in the symbolic model this might require n *Pre* and n *Post* operations, each on an singleton set. Thus, determining (and storing) all edges of a symbolically represented graph is expensive and we would ideally like to have algorithms that use sub-linear (in the number of vertices) many symbolic one-step operations.

Objectives. First we formally introduce the most relevant graph-algorithmic questions from the analysis of reactive systems [21]. Given a graph $G = (V, E)$ and a starting vertex $s \in V$, let $\mathcal{P}_s$ be the set of infinite paths in G starting from s. Each objective corresponds to a set of requirements on an infinite path and the question that needs to be decided by the algorithm is whether there is a path in $\mathcal{P}_s$ that satisfies these requirements, in which case we say the path *satisfies the objective.* An objective A is the *dual* of an objective B if a path satisfies A iff it does not satisfy B.

Let $T \subseteq V$ be a set of target vertices given as input. The most basic objective is *reachability* where an infinite path satisfies the objective if the path visits a vertex of T *at least once.* The dual *safety* objective is satisfied by infinite paths that only visit vertices of T. The next interesting objective is the *liveness* (aka *Büchi*) objective that requires an infinite path to visit some vertex of T *infinitely often.* The dual *co-liveness* (aka *co-Büchi*) objective requires an infinite path to eventually only visit vertices in T. Verifying these objectives are the most fundamental graph-algorithmic questions in the analysis of reactive systems.

Computing the strongly connected components (SCCs) is at the heart of the fastest algorithms for liveness and co-liveness: For example, there is a reduction from liveness to the computation of SCCs that takes symbolic steps in the order of the diameter of the graph. Thus, determining the symbolic complexity of SCCs also settles the symbolic complexity of liveness.

Furthermore, the diameter computation plays a crucial role in applications such as bounded model-checking [4], where the goal is to analyze the system for a bounded number of steps, and it suffices to choose the diameter of the graph as bound. Second, in many scenarios, such as in hardware verification, the graphs have small diameter, and hence algorithms that can detect if this is the case and then exploit the small diameter are relevant [4]. Motivated by these applications, we define the diameter of a graph as the largest finite distance in the graph, which coincides with the usual graph-theoretic definition on strongly connected graphs and is more general otherwise.

Note that linear lower bounds for the number of symbolic operations are non-trivial, since a one-step operation can involve *all* edges. For example, to determine all the neighbors of a given vertex v takes one symbolic operation, while it takes $O(\deg(v))$ many operations in the classic setting. In the following we use n to denote the number of vertices of a graph G and $D(G) = D$ to denote its diameter.

Previous results. To the best of our knowledge, no previous work has established lower bounds for symbolic computation.

There is some prior work on establishing upper bounds on the number of symbolic operations: In [15] a

Table 1: Bounds on the number of required symbolic operations for different tasks. $\Theta(n)$ bounds hold even for graphs with constant diameter D.

Reach	SCC	Safety	Büchi	coBüchi
$\Theta(D)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$

Table 2: Bounds on the number of symbolic operations for approximating the diameter of a graph. The lower bounds even hold for strongly connected graphs with constant diameter D.

approx.	exact	$1 + \varepsilon$	$3/2 - \varepsilon$	2
upper b.	$O(n \cdot D)$	$\widetilde{O}(n\sqrt{D})$	$\widetilde{O}(n\sqrt{D})$	$O(D)$
lower b.	$\Omega(n)$	$\Omega(n)$	$\Omega(n)$	

symbolic algorithm that computes the SCCs with $O(n)$ symbolic operations is presented. This algorithm leads to an algorithm for liveness and co-liveness with $O(n)$ symbolic operations and improves on earlier work by [5], which requires $O(n \log n)$ symbolic operations.

Note that for the reachability objective the straightforward algorithm requires $O(D)$ symbolic operations: Starting from the set containing only the start vertex s, repeatedly perform a *Post*-operation until T is reached. For safety the straightforward algorithm takes $O(n)$ symbolic operations: Iteratively remove from F vertices that do not have an outgoing edge to another vertex of F, i.e., vertices of $F \setminus \mathsf{Pre}(F)$, until a fixed point is reached.

Finally, there is a trivial algorithm for computing the diameter of the graph: Simply determine the depth of a breadth-first search from every vertex and output the maximum over all depths. Computing the depth of a breadth-first search can be done with $O(D)$ many symbolic steps, thus this requires $O(nD)$ many symbolic steps in total. In a strongly connected graph a 2-approximation of the diameter of the graph can be obtained by computing one breadth-first search from and one to some arbitrary vertex and output the sum of the depths. This takes $O(D)$ symbolic steps.

Our contributions. Our main contributions are novel lower bounds for the number of symbolic operations for many of the above graph-algorithmic questions, leading to an interesting separation between seemingly similar problems.

1. For reachability objectives, the basic symbolic algorithm requires $O(D)$ symbolic operations. Quite surprisingly, we show that such diameter-based upper bounds are *not possible* for its dual problem, namely safety, and are also not possible for liveness and co-liveness objectives. Specifically, we present tight lower bounds to show that, even for constant-diameter graphs, $\Omega(n)$ one-step symbolic operations are required for safety, liveness, and co-liveness objectives. In addition we establish tight bounds for symbolic operations required for the computation of SCCs, showing a lower bound of $\Omega(n)$ for constant-diameter graphs. See Table 1 for a summary of these results.

2. We show that even for strongly-connected constant-diameter graphs approximating the diameter re-

quires $\Omega(n)$ symbolic steps. More precisely, any $(3/2 - \varepsilon)$-approximation algorithm requires $\Omega(n)$ symbolic one-step operations, even for undirected and connected graphs with constant diameter. We also give a novel upper bound: We present a $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$ that takes $\widetilde{O}(n\sqrt{D})$ symbolic steps. This can be compared to the trivial $O(D)$ 2-approximation algorithm and the $O(nD)$ exact algorithm. Notice that for explicitly represented graphs the approximation of the diameter is already hard for constant-diameter graphs while in the symbolic model there exists a trivial $O(n)$ upper bound in this case, thus showing a lower bound of $\Omega(n)$ is non-trivial. See Table 2 for a summary of these results.

3. Finally we give a refined analysis of the number of symbolic steps required for computing strongly connected components based on a different problem parameter. Let $SCCs(G)$ be the set of all SCCs of G and D_C the diameter of the strongly connected component C. We give matching upper and lower bounds showing that the SCCs can be computed with $\Theta(\sum_{C \in SCCs(G)}(D_C + 1))$ symbolic steps. Note that $\sum_{C \in SCCs(G)}(D_C + 1)$ can be a factor n larger than $D(G)$.

Key technical contribution. Our key technical contribution is based on the novel insight that lower bounds for communication complexity can be used to establish lower bounds for symbolic computation. We feel that this connection is of interest by itself and might lead to further lower bounds for symbolic algorithms.

Our lower bounds are by two kinds of reductions, both from the communication complexity problem of Set Disjointness with k elements. First, we give reductions that construct graphs such that one-step operations can be computed with $O(1)$ bits of communication between Alice and Bob and thus allow for linear lower bounds on the number symbolic operations. Second, we give a reduction that constructs a graph with only $\sqrt{k}$ many vertices, i.e., $n = \sqrt{k}$, but allows one-step operations to require $O(n)$ bits of communication. This again results in linear lower bounds on the number of symbolic operations.

2 Preliminaries

Symbolic Computation. We consider symbolic computation on graphs. Given an input graph $G = (V, E)$ and a set of vertices $S \subseteq V$, the graph G can be accessed only by the following two types of operations:

1. *Basic set operations* like $\cup, \cap, \setminus, \subseteq$, and $=$;
2. *One-step operations* to obtain the predecessors or successors of the vertices of S in G. In particular we define the operations

$$\mathsf{Pre}(S) = \{v \in V \mid \exists s \in S : (v, s) \in E\} \quad \text{and}$$
$$\mathsf{Post}(S) = \{v \in V \mid \exists s \in S : (s, v) \in E\}.$$

In the applications the basic set operations are much cheaper as compared to the one-step operations. Thus we aim for lower bounds on the number of one-step operations, while not accounting for set operations. In all our upper bounds the number of set operations is at most of the same order as the number of one-step operations. Note that there is a one-to-one correspondence between a one-step operation and a Boolean matrix-vector multiplication with the adjacency matrix and that for undirected graphs Pre and Post are equivalent.

Communication Complexity Lower Bound for Set Disjointness. Our lower bounds are based on the known lower bounds for the communication complexity of the *Set Disjointness* problem. The classical symmetric two-party communication complexity model is as follows [19]. There are three finite sets X, Y, Z, the former two are possible inputs for a function $f : X \times Y \to Z$, where the actual input $x \in X$ is only known by Alice, and the actual input $y \in Y$ is only known by Bob. Alice and Bob want to evaluate a function $f(x, y)$ while sending as few bits as possible to each other. The communication happens according to a fixed protocol, known to both players beforehand, that determines which player sends which bits when, and when to stop.

Set Disjointness. In the Set Disjointness problem we have a universe $U = \{0, 1, \ldots, k-1\}$ of k elements and both sets X, Y contain all bit vectors of length k, i.e., they represent all possible subsets of U and are of size 2^k. Alice has a vector $x \in X$ and Bob has a vector $y \in Y$, and the function f is defined as $f(x, y) = 1$ if for all $0 \leq i \leq k-1$ either $x_i = 0$ or $y_i = 0$, and $f(x, y) = 0$ otherwise. We will sometimes use S_x and S_y to denote the sets corresponding to the vectors x and y, i.e., $S_x = \{i \mid x_i = 1\}$ and $S_y = \{i \mid y_i = 1\}$ and $f(x, y) = 1$ iff $S_x \cap S_y = \emptyset$. We next state a fundamental lower bound for the communication complexity of the Set Disjointness problem which will serve as basis for our lower bounds on the number of symbolic operations.

THEOREM 2.1. ([18, 22, 3, 16, 19]) *Any (probabilistic bounded error or deterministic) protocol for the Set Disjointness problem sends $\Omega(k)$ bits in the worst case over all inputs.*

3 Lower Bounds

In this section we present our lower bounds, which are the main results of the paper.

3.1 Lower Bounds for Computing Strongly Connected Components.

We first consider the problem of computing the strongly connected components (SCCs) of a symbolically represented graph. The best known symbolic algorithm is by Gentilini et al. [15] and computes the SCCs of a Graph G with $O(\min(n, D \cdot |SCCs(G)|))$ symbolic one-step operations and thus matches the linear running time of the famous Tarjan algorithm [25] in the non-symbolic world.

We provide lower bounds showing that the algorithm is essentially optimal, in particular we show that $O(D)$ algorithms are impossible. These lower bounds are by reductions from the communication complexity problem of Set Disjointness to computing SCCs in a specific graph. In particular, we show that any algorithm that computes SCCs with $o(n)$ symbolic one-step operations would imply a communication protocol for the Set Disjointness problem with $o(k)$ communication.

REDUCTION 3.1. *Let (x, y) be an instance of Set Disjointness and let w.l.o.g. $k = \ell \cdot \bar{k}$ for some integers $\ell, \bar{k}$. We construct a directed graph $G = (V, E)$ with $n = k + \ell$ vertices and $O(n^2)$ edges as follows. (1) The vertices are given by $V = \bigcup_{i=0}^{\ell-1} V_i$ with $V_i = \{v_{i,0}, \ldots, v_{i,\bar{k}}\}$. (2) There is an edge from $v_{i,j}$ to $v_{i',j'}$ if either $i < i'$ or $i = i'$ and $j < j'$. (3) For $0 \leq i < \ell$, $0 \leq j < \bar{k}$ there is an edge from $v_{i,j+1}$ to $v_{i,j}$ iff $x_{i \cdot \bar{k}+j} = 0$ or $y_{i \cdot \bar{k}+j} = 0$.*

In our communication protocol both Alice and Bob compute the number of SCCs on the graph from Reduction 3.1, according to a given algorithm. While both know all the vertices of the graph, they do not know all the edges (some depend on both x and y) and thus whenever such an edge is relevant for the algorithm, Alice and Bob have to communicate with each other. We show that the graph is constructed such that for each subset $S \subseteq V$ the operations $\mathsf{Pre}(S)$ and $\mathsf{Post}(S)$ can be computed with only four bits of communication between Alice and Bob.

THEOREM 3.1. *Any (probabilistic bounded error or deterministic) symbolic algorithm that computes the SCCs of graphs with n vertices needs $\Omega(n)$ symbolic one-step operations. Moreover, for a graph with the set*
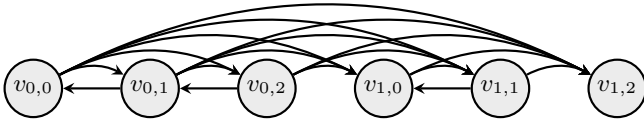
Figure 1: Reduction 3.1 for $k = 4, \ell = 2, S_x = \{2,3\}, S_y = \{0,1,3\}$

$SCCs(G)$ of SCCs and diameter D any algorithm needs $\Omega(|SCCs(G)| \cdot D)$ symbolic one-step operations.

We first show that Reduction 3.1 is a valid reduction from the Set Disjointness problem to an SCC problem. The missing proofs are given in Section 6.1.

LEMMA 3.1. $f(x,y) = 1$ iff the graph constructed in Reduction 3.1 has exactly ℓ SCCs.

The critical observation for the proof of Theorem 3.1 is that any algorithm that computes SCCs with N many symbolic one-step operations implies the existence of a communication protocol for Set Disjointness that only requires $O(N)$ communication.

LEMMA 3.2. For any algorithm that computes SCCs with N symbolic one-step operations there is a communication protocol for Set Disjointness that requires $O(N)$ communication.

Proof. In our communication protocol both Alice and Bob consider the graph from Reduction 3.1. We call edges of the graph that are present independently of x and y *definite* edges and edges whose presence depends on x and y *possible* edges.

Both Alice and Bob execute the given symbolic algorithm to decide whether the graph has ℓ SCCs (cf. Lemma 3.1). As both know all the vertices, they can execute set operations without communicating. Communication is only needed when executing symbolic one-step operations, since for these some of the possible edges might affect the outcome.

We next argue that each symbolic one-step operations can be executed with a constant number of bits of communication. First notice that as both Alice and Bob execute the same algorithm simultaneously, they both know the input set to an operation and they only need communication about the possible edges that can change the output. Both can independently identify these possible edges and they can decide whether such an edge exists by sending one bit each. We next argue that for each one-step operation we need to consider at most two possible edges. For this we consider the vertices $v_{i,j}$ in their linear ordering given by $i \cdot (\bar{k}+1)+j$, e.g., $v_{0,0} = v_0$ and $v_{\ell-1,\bar{k}} = v_{k+\ell-1}$.

Post operation: Let S be the input set and let $v_{\min}$ the vertex with the *minimum* index in S. Then we have $\{v_{\min+1}, \ldots v_{k+\ell-1}\} \subseteq \mathsf{Post}(S)$ and potentially also $v_{\min}$ and $v_{\min-1}$ can be in $\mathsf{Post}(S)$, but no other vertices. That is, we have $\{v_{\min+1}, \ldots v_{k+\ell-1}\} \subseteq \mathsf{Post}(S) \subseteq \{v_{\min-1}, \ldots v_{k+\ell-1}\}$. To decide whether $v_{\min}$ is in $\mathsf{Post}(S)$, we first check whether $v_{\min+1} \in S$ and if so we check whether the edge $(v_{\min+1}, v_{\min})$ is present. To decide $v_{\min-1} \in \mathsf{Post}(S)$, we check whether the edge $(v_{\min}, v_{\min-1})$ is present. That is, to compute $\mathsf{Post}(S)$ we only access two possible edges.

Pre operation: Let S be the input set and let $v_{\max}$ the vertex with the *maximum* index in S. Then we have $\{v_0, \ldots v_{\max-1}\} \subseteq \mathsf{Pre}(S)$ and potentially also $v_{\max}$ and $v_{\max+1}$ can be in $\mathsf{Pre}(S)$, but no other vertices. That is, we have $\{v_0, \ldots v_{\max-1}\} \subseteq \mathsf{Pre}(S) \subseteq \{v_0, \ldots v_{\max+1}\}$. To decide whether $v_{\max}$ is in $\mathsf{Pre}(S)$, we first check whether $v_{\max-1} \in S$ and if so we check whether the edge $(v_{\max}, v_{\max-1})$ is present. To decide if $v_{\max+1} \in \mathsf{Pre}(S)$, we check whether the edge $(v_{\max+1}, v_{\max})$ is present. That is, we can compute $\mathsf{Pre}(S)$ with accessing only two possible edges.

By the above we have that a symbolic algorithm with N one-step operations gives rise to a communication protocol for Set Disjointness with $O(N)$ bits of communication. $\square$

By Lemma 3.2 we have that any algorithm computing SCCs with $o(n)$ symbolic one-step operations would contradict Theorem 2.1. Now inspecting the graph of Reduction 3.1, we observe that its diameter is equal to $\bar{k}$, which leads to the following lower bounds. For $\ell = k/2$ the graph has diameter 2 and thus the $\Omega(n)$ holds even for graphs of constant diameter. On the other side, for $\ell = 1$ disjoint sets S_x and S_y correspond to strongly connected graphs and thus the $\Omega(n)$ lower bounds also holds for graphs with a bounded number of SCCs, i.e., there are no $O(|SCCs(G)|)$ symbolic algorithms. Finally for $\ell = \sqrt{k}$ we obtain a $\Omega(|SCCs(G)| \cdot D)$ lower bound.

REMARK 3.1. *By the above no algorithm can compute SCCs with $f(D) \cdot n^{o(1)}$ or $f(|SCCs(G)|) \cdot n^{o(1)}$ symbolic one-step operations for any function f. In contrast, if we consider both parameters simultaneously, there is an $O(|SCCs(G)| \cdot D)$ symbolic algorithm.*

The above lower bounds for computing SCCs match the $O(\min(n, D \cdot |SCCs(G)|))$ bound by the algorithm of Gentilini et al. [15]. One way to further improve the algorithm is to not consider the diameter of the whole graph but the diameter D_C of each single SCC C. In that direction the previous reduction already gives us an $\Omega(\sum_{C \in SCCs(G)}(D_C))$ lower bound and we will next improve it to an $\Omega(\sum_{C \in SCCs(G)}(D_C + 1))$ lower
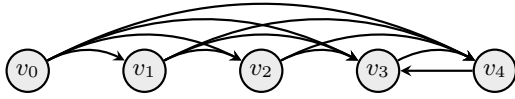
Figure 2: Reduction 3.2 for $k = 4, \ell = 2, S_x = \{2,3\}, S_y = \{0,1,3\}$

bound (i.e., it is $\Omega(n)$ even if $\sum_{C \in SCCs(G)}(D_C) \in O(1)$). These two bounds differ if the graph has a large number of trivial SCCs. Thus we next give a reduction that constructs a graph that has only trivial SCCs if S_x and S_y are disjoint.

REDUCTION 3.2. *Given an instance (x, y) of Set Disjointness, we construct a directed graph $G = (V, E)$ with $n = k + 1$ vertices and $O(n^2)$ edges as follows. (1) The vertices are given by $V = \{v_0, v_1, \ldots, v_k\}$. (2) There is an edge from v_i to v_j for $i < j$. (3) For $0 \le j \le k - 1$ there is an edge from v_{j+1} to v_j iff $x_j = 1$ and $y_j = 1$.*

THEOREM 3.2. *Any (probabilistic bounded error or deterministic) symbolic algorithm that computes the SCCs needs $\Omega(|SCCs(G)| + \sum_{C \in SCCs(G)} D_C)$ symbolic one-step operations.*

3.2 Lower Bounds for Liveness, Reachability, and Safety Objectives. In this section we extend our lower bounds for SCC computation to Liveness, Reachability, and Safety Objectives on graphs.

Lower Bounds for Reachability. The lower bounds for Reachability are an immediate consequence from our lower bounds for SCC computation in Theorem 3.1. When setting $\ell = 1$ in Reduction 3.1 then the vertex $v_{0,0}$ is reachable from all vertices iff the graph is strongly connected iff the sets S_x and S_y are disjoint.

THEOREM 3.3. *Any (probabilistic bounded error or deterministic) symbolic algorithm that solves Reachability in graphs with diameter D requires $\Omega(D)$ symbolic one-step operations.*

Lower Bounds for Liveness. To show an $\Omega(n)$ lower bound for Liveness objectives which holds even for graphs of bounded diameter, we introduce another reduction. This reduction is again from the Set Disjointness Problem and also constructs a graph such that Pre and Post operations can be executed with a constant number of bits of communication between Alice and Bob.

REDUCTION 3.3. *Given an instance (x, y) of Set Disjointness, we construct a directed graph $G = (V, E)$ with $n = k + 1$ vertices and $O(n^2)$ edges as follows. (1) The*

vertices are given by $V = \{v_0, v_1, \ldots, v_k\}$. *(2) There is an edge from v_i to v_j for $i < j$ and there is a loop edge (v_k, v_k). (3) For $0 \le j \le k - 1$ there is a loop edge (v_j, v_j) iff $x_j = 1$ and $y_j = 1$.*

Notice that the graph in Reduction 3.3 has diameter $D = 1$ and thus allows to show the lower bounds stated in Theorem 3.4 when considering $T = \{v_0, v_1, \ldots, v_{k-1}\}$, with the exception of (2) which is by Reduction 3.1 and $T = \{v_0\}$.
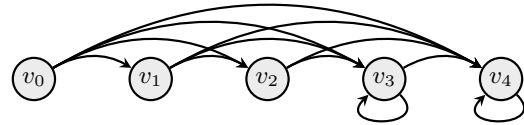

Figure 3: Reduction 3.3 for $k = 4, \ell = 2, S_x = \{2,3\}, S_y = \{0,1,3\}$

THEOREM 3.4. *For any (probabilistic bounded error or deterministic) symbolic algorithm that solves Büchi(T) the following lower bounds on the required number of symbolic one-step operations hold: (1) $\Omega(n)$ even for instances with constant D; (2) $\Omega(D)$ even for instances with $|T| = 1$; (3) $\Omega(|T|)$ even for instances with constant D; (4) $\Omega(|T| + D)$; and (5) $\Omega(|SCCs(G)| + \sum_{C \in SCCs(G)} D_C)$.*

Lower Bounds for co-Liveness and Safety. The following lower bounds are by Reduction 3.3 (and variations of it) and the set of safe vertices $T = \{v_0, v_1, \ldots, v_{k-1}\}$.

THEOREM 3.5. *For any (probabilistic bounded error or deterministic) symbolic algorithm that solves Safe(T) or coBüchi(T) the following lower bounds on the required number of symbolic one-step operations hold: (1) $\Omega(n)$ even for constant diameter graphs; (2) $\Omega(|T|)$ even for constant diameter graphs; and (3) $\omega(\sum_{C \in SCCs(G)}(D_C + 1))$ even for constant diameter graphs.*

Notice that the parameters diameter, number of SCCs, or diameters of SCCs do not help in the case of Safety. This is because every graph can be reduced to a strongly connected graph with diameter 2 without changing the winning set as follows: Add a new vertex v that has an edge to and an edge from all original vertices but do not add v to the safe vertices T.

We complete this section with a $\Omega(D)$ lower bound for coBüchi(T) which is by a variant of Reduction 3.1.

PROPOSITION 3.1. *Any (probabilistic bounded error or deterministic) symbolic algorithm that solves coBüchi(T) on graphs with diameter D needs $\Omega(D)$ symbolic one-step operations.*

3.3 Lower Bound for Approximate Diameter.

The Approximate Diameter Problem. Let $G = (V, E)$ be a directed graph with n vertices V and m edges E. Let $d(u, v)$ denote the shortest distance from $u \in V$ to $v \in V$ in G, i.e., the smallest number of edges of any path from u to v in G. Recall that we define the *diameter* of G as the maximum of $d(u, v)$ over all pairs u, v for which u can reach v in G. [1] We consider the problem of approximating the diameter D of a graph by a factor c, where the goal is to compute an estimate $\widetilde{D}$ such that $D/c \leq \widetilde{D} \leq D$. As undirected graphs are special cases of directed graphs, the lower bound is presented for undirected graphs and the upper bound for directed graphs (see Section 4), i.e., both hold for undirected and directed graphs.

Result. We show a lower bound of $\Omega(n)$ on the number of symbolic steps needed to distinguish between a diameter of 2 and a diameter of 3, even in an undirected connected graph. The basic symbolic algorithm for computing the diameter exactly takes $O(n \cdot D)$ many symbolic steps. Thus our lower bound is tight for constant-diameter graphs.

Outline Lower Bound. We show how to encode an instance of the Set Disjointness Problem with a universe of size k in an (undirected, connected) graph with $\Theta(\sqrt{k})$ vertices and $\Theta(k)$ edges such that 1) in a communication protocol any symbolic one-step operation can be simulated with $\Theta(\sqrt{k})$ bits and 2) the graph has diameter 2 if the two sets are disjoint and diameter 3 otherwise. Thus the communication complexity lower bound of $\Omega(k)$ for Set Disjointness implies a lower bound of $\Omega(\sqrt{k}) = \Omega(n)$ for the number of symbolic one-step operations to compute a $(3/2 - \varepsilon)$-approximation of the diameter of a graph with n vertices.

REDUCTION 3.4. *Let (x, y) be an instance of the Set Disjointness problem of size k and let $s = \sqrt{k}$. We construct an undirected graph $G = (V, E)$ with $n = 3s + 2$ vertices and $O(k)$ edges as follows. (1) There are three sets A, B, C with s vertices each and two auxiliary vertices u and t. We denote the i-th vertex of each of A, B, C with a lowercase letter indicating the set and subscript i. (2) There is an edge between u and t and between u and each vertex of A and B and between t and each vertex of C. (3) For each $0 \leq i < s$ there is*
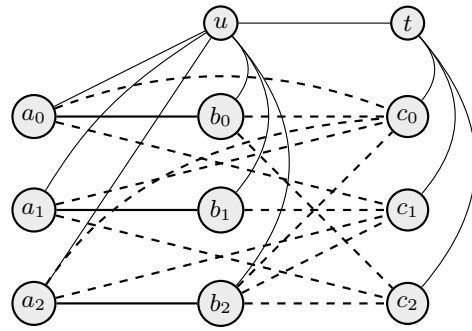

Figure 4: Reduction 3.4 for $k = 9, s = 3, S_x = \{2, 4, 8\}, S_y = \{1, 3, 5\}$.

an edge between $a_i \in A$ and $b_i \in B$. (4) For $0 \leq \ell < k$ let $i, j < s$ be such that $\ell = i \cdot s + j$. There is an edge between $a_i \in A$ and $c_j \in C$ iff $x_\ell = 0$ and there is an edge between $b_i \in B$ and $c_j \in C$ iff $y_\ell = 0$.

We first show that this graph has diameter 2 if S_x and S_y are disjoint and diameter 3 otherwise and then show how Alice can obtain a communication protocol for the Set Disjointness problem from any symbolic algorithm that can distinguish these two cases.

LEMMA 3.3. *Let $G = (V, E)$ be the graph given by Reduction 3.4 and let D denote its diameter. If $S_x \cap S_y = \emptyset$, then $D = 2$, otherwise $D = 3$.*

In the graph $G = (V, E)$ given by Reduction 3.4 Alice knows all the vertices of the graph and all the edges except those who are constructed based on y, i.e., Alice does not know the edges between B and C. To take into account the edges between B and C, Alice has to communicate with Bob. To show a lower bound on the number of symbolic steps, we show next an upper bound on the number of bits of communication between Alice and Bob needed to simulate a symbolic one-step operation on G. With the simulation of the one-step operations, the symbolic algorithm can be used as a communication protocol for distinguishing whether G has diameter 2 or 3 and thus by Lemma 3.3 to solve the Set Disjointness problem. Whenever the symbolic algorithm performs a Pre or Post operation (which are equivalent on undirected graphs) for a set S that contains vertices of B or C, then Alice can simulate this one-step operation by specifying the vertices of B and C that are in S with a bit vector of size $2s$, where Bob answers with a bit vector, again of size $2s$, that indicates all vertices that are adjacent to $(B \cup C) \cap S$. Thus the communication protocol can simulate a symbolic algorithm that performs T one-step operations with at most $4sT$ bits of communication. Hence we have by Theorem 2.1 that $4sT \geq \Omega(k) = \Omega(s^2)$ and thus

---

[1] Usually the diameter is defined over all pairs of vertices, not just the reachable ones, and is therefore ∞ if G is not strongly connected. Our definition is more general since determining whether the graph is strongly connected takes only $O(D)$ symbolic steps and additionally our definition is more natural in the symbolic setting as it provides an upper bound on the number of one-step operations needed until a fixed point is reached, which is an essential primitive in symbolic graph algorithms.

$T \geq \Omega(s) = \Omega(n)$. Together with Lemma 3.3, this proves the following theorem. Note that any $(3/2 - \varepsilon)$-approximation algorithm for the diameter of a graph can distinguish between diameter 2 and 3.

THEOREM 3.6. *Any (probabilistic bounded error or deterministic) symbolic algorithm that computes a $(3/2-\varepsilon)$-approximation of the diameter of an undirected connected graph with n vertices needs $\Omega(n)$ symbolic one-step operations.*

4 Upper Bounds

In this work we present the following upper bounds.

4.1 Upper Bounds for Computing Strongly Connected Components.
We revisit the symbolic algorithm of Gentilini et al. [15] that computes the SCCs and present a refined analysis to show that it only requires $O(\sum_{C \in SCCs(G)}(D_C+1))$ symbolic operations, improving the previously known $O(\min(n, D \cdot |SCCs(G)|))$ bound and matching the lower bound of Theorem 3.2 (details in Section 6.4).

THEOREM 4.1. *The algorithm of Gentilini et al. [15] computes the SCCs of a graph G with $O(|SCCs(G)| + \sum_{C \in SCCs(G)} D_C)$ symbolic operations.*

4.2 Upper Bounds for Liveness, Reachability, and Safety Objectives.
The upper bounds for Reachability, Safety, Liveness, and co-Liveness, are summarized in the following proposition, which is straightforward to obtain as discussed below.

PROPOSITION 4.1. *Let SCC be the number of symbolic steps required to compute the SCCs of the graph and let T be the set of target/safe vertices. Then $Reach(T)$ can be solved with $O(D)$ symbolic operations; $Büchi(T)$ can be solved with $O(\min(SCC + D, |T| \cdot D))$ symbolic operations; $coBüchi(T)$ can be solved with $O(|T| + D)$ symbolic operations; and $Safe(T)$ can be solved with $O(|T|)$ symbolic operations.*

Algorithm for Reachability. Given a target set T, we can easily compute the vertices that can reach T by iteratively applying Pre operations until a fixed-point is reached. By the definition of diameter, this requires only $O(D)$ symbolic operations.

Algorithms for Liveness. A simple algorithm for Liveness first starts an algorithm for computing SCCs and whenever an SCC is reported it tests whether the SCC contains one of the target vertices and if so adds all vertices of the SCC to the winning set. Finally, after all SCCs have been processed, the algorithm computes all vertices that can reach the current winning set and adds them to the winning set. That is, in total the algorithm only needs $O(SCC+D)$ many symbolic operations where SCC is the number of symbolic operations required by the SCC algorithm. An alternative algorithm for Liveness with $O(|T|\cdot D)$ symbolic operations is as follows. For each $v \in T$ check with $O(D)$ symbolic one-step operations whether the vertex can reach itself and if not remove the vertex from T. Then do a standard reachability with the updated set T as target, again with $O(D)$ symbolic one-step operations.

Algorithm for co-Liveness. Given a set T of safe vertices, an algorithm for co-Liveness first restricts the graph to the vertices of T; in the symbolic model this can be done by intersecting the outcome of each Pre and Post operation with T. One then uses an SCC algorithm and whenever a non-trivial SCC is reported, all its vertices are added to the winning set. Finally, after all SCCs have been processed, all vertices that can reach the current winning set in the original graph are added to the winning set. That is, in total the algorithm only needs $O(|T| + D)$ many symbolic operations, where $|T|$ comes from the linear number of symbolic operations required by the SCC algorithm for the modified graph.

Algorithm for Safety. Given a set T of safe vertices, an algorithm for safety first restricts the graph to the vertices of T. One then uses an SCC algorithm and whenever a non-trivial SCCs is reported, all its vertices are added to the winning set. Finally, after all SCCs have been processed, all vertices that can reach, within T, the current winning set are added to the winning set. That is, in total the algorithm only needs $O(|T|)$ symbolic operations as both the SCC algorithm for the modified graph and reachability in the modified graph are in $O(T)$. Also notice that reachability is not bounded by $O(D)$ as restricting the graph to vertices of T can change the diameter.

Notice that none of the above algorithms stores all the SCCs, but processes one SCC at a time. That is, the algorithms themselves only need a constant number of sets plus the sets stored in the algorithm for computing SCCs (which can be done with $O(\log n)$ many sets).

4.3 Upper Bounds for Approximate Diameter.
We present a $(1 + \varepsilon)$-approximation algorithm for the diameter (for any constant $\varepsilon > 0$) that takes $\widetilde{O}(n\sqrt{D})$ symbolic operations (the $\widetilde{O}$-notation hides logarithmic factors).

THEOREM 4.2. *A $(1+\epsilon)$-approximation of the diameter of a directed graph for any constant $\epsilon > 0$ can be obtained with $\widetilde{O}(n\sqrt{D})$ symbolic operations, using $O(1)$ many sets.*

Technical Overview $(1 + \varepsilon)$-*Approximation Algorithm.*
The symbolic algorithm is based on the 3/2-approximation algorithm by Aingworth et al. [1] for explicit graphs, we give a high-level overview of the differences here. An expensive step in the algorithm of [1] is the computation of s-partial BFS trees that contain s vertices closest to a root vertex v and can be determined with $O(s^2)$ explicit operations (this part was later replaced and improved upon by [23, 10]). In the symbolic model computing s-partial BFS trees would be less efficient, however, we can compute *all* vertices with distance at most x from v with only $O(x)$ many symbolic operations. The limitation of the approximation ratio c to 3/2 in the algorithm of [1] comes from having to deal with vertices for which less than s vertices are within distance at most $D/(2c)$. In the symbolic model we do not have to consider this case since with a budget of $O(x)$ operations we can always reach at least x vertices (assuming for now that the graph is strongly connected and $x < D$). Thus the algorithm simplifies to the second part of their algorithm, whose core part is to find a vertex within distance at most x for each vertex of the graph by using a greedy approximation algorithm for dominating set. However, in the symbolic model storing a linear number of sets is too costly, hence we inherently use that we can recompute vertices at distance at most x efficiently when needed. Details are presented in Section 6.5.

5 Discussion

5.1 SCCs and Verification Objectives.
First, our results show that the symbolic SCC algorithm by Gentilini et al. [15] is essentially optimal. That is, we have the three upper bounds of $O(n)$, $O(|SCCs(G) \cdot D|)$ and $O(\sum_{C \in SCCs(G)}(D_C + 1))$ and matching lower bounds of $\Omega(n)$ (Theorem 3.1), $\Omega(|SCCs(G)| \cdot D)$ (Theorem 3.1), and $\Omega(|SCC| + \sum_{C \in SCCs(G)} D_C)$ (Theorem 3.2).

Table 3: Results

symbolic operations for SCC computation		
$\Theta(n)$	$\Theta(|SCCs| \cdot D)$	$\Theta(\sum_{C \in SCCs}(D_C + 1))$

Our results for the different kinds of verification objectives are summarized in Table 4. We have an interesting separation between the reachability objective and the other objectives in terms of the diameter D. While reachability can be solved with $O(D)$ symbolic operations, all the other objectives require $\Omega(n)$ symbolic one-step operation on graphs of constant diameter.

When considering the diameters D_C of the SCCs we get another separation. There we have that Liveness and

Reachability can be solved with $O(\sum_{C \in SCCs(G)}(D_C + 1))$ many symbolic operations, while Safety and co-Liveness requires $\Omega(n)$ symbolic one-step operations on strongly connected graphs with constant diameter. This reflects the fact that in the standard algorithm for Safety and co-Liveness the SCC computation is performed on a modified graph.

5.2 Approximate Diameter.
For explicitly represented graphs a 3/2-approximation of the diameter can be computed in $\widetilde{O}(m\sqrt{n})$ time [23, 10], while under the strong exponential time hypothesis no $O(n^{2-o(1)})$ time algorithm exists to distinguish graphs of diameter 2 and 3 (i.e., no $(3/2 - \varepsilon)$-approximation can be obtained) [23]. The fastest exact algorithms take $\widetilde{O}(mn)$ time. While for explicitly represented graphs small, constant diameters are a hard case, the current results suggest that in the symbolic model the diameter of graphs with constant diameter can be determined more efficiently than for large diameters, as both the upper bound for exact and approximate computation of the diameter depend on the diameter of the graph and are linear when the diameter is constant. While the threshold of an approximation ratio of 3/2 appears in our (linear) lower bound, the current symbolic upper bounds do not show this behavior. Several interesting open questions remain: Is there a $o(n)$ c-approximation algorithm when $c \in [3/2, 2)$? Is there a linear $(1+\varepsilon)$-approximation algorithm for graphs with super-constant diameter? Or are there better lower bounds?

6 Detailed Proofs

6.1 Proofs of Section 3.1.

Proof. [Proof of Lemma 3.1.] We have to show that $f(x, y) = 1$ iff the graph constructed in Reduction 3.1 has exactly ℓ SCCs.

First notice that there are no edges from a set V_j to a set V_i if $i < j$ and thus there are at least ℓ SCCs, independently of the actual values of x and y.
$\Rightarrow$: If $f(x, y) = 1$ then all possible edges exists and it is easy to verify that the SCCs of the graphs are exactly the sets V_i for $0 \le i < \ell$.
$\Leftarrow$: If $f(x, y) = 0$ then there are $0 \le j \le \bar{k}$, $0 \le i < \ell$ such that there is no edge from $v_{i,j+1}$ to $v_{i,j}$. Now the set V_i splits up in at least two SCCs and thus there are at least $\ell + 1$ SCCs. $\qquad\square$

Proof. [Proof of Theorem 3.2.] We have to show that any (probabilistic bounded error or deterministic) symbolic algorithm that computes the SCCs needs $\Omega(|SCCs(G)| + \sum_{C \in SCCs(G)} D_C)$ symbolic one-step operations.

First consider Reduction 3.1 and notice that for the

Table 4: Results

Objective	number of symbolic operations in terms of		
	n	D & $\|T\|$	D_C
Reach(T)	$\Theta(n)$	$\Theta(D)$	$\Theta(D)$
Safe(T)	$\Theta(n)$	$\Theta(\|T\|)$	$\omega(\sum_{C \in SCCs(G)}(D_C + 1))$
Büchi(T)	$\Theta(n)$	$O(\|T\| \cdot D) \: / \: \Omega(\|T\| + D)$	$\Theta(\sum_{C \in SCCs(G)}(D_C + 1))$
coBüchi(T)	$\Theta(n)$	$\Theta(\|T\| + D)$	$\omega(\sum_{C \in SCCs(G)}(D_C + 1))$

constructed graph $\sum_{C \in SCCs(G)} D_C \in \Theta(n)$. Then by Theorem 3.1 we already have a $\Omega(\sum_{C \in SCCs(G)}(D_C))$ bound. Now consider Reduction 3.2 and notice that the constructed graph has n SCCS iff x and y are disjoint. Now we can use the same argument as in the proof of Theorem 3.1 that each symbolic one-step operations just needs constant communication. The instances where x and y are disjoint have n SCCs and $\sum_{C \in SCCs(G)} D_C = 0$. Hence an algorithm with $o(|SCCs(G)|)$ symbolic one-step operations would imply a communication protocol with $o(k)$ communication, a contradiction to Theorem 2.1. By combining the two lower bounds we get the desired $\Omega(|SCCs(G)| + \sum_{C \in SCCs(G)} D_C)$ bound. □

6.2 Proofs of Section 3.2.

Proof. [Proof of Theorem 3.3.] We have to show that any (probabilistic bounded error or deterministic) symbolic algorithm that solves Reachability in graphs with diameter D requires $\Omega(D)$ symbolic one-step operations.

Consider the graph from Reduction 3.1 with parameter $\ell = 1$. We have that v_0 is reachable from all vertices iff the graph is strongly connected. From the proof of Theorem 3.1 we have that testing whether the graph is strongly connected requires $\Omega(k)$ symbolic one-step operations. Now notice that (a) the graph is strongly connected iff v_k can reach v_0 and that (b) if the graph is strongly connected then $D = k$. □

Proof. [Proof of Theorem 3.4.] For (1) & (3) consider the graph constructed in Reduction 3.3 and the target set $T = \{v_0, v_1, \ldots, v_{k-1}\}$. We have a valid reduction from the Set Disjointness problem as the vertex v_0 is winning for Büchi(T) iff there is a loop for one of the vertices in T iff $S_x \cap S_y \neq \emptyset$. By the same argument as in the proof of Theorem 3.1 we have that each symbolic one-step operations just needs constant communication. Hence an algorithm with $o(n)$, $o(|T|)$ or $o(|SCCs(G)|)$ symbolic one-step operations would imply a communication protocol with $o(k)$ communication, a contradiction.

For (2) consider the graph constructed in Reduction 3.1 with $\ell = 1$ and the target set $T = \{v_0\}$. It is easy to verify that the vertex v_k is winning if it can reach T. Thus the $\Omega(D)$ lower bound for reachability also applies here. Notice that this also gives a $\Omega(\sum_{C \in SCCs(G)} D_C)$ lower bound.

The $\Omega(|T| + D)$ lower bound in (4) is a direct consequence of the $\Omega(D)$ lower bound for instances with constant size target sets T, and the $\Omega(|T|)$ lower bound for instances with constant diameter D.

Finally, (5) is by the $\Omega(|SCCs(G)|)$ bound from Reduction 3.3 and the $\Omega(\sum_{C \in SCCs(G)} D_C)$ bound by Reduction 3.1 with $\ell = 1$. □

Proof. [Proof of Theorem 3.5.] 1) & 2) Consider the graph constructed in Reduction 3.3 and the set of safe vertices $T = \{v_0, v_1, \ldots, v_{k-1}\}$. We have a valid reduction from the Set Disjointness problem as the vertex v_0 is winning for Safe(T) iff there is a loop for one of the vertices in T iff $S_x \cap S_y \neq \emptyset$. By the same argument as in the proof of Theorem 3.1 we have that each symbolic one-step operations just needs constant communication. Thus an algorithm with $o(n)$ or $o(|T|)$ symbolic one-step operations would imply a communication protocol with $o(k)$ communication, a contradiction.

3) Consider the graph constructed in Reduction 3.3 but replace the edge (v_k, v_k) by the edge (v_k, v_0). When considering the set of safe vertices $T = \{v_0, v_1, \ldots, v_{k-1}\}$ the same arguments as above apply and thus we get a $\Omega(n)$ lower bound. However, the graph is strongly connected and has diameter 2 and thus $|SCCs(G)| + \sum_{C \in SCCs(G)} D_C = O(1)$. □

Proof. [Proof of Proposition 3.1.] Consider the graph constructed in Reduction 3.1 with $\ell = 1$, add an additional loop edge (v_0, v_0), and consider the set $T = \{v_0\}$ of safe vertices, i.e., v_0 is the only safe vertex. It is easy to verify that the vertex v_k is winning in coBüchi(T) if it can reach T. Thus the $\Omega(D)$ lower bound for reachability also applies here. □

6.3 Proofs of Section 3.3.

Proof. [Proof of Lemma 3.3.] Let $G = (V, E)$ be the

graph given by Reduction 3.4 and let D denote its diameter. We have to show that if $S_x \cap S_y = \emptyset$, then $D = 2$, otherwise $D = 3$.

First note that through the edges adjacent to the auxiliary vertices the diameter of G is at most 3. Furthermore we have for all $0 \leq i, j < s$ that $d(a_i, b_j) \leq 2$, $d(u, c_j) = 2$, and $d(t, c_j) = 1$, for all $i \neq j$ additionally $d(a_i, a_j) = d(b_i, b_j) = d(c_i, c_j) = 2$, and for all $v \in A \cup B$ it holds that $d(u, v) = 1$ and $d(t, v) = 2$. Thus whether D is 2 or 3 depends only on the maximum over all $0 \leq i, j < s$ of $d(a_i, c_j)$ and $d(b_i, c_j)$.

If $S_x \cap S_y = \emptyset$, then for each pair of indices $0 \leq i, j < s$ at least one of the edges (a_i, c_j) and (b_i, c_j) exists. Since $(a_i, b_i) \in E$, we have for all $0 \leq j < s$ and all $v \in A \cup B$ that $d(v, c_j) \leq 2$ and hence $D = 2$.

If $S_x \cap S_y \neq \emptyset$, let $\ell \in S_x \cap S_y$ and let $0 \leq i, j < s$ be such that $\ell = i \cdot s + j$. Then neither the edge (a_i, c_j) nor the edge (b_i, c_j) exists. Thus the vertex a_i, and analogously the vertex b_i, has edges to the following vertices only: the auxiliary vertex u, the vertex b_i, and vertices $c_{j'} \in C$ with $j' \neq j$. The vertex c_j has edges only to the auxiliary vertex t and to vertices $a_{i'} \in A$ and $b_{i'} \in B$ with $i' \neq i$. Hence none of the vertices adjacent to a_i, and respectively for b_i, is adjacent to c_j and thus we have $D = d(a_i, c_j) = d(b_i, c_j) = 3$. $\qquad\square$

6.4 An Improved Upper Bound for Strongly Connected Components.

Gentilini et al. [15] provide a symbolic algorithm for computing the strongly connected components (SCCs) and show a bound of $O(\min(n, D \cdot |SCCs(G)|))$ on the number of its symbolic operations for a directed graph with n vertices, diameter D, and $|SCCs(G)|$ many SCCs. Let D_C be the diameter of an SCC C. We give a tighter analysis of the algorithm of [15] that shows an upper bound of $O(\sum_{C \in SCCs(G)} (D_C + 1))$ symbolic operations that matches our lower bound (Theorem 3.2). We have both $\sum_{C \in SCCs(G)} (D_C + 1) \leq (D + 1) \cdot |SCCs(G)|$ and $\sum_{C \in SCCs(G)} (D_C + 1) \leq n + |SCCs(G)| \leq 2n$ and thus our upper bound is always at most the previous one. We additionally observe that the algorithm can be implemented with $O(\log n)$ many sets (when the SCCs are output immediately and not stored). We first explain the intuition behind the algorithm of [15] and then present the improved analysis of its number of symbolic steps.

Symbolic Breadth-First Search. While explicit algorithms for SCCs are based on depth-first search (DFS), DFS is impractical in the symbolic model. However, breadth-first search (BFS) from a set $U \subseteq V$ can be performed efficiently symbolically, namely proportional to its depth, as defined below.

DEFINITION 6.1. (SYMBOLIC BFS) *A forward search*

from a set of vertices $U = U_0$ is given by a sequence of Post operations such that $U_i = U_{i-1} \cup \mathsf{Post}(U_{i-1})$ for $i > 0$ until we have $U_i = U_{i-1}$. We call $U_i \setminus U_{i-1}$ the i-th level *of the forward search and the index of the last non-empty level the* depth $\overrightarrow{B}(U)$ *of the forward search. Let $FW(U) = U_{\overrightarrow{B}(U)}$ be the* forward set*, which is equal to the vertices reachable from U. Analogously we define the* backward search *of depth $\overleftarrow{B}(U)$ and the* backward set $BW(U)$ *for Pre operations. We denote a singleton set $U = \{u\}$ by u.*

There is a simple algorithm for computing the SCCs symbolically with BFS that takes $O(D \cdot |SCCs(G)|)$ many symbolic steps: Start with an arbitrary vertex v. Compute the SCC containing v by taking the intersection of $FW(v)$ and $BW(v)$, remove the obtained SCC from the graph, and repeat.

Skeleton-based Ordering. The importance of DFS for SCCs lies in the *order* in which the SCCs are computed. Starting from a vertex v that lies in an SCC without outgoing edges (i.e. a sink in the DAG of SCCs of the graph), the forward search does not leave the SCC and for computing SCCs the backward search can be restricted to the vertices of $FW(v)$, i.e., the SCC of v can be determined with proportional to the diameter of the SCC many symbolic steps. The DFS-based SCC algorithm of [25] finds such an SCC first. The algorithm of [15] is based on an ordering obtained via BFS that achieves a DFS-like ordering suitable for computing SCCs symbolically. Our tighter analysis essentially shows that their approach achieves the best ordering we can hope for. The ordering is given by so-called *skeletons*.

DEFINITION 6.2. *A pair (S, v) with $v \in V$ and $\mathsf{S} \subseteq V$ is a* skeleton *of $FW(u)$ for $u \in V$ if v has maximum distance from u and the vertices of S form a shortest path from u to v.*

Let $SCC(v)$ denote the SCC containing $v \in V$. The SCCs of the vertices of S will be computed in the following order: First $SCC(u)$ is computed by performing a backward search from u within $FW(u)$. The remaining SCCs of of the vertices of S are then computed in the reverse order of the path induced by S, starting with $SCC(v)$. We now describe the overall algorithm, where in addition to $SCC(\mathsf{S}, v)$ the SCCs of $V \setminus FW(u)$ are computed (potentially using a different, previously computed skeleton).

The Algorithm. The pseudo-code of the algorithm is given in Algorithm SCC-Find, the pseudo-code for the sub-procedure for computing a forward set including a skeleton in Algorithm Skel-Forward. Processing a graph G, the algorithm proceeds as follows: it starts

Algorithm SCC-Find: Symbolic SCC Algorithm

Input : *Graph* $G = (V, E)$, *Skeleton* $(\mathcal{S}, v)$

1 **if** $V = \emptyset$ **then**
2   return $\emptyset$;
3 **if** $\mathcal{S} = \emptyset$ **then**
4     $v \leftarrow Pick(V)$;   // If there is no skeleton, pick arbitrary vertex
5 $(FW, \mathcal{S}', v') \leftarrow$ Skel_Forward(V, E, v) ; // Forward search incl. skeleton

    /* Compute the SCC containing v   */
6 $SCC \leftarrow \{v\}$;
7 **while** $(\mathsf{Pre}(SCC) \cap FW) \setminus SCC \neq \emptyset$ **do**
8     $SCC \leftarrow SCC \cup (\mathsf{Pre}(SCC) \cap FW)$
9 output SCC as an SCC;

    /* Recursive calls   */
10 SCC-Find$(G_{V \setminus FW}, (\mathcal{S} \setminus SCC, (pre(SCC \cap \mathcal{S}) \setminus SCC) \cap \mathcal{S}))$;
11 SCC-Find$(G_{FW \setminus SCC}, (\mathcal{S}' \setminus SCC, v'))$;
12 **return** SCCs;

Algorithm Skel-Forward: Skeleton Forward Search Algorithm

Input : *Graph* $G = (V, E)$, *Node* v
Output: FW Set of vertices reachable from v;
         $(\mathcal{S}', v')$ Skeleton for FW

1 $FW \leftarrow \emptyset$; $i \leftarrow 0$; LEVEL$[0] \leftarrow \{v\}$;

    /* Forward Search   */
2 **while** LEVEL$[i] \neq \emptyset$ **do**
3     $FW \leftarrow FW \cup$ LEVEL$[i]$;
4     $i \leftarrow i + 1$;
5   LEVEL$[i] \leftarrow \mathsf{Post}($LEVEL$[i-1]) \setminus FW$;

    /* Compute Skeleton   */
6 $i \leftarrow i - 1$;
7 $v' \leftarrow Pick($LEVEL$[i])$;
8 $\mathcal{S}' \leftarrow \{v'\}$;
9 **while** $i \geq 1$ **do**
10     $i \leftarrow i - 1$;
11     $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{Pick(\mathsf{Pre}(\mathcal{S}') \cap$ LEVEL$[i])\}$;
12 **return** $(FW, \mathcal{S}', v')$;

---

from some vertex v and computes the set of reachable vertices, i.e, the forward set $FW(v)$, including a skeleton that includes exactly one vertex of each level of the forward search and forms a shortest path in G. It then starts a backward search starting from v in the subgraph induced by $FW(v)$. Clearly the SCC of v is given by the vertices that are reached by the backward search. The algorithm returns this SCC as an SCC of G and recurses on (a) the subgraph $G_{V \setminus FW(v)}$ induced by the vertices of $V \setminus FW(v)$ and (b) the subgraph $G_{FW(v) \setminus SCC(v)}$ induced by the vertices of $FW(v) \setminus SCC(v)$. For the recursion on (a) we update a potentially already existing skeleton by removing all vertices that are in the current SCC (initially we have an empty skeleton) while for the recursion on (b) we use the skeleton computed by the forward search (but also remove vertices of the current SCC). The skeleton is then used to select the starting vertex in the consecutive steps of the algorithm: when the algorithm is called with skeleton (S, v), then the forward search is started from v; when the skeleton was computed in a forward search from a vertex u, then this corresponds to the vertex of the skeleton that is furthest away from u and contained in this recursive call.

A Refined Analysis. The correctness of the algorithm is by [15]. Notice that the algorithm would be correct even without the usage of skeletons but the skeletons are necessary to make it efficient, i.e., to avoid unnecessarily

long forward searches. We show the following theorem.

THEOREM 6.1. (RESTATEMENT OF THEOREM 4.1)
With input $(G, \emptyset)$ *Algorithm SCC-Find computes the SCCs of* G *and requires* $O(\sum_{C \in SCCs(G)}(D_C + 1))$ *symbolic operations.*

The analysis of [15] of the number of symbolic steps of Algorithm SCC-Find uses that (1) each vertex is added to at most two skeletons, (2) the steps of the forward searches can be charged to the vertices in the skeletons, and (3) backward searches are only performed to immediately identify an SCC and thus can be charged to the vertices of the SCC; hence both the steps of the forward and of the backward searches can be bounded with $O(n)$. For the backward searches it can easily be seen that the number of Pre operations to identify the SCC C is also bounded by $D_C + 1$. For the forward searches we show that each part of the skeleton (that in turn is charged for the forward search) can be charged to $D_C + 1$ for some SCC C; this in particular exploits that skeletons are shortest paths (in the graph in which they are computed).

LEMMA 6.1. ([15]) *For each recursive call of SCC-Find with input* $G, (\mathcal{S}, v)$ *we have that* $\mathcal{S}$ *is a set of vertices that induces a shortest path in the graph* G *and* v *is the last vertex of this path.*

We first recall the result from [15] that shows that the number of symbolic operations in an execution of

Skel-Forward is proportional to the size of the computed skeleton.

LEMMA 6.2. ([15]) *Skel-Forward only requires $O(\mathcal{S}')$ symbolic operations, i.e., is linear in the output, and can be implemented using only constantly many sets.*

Proof. For each level of the forward search we need one Post operation in the first while loop and one Pre operation in the second while loop, and the number of set operations is in the order of one-step operations. As for each level we add one vertex to $\mathcal{S}'$, the result follows. Moreover, there is no need to explicitly store all the levels as they can be easily recomputed from the next level when needed, increasing the number of symbolic operations only by a constant factor. □

REMARK 6.1. *Given Lemma 6.2 we can implement SCC-Find using $O(\log n)$ many sets at a time by recursing on the smaller of the two sub-graphs $G_{V\setminus FW}$ and $G_{FW\setminus SCC}$ first.*

We split the cost of $O(\mathcal{S}')$ symbolic operations for Skel-Forward into two parts: the part $\mathcal{S}'\cap SCC(v)$ where $SCC(v)$ is the SCC identified in this level of recursion and the part $\mathcal{S}'\setminus SCC(v)$ that is passed to one of the recursive calls. The following lemma shows that the first part and the subsequent backward search can be charged to $D_{SCC(v)}+1$, using that $\mathcal{S}'$ is a shortest path in $FW(v)$.

LEMMA 6.3. *Without accounting for the recursive calls, each call of SCC-Find takes $O(D_{SCC(v)}+1+|\mathcal{S}'\setminus SCC(v)|)$ symbolic operations, where $\mathcal{S}'$ is the new skeleton computed by Skel-Forward.*

Proof. By Lemma 6.2, the call to Skel-Forward takes $O(|\mathcal{S}'|)$ symbolic operations. In the i-th iteration of the while loop (Line 7) we add those vertices of $SCC(v)$ that can reach v in i steps. That is, the loop terminates after $D_{SCC(v)}$ iterations and thus only requires $O(D_{SCC(v)}+1)$ many symbolic operations. All the other steps just need a constant number of symbolic operations. That is, we have an upper bound of $O(D_{SCC(v)}+1+|\mathcal{S}'|)$. Now as $\mathcal{S}'$ induces a path starting at v in $FW(v)$, we have that whenever a vertex $u\neq v$ of $\mathcal{S}'$ can reach v, then also all vertices on the path from v to u can reach v and are therefore in the same SCC as v. Since the path is a shortest path, also every sub-path is a shortest path and thus we have that $|\mathcal{S}'\cap SCC(v)|\leq D_{SCC(v)}+1$, i.e., $|\mathcal{S}'|\in O(D_{SCC(v)}+1+|\mathcal{S}'\setminus SCC(v)|)$. Hence we obtain the desired bound of $O(D_{SCC(v)}+1+|\mathcal{S}'\setminus SCC(v)|)$ for the number of symbolic operations. □

Note that at each level of recursion we only charge the diameter of the SCC that is output and the vertices of the newly computed skeleton. Thus we do not charge vertices of a skeleton again until they are contained in an SCC that is identified. The following lemma shows that in this case we can charge the symbolic steps that were charged to the vertices of the skeleton to D_C+1, where C is the SCC the part of the skeleton belongs to. Notice that $\mathcal{S}\setminus SCC(v)$ is the skeleton for the first recursive call and $\mathcal{S}'\setminus SCC(v)$ is the skeleton for the second recursive call, i.e., all vertices of a skeleton are finally assigned to an SCC. That is, we can bound the total number of symbolic steps by $O(\sum_{C\in SCCs(G)}(D_C+1))$.

LEMMA 6.4. *Whenever SCC-Find is called for a graph H and a skeleton $(\mathcal{S},v)$, then $|\mathcal{S}\cap SCC(v)|\leq D_{SCC(v)}+1$.*

Proof. By Lemma 6.1 the set $\mathcal{S}$ induces a shortest path in H that ends at v. Thus if v can reach a vertex $u\neq v$ of $\mathcal{S}$, then it can also reach all vertices of $\mathcal{S}$ that are on the path from u to v and all vertices on this sub-path are in the same SCC as v. Furthermore, the sub-path is a shortest path as well and thus the vertices $|\mathcal{S}\cap SCC(v)|$ form a shortest path in $SCC(v)$ and hence the diameter $D_{SCC(v)}$ of $SCC(v)$ is at least $|\mathcal{S}\cap SCC(v)|-1$. □

6.5 $(1+\varepsilon)$-Approximation of Diameter with $\widetilde{O}(n\sqrt{D})$ Symbolic Operations. Given a vertex $u\in V$, let $\overrightarrow{N}_x(u)$ denote the vertices with distance at most x *from* u and let $\overleftarrow{N}_y(u)$ be the set of vertices with distance at most y *to* u. We have that $\{u\}\cup\mathsf{Post}(\{u\})=\overrightarrow{N}_1(u)$ and $\overrightarrow{N}_x(u)\cup\mathsf{Post}(\overrightarrow{N}_x(u))=\overrightarrow{N}_{x+1}(u)$. The maximum distance from the vertex u to any other vertex $v\in V$ is given by the smallest x for which $\overrightarrow{N}_x(u)\cup\mathsf{Post}(\overrightarrow{N}_x(u))=\overrightarrow{N}_x(u)$. Note that x is at most D and that computing x in this way corresponds to performing a breadth-first-search (BFS) from x on explicitly represented graphs; thus following [1], we denote the smallest x for which $\overrightarrow{N}_x(u)\cup\mathsf{Post}(\overrightarrow{N}_x(u))=\overrightarrow{N}_x(u)$ with $\overrightarrow{B}(u)$ and the smallest y for which $\overleftarrow{N}_y(u)\cup\mathsf{Pre}(\overleftarrow{N}_y(u))=\overleftarrow{N}_y(u)$ with $\overleftarrow{B}(u)$. The set of vertices reachable from $u\in V$ is given by $\overrightarrow{N}_{\overrightarrow{B}(u)}(u)$.

The Basic Exact Algorithm. The maximum $d(u,v)$ over all pairs $u,v\in V$ for which u can reach v can be computed by taking the maximum of $\overrightarrow{B}(u)$ over all $u\in V$. Computing $\overrightarrow{B}(u)$ for all $u\in V$ takes $O(n\cdot D)$ many Post operations. To obtain only the value of D, only a constant number of sets have to be stored. Note that this basic algorithm only uses a linear number of symbolic operations for graphs with constant diameter. See Section 3.3 for a matching lower bound for this case.

A Simple 2-Approximation Algorithm. If the graph G is strongly connected, then a 2-approximation of D is

given by $(\overrightarrow{B}(u) + \overleftarrow{B}(u))/2$ for any vertex $u \in V$. This follows from the triangle inequality and takes $O(D)$ many symbolic steps to compute.

Result. We present an algorithm that computes an estimate $\widetilde{D}$ of the diameter D of the input graph G such that $\widetilde{D} \in [D - x, D]$ for a parameter $x \leq \sqrt{D}$ and takes $O(n \cdot D/x \log n)$ symbolic steps and uses a constant number of sets. For $x = \sqrt{D}$ this implies a bound of $O(n\sqrt{D} \log n)$ on the number of symbolic steps and an approximation guarantee that is better than a $(1 + \varepsilon)$-approximation for any constant $\varepsilon > 0$ (here we assume $\sqrt{D} \geq (1 + \varepsilon)/\varepsilon$; otherwise D is constant anyway and thus the exact algorithm only takes $O(n)$ symbolic steps). To pick the parameter x correctly, one can use the 2-approximation algorithm if the graph is strongly connected or use doubling search at the cost of an additional factor of $\log n$.

Searching from Neighborhood. Let a and b be two vertices with maximum distance in G, i.e., $d(a, b) = D$. We start with the simple observation that it is sufficient to determine the depth of a BFS from a vertex with distance at most x from a to obtain an estimate that is at most x smaller than D.

OBSERVATION 6.1. (SEE ALSO [1]) *Let $a, b \in V$ be such that $d(a, b) = D$. Then $\overrightarrow{B}(v) \geq D - x$ for $v \in \overrightarrow{N}_x(a)$ and $\overleftarrow{B}(u) \geq D - y$ for $u \in \overleftarrow{N}_y(b)$.*

Thus to obtain an estimate for the diameter, it is certainly sufficient to find a vertex u in $\overrightarrow{N}_x(v)$ for *every* vertex $v \in V$ and compute $\overrightarrow{B}(u)$ for all these vertices. If the graph is not strongly connected, it can happen that some vertices v can not reach x vertices and hence $\overrightarrow{N}_x(v)$ might contain less than x vertices. In this case we know that $\overrightarrow{B}(v) < D$. Thus it also suffices to find a vertex u in $\overrightarrow{N}_x(v)$ for every vertex $v \in V$ for which $|\overrightarrow{N}_x(v)| \geq x$; we denote this set of vertices with V_x.

COROLLARY 6.1. *Let S be a set of vertices such that $S \cap \overrightarrow{N}_x(v) \neq \emptyset$ for all $v \in V_x$ for $x < D$. Let $\widetilde{D} = \max_{u \in S} \overrightarrow{B}(u)$. Then $\widetilde{D} \in [D - x, D]$. Given S, computing $\widetilde{D}$ takes $O(|S| \cdot D)$ symbolic operations and storing $O(1)$ many sets.*

Dominate each Neighborhood. An *out-dominating set* for a set of vertices $A \subseteq V$ contains for each vertex of A either the vertex or one of its successors. Finding a set $S \subseteq V$ such that $S \cap \overrightarrow{N}_x(v) \neq \emptyset$ for all $v \in V_x$ is equivalent to find an out-dominating set for all vertices with degree at least x in the following graph: Let $\hat{G}$ be the graph obtained from G by adding an edge from v to each vertex of $\overrightarrow{N}_x(v) \setminus \{v\}$ for all $v \in V$. In $\hat{G}$

every vertex of V_x has out-degree at least x. Thus an out-dominating set for V_x in $\hat{G}$ contains a vertex of $\overrightarrow{N}_x(v)$ for all $v \in V_x$, i.e., for all vertices $v \in V$ with $|\overrightarrow{N}_x(v)| \geq x$. We adopt the classical greedy algorithm for dominating set to compute an out-dominating set in $\hat{G}$ with the following guarantees. We prove Lemma 6.5 in the following subsection.

LEMMA 6.5. *An out-dominating set S for the vertices of V_x in $\hat{G}$ with $|S| \in O(n/x \cdot \log n)$ can be found with $O(n \cdot x \cdot \log n)$ symbolic operations on G, storing $O(1)$ many sets.*

Overall Algorithm. Hence our algorithm is as follows. First we find a set S of size $O(n/x \cdot \log n)$ that contains a vertex of $\overrightarrow{N}_x(v)$ for every $v \in V_x$ in $O(n \cdot x \cdot \log n)$ symbolic steps (Lemma 6.5). Then we compute $\overrightarrow{B}(u)$ for all $u \in S$ with $O(n \cdot D/x \cdot \log n)$ many symbolic steps and return the maximum value of $\overrightarrow{B}(u)$ that was found (Corollary 6.1). Together with the observations at the beginning of this section we obtain the following theorem. The $\widetilde{O}$-notation hides the logarithmic factors.

THEOREM 6.2. (RESTATEMENT OF THEOREM 4.2) *A $(1 + \epsilon)$-approximation of the diameter of a directed graph for any constant $\epsilon > 0$ can be obtained with $\widetilde{O}(n\sqrt{D})$ symbolic operations, using $O(1)$ sets.*

6.5.1 Proof of Lemma 6.5. A *fractional* out-dominating set of a set $A \subseteq V$ is a function that assigns a weight $w_v \in [0, 1]$ to each $v \in V$ such that for every $v \in A$ the sum of the weights over v and its successors is at least one. The size of a fractional out-dominating set is the sum of all weights w_v. For Lemma 6.5 we want to obtain an out-dominating set of V_x. The vertices of V_x have out-degree at least x in $\hat{G}$. Thus a fractional out-dominating set of V_x in $\hat{G}$ is obtained by assigning each vertex a weight of $1/x$. The size of this fractional out-dominating set is $O(n/x)$. We show a greedy algorithm that finds an out-dominating set of V_x in $\hat{G}$ of size within a logarithmic factor of the optimal fractional out-dominating set, i.e., of size $O(n/x \cdot \log n)$. The greedy algorithm is given in Algorithm Dominating Set and is a modification of the greedy algorithm by [17, 20, 11] using an idea from [2]. We first describe the algorithm and show that it takes $O(n \cdot x \cdot \log n)$ symbolic steps to output an out-dominating set of V_x and then prove that the size of the obtained out-dominating set for V_x is within $O(\log n)$ of the optimal fractional solution.

Algorithm Dominating Set takes the graph $G = (V, E)$ and the parameter x as input. Constructing the graph $\hat{G}$, i.e., storing the sets $\overrightarrow{N}_x(v)$ for all $v \in V$, is too costly. Note that there is a one-to-one correspondence

Algorithm Dominating Set: Algorithm for Out-Dominating Set in $\hat{G}$

Input : Graph $G = (V, E)$, parameter x
Output: Set $S \subseteq V$ that contains a vertex of $\overrightarrow{N}_x(v)$ for all v with $|\overrightarrow{N}_x(v)| \geq x$

1 $S \leftarrow \emptyset$;   /* dominating set */
2 $C \leftarrow \emptyset$;   /* covered vertices */
3 $j \leftarrow \lfloor \log_2 n \rfloor$;   /* size threshold */
4 **for** $v \in V$ **do**   /* don't have to cover vertices that reach $< x$ vertices */
5   **if** $|\overrightarrow{N}_x(v)| < x$ **then**
6        $C \leftarrow C \cup \{v\}$;

7 **while** $j \geq 0$ **do**
8   **for** $v \in V \setminus S$ **do**
9   **if** $|\overleftarrow{N}_x(v) \setminus C| \geq 2^j$ **then**
10           $S \leftarrow S \cup \{v\}$;
11           $C \leftarrow C \cup \overleftarrow{N}_x(v)$;
12     $j \leftarrow j - 1$;
13 **return** S;

---

between the edges of $\hat{G}$ and the paths of length $\leq x$ in G and that the union of a vertex v with its successors in $\hat{G}$ is given by $\overrightarrow{N}_x(v)$ and the union of a vertex v with its predecessors in $\hat{G}$ is given by $\overleftarrow{N}_x(v)$. We recompute the sets $\overleftarrow{N}_x(v)$ in $O(x)$ symbolic steps per set when needed by the algorithm.

Observe that the set S is an out-dominating set of V_x in $\hat{G}$ if and only if $\cup_{v \in S} \overleftarrow{N}_x(v) \supseteq V_x$. We say that the vertices of $\cup_{v \in S} \overleftarrow{N}_x(v)$ are covered by the set S. In the algorithm the set S denotes the set of vertices added to the out-dominating set so far and the set C denotes the vertices that are covered by S; we additionally add the vertices of $V \setminus V_x$ to C, as they do not have to be covered (lines 4–6).

The main part of the algorithm consists of a while-loop with $O(\log n)$ many iterations. The variable j is initialized with $\lfloor \log_2 n \rfloor$ and is decreased after each iteration of the while-loop; in the last iteration of the while-loop we have $j = 0$. In each iteration every vertex that is not yet in S is considered one after the other. For each vertex $v \in V \setminus S$ the set $\overleftarrow{N}_x(v)$ is computed and the vertex is added to S if the set $\overleftarrow{N}_x(v)$ contains more than 2^j vertices that are not yet covered. When a vertex v is added to S, the vertices of $\overleftarrow{N}_x(v)$ are marked as covered by adding them to C. In the last iteration we have $2^j = 1$ and thus all vertices of V_x that were not covered yet are added to S. Hence the returned set S

is an out-dominating set of V_x. In each iteration of the while-loop $O(n \cdot x)$ symbolic operations are used, thus the algorithm takes $O(n \cdot x \log n)$ symbolic steps in total, storing $O(1)$ many sets at a time.

It remains to show that the size of S is within a factor of $O(\log n)$ of the size of an optimal fractional out-dominating set S^* of V_x in $\hat{G}$ with weights $w_v \in [0, 1]$ for all $v \in V$. The outline of the proof is as follows: For each vertex v that the greedy algorithm adds to S, we charge a total cost ≥ 1 to the weights w_u and show that the total cost charged to w_u for each $u \in V$ is at most $2 \cdot H_{\Delta_u} \cdot w_u$, where $H_i \in O(\log i)$ is the i-th harmonic number and Δ_u is the in-degree of u in $\hat{G}$. This implies that the number of vertices in S is bounded by the sum of $2 \cdot w_u \cdot H_n$ over all $u \in V$, which proves the claim.

We charge the weights when adding v to S in Algorithm Dominating Set as follows: For each vertex of $\overleftarrow{N}_x(v) \setminus C$ (i.e. the newly covered vertices) we consider all vertices u that contribute to the cover of this vertex and charge them $w_u / |\overleftarrow{N}_x(v) \setminus C|$. Note that all vertices of $V \setminus V_x$ are contained in C and thus $\overleftarrow{N}_x(v) \setminus C \subseteq V_x$, hence each vertex of $\overleftarrow{N}_x(v) \setminus C$ is covered by the optimal fractional out-dominating set with a weight of at least one. Thus this charges at least $1 / |\overleftarrow{N}_x(v) \setminus C|$ per vertex of $\overleftarrow{N}_x(v) \setminus C$ and hence at least one per vertex added to S. We have that for a fractional out-dominating set the set $\overleftarrow{N}_x(u)$ is the set of vertices to whose covering the weight w_u contributes. Thus the charge for vertex u when adding v is given by

$$\frac{\left| \left(\overleftarrow{N}_x(v) \setminus C \right) \cap \overleftarrow{N}_x(u) \right|}{\left| \overleftarrow{N}_x(v) \setminus C \right|} \cdot w_u.$$

We finally show that each vertex v is charged at most $w_v \cdot H_{n+1}$. Let j' be the value of j when v is added to S, i.e., $|\overleftarrow{N}_x(v) \setminus C| \geq 2^{j'}$. Note that if a vertex u is charged a non-zero amount, then it is not contained in C and therefore not in S. Hence we have that u was not added to S in the previous iteration of the while-loop and thus by the greedy condition $|\overleftarrow{N}_x(u) \setminus C| \leq 2^{j'+1}$. Hence whenever u is charged for a vertex v, we have

$$\left| \overleftarrow{N}_x(v) \setminus C \right| \geq \frac{1}{2} \left| \overleftarrow{N}_x(u) \setminus C \right|,$$

and thus

$$\frac{\left| \left(\overleftarrow{N}_x(v) \setminus C \right) \cap \overleftarrow{N}_x(u) \right|}{\left| \overleftarrow{N}_x(v) \setminus C \right|} \cdot w_u$$

$$\leq \frac{2 \cdot \left| \left(\overleftarrow{N}_x(v) \setminus C \right) \cap \overleftarrow{N}_x(u) \right|}{\left| \overleftarrow{N}_x(u) \setminus C \right|} \cdot w_u.$$

Now consider the vertex u over the whole algorithm. The vertex u is charged for each vertex in $\overleftarrow{N}_x(u)$ at most once. By the above it is charged at most $2w_u/|\overleftarrow{N}_x(u)|$ for the first vertex it is charged for, at most $2w_u/(|\overleftarrow{N}_x(u)|-1)$ for the second vertex, and at most $2w_u/(|\overleftarrow{N}_x(u)|-i+1)$ for the i-th vertex. Thus a vertex u with $|\overleftarrow{N}_x(u)| = \Delta_u$ is charged at most $2w_u \cdot H_{\Delta_u} \in O(w_u \log(\Delta_u))$, and hence we obtain an $O(\log(n))$ approximation of S^*.

References

[1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. Announced at SODA'96.

[2] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *SODA*, pages 1497–1514, 2014.

[3] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. Announced at FOCS'02.

[4] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. *Advances in Computers*, 58:117–148, 2003.

[5] R. Bloem, H. N. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. *Form. Methods Syst. Des.*, 28(1):37–56, 2006. Announced at FMCAD'00.

[6] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, C-35(8):677–691, 1986.

[7] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[8] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^20 states and beyond. In *LICS*, pages 428–439, 1990.

[9] K. Chatterjee, M. Henzinger, M. Joglekar, and N. Shah. Symbolic algorithms for qualitative analysis of Markov decision processes with büchi objectives. *Form. Methods Syst. Des.*, 42(3):301–327, 2013. Announced at CAV'11.

[10] S. Chechik, D. H. Larkin, L. Roditty, G. Schoenebeck, R. E. Tarjan, and V. Vassilevska Williams. Better approximation algorithms for the graph diameter. In *SODA*, pages 1041–1052, 2014.

[11] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[12] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, September 2003. Announced at CAV'00.

[13] E. M. Clarke, K. L. McMillan, S. V. Aguiar Campos, and V. Hartonas-Garmhausen. Symbolic model checking. In *CAV*, pages 419–427, 1996.

[14] E.M. Clarke, O. Grumberg, and D. Peled. Symbolic model checking. In *Model Checking*. MIT Press, 1999.

[15] R. Gentilini, C. Piazza, and A. Policriti. Symbolic graphs: Linear solutions to connectivity related problems. *Algorithmica*, 50(1):120–158, 2008. Announced at SODA'03.

[16] J. Håstad and A. Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

[17] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[18] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[19] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, New York, NJ, USA, 1997.

[20] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.

[21] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992.

[22] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[23] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *STOC*, pages 515–524, 2013.

[24] F. Somenzi. Binary Decision Diagrams. In M. Broy and R. Steinbrüggen, editors, *Calculational System Design*, F]: [Nato ASI series, pages 303–366. IOS Press, 1999.

[25] R. E. Tarjan. Depth First Search and Linear Graph Algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.