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 induces a directed graph: Each vertex represents a state of the system and each directed edge represents a possible state transition.

- The resulting graphs are huge (exponential in the number of variables)
- Explicit representation of graphs is infeasible
- Graphs are implicitly represented, e.g., by binary decision diagrams (BDDs)

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.

Symbolic Algorithms allow the same operations as standard RAM algorithms, except:

- for access to the vertices and edges of the input graph, and
- for manipulation of sets of vertices.

Our lower bounds are by reductions from the Set Disjointness Problem from communication complexity.

### Set Disjointness Problem (with parameter $k$)

- Universe $U = \{0, \ldots, k-1\}$
- Alice’s input: bit vector $x$ of length $k$
- Bob’s input: bit vector $y$ of length $k$
- Function $f(x,y) = 1$ iff all $0 \leq i \leq k-1$ either $x_i = 0$ or $y_i = 0$

**Thm 1.** ([3]) Any protocol for Set Disjointness sends $\Omega(k)$ bits in the worst case.

We exemplify our technique by one result for SCCs. Given an instance $(x,y)$ of Set Disjointness with $k = \ell \cdot \ell$ we define the graph $G = (V,E)$ with:

- $V = \bigcup_{i=0}^{\ell-1} V_i$ with $V_i = \{v_{i,0}, \ldots, v_{i,\ell-1}\}$
- Forward Edges $\{v_i,v_j\}$ if $i < \ell$ or $i = \ell$ and $j < \ell$
- Backward Edges $\{v_{i+j},v_i\}$ if $x_{i+j} = 0$ or $y_{i+j} = 0$

The graph $G$ for $k = 4, \ell = 2$, and $x = (0,0,1,1), y = (1,1,0,1)$ is given below:

![Graph diagram](image)

**Lemma 2.** $f(x,y) = 1$ iff $G$ has exactly $\ell$ SCCs.

One can show that each symbolic operation can be performed in a communication protocol with $O(1)$ bits of communication.

**Lemma 3.** For any algorithm that computes SCCs with $N$ symbolic operations there is a communication protocol for Set Disjointness that requires $O(N)$ communication. Thus by Thm 1 we have a $\Omega(n)$ lower bound for computing SCCs, even in graphs of constant diameter.

**Thm 2.** Any symbolic algorithm that computes the SCCs of graphs with $n$ vertices needs $\Omega(n)$ symbolic one-step operations.

### Communications Complexity

**Two-party Communication Complexity Model**

- Two parties Alice, Bob
  - Alice and Bob need to compute a function $f(x,y)$, but $x \in X$ is only known to Alice and $y \in Y$ is only known to Bob.
  - Aim: send as few bits as possible between Alice and Bob.
  - We do not count computation but only communication.

- Communication protocol ("the algorithm")
  - Determines which player sends which bits when.
  - Is fixed beforehand, and is known to both Alice and Bob.

### Set-based Symbolic Model of Computation

Symbolic Algorithms allow the same operations as standard RAM algorithms, except:

- for access to the vertices and edges of the input graph, and
- for manipulation of sets of vertices.

**Symbolic Operations**

- Access to edges: Only through One-step operations: Pre and Post:
  - Successor Operation: $\text{Post}(S) = \{v \in V \mid \exists s \in S : (v,s) \in E\}$
  - Predecessor Operation: $\text{Pre}(S) = \{v \in V \mid \exists s \in S : (s,v) \in E\}$

- Manipulation of sets of vertices via basic set operations: Given one or two sets of vertices, we can perform basic set operations like union, intersection or complement.

**Symbolic Space**

The Symbolic Space requirement of an algorithm is the number of sets simultaneously stored by the algorithm. As we deal with compact representation of huge graphs and the number of stored sets should be small w.r.t. size of the graph, i.e., $O(\log n)$.

### Our Results

We provide the first lower bounds for the Set-based Symbolic Model of Computation and provide matching upper and lower bounds for fundamental problems.

### Computing the (Approximate) Diameter of a Graph

The diameter $D$ of a graph is defined as the largest finite distance in the graph. Many graphs, e.g., in hardware verification, have small diameter $D$ which can be exploited for more efficient algorithms. We provide a lower bound and a symbolic approx. scheme.

<table>
<thead>
<tr>
<th>Symbolic Operations</th>
<th>Exact $1 + \varepsilon$</th>
<th>$3/2 - \varepsilon$</th>
<th>$2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Bound</td>
<td>$O(n \cdot D)$</td>
<td>$O(\sqrt{D})$</td>
<td>$O(D)$</td>
</tr>
<tr>
<td>Lower Bound</td>
<td>$\Omega(n)$</td>
<td>$\Omega(n)$</td>
<td>$\Omega(n)$</td>
</tr>
</tbody>
</table>

### Deciding Fundamental Objectives

Starting from a vertex we want to decide whether there exists an infinite path satisfying certain objectives in the analysis of reactive systems, e.g., Reachability, Safety, Liveness (Büchi), and co-liveness (coBüchi).

<table>
<thead>
<tr>
<th>Symbolic Operations</th>
<th>Reach</th>
<th>Safety</th>
<th>Büchi</th>
<th>coBüchi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Bound</td>
<td>$\Omega(n)$</td>
<td>$\Omega(2n)$</td>
<td>$\Omega(n)$</td>
<td>$\Omega(2n)$</td>
</tr>
<tr>
<td>Lower Bound</td>
<td>$\Omega(D)$</td>
<td>$\Omega(D)$</td>
<td>$\Omega(D)$</td>
<td>$\Omega(D)$</td>
</tr>
</tbody>
</table>

### Strongly Connected Components

Computing Strongly Connected Components (SCCs) is at the heart of the algorithms for the above objectives. The best known upper bound $O(\min(n, D \cdot |SCC(G)|))$ is by the algorithm of Gentilini et al. [2]. We give matching lower bounds and a refined analysis in terms of the diameters of the SCCs. Here $|SCC(G)|$ is the set of SCCs of $G$ and $D_C$ is the diameter of the SCC $C$. We obtain that computing SCC is in $O(\min\{n, D \cdot |SCC(G)|, \sum_{C \in SCC(G)}|D_C| + 1\})$.

### References