Theory of Parallel and Distributed Systems (WS2016/17)
Kapitel 4
Lower Bounds (V4 Part)

Walter Unger

Lehrstuhl für Informatik 1

Inhalt I

1 Motivation
2 Coloring Cycles
   - Preparations
   - Results
3 P-Completeness
   - NP-hard
   - Poly-Logarithmic Time versus Memory
4 First Reduction
5 More Reductions
   - Definition
   - Generability
   - Remarks
   - CVP, MCVP, TSMCVP
   - CFE
   - LFMIS
   - LFMC
   - DFS
   - MAXFLOW
Motivation

- Shows the quality of any algorithm.
- Interesting property of any problem.
- Interesting techniques to prove lower bounds.
  - No assumption about the used algorithms
  - Have to show a property for all algorithms and some inputs.
  - For all algorithms there is an input, such that the running time is at least....
  - Typically more complicated than upper bounds.
- Here we start with lower bounds for coloring cycles.
Ideas

- Model distributed computers, connected in a cycle.
- No assumption about structure of the algorithm.
- Assume the running time is $t$ on a cycle of length $n$.
- Step one: Normalize the behavior of the algorithm.
- Step two: Extend the possible inputs for the algorithms, such that the algorithm works still correct.
- Step three: find some contradiction.
Step one: Normalize the behavior of the algorithm

- After \( t \) steps a node may know the identifiers of \( 2t + 1 \) nodes. Let

\[
W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n \land x_i = x_j \Rightarrow i = j\}
\]

be the set of possible surroundings.

- It is not necessary to color any node before step \( t \):
  - Each node may simulate the behavior of the \( 2t + 1 \) nodes in the surrounding.
  - Or each nodes sends also the history of colors.

- Thus after \( t \) rounds node \( v \) has the topological information \( \zeta(v) \):

\[
\zeta(v) = (x_1, x_2, \ldots, x_s) \in W_{s,n} \text{ with } s = 2t + 1.
\]

- Any algorithm will use some deterministic strategy \( \pi \) to find a coloring:

\[
c(v) \leftarrow \Phi_\pi(\zeta(v)) \text{ with } \Phi_\pi : W_{s,n} \mapsto \{1, 2, \ldots, c_{\text{max}}\}.
\]
Step two: Extend the possible inputs

- The set of nodes is $W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n\}$.
- The set of edges is $E_{s,n}$. They contain any possible edge in any cycle:
  \[
  E_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\}
  \]
- This graph $B_{s,n} = (W_{s,n}, E_{s,n})$ has $\binom{n}{s} s!$ nodes of degree $n - s$. Thus it has $(n - s)\binom{n}{s} s!$ edges.

Theorem (Coloring $B_{s,n}$)

If an algorithm $\pi_t$ colors any cycle of length $n$ with $c$ colors in $t$ steps, then it will define a legal coloring of $B_{s,n}$. 
Step two: Extend the possible inputs

\[ W_{s, n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n\} \quad \text{and} \quad E_{s, n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\} \]

Theorem (Coloring \(B_{s,n}\))

If an algorithm \(\pi_t\) colors any cycle of length \(n\) with \(c\) colors in \(t\) steps, then it will define a legal coloring of \(B_{s,n}\).

- Assume algorithm \(\pi_t\) colors cycle of length \(n\) correct, but not the \(B_{s,n}\).
- Thus there is an edge \(e = ((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \in E_{s,n}\) which is not colored correctly.
- Take this edge and extend it to a cycle of length \(n\) using the missing identifiers.
- This cycle with this order is not colored correctly.
- Contradiction.
Lower Bound for even length cycle

\[ W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n\} \text{ and } E_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\} \]

**Theorem (Distributed Coloring C_{2n})**

*Any deterministic distributed algorithm uses* \( n - 1 \) *rounds to color a cycle of length* \( 2n \) *with 2 colors.*

- Assume the algorithm runs in time *\( t \leq n - 2 \).*
- Then this algorithm will color the graph \( B_{2t+1,2n} \) with 2 colors.
- \( B_{2t+1,2n} \) is bipartite for *\( t \leq n - 2 \).*
- We will now construct the following cycle:

\[
\begin{align*}
(1, 2, 3, \ldots, 2t + 1) & \rightarrow (2, 3, 4, \ldots, 2t + 2) \\
\rightarrow (3, 4, 5, \ldots, 2t + 3) & \rightarrow (4, \ldots, 2t + 3, 1) \\
\rightarrow \ldots & \rightarrow (2t + 2, 2t + 3, 1, 2, \ldots, 2t - 1) \\
\rightarrow (2t + 3, 1, 2, \ldots, 2t) & \rightarrow (1, 2, 3, \ldots, 2t + 1)
\end{align*}
\]
Lower Bound for even length cycle

\[ W_{s,n} = \{(x_1, x_2, \ldots, x_s) | 1 \leq x_i \leq n \} \text{ and } E_{s,n} = \{(x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1}) | x_1 \neq x_{s+1} \} \]

**Theorem (Parallel Coloring \( C_{2n} \))**

*Any deterministic parallel algorithm uses \( \log n \) rounds to color a cycle of length \( 2n \) with 2 colors.*

- Assume the algorithm runs in time \( t \leq \log n \).
- The best way to collect information is doubling (see lower bound for broadcast/accumulation).
- Then we may use its strategy to construct a distributed version running in \( t \) time.
- Contradiction.
Step four: find some contradiction

\[ W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n\} \text{ and } E_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\} \]

- We want a lower bound for the 3-coloring of cycles.
- Step a) Show \( \chi(B_{2t+1}, n) \geq \log^2 t n \).
- Step b) Show \( \chi(\tilde{B}_s, n) \leq \chi(B_s, n) \).
- Step c) Use the line-graph construction.
- Step d) Show property for coloring a line-graph.
- Step e) Put everything together.
Construction of $\tilde{B}_{s,n}$

$W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n\}$ and $E_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\}$

- **Remember:**
  - $W_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_i \leq n \land x_i = x_j \Rightarrow i = j\}$
  - $E_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\}$
  - $B_{s,n} = (W_{s,n}, E_{s,n})$

- **Construct now:**
  - $\tilde{W}_{s,n} = \{(x_1, x_2, \ldots, x_s) \mid 1 \leq x_1 < x_2 < \ldots < x_s \leq n\}$
  - $\tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\}$
  - $\tilde{B}_{s,n} = (\tilde{W}_{s,n}, \tilde{E}_{s,n})$

- Thus $\tilde{B}_{s,n}$ is by definition a non-directed sub-graph of $B_{s,n}$.

**Lemma**

We have: $\chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n})$. 

Line-Graphs

\[ W_{s,n} = \{ (x_1, \ldots, x_s) \mid x_1 < \ldots < x_s \}, \quad E_{s,n} = \{ (x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1}) \mid x_1 \neq x_{s+1} \}, \quad \chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n}) \]

Definition (Line-Graphs)

Let \( G = (V, E) \) be an directed graph. \( DL(G) = (E, E') \) is called line-graph of \( G \), iff

\[ E' = \{ (e, e') \mid e, e' \in E \land e \cap e' \neq \emptyset \}. \]

A graph \( H \) is called directed line-graph, iff a graph \( G \) exists, with \( DL(G) = H \).
Line-Graphs

$\tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}, \tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1})) \mid x_1 \neq x_{s+1}\}, \chi(\tilde{B}_{s,n}) \leq \chi(\tilde{B}_{s,n})$

**Definition (Line-Graphs)**

Let $G = (V, E)$ be an undirected graph. $L(G) = (E, E')$ is called line-graph of $G$, iff

$$E' = \{\{e, e'\} \mid e, e' \in E \land e \cap e' \neq \emptyset\}.$$ 

A graph $H$ is called line-graph, iff a graph $G$ exists, with $L(G) = H$. 

\[\begin{array}{ccc}
a & \rightarrow & b \\
| & \downarrow & | \\
x & \rightarrow & y \\
| & \downarrow & | \\
b & \rightarrow & c
\end{array}\]
Beispiel 1

\[ \tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}, \quad \tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1})) \mid x_1 \neq x_{s+1}\}, \quad \chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n})\]
Beispiel 2

\[ \tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}, \quad \tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1})) \mid x_1 \neq x_{s+1}\}, \quad \chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n}) \]
Beispiel 3

\[ \tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}, \quad \tilde{E}_{s,n} = \{(x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1}) \mid x_1 \neq x_{s+1}\}, \quad \chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n}) \]
DeBruijn network of dimension $d$

$\tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}$, $\tilde{E}_{s,n} = \{(x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1}) \mid x_1 \neq x_{s+1}\}$, $\chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n})$

- DeBruijn network:
  
  \[
  DB(d) = (V_{DB(d)}, E_{DB(d)}^s \cup E_{DB(d)}^{se})
  \]
  
  \[
  V_{DB(d)} = \{0, 1\}^d
  \]
  
  \[
  E_{DB(d)}^s = \{(aw, wa) \mid a \in \{0, 1\}, aw, wa \in V_{DB(d)}\}
  \]
  
  \[
  E_{DB(d)}^{se} = \{(aw, wb) \mid a \in \{0, 1\}, b = 1 - a, aw, wb \in V_{DB(d)}\}
  \]

  - Number of nodes: $2^d$
  - Degree: $2 + 2$
  - Number of edges: $2^{d+1}$
  - Diameter: $d$

**Lemma**

We have: $DB(d + 1) = DL(DB(d))$ for $d \geq 1$. 
Line-Graph Properties of $\tilde{B}_{s,n}$

$\tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}$, $\tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1})) \mid x_1 \neq x_{s+1}\}$, $\chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n})$

**Lemma**

1. $\tilde{B}_{1,n}$ is the complete directed graph of $n$ nodes.
2. We have $\tilde{B}_{s+1,n} = LG(\tilde{B}_{s,n})$ for $s \geq 1$.

**Proof:**

1. By definition: $\tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_s, x_{s+1})) \mid x_1 \neq x_{s+1}\}$.

2. By construction:

   - In $\tilde{B}_{s,n}$: $(x_1, x_2, \ldots, x_s) \rightarrow (x_2, x_3, \ldots, x_{s+1})$ and $(x_2, x_3, \ldots, x_{s+1}) \rightarrow (x_3, x_4, \ldots, x_{s+2})$.
   - In $V(DL(\tilde{B}_{s+1,n}))$: $((x_1, x_2, \ldots, x_s), (x_2, x_3, \ldots, x_{s+1}))$ and $((x_2, x_3, \ldots, x_{s+1}), (x_3, x_4, \ldots, x_{s+2}))$.
   - In $V(DL(\tilde{B}_{s+1,n}))$: $(x_1, x_2, \ldots, x_s, x_{s+1})$ and $(x_2, x_3, \ldots, x_{s+1}, x_{s+2})$ (simplified).
   - In $E(DL(\tilde{B}_{s+1,n}))$: $((x_1, x_2, \ldots, x_s, x_{s+1}), (x_2, x_3, \ldots, x_{s+1}, x_{s+2}))$. 
Bounds for Coloring Line-Graphs

\[ \tilde{W}_{s,n} = \{(x_1, \ldots, x_s) \mid x_1 < \ldots < x_s\}, \quad \tilde{E}_{s,n} = \{((x_1, x_2, \ldots, x_s), (x_2, \ldots, x_{s+1})) \mid x_1 \neq x_{s+1}\}, \quad \chi(\tilde{B}_{s,n}) \leq \chi(B_{s,n}) \]

**Lemma**

Let \( H \) be any directed graph, then we have \( \chi(DL(H)) \geq \log(\chi(H)) \).

**Proof:**

- Let \( k = \chi(DL(H)) \), thus we can color the nodes from \( DL(H) \) with \( k \) colors.
- Thus we may color the edges from \( H \) with \( k \) colors: \( \chi'(H) \leq k \).
- For any edge \( e = (v, w) \) of \( H \) let \( c'(e) \) be the color of \( e \).
- Define now a coloring of the nodes \( v \) of \( H \):
  \[ c(v) = \bigcup_{v \in e} c'(e) \]
  This is a correct \( 2^k \) node-coloring of \( H \).
- Thus \( \chi(H) \leq 2^k = 2^{\chi(DL(H))} \).
- Thus \( \log(\chi(H)) \leq \chi(DL(H)) \).
Lemma

We have $\chi(\tilde{B}_s, n) \geq \log^{(s-1)} n$.

Proof:

- $\tilde{B}_1, n$ is the complete directed graph of $n$ nodes.
- $\chi(\tilde{B}_1, n) = n$.
- We have $\tilde{B}_{s+1}, n = LG(\tilde{B}_s, n)$ for $s \geq 1$.
- We have already: $\chi(DL(H)) \geq \log(\chi(H))$.
- Thus we get $\chi(\tilde{B}_{s+1}, n) \geq \log(\chi(\tilde{B}_s, n))$.
- Thus we get $\chi(\tilde{B}_s, n) \geq \log^{(s-1)}(\chi(\tilde{B}_1, n))$.
- Thus we get $\chi(\tilde{B}_s, n) \geq \log^{(s-1)}(n)$. 
Theorem

Any deterministic distributed algorithm needs at least $\frac{1}{2}(\log^* n - 1)$ rounds to color a cycle of length $n$ with 3 colors.

Proof:

- We have already: $\chi(\tilde{B}_s, n) \geq \log^{(s-1)} n$, resp.:
- We have already: $\chi(\tilde{B}_{2t+1}, n) \geq \log^{(2t)} n$.
- We also have: $\chi(\tilde{B}_{2t+1}, n) \leq 3$.
- Thus we get: $\log^{(2t)} n \leq 3$ and finally
- $2t \geq \log^* n - 1$. 

Comparison with NP-complete

- NP-hard: the “most complicated” problems for the class $\mathcal{NP}$.
- Theory of NP-complete problems was developed, to “explain” that for many problems no polynomial time deterministic algorithm is known.
- A problem is NP-hard $\iff$
  - It is possible in polynomial time to reduce any other problem from NP to a NP-hard problem.
  - First NP-hard problem: Does a non-deterministic TM stop in polynomial time?
  - All other NP-hard problems were reduced from this.
- We assume (proof is still missing), that for these NP-hard problem no deterministic polynomial time algorithms exist.
- Thus we may assume, that for NP-complete problems no polynomial time deterministic parallel algorithm will be known using a polynomial number of processors.
Some Observations about Problems from $\mathcal{P}$

- Any problem from $\mathcal{P}$ is a candidate for a parallel algorithm.
- A problem is well to parallelize, if there is a parallel deterministic algorithm
  - which uses a polynomial number of processors
  - and runs in poly-logarithmic time.
- These class is called $\mathcal{NC}$ (Nick’s Class).
- We have by definition: $\mathcal{NC} \subset \mathcal{P}$.
- Important Question: $\mathcal{NC} \equiv \mathcal{P}$?
- It is assumed, $\mathcal{NC} \neq \mathcal{P}$
- Thus the theory of $\mathcal{P}$-complete problems was developed.
- And it follows just the technique of $\mathcal{NPC}$.
Recall the situation for \( \mathcal{NPC} \) (try to separate \( \mathcal{NP} \) from \( \mathcal{P} \)):
- Hard problem: stops a non-deterministic TM in polynomial time?
- Reduction: runs deterministic in polynomial time.

Or in other words:
- Hard problem: a nice candidate from the “hard class”.
- Reduction by computation within the “easy class”.

Uses the analogous technique for \( \mathcal{P} \) (try to separate \( \mathcal{P} \) from \( \mathcal{NC} \)):
- Hard problem: stops a deterministic TM in polynomial time?
- Reduction: runs deterministic in time poly-logarithmic time.
- Analog reduction: using poly-logarithmic memory.
We had:

- Reduction: runs deterministic in time poly-logarithmic time.
- Analog reduction: using poly-logarithmic memory.

We will transform an algorithm running deterministic in time poly-logarithmic time into one using poly-logarithmic memory.

- From the parallel algorithm running deterministic in time poly-logarithmic
- we build a circuit network.
- This has poly-logarithmic depth and polynomial width.
- To compute any value within this circuit network we only need to store the values on a path towards the input.
- Thus we have poly-logarithmic memory (and do not care about the running time).
A problem $X$ is called $\mathcal{P}$-complete, iff:

- $X$ is in $\mathcal{P}$.
- Any problem $Y$ from $\mathcal{P}$ could be reduced to $X$ with poly-logarithmic memory.
- I.e. there is a function $f$ computable with poly-logarithmic memory, such that
  $\forall w \in \Sigma^* : w \in X \Leftrightarrow f(w) \in Y$
Definition (Generability)

- **Input:** Set $X$ with binary operator $\circ$, $T \subseteq X$ and $s \in X$.
- **Output:** Is $s$ in the closure of $T$ in terms of $\circ$.

Let $S \circ S := \{a \circ b \mid a, b \in S\}$.

Algorithm for $\text{Generability}(X, \circ, S, s)$ in $\mathcal{P}$:
- while $S \neq S \circ S$ do $S = S \circ S$
- return $s \in S$.

We will first show $\mathcal{P}$-completeness for a ternary operation.
- i.e. $\circ$ will be substituted by $\text{next}(u, v, w)$.
- Reduction from the halting problem of a deterministic TM.
First Reduction

**Definition (Generability’)**

- Input: Set $X$ with ternary operator $\text{next}(u, v, w)$, $T \subset X$ and $s \in X$.
- Output: Is $s$ in the closure of $T$ in terms of $\circ$.

**Definition (TM)**

- Input band with postitions $0, 1, 2, \cdot T(n) + 1$.
- By $c(i, j) \in \Sigma$ we denote the contents at position $i$ at time $j$.
- Let $c(0, j) = c(T(n) + 1, j) = \$ $ for all time points $j$.
- The function $\text{trans}$ defines the transitions for the TM.
- I.e. $c(p, t + 1) = \text{trans}(c(p - 1, t), c(p, t), c(p + 1, t))$.
- Input given at positions $c(p, 0)$ ($\forall p : 1 \leq p \leq T(n)$).
- Output placed at $c(1, T(n))$ where $\#$ encodes a “true”.
First Reduction (Generability’) 

Theorem:

Generability’ is $\mathcal{P}$-complete.

Proof:

- A TM may be transformed in $\mathcal{NC}$ into the above form.
- The triple $(t, p, sym)$ encodes that the contents at position $p$ and time $t$ is $sym$.
- We will now compute the input for Generability’ from the above TM:
  - $X = \{0, 1, \ldots, T(n)\} \times \{0, 1, \ldots, T(n) + 1\} \times \Sigma$.
  - $T = \{(0, i, c(0, i)) | 0 \leq i \leq T(n) + 1\}$
  - $s = (T(n), 1, \#)$
  - $next = trans$
- This can be done in $\mathcal{NC}$.
- $s$ is in the closure of $next$ iff TM stops with “True”.
First Reduction (Generability)

Theorem:
Generability ist \( P \)-complete.

Proof:
- Reduktion von Generability’
- \( X' := X \cup X^2 \) (\( X \) form above)
- \( T = \{(0, i, c(0, i)) \mid 0 \leq i \leq T(n) + 1\} \)
- \( s = (T(n), 1, \#) \)
- It remains to define \textit{next} as a binary Operator \( \odot \).
- \( u \odot v := (u, v) \) and
- \( (u, v) \odot w := \text{next}(u, v, w) \)
Lemma:
If $\odot$ is associative, the is the corresponding Generability-Problem in $NC$.

Proof:
- We transform this problem into the reachability problem on a graph $G$.
- If $x \odot z = y$ then generate an edge $(x, y)$ with label $z$.
- $G = (X, E)$ with $E = \{(x, y) \mid \exists z \in X : x \odot z = y\}$
- and $\forall (x, y) \in E : l(x, y) := \{z \in X \mid x \odot z = y\}$.
- If there is a path from $a \in T$ to $s$ using edges with labels $b, c, d, \cdots$, then we may generate $s$ by $((\cdots (a \odot b) \odot c) \odot d) \cdots$.
- If $s$ may be generated by using elements from $T$ with $\odot$, then we may have also the form $((\cdots (a \odot b) \odot c) \odot d) \cdots$.
- This will give us a path in the above constructed graph $G$. 
Reduktion (CVP)

Definition (CVP)
- Input: A boolean circuit with some input.
- Output: Is the output value *true*.

Theorem:
The problem CVP is \( \mathcal{P} \)-complete.

Proof
- Reduction form the Generability Problem.
- The elements from \( T \) are the inputs for the circuit with value *true*.
- The output will be the element \( s \).
Details for the Reduction (CVP)

- For each element \( x \) from \( X \setminus T \) do:
  - Compute pairs from \( X \times X \) which will give \( x \):
    \[
    (y_1, z_1), (y_2, z_2), (y_3, z_3), \ldots, (y_{k_x}, z_{k_x})
    \]
  - I.e. \( y_i \odot z_i = x \) for all \( 1 \leq i \leq k_x \).
  - This is one part of the circuit:
    \[
    x = \bigvee_{i=1}^{k_x} y_i \land z_i
    \]
  - Thus \( x \) will have the value \( true \) iff \( x \) may be generated.
  - Thus \( s \) will have the value \( true \) iff \( s \) may be generated.
  - This construction is in \( \mathcal{NC} \).
### Definition (MCVP)

- **Input:** A boolean circuit with some input and only operators $\lor$ und $\land$.
- **Output:** Is the output value $true$.

### Theorem:

The MCVP is $\mathcal{P}$-complete.

### Proof:

- Similar proof to the CVP problem.
Reduktion (TSMCVP)

**Definition (TSMCVP)**
- Input: A boolean circuit with some input and only operators \(\lor\) und \(\land\) and a topological sorting of the values.
- Output: Is the output value \(true\).

**Theorem:**
The TSMCVP is \(P\)-complete.

**Proof:**
- Similar proof to the CVP problem.
- Note: the proof for Generability’ did contain a topological sorting.
- This was the lexicographical order of the elements \((t, p, sym)\).
- This order could easily be preserved during the following step of the reduction.
Reduktion (CFE)

**Definition (CFE)**

- **Input:** a context-free grammar $G$.
- **Output:** will $G$ generate the empty word $\varepsilon$.

**Theorem:**

The CFE is $\mathcal{P}$-complete.

**Proof (Reduktion from Generability Problem):**

- Let $(X, T, \odot, s)$ be the input for the Generability problem.
- Let $X$ be the non-terminals of $G$.
- Let $s$ be the start symbol.
- For each $x \in T$ generate the rule: $x \rightarrow \varepsilon$.
- If $y \odot z = x$ generate the rule: $x \rightarrow yz$.
- Note: If $G$ contains no $\varepsilon$-rules, then is CFE in $\mathcal{NC}$.
Reduction (LFMIS)

Definition (LFMIS)
- Input: non-directed graph $G = (V, E)$.
- Output: lexicographical first maximum independent set (IS) of $G$.

Theorem:
The LFMIS is $\mathcal{P}$-complete.

Proof (Reduction from MCVP problem)
- Consider the greedy-strategy for the LFMIS problem.
- Let $V = \{v_1, v_2, \ldots, v_n\}$ nodes for the MCVP Problems in their topological sorting.
- Let $\{v_1, v_2, \ldots, v_e\}$ be the input nodes and $v_n$ be the output node.
- We construct $G = (V', E')$ as input for LFMIS.
Continuation of the Reduction (LFMIS)

- Let $V' = \{v'_1, v''_1, v'_2, v''_2, \ldots, v'_n, v''_n\}$ be numbered from 1 till $2n$.
- The numbers of $v'_i, v''_i$ are exchanged, if
  - $v_i$ is an or-node or
  - $v_i$ is an input node with the value `false`.
- For all $1 \leq i \leq n$ generate an edge $\{v'_i, v''_i\}$.
- Thus only one of the nodes $v'_i, v''_i$ is in the IS.
- If $v$ is an and-node $G$ with input $u$ and $w$, then add the edges $\{v', u''\}$ and $\{v', w''\}$.
- Thus $v'$ will be in the IS iff non of the nodes $u'', w''$ are in the IS.
- If $v$ is an or-node $G$ with inputs $u$ and $w$, then add the edges $\{v'', u'\}$ and $\{v'', w'\}$.
- Thus $v''$ will be in the IS iff if non of the nodes $u', w'$ are in the IS.
- Thus LFMIS is simulating correctly the boolean circuit.
Reduction (LFMC)

Definition (LFMC)

- Input: non-directed graph $G = (V, E)$.
- Output: lexicographical first maximum clique of $G$.

Theorem:

Das LFMC is $\mathcal{P}$-complete.

Proof

- Reduction from LFMIS problem.
- Let $G = (V, E)$ be the input for LFMIS problem.
- Then $G = (V, \overline{E})$ will be input for the LFMC problem.
Given $G = (V, E)$

Procedure $\text{DFS}(v)$:

- if $\text{DFI}(v) = 0$ then
  - $\text{counter} := \text{counter} + 1$
  - $\text{DFI}(v) := \text{counter}$
  - $\forall w \in V : (v, w) \in E$ do $\text{DFS}(w)$
Reduction (DFS)

**Definition (DFS)**
- Input: directed graph $G = (V, E)$ and $v \in V$.
- Output: The values $DFI(w)$ of the call $DFS(v)$ for all $w \in V$.

**Theorem:**
The DFS is $\mathcal{P}$-complete.

**Proof**
- Reduction from CVP problem with $\odot := \overline{x} \lor \overline{y} = \overline{x} \land \overline{y}$
- It is easy to see, that this version of CVP Problem is also $\mathcal{P}$-complete.
- Idea: for each value of $v$ in the input of CVP, will be in $G = (V, E)$ two nodes $s$ and $t$, with $v$ is true iff $DFI(s) < DFI(t)$. 
Continuation of the Reduction (DFS)

- Let $v_1, v_2, \cdots, v_n$ be the nodes of the circuit.
- For each $v_i$ we will build a sub-graph $G_i$.
- These sub-graphs $G_i$ will be edge-disjoint, but not node-disjoint.
- $G_i$ and $G_j$ ($i < j$) may have common nodes $i \neq j$.
- $v_i$ has $v_{i_1}$ and $v_{i_2}$ as input nodes
- and the nodes $v_{o_1}, v_{o_2}, v_{o_3}, \cdots, v_{o_k}$ use $v_i$ as input.
- Then has $G_i$ for $k = 3$ the following structure.
- We indicate the order of the edges in the adjacency list by the number of arrow heads.
- If $v_i$ is an input node in the circuit and the nodes $v_{o_1}, v_{o_2}, v_{o_3}, \cdots, v_{o_k}$ use $v_i$ as input, then we will have a simplified graph $G_i$. This is seen as the second one.
Continuation of the Reduction (DFS)

$last(i - 1)$

$first(i)$

$v_i$ ist intern

$t(i)$

$last(i)$

$last(i)$

$i_1 \neq i$

$i_2 \neq i$

$s(i)$

$i \neq o_1$

$i \neq o_2$

$i \neq o_3$
Continuation of the Reduction (DFS)

\( last(i - 1) \)

\( first(i) \)

\( s(i) \)

\( v_i \) ist Input

\( last(i) \)

\( t(i) \)

\( \text{last}(i - 1) \)

\( \text{first}(i) \)

\( s(i) \)

\( v_i \) ist Input

\( \text{last}(i) \)

\( t(i) \)
Continuation of the Reduction (DFS)

- The DFS run starts at $first(1)$.
- After $last(i)$ will be the next visited node $first(i + 1)$.
- The order how $s(i)$ and $t(i)$ in $G_i$ are visited, will be given by the value of $v_i$.
- After $last(n)$ is visited, is each graph $G_i$ is also visited, excluding some minor parts.
Continuation of the Reduction (DFS)

**Lemma**

We consider a DFS-run in $G$ stating in node $\text{first}(1)$:

- If $v_i$ has the value $\text{true}$, then $s(i)$ will be visited before $t(i)$ and the nodes $i\#o_1, i\#o_2, \cdots, i\#o_k$ are visited after $\text{first}(i)$ and before $\text{last}(i)$.

- If $v_i$ has the value $\text{false}$, then the node $t(i)$ will be visited before $s(i)$ and none of the nodes $i\#o_1, i\#o_2, \cdots, i\#o_k$ will be visited in the interval between $\text{first}(i)$ and $\text{last}(i)$ visits.

**Proof:**

- By induction:

- Start of induction, consider all input-nodes.

- Induction-step, Assume above statement holds for all graphs $G_j$ ($1 \leq j < i$).
Continuation of the Reduction (Start of Induction)

- If $v_i$ has the value *true*, then we visit $s(i)$ before $t(i)$ and the nodes $i\#o_1, i\#o_2, \ldots, i\#o_k$ are visited after $first(i)$ and before $last(i)$.
Continuation of the Reduction (Induction-Step)

- If $v_i$ has the value *true*, then $s(i)$ will be visited before $t(i)$ and the nodes $i\#o_1, i\#o_2, \ldots, i\#o_k$ are visited after $\textit{first}(i)$ and before $\textit{last}(i)$.
- Then the nodes $v_{i_1}$ and $v_{i_2}$ have the value *false*.
Continuation of the Reduction (Induction-Step)

- If \( v_i \) has the value \textit{false}, then the node \( t(i) \) will be visited before \( s(i) \) and none of the nodes \( i \neq o_1, i \neq o_2, \ldots, i \neq o_k \) will be visited in the interval between \( \text{first}(i) \) and \( \text{last}(i) \) visits.

- Then one of the nodes \( v_{i_1} \) or \( v_{i_2} \) has the value \textit{true}.
Continuation of the Reduction (DFS)

- The construction is a NC-Reduction.
- The construction is the direct simulation of the operations of the circuit.
- The construction may be also given for non-directed graphs.
Reduction (MAXFLOW)

**Definition (MAXFLOW)**
- Input: directed graph $G = (V, E)$, $s, t \in V$ and capacity function $c : E \mapsto \mathbb{N}$.
- Output: Maximal flow from $s$ to $t$, i.e. function $f : E \mapsto \mathbb{N}$.
  - with: $\forall e \in E : f(e) \leq c(e)$
  - and: $\forall v \in V \setminus \{s, t\} : \sum_{e = (a, v) \in E} f(e) = \sum_{e = (v, a) \in E} f(e)$

**Theorem:**
The MAXFLOW problem is $\mathcal{P}$-complete.

**Proof:**
- Reduction from the problem CVP.
- Show, even to compute the parity of a flow (PMAXFLOW), is $\mathcal{P}$-complete.
Continuation of the Reduction (MAXFLOW)

- W.l.o.g. out-degree of a input node 1.
- W.l.o.g. out-degree of a node is at most 2.
- W.l.o.g. circuit is revers topological sorted, i.e. $v_0$ is the output node.
- W.l.o.g. $v_0$ is an or.
- Given is the circuit graph $G = (V, E)$.
- Input for PMAXFLOW: $G' = (V \cup \{s, t\}, E')$.
- $E \subset E'$.
- $E' \subset E \cup \{(s, v), (v, t) \mid v \in V\}$
Continuation of the Reduction (MAXFLOW)

- \(\forall (i, j) \in E : c((i, j)) = 2^i\).
- If the value of \(v_i\) is true then let: \(f((i, j)) = 2^i \quad (\forall (i, j) \in E)\).
- If the value of \(v_i\) is false then let: \(f((i, j)) = 0 \quad (\forall (i, j) \in E)\).
- Let \(d(0) = 1\) and otherwise let \(d(i)\) be the out-degree of \(v_i\).
- Let \((k, i), (j, i) \in E\), and let \(\text{surplus}(i) := 2^k + 2^j - d(i)2^i\).
- \(\forall i \in V : c(s, i) = 2^i\) if the value of \(v_i\) is true.
- \(\forall i \in V : c(s, i) = 0\) if the value of \(v_i\) is false.
- \(\forall i \in V : c(i, t) = \text{surplus}(i)\) if \(v_i\) is an and-node.
- \(\forall i \in V : c(i, s) = \text{surplus}(i)\) if \(v_i\) is an or-node.
- \(c(0, t) = 1\).
Continuation of the Reduction (MAXFLOW)

- $\forall i \in V : f(s, i) = c(s, i)$.
- $\forall i \in V : f(i, j) = c(i, j)$ if $v_i$ is an input-node.
- $\forall (i, j) \in E : f(i, j) = c(i, j) = 2^i$ if the value of $v_i$ is true.
- $\forall (i, j) \in E : f(i, j) = 0$ if the value of $v_i$ is false.
- $f(0, t) = 1$ if $v_0$ has the value true.
- Let $overflow(i)$ be the difference between the current input-flow and the output-flow.
  - $f((i, t)) = overflow(i)$ if $v_i$ is an and-node.
  - $f((i, s)) = overflow(i)$ if $v_i$ is an or-node.
- Note: the defined function $f$ is a flow.
Lemma

The defined flow is optimal.

- Use enlarging pathes from \( s \) to \( t \):
  - An edge \( e = (i, j) \) in the path is called forward-edge if \( f(e) < c(e) \).
  - An edge \( e = (j, i) \) in the path is called backward-edge if \( f(e) > 0 \).

- Known: Flow is maximal if there is no enlarging path.

- Assume: there is an enlarging path.
  - A path starts at \( s \) with a backward-edge.
  - A path ends at \( t \) with a forward-edge.
Continuation of the Reduction (MAXFLOW)

Thus we have three consecutive nodes $j, i, k$ with:

- $j \neq t$.
- $k \neq s$.
- $(j, i)$ is a backward-edge.
- $(i, k)$ is a forward-edge.
- $(i, j), (i, k)$ are edges in $E'$.
- $f((i, j)) > 0$ and $f((i, k)) < c((i, k))$.

- $v_i$ may not be an input-node.
- $v_i$ may not be an and-node, because from $j \neq t$ and $f((i, j)) > 0$ we get that all outgoing edges are full.
- $v_i$ may not be an or-node, because from $k \neq s$ and $f((i, k)) < c((i, k))$ be get that all outgoing edges are without flow.