The Complexity of the Comparator Circuit Value Problem

Stephen Cook

Joint work with Yuval Filmus and Dai Tri Man Lê

Department of Computer Science
University of Toronto
Canada

Banff 2013

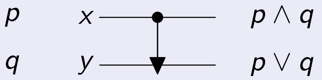
Outline of the talk

1. Define comparator circuits
2. Define CC as the class of problems reducible to $\textsc{Ccv}$ (the comparator circuit value problem)
3. Give interesting complete problems for CC
4. Introduce universal comparator circuits, with resulting robustness properties of CC.
5. Support the conjecture that CC and NC are incomparable using oracle separations.

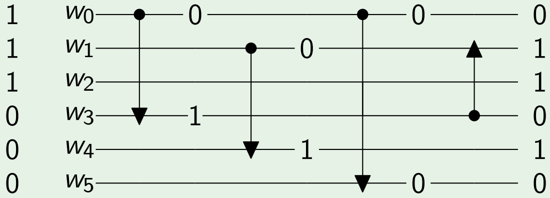
Comparator Circuits

- Originally invented for sorting, e.g.,
  - Batcher's $\mathcal{O}(\log^2 n)$-depth sorting networks ('68)
  - Ajtai-Komlós-Szemerédi (AKS) $\mathcal{O}(\log n)$-depth sorting networks ('83)
- Can also be considered as Boolean circuits.

Comparator gate

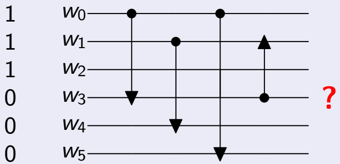


Example



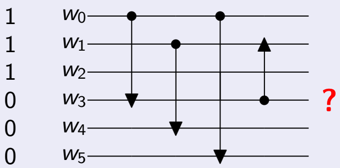
Comparator Circuit Value (CCV) Problem (decision)

Given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire.



Comparator Circuit Value ($\textsc{Ccv}$) Problem (decision)

Given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire.

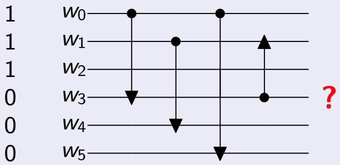


Comparator Circuit complexity class

1. $\mathsf{CC} = \{$decision problems AC^0 many-one-reducible to $\textsc{Ccv}\}$

Comparator Circuit Value (Ccv) Problem (decision)

Given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire.

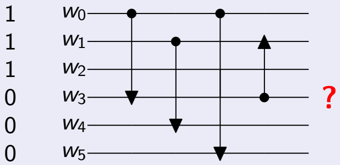


Comparator Circuit complexity class

1. CC $= \{$decision problems AC^0 many-one-reducible to Ccv$\}$
2. Subramanian ['90] Defined CC using log space many-one reducibility
3. We introduce universal comparator circuits and use them to show that the two definitions coincide.

Comparator Circuit Value ($\textsc{Ccv}$) Problem (decision)

Given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire.



Comparator Circuit complexity class

1. CC $= \{$decision problems AC^0 many-one-reducible to $\textsc{Ccv}\}$
2. Subramanian ['90] Defined CC using log space many-one reducibility
3. We introduce universal comparator circuits and use them to show that the two definitions coincide.
4. Subramanian showed

$$\boxed{\mathsf{NL} \subseteq \mathsf{CC} \subseteq \mathsf{P}}$$

NL is nondeterministic log space

- Recall NL $\subseteq$ CC $\subseteq$ P
- But also NL $\subseteq$ NC $\subseteq$ P
  where NC (the parallel class) contains the problems solvable by uniform polysize polylog depth Boolean circuit families.
- NC contains all context-free languages, and matrix powering and determinants over $\mathbb{Z}, \mathbb{Q}$ etc.

- Recall $\mathsf{NL} \subseteq \mathsf{CC} \subseteq \mathsf{P}$
- But also $\mathsf{NL} \subseteq \mathsf{NC} \subseteq \mathsf{P}$
  where NC (the parallel class) contains the problems solvable by uniform polysize polylog depth Boolean circuit families.
- NC contains all context-free languages, and matrix powering and determinants over $\mathbb{Z}, \mathbb{Q}$ etc.

Conjecture

NC and CC are incomparable. (So in particular $\mathsf{CC} \subsetneq \mathsf{P}$.)

- Recall $\mathsf{NL} \subseteq \mathsf{CC} \subseteq \mathsf{P}$
- But also $\mathsf{NL} \subseteq \mathsf{NC} \subseteq \mathsf{P}$
  where NC (the parallel class) contains the problems solvable by uniform polysize polylog depth Boolean circuit families.
- NC contains all context-free languages, and matrix powering and determinants over $\mathbb{Z}, \mathbb{Q}$ etc.

Conjecture

NC and CC are incomparable. (So in particular $\mathsf{CC} \subsetneq \mathsf{P}$.)

Intuitively, we think $\mathsf{CC} \subsetneq \mathsf{P}$ because each of the two comparator gate outputs in a comparator circuit is limited to fan-out one. (More later. . .)

Example Complete Problems for CC

- CCV
- Stable Marriage Problem
- Lexicographical first maximal matching
- Telephone connection problem
- Others . . .

Example Complete Problems for CC

- Ccv
- Stable Marriage Problem
- Lexicographical first maximal matching
- Telephone connection problem
- Others . . .

If our conjecture (that NC and CC are incomparable) is correct then none of these complete problems has an efficient parallel algorithm.

Stable Marriage Problem (search version) (Gale-Shapley '62)

- Given n men and n women together with their preference lists
- Find a stable marriage between men and women, i.e.,
  1. a perfect matching
  2. satisfies the stability condition: no two people of the opposite sex like each other more than their current partners
  3. A stable marriage always exists, but may not be unique.

Stable Marriage Problem (search version) (Gale-Shapley '62)

- Given n men and n women together with their preference lists
- Find a stable marriage between men and women, i.e.,
  1. a perfect matching
  2. satisfies the stability condition: no two people of the opposite sex like each other more than their current partners
  3. A stable marriage always exists, but may not be unique.

Stable Marriage Problem (decision version)

Is a given pair of (m, w) in the man-optimal (woman-optimal) stable marriage?

Stable Marriage Problem (search version) (Gale-Shapley '62)

- Given n men and n women together with their preference lists
- Find a stable marriage between men and women, i.e.,
  1. a perfect matching
  2. satisfies the stability condition: no two people of the opposite sex like each other more than their current partners
  3. A stable marriage always exists, but may not be unique.

Stable Marriage Problem (decision version)

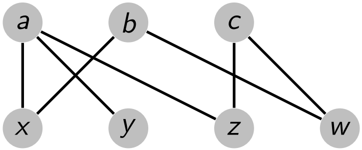
Is a given pair of (m, w) in the man-optimal (woman-optimal) stable marriage?

The Stable Marriage problem has been used to pair medical interns with hospital residencies in the USA.

Lex-first maximal matching problem (CC-Complete)

Lex-first maximal matching

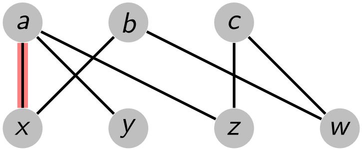
- Let G be a bipartite graph.
- Successively match the bottom nodes $x, y, z, \dots$ to the least available top node



Lex-first maximal matching problem (CC-Complete)

Lex-first maximal matching

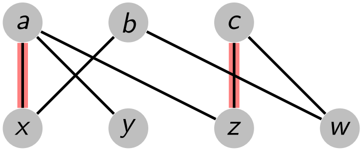
- Let G be a bipartite graph.
- Successively match the bottom nodes $x, y, z, \ldots$ to the least available top node



Lex-first maximal matching problem (CC-Complete)

Lex-first maximal matching

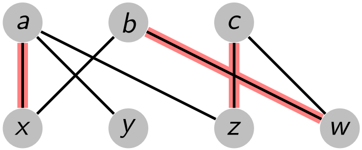
- Let G be a bipartite graph.
- Successively match the bottom nodes $x, y, z, \ldots$ to the least available top node



Lex-first maximal matching problem (CC-Complete)

Lex-first maximal matching

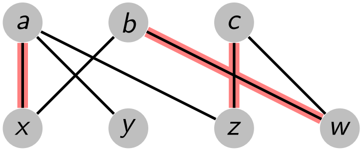
- Let G be a bipartite graph.
- Successively match the bottom nodes $x, y, z, \ldots$ to the least available top node



Lex-first maximal matching problem (CC-Complete)

Lex-first maximal matching

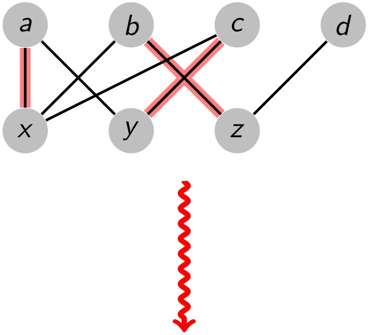
- Let G be a bipartite graph.
- Successively match the bottom nodes $x, y, z, \ldots$ to the least available top node

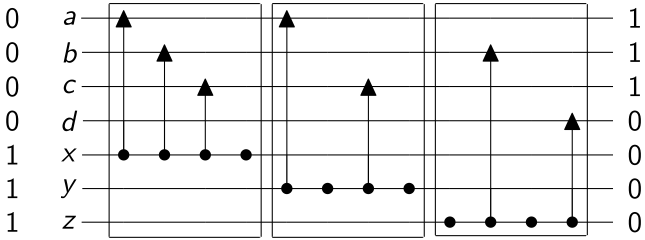


Lex-first maximal matching decision problems

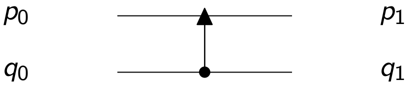
- Edge Is a given edge $\{u, v\}$ in the lex-first maximal matching of G?
- Vertex Is a given (top) vertex v in the lex-first maximal matching of G?
- The problems are equivalent.

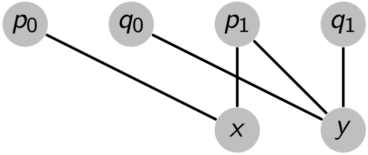
Reducing vertex lex-first maximal matching to $\textsc{Ccv}$



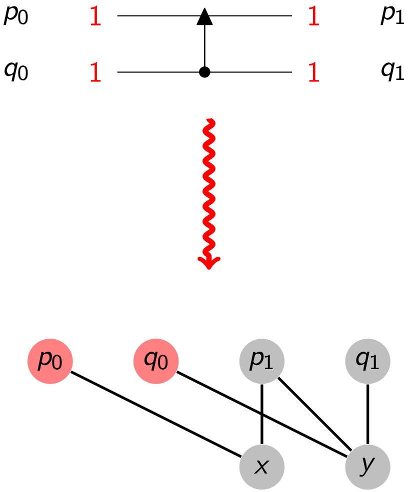


Reducing CCV to lex-first maximal matching

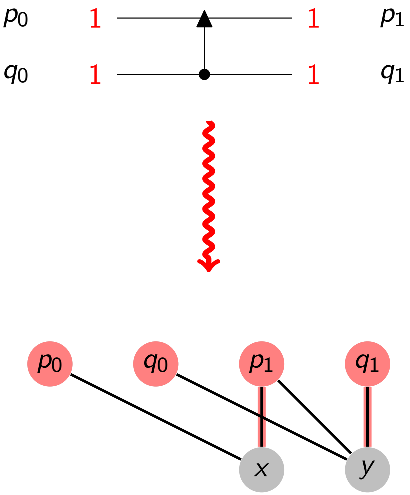




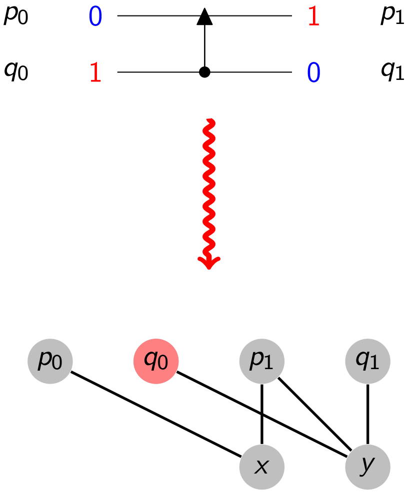
Reducing CCV to lex-first maximal matching



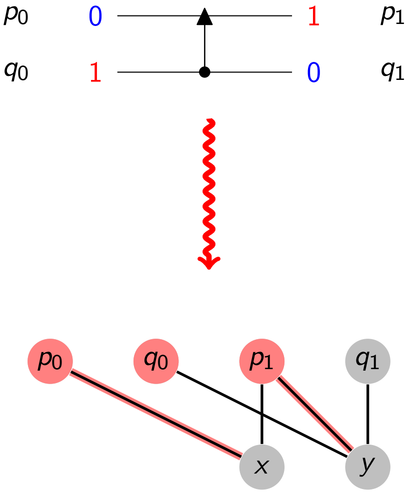
Reducing $\textsc{Ccv}$ to lex-first maximal matching



Reducing CCV to lex-first maximal matching



Reducing $\textsc{Ccv}$ to lex-first maximal matching



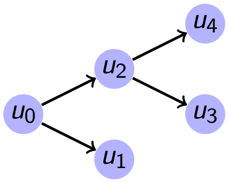
NL $\subseteq$ CC

- This result is due to Feder [1992].
- Dai Lê has a neat proof (See the appendix to our recent arXiv paper.)

NL $\subseteq$ CC

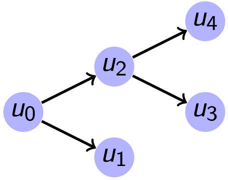
- This result is due to Feder [1992].
- Dai Lê has a neat proof (See the appendix to our recent arXiv paper.)
- Show $stCONN \leq_m^{\mathrm{AC}^0} \mathrm{C{\small CV}}$.
- May assume that the given directed graph $G = (V, E)$ has edges of the form (u_i, u_j), where $i < j$.

NL $\subseteq$ CC

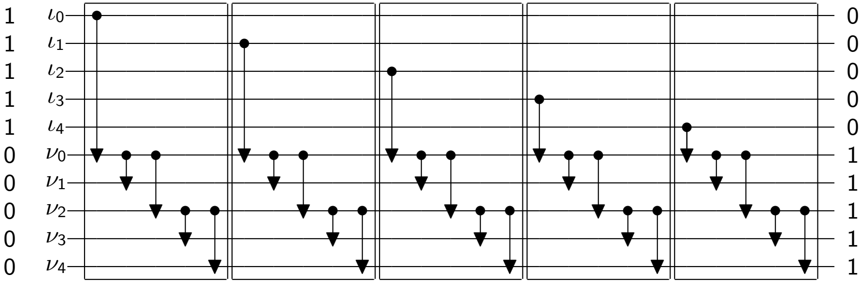


- This result is due to Feder [1992].
- Dai Lê has a neat proof (See the appendix to our recent arXiv paper.)
- Show $stCONN \leq_m^{\mathrm{AC}^0} \mathrm{CCV}$.
- May assume that the given directed graph $G = (V, E)$ has edges of the form (u_i, u_j), where $i < j$.

NL $\subseteq$ CC



- This result is due to Feder [1992].
- Dai Lê has a neat proof (See the appendix to our recent arXiv paper.)
- Show $stCONN \leq_m^{AC^0}$ CCV.
- May assume that the given directed graph $G = (V, E)$ has edges of the form (u_i, u_j), where $i < j$.



Notation

- $x, y, z, \ldots$ denote elements of $\mathbb{N}$ (presented in unary)
- $X, Y, Z, \ldots$ denote binary strings
- $|X|$ denotes the length of X.
- A complexity class is a set of relations of the form $R(\vec{x}, \vec{X})$

Notation

- $x, y, z, \ldots$ denote elements of $\mathbb{N}$ (presented in unary)
- $X, Y, Z, \ldots$ denote binary strings
- $|X|$ denotes the length of X.
- A complexity class is a set of relations of the form $R(\vec{x}, \vec{X})$
- AC^0 many-one reducibility
  $R_1(X) \leq_m^{\mathrm{AC}^0} R_2(X)$ iff there exists an AC^0 function $F(X)$ such that

$$R_1(X) \leftrightarrow R_2(F(X))$$

Notation

- $x, y, z, \ldots$ denote elements of $\mathbb{N}$ (presented in unary)
- $X, Y, Z, \ldots$ denote binary strings
- $|X|$ denotes the length of X.
- A complexity class is a set of relations of the form $R(\vec{x}, \vec{X})$
- AC^0 many-one reducibility
  $R_1(X) \leq_m^{AC^0} R_2(X)$ iff there exists an AC^0 function $F(X)$ such that
  $$R_1(X) \leftrightarrow R_2(F(X))$$
- Thus CC is the class of relations $R(\vec{x}, \vec{X})$ that are AC^0 many-one reducible to $\textsc{Ccv}$.

Function Classes

- Given a class C of relations, we associate a class FC of functions as follows.

- A function F taking strings to strings is in FC iff
  1. $|F(X)| = |X|^{O(1)}$ (p-bounded)
  2. The bit graph $B_F(i, X)$ is in C

- Here $B_F(i, X)$ holds iff the ith bit of $F(X)$ is 1.

Is FCC closed under composition?

- This question was left open in our earlier paper in CSL 2011 paper (before Yuval Filmus joined our project)

Is FCC closed under composition?

- This question was left open in our earlier paper in CSL 2011 paper (before Yuval Filmus joined our project)
- Suppose $F(X) = G(H(X))$. Let $Y = H(X)$.
- The bit graph of $G(Y)$ is AC^0-reducible to $\textsc{Ccv}$.
- Thus the circuit computing $G(Y)$ is described by $Y' = \mathsf{AC}^0(Y)$.

Is FCC closed under composition?

- This question was left open in our earlier paper in CSL 2011 paper (before Yuval Filmus joined our project)
- Suppose $F(X) = G(H(X))$. Let $Y = H(X)$.
- The bit graph of $G(Y)$ is AC^0-reducible to $\mathrm{C{\scriptsize CV}}$.
- Thus the circuit computing $G(Y)$ is described by $Y' = \mathsf{AC}^0(Y)$.
- But $Y = H(X)$ is the output of another comparator circuit.

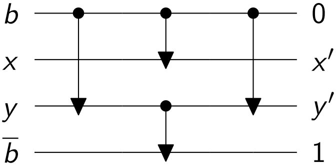
Is FCC closed under composition?

- This question was left open in our earlier paper in CSL 2011 paper (before Yuval Filmus joined our project)
- Suppose $F(X) = G(H(X))$. Let $Y = H(X)$.
- The bit graph of $G(Y)$ is AC^0-reducible to $\textsc{Ccv}$.
- Thus the circuit computing $G(Y)$ is described by $Y' = \mathsf{AC}^0(Y)$.
- But $Y = H(X)$ is the output of another comparator circuit.

So we need a universal comparator circuit, taking Y' as input, to compute $G(Y)$.

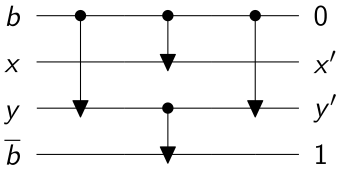
Universal comparator circuits [Filmus]

Here is a gadget which allows a conditional application of a comparator to two of its inputs x, y, depending on whether b is 0 or 1.

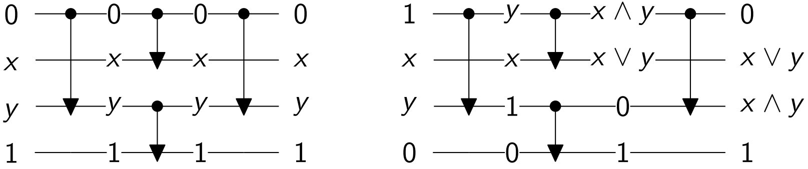


Universal comparator circuits [Filmus]

Here is a gadget which allows a conditional application of a comparator to two of its inputs x, y, depending on whether b is 0 or 1.



Operation of the gadget:



Universal comparator circuits

- In order to simulate a single arbitrary comparator in a circuit with m wires we put in $m(m-1)$ gadgets in a row, for the $m(m-1)$ possible comparators.

Universal comparator circuits

- In order to simulate a single arbitrary comparator in a circuit with m wires we put in $m(m-1)$ gadgets in a row, for the $m(m-1)$ possible comparators.
- Simulating n comparators requires $m(m-1)n$ gadgets.

Universal comparator circuits

- In order to simulate a single arbitrary comparator in a circuit with m wires we put in $m(m-1)$ gadgets in a row, for the $m(m-1)$ possible comparators.
- Simulating n comparators requires $m(m-1)n$ gadgets.
- Thus there is an AC^0 function UNIV such that if m, n are arbitrary parameters, then

$$U = \mathrm{UNIV}(m, n) = \langle m', n', U' \rangle$$

is a universal circuit with m' wires and n' gates which simulates all comparator networks with at most m wires and at most n comparators.

$$\begin{aligned} m' &= 2m(m-1)n + m \\ n' &= 4m(m-1)n \end{aligned}$$

Applications of universal comparator circuits

- FCC is closed under composition.

Applications of universal comparator circuits

- FCC is closed under composition.
- CC is closed under (many-one) log-space reducibility.

Applications of universal comparator circuits

- FCC is closed under composition.
- CC is closed under (many-one) log-space reducibility.
- This is becasue NL $\subseteq$ CC, so FCC includes all log space functions. And FCC is closed under composition.
- If $R(X) \leftrightarrow \mathrm{Ccv}(F(X))$, where F is log-space computable, then

$$\chi_R(X) = \chi_{\mathrm{Ccv}}(F(X))$$

where χ_R is the characteristic function of R.

Applications of universal comparator circuits Cont'd

- $R(X)$ is in CC iff there is an AC^0-uniform family $\{C_k^R\}_{k\in\mathbb{N}}$ of comparator circuits, where C_k computes $R(X)$ for $|X| = k$.

Applications of universal comparator circuits Cont'd

- $R(X)$ is in CC iff there is an AC^0-uniform family $\{C_k^R\}_{k\in\mathbb{N}}$ of comparator circuits, where C_k computes $R(X)$ for $|X| = k$.
- The direction $\Leftarrow$ is immediate.

Applications of universal comparator circuits Cont'd

- $R(X)$ is in CC iff there is an AC^0-uniform family $\{C_k^R\}_{k\in\mathbb{N}}$ of comparator circuits, where C_k computes $R(X)$ for $|X| = k$.
- The direction $\Leftarrow$ is immediate.
- Proof of direction $\Rightarrow$: This is clear if $R(X)$ is in AC^0. (An AC^0 circuit converts into a polysize tree circuit, which converts to a comparator circuit.)

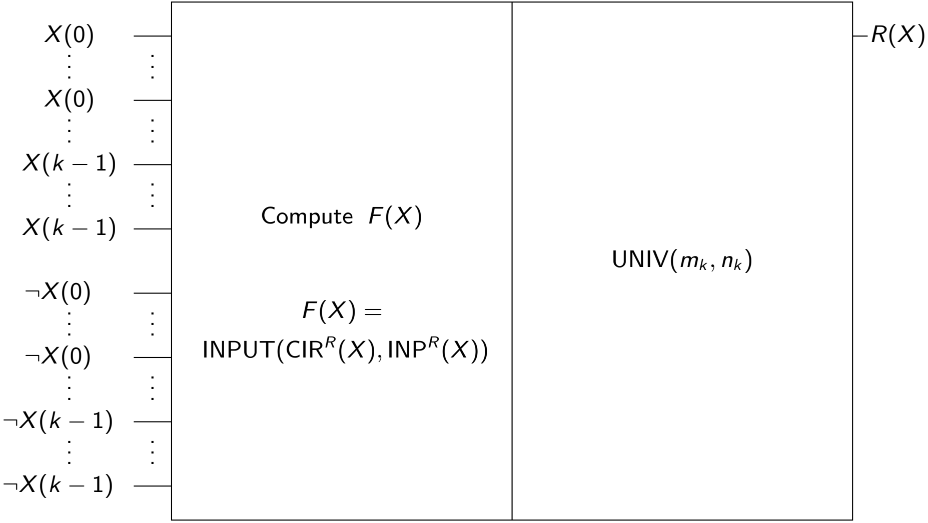
Applications of universal comparator circuits Cont'd

- $R(X)$ is in CC iff there is an AC^0-uniform family $\{C_k^R\}_{k\in\mathbb{N}}$ of comparator circuits, where C_k computes $R(X)$ for $|X| = k$.
- The direction $\Leftarrow$ is immediate.
- Proof of direction $\Rightarrow$: This is clear if $R(X)$ is in AC^0. (An AC^0 circuit converts into a polysize tree circuit, which converts to a comparator circuit.)
- If $R(X) \in \mathrm{CC}$, then

$$R(X) \leftrightarrow \textsc{Ccv}(F(X))$$

for some AC^0 function $F(X)$. Apply a universal circuit to the output of $F(X)$.

The circuit C_k computing $R(X)$ for $|X| = k$



Conjecture: NC and CC are incomparable

- Lex-First Max Matching (LFMM) is in CC.

Conjecture

LFMM is not in NC.
(The obvious algorithm for LFMM is sequential.)

Conjecture: NC and CC are incomparable

- Lex-First Max Matching (LFMM) is in CC.

Conjecture

LFMM is not in NC.
(The obvious algorithm for LFMM is sequential.)

- The function $A \rightsquigarrow A^n$ (where A is an $n \times n$ integer matrix) is in NC^2 , but we do not know how to put it in CC.

Why do we think $\mathsf{NC}^2 \subsetneq \mathsf{CC}$?

- NC^2-gates have multiple fan-out, but each end of a comparator gate has fan-out one.

Why do we think $\mathsf{NC}^2 \subsetneq \mathsf{CC}$?

- NC^2-gates have multiple fan-out, but each end of a comparator gate has fan-out one.
- If either input of a comparator gate is 'flipped', then exactly one output is flipped.
  Thus comparator gates are 1-Lipschitz.

Why do we think $\mathsf{NC}^2 \subsetneq \mathsf{CC}$?

- NC^2-gates have multiple fan-out, but each end of a comparator gate has fan-out one.
- If either input of a comparator gate is 'flipped', then exactly one output is flipped.
  Thus comparator gates are 1-Lipschitz.
- Flipping an input to a gate generates a unique flip-path in the circuit from that gate to some output of the circuit.

Why do we think $\mathsf{NC}^2 \subsetneq \mathsf{CC}$?

- NC^2-gates have multiple fan-out, but each end of a comparator gate has fan-out one.
- If either input of a comparator gate is 'flipped', then exactly one output is flipped.
  Thus comparator gates are 1-Lipschitz.
- Flipping an input to a gate generates a unique flip-path in the circuit from that gate to some output of the circuit.
- But flipping an input to an NC^2-gate can generate many parallel flip-paths.

Relativized CC and NC are incomparable

Oracle gates for comparator circuits

- The oracle $\alpha : \{0,1\}^* \to \{0,1\}^*$ is length preserving.
- $\alpha_n : \{0,1\}^n \to \{0,1\}^n$ is the restriction of α to n.
- An oracle gate α_n can be inserted anywhere in a relativized comparator circuit: select any n wires as inputs to the gate and any n wires as outputs.

Relativized CC and NC are incomparable

Oracle gates for comparator circuits

- The oracle $\alpha : \{0,1\}^* \to \{0,1\}^*$ is length preserving.
- $\alpha_n : \{0,1\}^n \to \{0,1\}^n$ is the restriction of α to n.
- An oracle gate α_n can be inserted anywhere in a relativized comparator circuit: select any n wires as inputs to the gate and any n wires as outputs.
- To make α_n gates look more like comparator gates, we require that α_n have the 1-Lipschitz property.

Relativized CC and NC are incomparable

Oracle gates for comparator circuits

- The oracle $\alpha : \{0,1\}^* \to \{0,1\}^*$ is length preserving.
- $\alpha_n : \{0,1\}^n \to \{0,1\}^n$ is the restriction of α to n.
- An oracle gate α_n can be inserted anywhere in a relativized comparator circuit: select any n wires as inputs to the gate and any n wires as outputs.
- To make α_n gates look more like comparator gates, we require that α_n have the 1-Lipschitz property.
- We allow $\neg$ gates in relativized CC(α) circuits.
  (We can allow them in comparator circuits without changing CC.)

Relativized CC and NC are incomparable

Oracle gates for comparator circuits

- The oracle $\alpha : \{0,1\}^* \to \{0,1\}^*$ is length preserving.
- $\alpha_n : \{0,1\}^n \to \{0,1\}^n$ is the restriction of α to n.
- An oracle gate α_n can be inserted anywhere in a relativized comparator circuit: select any n wires as inputs to the gate and any n wires as outputs.
- To make α_n gates look more like comparator gates, we require that α_n have the 1-Lipschitz property.
- We allow $\neg$ gates in relativized CC(α) circuits. (We can allow them in comparator circuits without changing CC.)
- Changing one input to one α_n gate produces a unique flip path in the circuit from that gate to the outputs of the circiut.

Theorem

There is a relation $R_1(\alpha)$ computable by a polysize family of comparator oracle circuits by which cannot be computed by any $\mathrm{NC}(\alpha)$ *circuit family (even when α is restricted to be 1-Lipschitz).*

Proof Idea.

- $\alpha_n^k(\vec{0})$ is easily computed by relativized comparator circuits, but requires depth k circuits [ACN 07].

Theorem

There is a relation $R_1(\alpha)$ computable by a polysize family of comparator oracle circuits by which cannot be computed by any $\mathsf{NC}(\alpha)$ circuit family (even when α is restricted to be 1-Lipschitz).

Proof Idea.

- $\alpha_n^k(\vec{0})$ is easily computed by relativized comparator circuits, but requires depth k circuits [ACN 07].
- The hard part is proving the depth lower bound when α is 1-Lipschitz.

$\square$

Theorem

There is a relation $R_2(\alpha)$ computable by an $\mathsf{NC}^3(\alpha)$ circuit family but not computable by any polysize family of comparator oracle circuits (even when α is restricted to be 1-Lipschitz).

Theorem

There is a relation $R_2(\alpha)$ computable by an $\mathsf{NC}^3(\alpha)$ circuit family but not computable by any polysize family of comparator oracle circuits (even when α is restricted to be 1-Lipschitz).

Proof Idea where α is weakly 1-Lipschitz

(At most one output bit flips when one input bit flips.)

- Let $a_i^k : \{0,1\}^{dn} \to \{0,1\}$ be a Boolean oracle.
- Let $A^k = (a_1^k, \ldots, a_n^k)$
- Define a function $y = f[A^1, \ldots, A^m]$ as follows:

$$\begin{aligned} x_i^k &= a_i^k(\overbrace{x_1^{k+1}, \ldots, x_1^{k+1}}^{d \text{ times}}, \ldots, \overbrace{x_n^{k+1}, \ldots, x_n^{k+1}}^{d \text{ times}}), && k \in [m],\ i \in [n], \\ x_i^{m+1} &= 0, && i \in [n], \\ y &= x_1^1 \oplus \cdots \oplus x_n^1. \end{aligned}$$

$$
\begin{aligned}
x_i^k &= a_i^k(\overbrace{x_1^{k+1}, \ldots, x_1^{k+1}}^{d \text{ times}}, \ldots, \overbrace{x_n^{k+1}, \ldots, x_n^{k+1}}^{d \text{ times}}), && k \in [m],\ i \in [n], \\
x_i^{m+1} &= 0, && i \in [n], \\
y &= x_1^1 \oplus \cdots \oplus x_n^1.
\end{aligned}
$$

- a_i^k has dn inputs and one output.
- Add $dn - 1$ zeros as extra outputs for each a_i^k.
- Each a_i^k computes a weakly 1-Lipschitz function.
- Let $X^k = (x_1^k, \ldots, x_n^k) \qquad A^k = (a_1^k, \ldots, a_n^k)$
- $y = f[A^1, \ldots, A^m]$

$$
\begin{aligned}
x_i^k &= a_i^k(\overbrace{x_1^{k+1},\dots,x_1^{k+1}}^{d \text{ times}},\dots,\overbrace{x_n^{k+1},\dots,x_n^{k+1}}^{d \text{ times}}), && k \in [m],\ i \in [n],\\
x_i^{m+1} &= 0, && i \in [n],\\
y &= x_1^1 \oplus \cdots \oplus x_n^1.
\end{aligned}
$$

- a_i^k has dn inputs and one output.
- Add $dn - 1$ zeros as extra outputs for each a_i^k.
- Each a_i^k computes a weakly 1-Lipschitz function.
- Let $X^k = (x_1^k, \dots, x_n^k) \qquad A^k = (a_1^k, \dots, a_n^k)$
- $y = f[A^1, \dots, A^m]$
- Set $m = \log^2 n$ and $d = 4$
- Then a depth $\log^2 n$ NC^3 oracle circuit computes f

$$
\begin{aligned}
x_i^k &= a_i^k(\overbrace{x_1^{k+1},\dots,x_1^{k+1}}^{d \text{ times}},\dots,\overbrace{x_n^{k+1},\dots,x_n^{k+1}}^{d \text{ times}}), && k \in [m],\ i \in [n],\\
x_i^{m+1} &= 0, && i \in [n],\\
y &= x_1^1 \oplus \cdots \oplus x_n^1.
\end{aligned}
$$

- a_i^k has dn inputs and one output.
- Add $dn - 1$ zeros as extra outputs for each a_i^k.
- Each a_i^k computes a weakly 1-Lipschitz function.
- Let $X^k = (x_1^k, \dots, x_n^k) \qquad A^k = (a_1^k, \dots, a_n^k)$
- $y = f[A^1, \dots, A^m]$
- Set $m = \log^2 n$ and $d = 4$
- Then a depth $\log^2 n$ NC^3 oracle circuit computes f
- **Claim:** Every oracle comparator circuit computing $f[A^1, \dots, A^m]$ has at least $\min(2^n, (d-2)^{m-1})$ gates. (Superpolynomial size)

Proof outline of Claim:

Every oracle comparator circuit computing $f[A^1, \ldots, A^m]$ has at least $\min(2^n, (d-2)^{m-1})$ gates.

Fix an oracle comparator circuit C computing $y = f[A^1, \ldots, A^m]$

- Def'n: An input to an oracle a_k^i is *regular* if it has the form $(b_1)^d \cdots (b_n)^d$.
  We say oracle a_i^k is *regular* if $a_i^k(Z) = 0$ for all irregular inputs Z.

Proof outline of Claim:

Every oracle comparator circuit computing $f[A^1, \ldots, A^m]$ has at least $\min(2^n, (d-2)^{m-1})$ gates.

Fix an oracle comparator circuit C computing $y = f[A^1, \ldots, A^m]$

- Def'n: An input to an oracle a_k^i is *regular* if it has the form $(b_1)^d \cdots (b_n)^d$.
  We say oracle a_i^k is *regular* if $a_i^k(Z) = 0$ for all irregular inputs Z.
- Let g be the total number of any of the gates a_i^k in C.
  Given an assignment to the oracles, we say a particular gate a_i^k is *active* if its input is correct.

Proof outline of Claim:

Every oracle comparator circuit computing $f[A^1, \ldots, A^m]$ has at least $\min(2^n, (d-2)^{m-1})$ gates.

Fix an oracle comparator circuit C computing $y = f[A^1, \ldots, A^m]$

- Def'n: An input to an oracle a^i_k is *regular* if it has the form $(b_1)^d \cdots (b_n)^d$.
  We say oracle a^k_i is *regular* if $a^k_i(Z) = 0$ for all irregular inputs Z.
- Let g be the total number of any of the gates a^k_i in C.
  Given an assignment to the oracles, we say a particular gate a^k_i is *active* if its input is correct.
- Let g_k be the expected total number of active gates $a^k_1, \ldots, a^k_n$ in C under a uniformly random *regular* setting of *a*ll oracles.
- $g_1 \geq n$ (because $y = x^1_1 \oplus \cdots \oplus x^1_n$)

Proof outline of Claim:

Every oracle comparator circuit computing $f[A^1, \ldots, A^m]$ has at least $\min(2^n, (d-2)^{m-1})$ gates.

Fix an oracle comparator circuit C computing $y = f[A^1, \ldots, A^m]$

- Def'n: An input to an oracle a^i_k is *regular* if it has the form $(b_1)^d \cdots (b_n)^d$.
  We say oracle a^k_i is *regular* if $a^k_i(Z) = 0$ for all irregular inputs Z.
- Let g be the total number of any of the gates a^k_i in C.
  Given an assignment to the oracles, we say a particular gate a^k_i is *active* if its input is correct.
- Let g_k be the expected total number of active gates $a^k_1, \ldots, a^k_n$ in C under a uniformly random *regular* setting of *a*ll oracles.
- $g_1 \geq n$ (because $y = x^1_1 \oplus \cdots \oplus x^1_n$)
- It suffices to show $g_{k+1} \geq (d-2)(g_k - g/2^n)$

Proof idea of final Claim:

$g_{k+1} \geq (d-2)(g_k - g/2^n)$

- Consequence of weakly 1-Lipschitz: If we change the definition of some gate a_i^k at its current input in C, this generates a unique flip-path which may end at some copy of some other gate, in which case we say that the latter gate *consumes* the flip-path.

Proof idea of final Claim:

$g_{k+1} \geq (d-2)(g_k - g/2^n)$

- Consequence of weakly 1-Lipschitz: If we change the definition of some gate a_i^k at its current input in C, this generates a unique flip-path which may end at some copy of some other gate, in which case we say that the latter gate *consumes* the flip-path.
- Let $G_1, \ldots, G_{2^n}$ be a Gray code listing all strings in $\{0,1\}^n$, starting at $G_1 = X_{k+1}$. We change the definition of the output of A^{k+1} (at its active input) successively from G_1 to G_{2^n} and count the number of flip paths generated.
- The Claim follows because every time a particular a_i^k gate is updated from one active input to the next, it will absorb as least $d-2$ flip paths.

Conclusion

The complexity class CC is interesting because

- It is robust: It has several alternative characterizations.
- It has interesting complete problems.
- It appears to be a proper subset of P and incomparable with NC (and SC).

Conclusion

The complexity class CC is interesting because

- It is robust: It has several alternative characterizations.
- It has interesting complete problems.
- It appears to be a proper subset of P and incomparable with NC (and SC).

Open Problems:

Are any of the following problems in in CC?

:

Integer matrix powering?

All context free languages?

maximum matching in graphs?