

Comparator Circuits over Finite Bounded Posets

Balagopal Komarath* Jayalal Sarma K.S. Sunil

Department of Computer Science & Engineering,
Indian Institute of Technology Madras, Chennai, India.

Email: {baluks|jayalal|sunil}@cse.iitm.ac.in

November 20, 2018

Abstract

The comparator circuit model was originally introduced in [5] (and further studied in [2]) to capture problems that are not known to be P-complete but still not known to admit efficient parallel algorithms. The class CC is the complexity class of problems many-one logspace reducible to the Comparator Circuit Value Problem and we know that $\text{NLOG} \subseteq \text{CC} \subseteq \text{P}$. Cook *et al.* [2] showed that CC is also the class of languages decided by polynomial size comparator circuit families.

We study generalizations of the comparator circuit model that work over fixed finite bounded posets. We observe that there are universal comparator circuits even over arbitrary fixed finite bounded posets. Building on this, we show the following :

- Comparator circuits of polynomial size over fixed finite *distributive* lattices characterize the class CC. When the circuit is restricted to be skew, they characterize LOG. Noting that (uniform) polynomial sized Boolean circuits (resp. skew) characterize P (resp. NLOG), this indicates a comparison between P vs CC and NLOG vs LOG problems.
- Complementing this, we show that comparator circuits of polynomial size over arbitrary fixed finite lattices characterize the class P even when the comparator circuit is skew.
- In addition, we show a characterization of the class NP by a family of polynomial sized comparator circuits over fixed *finite bounded posets*. As an aside, we consider generalizations of Boolean formulae over arbitrary lattices. We show that Spira's theorem [6] can be extended to this setting as well and show that polynomial sized Boolean formulae over finite fixed lattices capture the class NC^1 .

These results generalize results in [2] regarding the power of comparator circuits. Our techniques involve design of comparator circuits and finite posets. We then use known results from lattice theory to show that the posets that we obtain can be embedded into appropriate lattices. Our results give new methods to establish CC upper bounds for problems and also indicate potential new approaches towards the problems P vs CC and NLOG vs LOG using lattice theoretic methods.

*Supported by TCS PhD Fellowship

1 Introduction

Completeness for the class P for a problem is usually considered to be evidence that it is hard to design an efficient parallel algorithm for the problem. However, there are many computational problems in the class P, which are not known to be P-complete, yet designing efficient parallel algorithms for them has remained elusive. Some of the classical examples of such problems include lex-least maximal matching problem and stable marriage problem [5].

Attempting to capture the exact complexity of computation in these problems using a variant of Boolean circuit model, Mayr and Subramanian [5] (see also [2]) studied the comparator circuit model. A comparator circuit is a sorting network working over the values 0 and 1. A comparator gate has two inputs and two outputs. The first output is the AND of the two inputs and the second output is the OR of the two inputs. A comparator circuit

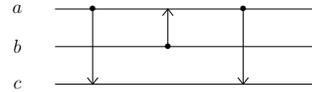


Figure 1: A Comparator Circuit

is a circuit that has only comparator gates. In particular, fan-out gates are not allowed. Without loss of generality, we can assume that NOT gates are used only at the input level. A graphical representation of a comparator circuit is shown in Figure 1. In this representation, we draw a set of parallel *lines*. Each line carries a logical value which is updated by gates incident on that line. Each gate is represented by a directed arrow from one line (say i) to another (say j) and the gate updates the values of lines as follows. The value of line i (j) is set to the AND (resp. OR) of values previously on lines i and j . The gates are evaluated from left to right. The output of the circuit is the final value of a line designated as the output line. We define the model formally in Section 2.

In order to study the complexity theoretic significance of comparator circuits, the corresponding circuit value problem was explored in [5]. That is, given a comparator circuit and an input, test if the output wire carries a 1 or not. The class CC is defined in [5] as the class of languages that are logspace many-one reducible to the comparator circuit value problem. They also observed that the class CC is contained in P. Feder's algorithm (described in [8]) for directed reachability proves that the class CC contains NLOG as a subclass. These are the best containments currently known about the complexity class CC.

There has been a recent spurt of activity in the characterization of CC. Cook et al. [2] showed that the class CC is robust even if the complexity of the many-one reduction to the comparator circuit value problem is varied from AC^0 to NLOG. They also gave a characterization of the class CC in terms of a computational model (comparator circuit families). Their main contribution in this regard is the introduction of a universal comparator circuit that can simulate the computation of a comparator circuit given as input (to the universal circuit). Comparison of CC with the class NC has interesting implications to the corresponding computational restrictions. For example, hardness for the class CC is conjectured to be evidence that the problem is not efficiently parallelizable. This intuition was further strengthened by Cook et al. [2] by showing that there are oracle sets relative to which CC and NC are incomparable (NC is the class of all languages efficiently solvable by parallel algorithms). In addition, it is conjectured in [2] that the classes NC, SC and CC are pairwise incomparable.

Our Results & Techniques: In this paper, we study the computational power of comparator circuits working over arbitrary fixed finite bounded posets. Informally, instead of 0 and 1, the

values used throughout the computation could be any element from the poset and the AND and OR gates compute maximal lower bounds and minimal upper bounds over the poset respectively. We define this model formally in section 3. We obtain the following results:

- There exist Universal Comparator Circuits for comparator circuits irrespective of the underlying bounded poset. (Proposition 3, Section 3.)
- Comparator circuits of polynomial size over fixed finite distributive lattices capture the class CC. (Theorem 4, Section 4). This leads to a new way to show that a problem is in the class CC. That is, by designing a comparator circuit over a fixed finite lattice and then showing that the lattice is distributive. (An application of this method to design CC algorithms for the stable matching problem can be found in [5]. See also Section 6.2 in [2]). Since there are lattice theoretic techniques known (cf. M_3-N_5 Theorem [3]) for showing that a lattice is distributive, this alternate definition of the class CC using comparator circuits over distributive lattices might be of independent interest.

- Going beyond distributivity, we show that comparator circuits of polynomial size over fixed finite lattices characterize the class P. (Theorem 5, Section 4). In particular, we design a fixed finite poset P over which, for any language $L \in P$, there is a polynomial size comparator circuit family over P computing L . During computation, we only use lubs and glbs that exist in the poset P . This enables the use of Dedekind-MacNeille completion (DM completion) to construct a fixed finite lattice completing the poset P while preserving existing lubs and glbs in the poset and that lattice can be used to perform all computations in P. A potential drawback of the lattice thus obtained is that the complexity class captured by comparator circuits over it may vary depending on the element in the lattice used as the accepting element. By using standard tools from lattice theory, we derive that there is a fixed constant $i \geq 3$, such that comparator circuits over Π_i (where Π_i is the i^{th} partition lattice - see Section 2 for a definition) with polynomial size can compute all functions in P. Moreover, we show that comparator circuits over the lattice Π_i capture P irrespective of the accepting element used.

However, both partition lattices for $i \geq 3$ and the lattice given by DM completion are non-distributive. Exploring the possibility of another completion of the poset P into a distributive lattice that preserves existing lubs and glbs (which will show $P = CC$), we arrive at the following negative result : the poset P cannot be embedded into any distributive lattice while preserving all existing lubs and glbs. (Theorem 6).

It is conceivable that the class P could be captured by a family of distributive lattices, while no finite fixed lattice capturing P can be distributive. Motivated by this, we also present an analogue of the main theorem using growing posets of much simpler structure (See appendix A). However, we argue that this poset family also cannot be embedded into a family of distributive lattices while preserving all existing lubs and glbs.

- Going beyond lattice structure, we show that comparator circuits over fixed finite bounded posets capture the class NP. (Theorem 7, Section 5). Here, we crucially use the fact that posets that are not lattices could have elements that do not have unique minimal upper bounds to simulate non-determinism. Hence, any completion of this poset into a lattice will fail to capture NP, unless $P = NP$. Note, that the DM completion of this poset would not be able to characterize NP as in the case of P because the DM completion would introduce elements so

that the elements in the poset that have non-unique minimal upper bounds and/or maximal lower bounds would end up having unique lubs and glbs.

- Restricting the structure of the comparator circuit, we obtain an exact characterization of the class LOG using skew comparator circuits (Theorem 8). Noting that the polynomial sized skew Boolean circuits characterize exactly the class NLOG, this leads to a comparison between CC vs P and NLOG vs LOG problem : *both problems address the power of polynomial size Boolean circuits vs comparator circuits in general and skew circuits respectively.*
- We further study generalizations of skew comparator circuits to arbitrary lattices. When the lattice is distributive, it follows that the circuits capture exactly LOG. Complementing this, we show that there are fixed finite lattices P over which the skew comparator circuits characterize exactly P.(Theorem 9).
- We study generalizations of Boolean formulas to arbitrary lattices where the AND and OR gates compute the \wedge and \vee of the lattices. We generalize Spira’s theorem [6] to this setting and show that polynomial sized Boolean formulae over finite fixed lattices capture exactly NC^1 (Theorem 10).

Thus, we observe that as the comparator circuit is allowed to compute over progressively general structures (from distributive lattices to arbitrary lattices to posets), the model captures classes of problems that are progressively harder to parallelize (From CC to P to NP). The table below indicates the results (known results are indicated by citations).

Lattices \implies	Boolean	Distributive	General	Posets
poly-sized	CC(see [2])	CC	P	NP
Skew, poly-sized	LOG	LOG	P	-
Formulae	NC^1 (see [6])	NC^1	NC^1	

The main technical contribution in our proofs is the design of posets and the corresponding comparator circuits for capturing complexity classes. We then use known ideas from lattice and order theory in order to derive lattices to which the constructed posets can be embedded.

2 Preliminaries

The standard definitions in complexity theory used in this paper can be found in standard textbooks [1]. All reductions in this paper are computable in logspace unless mentioned otherwise. By (standard) Boolean circuits, we mean circuits over the basis $\{\vee, \wedge\}$ where NOT gates are only allowed at the input level. In this section, we define comparator circuits, certain restrictions on comparator circuits and complexity classes based on those restrictions.

A *comparator circuit* has a set of n lines $\{w_1, \dots, w_n\}$ and an ordered list of gates (w_i, w_j) . Each line can be fed as input a value that is either (Boolean) 0 or 1. We define $val(w_i)$ to be the value of the line w_i . Each gate (w_i, w_j) updates $val(w_i)$ to $val(w_i) \wedge val(w_j)$ and $val(w_j)$ to $val(w_i) \vee val(w_j)$ in order. After all gates have updated the values, the value of the line w_1 is the output of the circuit.

The COMPARATOR CIRCUIT VALUE PROBLEM is: Given (C, x) as input find the output of the comparator circuit C when fed x as input. We can think of C being encoded according to the

above definition of comparator circuits. We call this the ordered list representation as the gates are presented as an ordered list. Mayr and Subramanian [5] defined the complexity class CC as the set of all languages logspace reducible to the Comparator Circuit Value problem. Cook et al. [2] characterized the class CC as languages computed by AC^0 -uniform families of annotated comparator circuits. In an annotated comparator circuit the initial value of a line could be an input variable x_i or its complement \bar{x}_i . In a family of annotated comparator circuits for a language L , the n^{th} comparator circuit in the family has exactly n input variables (x_1, \dots, x_n) and the circuit computes $L \cap \{0, 1\}^n$.

Skew Comparator Circuits: We now define skewness in comparator circuits. To begin with, we present an alternate definition of comparator circuits that is closer to the definition of standard Boolean circuits. A comparator gate is a 2-input, 2-output gate that takes a and b as inputs and outputs $a \wedge b$ and $a \vee b$. Then the comparator circuit is simply a circuit (in the usual sense) that consists of only comparator gates (In particular, fan-out gates are not allowed). Using this definition, we can encode comparator circuits by using DAGs as we encode standard Boolean circuits. It is easy to see that given a comparator circuit encoded as an ordered list of gates, we can obtain the DAG encoding the comparator circuit in logspace. Using this definition, we can talk about *wires* in the comparator circuit.

We say that an AND gate in a comparator gate is *used* if the AND output wire of that comparator gate has a path, through comparator gates, to the output wire. An AND gate in the circuit is called *skew* if and only if at least one input to that gate is the constant 0 or the constant 1 or (in the case of annotated circuits) an input bit x_i or \bar{x}_i for some i .

A comparator circuit is called a *skew comparator circuit* if and only if all used AND gates in the circuit are *skew*. The complexity class SkewCC consists of all languages that can be decided by poly-size skew comparator circuit families. We define SkewCCVP to be the circuit evaluation problem for skew comparator circuits. Note that given the ordered list representation of a comparator circuit, it is easy to check whether an AND gate is used or not. For ex., if the i^{th} gate is (w_1, w_2) , then the AND output of this gate is unused if and only if there is no element in the list of gates with w_1 as a member at a position greater than i in the list and if the AND output of this gate is not the output wire.

The circuit family is LOG-uniform if and only if there exists a TM M that outputs the n^{th} circuit in the family in $O(\log(n))$ space given 1^n as input. All circuits in this paper are LOG-uniform unless mentioned otherwise.

Lattice and Order Theory: We include some basic definitions and terminology from standard lattice and order theory that are required later in the paper. A more detailed treatment can be found in standard textbooks [3].

A set P along with a reflexive, anti-symmetric and transitive relation denoted by \leq_P is called a *poset*. An element $m \in P$ is called the *greatest element* if $x \leq m$ for all x in P . An element $m \in P$ is called the *least element* if $m \leq x$ for all x in P . A poset is called *bounded* if it has a greatest and a least element. Note that any finite poset can be converted into a finite bounded poset by adding two new elements 0 and 1 and adding the relations $m \leq 1$ and $0 \leq m$ for every element m in the poset. *Minimal upper bounds* of two elements x, y in P , denoted by $x \vee y$, is the set of all $m \in P$ such that $x \leq m, y \leq m$ and there exists no m' distinct from m such that $x \leq m', y \leq m'$ and $m' \leq m$. *Maximal lower bounds* of two elements x, y in P , denoted by $x \wedge y$, is the set of all

$m \in P$ such that $m \leq x$, $m \leq y$ and there exists no m' distinct from m such that $m' \leq x$, $m' \leq y$ and $m \leq m'$. A poset P is called a *lattice* if every pair of elements x and y has a unique maximal lower bound and a unique minimal upper bound. In a lattice, the minimal upper bound (maximal lower bound) of two elements is also known as the *join* (*meet*). Since minimal upper bound and maximal lower bound are unique in a lattice, we drop the set notation when describing them, i.e., instead of writing $a \vee b = \{x\}$, we simply write $a \vee b = x$. A lattice L is called *distributive* if for every elements $a, b, c \in L$ we have $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. An *order embedding* of a poset P into a poset P' is a function $f : P \mapsto P'$ such that $f(x) \leq_{P'} f(y) \iff x \leq_P y$. We say that the lattice L is a sub-lattice of L' if $L \subseteq L'$ and L is also a lattice under the meet and join operations inherited from L' . In this case, we say that L' *embeds* L .

We now state some technical theorems from the theory which we crucially use. The following theorem shows that given a poset one can find a lattice that contains the poset.

Theorem 1 (Dedekind-Macneille Completion[3]). *For any poset P , there always exists a smallest lattice L that order embeds P . This lattice L is called the Dedekind-MacNeille completion of P .*

One crucial property of Dedekind-MacNeille completion is that it preserves all meets and joins that exist in the poset, i.e., if a and b are two elements in the poset and $a \vee b = x$ in the poset, then we have $f(a) \vee f(b) = f(x)$ in the Dedekind-MacNeille completion of the poset, where f is the embedding function that maps elements in P to elements in L .

We now state a very important theorem that concerns the structure of distributive lattices.

Theorem 2 (Birkhoff's Representation Theorem[3]). *The elements of any finite distributive lattice can be represented as finite sets, in such a way that the join and meet operations over the finite distributive lattice correspond to unions and intersections of the finite sets used to represent those elements.*

The n^{th} *partition lattice* for $n \geq 2$, denoted Π_n , is the lattice where elements are partitions of the set $\{1, \dots, n\}$ ordered by refinement. Equivalently, the elements are equivalence relations on the set $\{1, \dots, n\}$ where the glb is the intersection and lub is the transitive closure of the union.

Theorem 3 (Pudlák, Tůma[7]). *For any finite lattice L , there exists an i such that L can be embedded as a sublattice in Π_i .*

We can describe elements of the partition lattice Π_n by using undirected graphs on the vertex set $\{1, \dots, n\}$. Given an undirected graph $G = (\{1, \dots, n\}, E)$, the corresponding element $A_G \in \Pi_n$ is the equivalence relation $A_G = \{(i, j) : j \text{ is reachable from } i \text{ in } G\}$. We may choose transitively closed graphs (disjoint union of cliques) as the canonical representation for elements of partition lattices. Figure 2 shows the lattice Π_4 and two undirected graphs representing two different elements in Π_4 .

Some Relations in Partition Lattices: A formula over a lattice is defined analogously to a Boolean formula. The Boolean AND and OR operations are generalized to glb and lub operations of the lattice and the formula may contain elements of the lattice as constants (Similar to Boolean values 0 and 1 in a Boolean formula). In this section, we prove the existence of a certain formula over partition lattices. The following statements hold¹ for any partition lattice Π_i where $i \geq 2$. In the following propositions, the element 0 refers to the least element of the lattice and the element 1 refers to the greatest element of the lattice.

¹Since we have not seen them explicitly in the literature, we include the proofs in this paper.

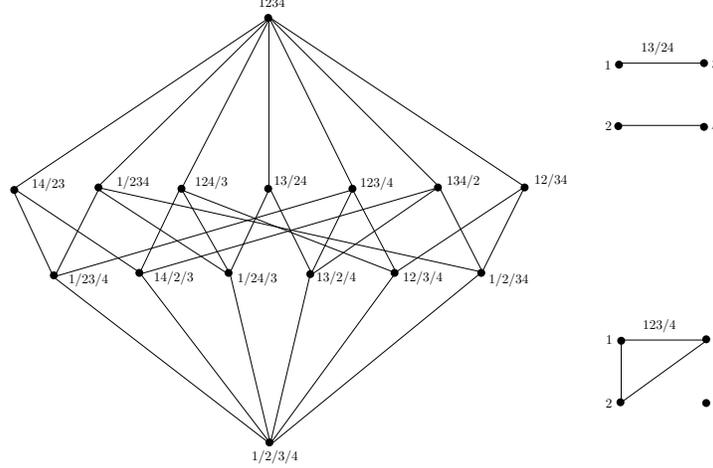


Figure 2: The lattice Π_4 and the undirected graph representation of $13/24$ and $123/4$

Proposition 1. *For any $A, B \in \Pi_i$ such that $A \not\leq B$, there exists a formula $\text{DIST}_{A,B}$ over Π_i such that $\text{DIST}_{A,B}(x) = 1$ if $x = A$ and strictly less than 1 if $x = B$.*

Proof. There are two cases to consider. Case when $[A > B]$: Let $P \in \Pi_i$ be the element corresponding to a path with exactly one vertex from each partition in A . We define $\text{DIST}_{A,B}(x) = x \vee P$. Case when $[A \not\geq B]$: Let e_1, \dots, e_m be the edges in $B \setminus A$ (using canonical representation) and let B_i denote the element in the partition lattice that correspond to the undirected graph having only the edge e_i . Let $g(x) = x \vee B_1 \vee \dots \vee B_m$. We have $g(A) > g(B) = B$. Then define $\text{DIST}_{A,B}(x) = \text{DIST}_{g(A),B}(g(x))$. \square

Proposition 2. *For any $A \in \Pi_i$, there exists a formula $\text{GE}_A(x)$ that is 1 iff $x \geq A$. In addition, there exists a formula $\text{GE}'_A(x)$ that evaluates to 1 if $x \geq A$ and evaluates to 0 otherwise.*

Proof. For the first part, simply define the formula $\text{GE}_A(x) = \bigwedge_{B \not\geq A} \text{DIST}_{A,B}(x)$ when $A \neq 0$ where $\text{DIST}_{A,B}(x)$ is as defined in Proposition 1. Define GE_0 as identically 1.

For the second part, consider the formula f_Z that is defined if and only if $Z \neq 1$ and it maps 1 to 1 and Z to 0 (the images of the rest of the elements in the lattice can be arbitrary). Let Z have $k \geq 2$ partitions. Let e_1, \dots, e_m be the edges of the complete k -partite graph on these k partitions. Let B_1, \dots, B_m be lattice elements such that B_i corresponds to the undirected graph that contains only the edge e_i . $f_Z(x) = \bigvee_{i=1}^m (x \wedge B_i)$. Now to complete the second part, define the formula $\text{GE}'_A(x) = \bigwedge_{B < 1} f_B(\text{GE}_A(x))$ (GE'_0 is identically 1). \square

3 Generalization to Finite Bounded Posets and Universal Circuits

In this section, we consider comparator circuit models over arbitrary fixed finite bounded posets instead of the Boolean lattice on two elements. We then prove the existence of universal circuits for these models. The existence of these generalized universal comparator circuits imply that the classes characterized by comparator circuit families over fixed finite bounded posets also have canonical complete problems – the comparator circuit evaluation problem over the same fixed finite bounded poset.

Definition 1 (Comparator Circuits over Fixed Finite Bounded Posets). A *comparator circuit family* over a finite bounded poset P with an accepting element² $a \in P$ is a family of circuits $C = \{C_n\}_{n \geq 0}$ where $C_n = (W, G, f)$ and $f : W \mapsto (P \cup \{(i, g) : 1 \leq i \leq n \text{ and } g : \Sigma \mapsto P\})$. Here $W = \{w_1, \dots, w_m\}$ is a set of lines and G is an ordered list of gates (w_i, w_j) .

On input $x \in \Sigma^n$, we define the output of the comparator circuit C_n as follows. Each line is initially assigned a value according to f as follows. We denote the value of the line w_i by $\text{val}(w_i)$. If $f(w) \in P$, then the value is the element $f(w)$. Otherwise $f(w) = (i, g)$ and the initial value is given by $g(x_i)$. A gate (w_i, w_j) (non-deterministically) updates the value of the line w_i into $\text{val}(w_i) \wedge \text{val}(w_j)$ and the value of the line w_j into $\text{val}(w_i) \vee \text{val}(w_j)$. The values of lines are updated by each gate in G in order and the circuit accepts x if and only if $\text{val}(w_1) = a$ at the end of the computation for some sequence of non-deterministic choices.

Let Σ be any finite alphabet. A comparator circuit family C over a bounded poset P with an accepting element $a \in P$ decides $\mathsf{L} \subseteq \Sigma^*$ if $C_{|x|}(x) = a$ if and only if $x \in \mathsf{L} \forall x \in \Sigma^*$.

All comparator circuit families in this paper are logspace-uniform unless mentioned otherwise.

Remark 1. Note that when the underlying poset is a lattice, the output of all gates in the comparator circuit is deterministic. In other words, the non-determinism in the circuit comes from the fact that two elements in a poset need not have unique lubs and glbs.

Note that we can generalize any circuit model that uses only AND and OR gates to work over arbitrary bounded posets. However, as we will see in this paper, the most interesting case is comparator circuits over arbitrary bounded posets as they lead to new characterizations of complexity classes other than CC.

Definition 2. We define the complexity class $(P, a)\text{-CC}$ as the set of all languages accepted by poly-size comparator circuit families over the finite bounded poset P with accepting element $a \in P$.

If the complexity class does not change with the accepting element, i.e., $(P, a)\text{-CC} = (P, b)\text{-CC}$ for any $a, b \in P$, we simply write $P\text{-CC}$ to refer to the complexity class $(P, a)\text{-CC}$.

We note that for any bounded poset P with at least 2 elements, we can simulate a Boolean lattice by using 0 (least element) and some $a > 0$ in P . Therefore, we have $\text{CC} \subseteq (P, a)\text{-CC}$.

Definition 3. For any finite bounded poset P and any $a \in P$, the comparator circuit evaluation problem $(P, a)\text{-CCVP}$ is defined as the set of all tuples (C, x) such that C on input x has a sequence of non-deterministic choices where it outputs $a \in P$ where C is a comparator circuit over P .

We now describe an encoding for the $(P, a)\text{-CCVP}$ problem. The input is encoded by a binary string of the form $1^n 0 1^m 0 \{0, 1\}^{n(n-1)m+n}$. Here the last $n(n-1)m$ bits of the string can be viewed as m blocks of $n(n-1)$ bits where the i^{th} block has exactly one set bit, say (k, j) where $k \neq j$, and it encodes the fact that the i^{th} gate is from line k to line j . The n bits prior to these bits encode the initial values of n lines. This encoding is logspace-equivalent to the ordered list representation. We call strings of this form (n, m) -valid. A given N -bit string can be valid for at most one (n, m) pair. We first prove that a universal comparator circuit exists even for comparator circuit model working over arbitrary finite fixed posets.

²In the definition of general Boolean circuits it is implicit that the element 1 is the accepting element. However, it does not make any difference even if we use 0 as the accepting element. This is because a Boolean circuit that accepts using 0 can be easily converted to one that accepts using 1 by complementing the output. This is not true for comparator circuits over bounded posets in general. Using different elements as accepting elements may change the power of the comparator circuit.

Proposition 3. For any bounded poset P , there exists a universal comparator circuit $U_{n,m}$ over P that when given (C, x) as input, where C is a comparator circuit over P with n lines and m gates, simulates the computation of C . That is, $U_{n,m}$ has a non-deterministic path that outputs $a \in P$ if and only if C has such a path, for any $a \in P$. Moreover, the size of $U_{n,m}$ is $\text{poly}(n, m)$.

Proof. We simply observe that the construction for a universal circuit for the class CC in [2] generalizes to arbitrary bounded posets. The gadget shown in Figure 3 simulates the comparator gate $g = (y, x)$ depending on the “enable” input e . Here the inputs e and \bar{e} satisfy the following property. If $e = 0$ ($e = 1$), then $\bar{e} = 1$ ($\bar{e} = 0$ resp) where 0 and 1 are the least and greatest elements of the bounded poset P respectively. If the enable input is 1, then gate g is active. If the enable input is 0, then the gate g acts as a pass-through gate, i.e., the lines labelled x and y retain their original values.

Now to simulate a single gate in the circuit C , the universal circuit uses $n(n - 1)$ such gadgets where n is the number of lines in C . The inputs e and \bar{e} for each gadget is set according to C . The circuit C can be simulated using $n(n - 1)m$ gates where m is the number of gates in C . \square

The following proposition is a generalization of the corresponding theorem for Boolean comparator circuits in [2].

Proposition 4. The language $(P, a)\text{-CCVP}$ is complete under logspace reductions for the class $(P, a)\text{-CC}$ for all finite bounded posets P and any $a \in P$.

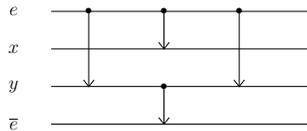


Figure 3: Conditional Comparator Gadget

Proof. The problem $(P, a)\text{-CCVP}$ is trivially hard for the class $(P, a)\text{-CC}$. Let $L \in (P, a)\text{-CC}$ via a logspace-uniform circuit family $\{C_n\}$.

Now given x as input, we output the tuple (C_n, x) by running the uniformity algorithm.

The fact that $(P, a)\text{-CCVP} \in (P, a)\text{-CC}$ follows from Proposition 3. Given a string, it can be checked in logspace whether it is (n, m) -valid once n and m are fixed. Let $V_{n,m}$ be a logspace uniform comparator circuit over the 0–1 lattice that takes an N bit string as input and outputs 1 iff the input is an (n, m) -valid string. Let $N = 2n + m + 2 + n(n - 1)m$ be the total length of the input. The uniformity machine on input N writes out the description of $\bigvee_{(n,m)} U_{n,m} \wedge V_{n,m}$ over all (n, m) pairs satisfying $N = 2n + m + 2 + n(n - 1)m$. \square

4 Comparator Circuits over Lattices

First, we show that comparator circuits over distributive lattices characterize the class CC .

Theorem 4. Let L be any non-trivial finite distributive lattice and $a \in L$ be an arbitrary element. Then $\text{CC} = (L, a)\text{-CC}$.

Proof. By Birkhoff’s representation theorem, every finite distributive lattice of k elements is isomorphic to a lattice where each element is some subset of $[k]$ (ordered by inclusion) and the join and meet operations in the original finite distributive lattice correspond to set union and set intersection operations in the new lattice. We will use this to simulate a circuit over an arbitrary finite

distributive lattice L of size k using a circuit over the 0–1 lattice. Each line w in the original circuit is replaced by k lines w_1, \dots, w_k . The invariant maintained is that whenever a line in the original circuit carries $a \in L$, these k lines carry the characteristic vector of the set corresponding to the element a . Now a gate (w, x) in the original circuit is replaced by k gates $(w_1, x_1), \dots, (w_k, x_k)$ in the new circuit. The correctness follows from the fact that meet and join operations in the original circuit correspond to set union and set intersection which in turn correspond to AND and OR operations of the characteristic vectors.

We now prove that $\text{CC} \subseteq (L, a)\text{-CC}$. First, we consider the case where $a \neq 0$. We replace the Boolean value 0 by the minimum element in L and the Boolean value 1 by the maximum element in L . The output wire of the new circuit is $o \wedge a$ where o is the original output wire. It is easy to see that this circuit output a if and only if the original circuit outputs 1. If $a \neq 0$, we can use the fact that the class CC is closed under complementation to construct a Boolean comparator circuit that accepts using 0. This can be easily translated to an $(L, a)\text{-CC}$ circuit as above. \square

Now we consider comparator circuits over fixed finite lattices. Note that when characterizing the class P in terms of Boolean circuits, the fan-out of gates is required to be at least 2. In fact, Mayr and Subramanian’s [5] primary motivation while introducing the class CC was to study fan-out restricted circuits. We show that if comparator circuits are given the freedom to compute over any lattice (as opposed to the Boolean lattice on 2 elements), then the fan-out restriction is irrelevant.

The following lemma describes a fixed finite lattice over which comparator circuits capture P . However, it is not clear whether the class captured by comparator circuits over this lattice is independent of the accepting element. In Theorem 5, we show that there exists a lattice that captures P irrespective of the accepting element. The language MCVP consists of all tuples (C, x) where C is a Boolean circuit with only AND, OR and input gates. Here $x \in \{0, 1\}^n$ where n is the number of input gates to C and x specifies the value of each of these input gates. In the proof, we will reduce in logspace the language MCVP which is complete for the class P under logspace reductions to the comparator circuit value problem over the finite lattice given in Figure 5.

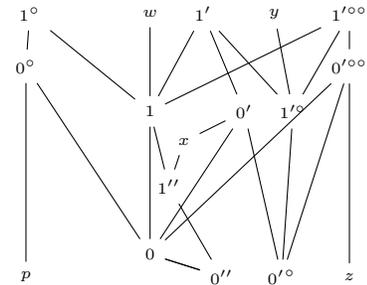


Figure 4: The poset for simulating P

Lemma 1. *Let L be the lattice in Figure 5. Then $\text{P} = (L, 1)\text{-CC}$ (Note that 1 is not the maximum element in the lattice).*

Proof. Let (C, x) be the input to MCVP . For each wire in C , we add a line to our comparator circuit. The initial value of the lines that correspond to the input wires of C are set to 0 or 1 of the poset P shown in Figure 4 according to whether they are 0 or 1 in x . The comparator circuit simulates C in a level by level fashion maintaining the invariant that the lines carry 0 or 1 depending on whether they carry 0 or 1 in C . We will show how our comparator circuit simulates a level 1 OR gate of fan-out 2. The proof then follows by an easy induction.

Since $0 \leq_P 1$ an AND (OR) gate in C can be simulated by a meet (join) operation in P . The gadget shown in Figure 6 is used to implement the fan-out operation. The idea is that the first gate in the gadget implements the AND/OR operation and the rest of the gates in this gadget “copy” the result of this operation into the lines o_1 and o_2 that correspond to the two output wires of the gate. The reader can verify that the elements of P satisfy the following meet and join identities.

Figure 6 shows how one could use the following identities to copy the output of $a \vee b$ into two lines (labelled o_1 and o_2).

The identity $0 \vee 1 = 1$ is used to implement the Boolean AND/OR operation. This is used by the first gate in Figure 6. Once the required value is computed. We add a gate between the line carrying the result of the AND/OR operation and a line with value x . As the following identities show, this makes two “copies” of the result of the Boolean operation. $0 \vee x = 0'$, $1 \vee x = 1'$, $0 \wedge x = 0''$, $1 \wedge x = 1''$

Now, the following identities can be used to convert the first copy ($0'$ or $1'$) into the original value (0 or 1). $0' \wedge y = 0'^{\circ}$, $1' \wedge y = 1'^{\circ}$, $0'^{\circ} \vee z = 0'^{\circ\circ}$, $1'^{\circ} \vee z = 1'^{\circ\circ}$, $0'^{\circ\circ} \wedge w = 0$, $1'^{\circ\circ} \wedge w = 1$

Similarly, the following identities can be used to convert the second copy ($0''$ or $1''$) into the original value (0 or 1). $0'' \vee p = 0^{\circ}$, $1'' \vee p = 1^{\circ}$, $0^{\circ} \wedge w = 0$, $1^{\circ} \wedge w = 1$

The lattice in Figure 5 is simply the Dedekind-MacNeille completion of P . Since the Dedekind-MacNeille completion preserves all existing meets and joins, the same computation can also be performed by this lattice.

To see that for any lattice L and any $a \in L$, (L, a) -CC is in \mathbf{P} , observe that in poly-time we can evaluate the n^{th} comparator circuit from the comparator circuit family for the language in (L, a) -CC. \square

Lemma 1 shows that the complexity class captured by the comparator circuit could change (Assuming $\text{CC} \neq \mathbf{P}$) depending on the underlying lattice and the accepting element. In the following theorem, we show that if we consider any partition lattice, say Π_i , that embeds L (in Lemma 1), then the complexity class is \mathbf{P} irrespective of the accepting element. We crucially use the fact that the circuit in the proof of Lemma 1 outputs only the elements 0 and 1 in L .

Theorem 5. *There exists a constant i such that Π_i -CC = \mathbf{P} .*

Proof. We know that there exists a finite lattice L and an $a, b \in L$ such that for any language $M \in \mathbf{P}$ there exists a comparator circuit family over L that decides M by using a to accept and b to reject. Also $b < a$. By Pudlák-Tůma theorem [7], we know that there exists a constant i such that L can be embedded in Π_i . It remains to show that the accepting element used does not change the complexity. In fact, we will show that for any $X, Y \in \Pi_i$ where $X \neq Y$, we can design a comparator circuit family over Π_i that accepts M using X and rejects using Y . Let A and B be the elements in Π_i that a and b gets mapped to by this embedding ($B < A$). Then there exists a circuit family C over Π_i , deciding M , that accepts using A and rejects using B . We will construct a circuit family C' over Π_i from C such that C' uses 1 to accept and 0 to reject. Here 1 and 0 are the maximum and minimum elements in Π_i . Now if we let x be the output of a circuit in the circuit family C , we can construct C' by computing $\text{GE}'_A(x)$ (See Proposition 2). Similarly, we can construct a circuit family C'' that accepts using 0 and rejects using 1 by reducing the language M to $\overline{\text{MCVP}}$ and then applying the construction in Lemma 1 and then computing $\text{GE}'_A(x)$ on the output of this circuit. The required circuit family is then the one computing $(X \wedge C') \vee (Y \wedge C'')$. \square

If we can show that there exists a finite distributive lattice such that the poset in Figure 4 can be embedded in that lattice while preserving all existing meets and joins, then $\mathbf{P} = \text{CC}$. In the following theorem, we show that such an embedding is not possible.

Theorem 6. *The poset in Figure 4 cannot be embedded into any distributive lattice while preserving all meets and joins.*

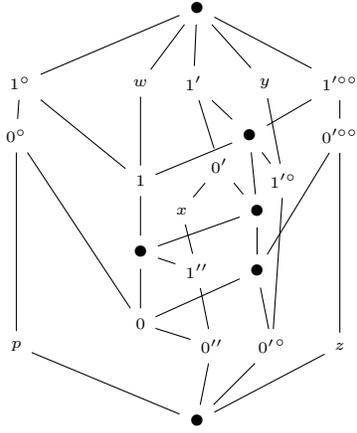


Figure 5: The lattice for simulating P

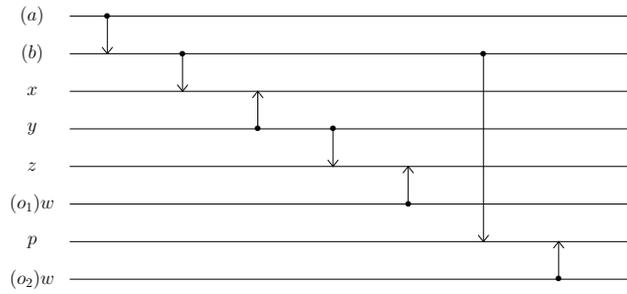


Figure 6: Copy $a \vee b$ into o_1 and o_2

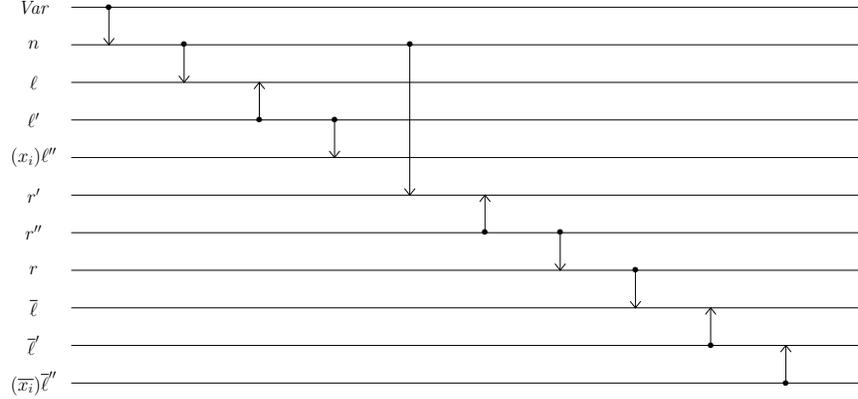


Figure 8: Nondeterministically generate x_i and \bar{x}_i

problem SAT into (P, a) -CCVP. Without loss of generality, we can assume that the circuit does not contain any NOT gates.

Note that the poset P contains the poset in the proof of Theorem 5. This is represented by the hexagon in Figure 7. The elements marked 0 and 1 inside this hexagon are the elements marked 0 and 1 in Figure 4. This containment ensures that we can implement all operations that we used while simulating MCVP to be used here as well. Let C be the input to the SAT problem. The 0 and 1 values carried by wires will be represented by 0 and 1 in P as in the proof of Theorem 5. The non-trivial part is to simulate the input variables x_1, \dots, x_n . These input variables to C are handled by non-deterministically generating 0 or 1 (of P) on the lines corresponding to the wires attached to these input gates. We also have to ensure that when we non-deterministically generate the values of input variables, the values generated for x_i and \bar{x}_i are consistent. This is ensured by generating x_i non-deterministically and then complementing the generated value to get \bar{x}_i . The fan-out operation is implemented as in the proof of Theorem 5.

Note that the minimal upper bounds for the elements Var and n in the poset P are n_0 and n_1 . These values stand for a non-deterministically generated 0 and 1 resp. Now for each variable x_i we take the minimal upper bound of these two elements in P to non-deterministically generate the value of x_i . The only thing that remains to be done is to make the corresponding \bar{x}_i variable consistent, i.e., when a 0 is generated non-deterministically for x_i , we have to ensure that all lines carrying \bar{x}_i in that non-deterministic path carry the value 0. The sequence of meet and join identities that we are going to describe can be used to implement this computation. Figure 8 shows how to generate x_i and \bar{x}_i consistently in a non-deterministic fashion using the identities given below.

The following identity enables us to non-deterministically generate a 0 or a 1. Note that we are only generating n_0 and n_1 at this point. But we will later convert this into 0 or 1 that are used for implementing the Boolean operations.

$$Var \vee n = \{n_0, n_1\}$$

Now we use the following identities to convert n_0 or n_1 into a 0 or a 1 respectively.

$$\begin{array}{ll}
\ell \vee n_0 = n'_0 & \ell \vee n_1 = n'_1 \\
\ell' \wedge n'_0 = n''_0 & \ell' \wedge n'_1 = n''_1 \\
\ell'' \vee n''_0 = 0 & \ell'' \vee n''_1 = 1
\end{array}$$

Note that the original n_0 or n_1 that was generated will be destroyed by the above sequence of operations (By doing $\ell \wedge n_0$ for ex.). Using the following identities, we ensure that the original value generated non-deterministically is restored.

$$\begin{array}{ll}
\ell \wedge n_0 = cn'_0 & \ell \wedge n_1 = cn'_1 \\
r' \vee cn'_0 = ccn'_0 & r' \vee cn'_1 = ccn'_1 \\
r'' \wedge ccn'_0 = cccn'_0 & r'' \wedge ccn'_1 = cccn'_1 \\
r \vee cccn'_0 = n_0 & r \vee cccn'_1 = n_1
\end{array}$$

Now we use the restored value along with the following identities to generate the value for the line carrying \bar{x}_i .

$$\begin{array}{ll}
\bar{\ell} \vee n_0 = \bar{n}_0 & \bar{\ell} \vee n_1 = \bar{n}_1 \\
\bar{\ell}' \wedge \bar{n}_0 = \bar{n}'_0 & \bar{\ell}' \wedge \bar{n}_1 = \bar{n}'_1 \\
\bar{\ell}'' \wedge \bar{n}'_0 = 1 & \bar{\ell}'' \wedge \bar{n}'_1 = 0
\end{array}$$

The reduction from SAT is as follows. First, we use the reduction from CVP to (P, a) -CC to construct a (P, a) -CC circuit, say C , that evaluates the input formula. Then, we construct a circuit for non-deterministically generating 0/1 values for all the variables in the formula. The wires of this circuit that carry the generated values are then connected to the input wires in C that take the values of variables in the formula as input. It is easy to see that the resulting circuit evaluates to 1 if and only if the formula is satisfiable.

To see that (P, a) -CC is in NP, observe that we can evaluate any (P, a) -CC circuit in NP by guessing the output value of each gate to be one of the possible values. i.e., if the gate is an OR (AND) gate taking a and b as input, we non-deterministically guess that the gate outputs one of the values in $a \vee b$ ($a \wedge b$). Finally, we simply check whether the value on the output wire is in the accepting set. \square

6 Skew Comparator Circuits

In this section, we study the skew comparator circuits defined in the preliminaries. We show that SkewCC is the class LOG. Recall that the class NLOG can be characterized as the set of all languages computed by logspace-uniform Boolean circuits with skewed AND gates. So the result in this section draws a parallel between the P vs CC problem and the NLOG vs LOG problem. It immediately follows that SkewCC over distributive lattices also characterize the class LOG.

We begin by considering a canonical complete problem for the class LOG. The language DGAP1 consists of all tuples (G, s, t) where $G = (V, E)$ is a directed graph where each vertex has out-degree at most one and $s, t \in V$ and there is a directed path from s to t . We use a variant of DGAP1 problem in our setting. The variant (called DGAP1') is that the out-degree constraint is not applied to s . It is easy to see that DGAP1' is also in LOG. Indeed, for each neighbour u of s , run the DGAP1 algorithm to check whether t is reachable from u .

Theorem 8. $\text{SkewCC} = \text{LOG}$

Proof. (\subseteq) Let $L \in \text{SkewCC}$. We will prove that $L \in \text{LOG}$ by reducing L to DGAP1'. The reduction is as follows. Observe that we can reduce the language L to SkewCCVP by a logspace reduction (using the uniformity algorithm). Then we reduce SkewCCVP to DGAP1' as follows. Let C be an instance of SkewCCVP. For each wire in C add a vertex to the graph G . The vertex corresponding to the output wire is the destination vertex t . Add a source vertex s . The edges of G are as follows. For each vertex v that corresponds to an input wire of C having value 1, add the edge (s, v) to the graph. Now consider a comparator gate g in C with input wires e_1 and e_2 and AND output wire e_3 and OR output wire e_4 . There are two cases.

Gate g has an AND output Without loss of generality, assume that e_2 is an input wire to C . If $e_2 = 1$, then add the edges (e_1, e_3) and (e_2, e_4) to G . If $e_2 = 0$, then add the edge (e_1, e_4) to the graph G .

Gate g has an unused AND output Add the edges (e_1, e_4) and (e_2, e_4) . Note that it is easy to check in logspace whether the AND output of a gate is used or not. Simply scan forward on the input to check whether any gate in the input after g is incident on the AND output line of g or not.

It is clear that G has an s - t path if and only if C outputs 1. This follows from the observation that every vertex v in G where $v \neq s$ corresponds to a wire in C and v is reachable from s if and only if the wire corresponding to v carries the value 1. All vertices other than s in G have out-degree at most 1. Furthermore, the reduction can be implemented in logspace.

(\supseteq) Let $L \in \text{LOG}$ and let B be a poly-sized layered branching program deciding L . We will design a skew comparator circuit C to simulate B . Let s be a state in B reading x_i and let the edge labelled 1 be directed towards a state t and let the edge labelled 0 be directed towards a state u . Then the gadget shown in Figure 9b simulates this part of the BP B (We say that this gadget corresponds to the state s). The truth table for this gadget is shown in the Table 9a. This table assumes that the lines t and u carry the value 0 initially. The value of the line labelled s will be 1 on input x just before the gates in this gadget are evaluated if and only if the input x reaches the state s in B . It is clear that after all the gates in this gadget are evaluated, the value of the line labelled t (or u) is 1 if and only if the input x reaches t (or u resp.) in B .

Now the circuit C is as follows. For each state in B introduce a line in C and for each state in each layer from the first layer to the last layer, in that order, add the gates in the gadgets corresponding to these states in the same order to C . Note that the lines annotated x_i and \bar{x}_i in a gadget are only used in that gadget. When these values are required again, new annotated lines are used. The line corresponding to the accepting state is the output line. The initial value of lines corresponding to each state other than the start state of B is 0 and the initial value of the line corresponding to the start state is 1. Also the circuit is a skew circuit since all used AND gates in

the gadget are skew. For establishing the correctness, we observe that the following claim holds. The circuit C outputs 1 on input x if and only if there is a path in B from the start state to the accepting state on input x . To complete the correctness proof, we prove the following claim:

Claim 1. *The circuit C outputs 1 on input x if and only if there is a path in B from the start state to the accepting state on input x .*

Proof. Let the i^{th} block of C include all the gadgets corresponding to all the states in layer i of B . We will prove the more general claim that after all gates up to and including the i^{th} block are evaluated, if we consider all the lines that correspond to states in the $(i + 1)^{\text{th}}$ layer of B , the only line that will have a value 1 will correspond to the state on $(i + 1)^{\text{th}}$ layer reached on input x . We will prove this by induction on the layer number.

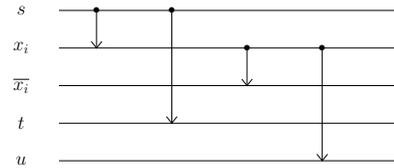
Base case: $i = 0$ Since there is only the start state in layer 1 and it is initialized to the value 1, the base case is true.

Induction Assume that the claim is true for i . Let s be the state in the $(i + 1)^{\text{th}}$ layer that is reached by x and let t be the state in the $(i + 2)^{\text{th}}$ that is reached by x . Now from the truth table in Table 9a it is clear that after the gadget for state s is evaluated the value of line t will become 1. Also, from the truth table, it is clear that the values of all the other lines that correspond to states in the $(i + 2)^{\text{th}}$ layer remains 0. Notice that all gates in block $i + 1$ incident on t are OR gates. So once the value of line t becomes 1, it remains so until block $i + 2$. \square

Let s be the number of states in B . Then the number of lines in C is at most $3s$ and the number of gates in C is at most $4s$. Since B is poly-size, so is C .

s	x_i	\bar{x}_i	t	u
0	1	0	0	0
0	0	1	0	0
1	1	0	1	0
1	0	1	0	1

(a) Truth Table for the gadget for BPs



(b) The gadget for simulating BPs

It is easy to see that this reduction can be implemented in NC^1 . \square

Since the construction in Theorem 4 preserves skewness of the circuit, we have the following corollary.

Corollary 1. *Let L be any distributive lattice and let a be any element in L , then $(L, a)\text{-SkewCC} = \text{LOG}$.*

We now look at skewed comparator circuits over arbitrary lattices and show that they also characterize the class P . We prove this by modifying the proof of Theorem 5. More specifically, we show that by changing the underlying lattice, we can simulate any AND gate using OR gates and skewed AND gates.

Theorem 9. *There exists an i such that $\Pi_i\text{-SkewCC} = \text{P}$.*

Proof. We will start with the comparator circuit in the proof of Theorem 5 and show how to replace AND gates in that circuit with OR gates and skewed AND gates. We start with the poset shown in Figure 4. We then add new elements q , r , and v to the poset that satisfies the following relations.

$$\begin{aligned} q \wedge 0 &= 0^q \\ q \wedge 1 &= 1^q \\ r \vee 0^q &= 0^r \\ r \vee 1^q &= 1^r \\ v \wedge 0^r &= 0+ \\ v \wedge 1^r &= 1+ \end{aligned}$$

Here, the elements $0+$ and $1+$ can be thought of as placeholders for 0 and 1 respectively. We then add four new elements to the poset $(a, b)+$ where $a, b \in \{0, 1\}$ satisfying $a+ \vee b+ = (a, b)+$. Then we introduce new elements s , t , and u such that

$$\begin{aligned} s \wedge (1, 1)+ &= 1^s \\ s \wedge (a, b)+ &= 0^s, \text{ otherwise} \\ t \vee 0^s &= 1^t \\ t \vee 1^s &= 0^t \\ u \wedge 0^t &= 0 \\ u \wedge 1^t &= 1 \end{aligned}$$

Now given an AND gate computing $x \wedge y \in \{0, 1\}$ in the circuit in the proof of Theorem 5 (Note that the non-skew AND gates in that circuit always take input from $\{0, 1\}$). We replace that AND gate with the following sequence of operations. First we compute $((x \wedge q) \vee r) \wedge v$ to yield $x+$. We then OR the wires containing $x+$ and y (This is the only non-skew gate used in this construction) to yield $(x, y)+$. Finally, we compute $((x, y)+ \wedge s) \vee t \wedge u$ to yield the required value $x \wedge y$. Note that all AND gates used in this construction are skewed. The complete set of relations added to the poset in Figure 4 is listed in Figure 10.

We use the same argument as in the proof of Theorem 5 to show that this can be simulated in a partition lattice irrespective of the accepting element. \square

7 Formulae over Lattices

It is well known that languages decided by poly-size formulae is the class NC^1 . By definition, the class NC^1 is also the class of languages decided by log-depth Boolean circuits with bounded fan-in AND and OR gates. We can modify Definition 1 to define formulae over finite bounded posets. We denote by (L, a) -Formulae, where L is a lattice and $a \in L$, the class of all languages decided by poly-size formulae over L using a as the accepting element. In this section, we show that the languages computed by poly-size formulae over any fixed finite lattice is the class NC^1 . The proof for the Boolean case is by [6] and it works by depth reducing an arbitrary formula of poly-size to a

$$\begin{array}{cccccc}
t \leq 0^t & 0^t \leq 1^t & 0^s \leq 0^t & 0^s \leq 1^s & 0^s \leq (0,0)+ \\
1^s \leq 1^t & 1^s \leq s & 1^s \leq (1,1)+ & (0,1)+ \leq (1,1)+ & (1,0)+ \leq (1,1)+ \\
(0,0)+ \leq (0,1)+ & (0,0)+ \leq (1,0)+ & 0+ \leq (0,0)+ & 0+ \leq 1+ & 0+ \leq 0^r \\
1+ \leq (0,1)+ & 1+ \leq v & 1+ \leq 1^r & 0^r \leq 1^r & r \leq 0^r \\
0^q \leq 0^r & 0^q \leq 1^q & 0^q \leq 0 & 1^q \leq 1^r & 1^q \leq q \\
1^q \leq 1 & 1 \leq u & 1 \leq 1^t & 1 \leq (1,0)+ & 0 \leq 0^t \\
0 \leq (0,0)+ & & & &
\end{array}$$

Figure 10: Relations added to the poset in Figure 4 to make the circuit skewed

Boolean formula of poly size and log depth. The depth reduction is done by identifying a separator vertex in the tree and then evaluating the separated components (which are smaller circuits) in parallel. We show that a similar argument can be extended to the case of finite lattices as well. Our main theorem in this section is the following.

Theorem 10. *Let L be any finite lattice and let a be an arbitrary element in L . We have $(L, a)\text{-Formulae} = \text{NC}^1$.*

Proof. (\supseteq) Any lattice with at least 2 elements contains the 0–1 lattice as a sublattice. Also since NC^1 is closed under complementation, the class does not change even if the acceptor is 0.

(\subseteq) Let F be a poly-size formula family over L . Let i be such that L can be embedded in Π_i . Let F' be the formula family over Π_i that corresponds to F . We will now construct a log-depth poly-size formula family F'' that computes the same language as F' . We will use F' to denote a formula in the family F' . Let v be the tree separator of the tree corresponding to F' . For each $a_i \in \Pi_i$, we will construct two formulae. The first one, say F_1^v , computes the value at the root of F' assuming that value at v is a_i and the other, say F_2^v computes the value at the node v and applies GE'_{a_i} (See Proposition 2) on that value. Then we compute the sub-formula $F_1^v \wedge F_2^v$. After that we take the lub over all such sub-formulae (one for each a_i). This construction is applied recursively on F_1^v and F_2^v to obtain a log-depth poly-size formula equivalent to F .

Suppose the correct value of the sub-formula of F' rooted at v is a_i . Then the only sub-formulae $F_1^v \wedge F_2^v$ outputting a non-zero value are the ones corresponding to $a_j \leq a_i$. The non-zero value output by such a sub-formula is b_j , the value obtained at the root when the value of v is a_j . But we know that b_i , the actual value of the original formula is greater than or equal to the value b_j of any sub-formula by monotonicity of lub and glb. So the topmost lub will always output the correct value b_i .

The final formula is log-depth, poly-size since the formulae GE'_a have constant depth. Now we can construct an NC^1 circuit from F'' by encoding each element in Π_i in binary and replacing each gate in F'' by constant-sized circuits computing the lub and glb over Π_i . \square

8 Discussion and Conclusion

We studied the computational power of comparator circuits over bounded posets. We provide alternative characterizations of P, LOG, NLOG and NP in terms of comparator circuits.

A natural open problem that comes out of our approach is about a possible dichotomy between P and CC with respect to lattice structure. More concretely, can we design comparator circuits over fixed lattices M_3 or N_5 (or powers of it) for all languages in P ? Noting that existence of M_3 or N_5 as a sublattice is a necessary and sufficient condition for non-distributivity (by the M_3 - N_5 theorem [3]), if we manage to show that $M_3\text{-CC} = (N_5, a)\text{-CC} = \mathsf{P}$ for any $a \in N_5$, this will show a dichotomy between P and CC .

In the context of NLOG vs LOG , there are two open problems. Firstly, it will also be interesting to see if a dichotomy theorem holds, with respect to the lattice structure. Secondly, we note that the upper bound of NLOG for the case of skew comparator circuits over finite lattices, uses the embeddability into partition lattices. The power of skew comparator circuits over finite bounded posets is unclear. It is not even clear whether they compute only languages in P .

Cook et al. [2] proposed the question whether membership testing for CFLs is in CC . Our characterization of CC in terms of distributive lattices leads to a concrete approach towards proving this. Namely, designing a lattice to decide membership testing for CFLs and showing that this lattice is distributive.

Acknowledgments: We thank the anonymous reviewers for their constructive comments, which helped us improve the paper. In particular, we thank the reviewer who pointed out an error in the proof of earlier Theorem 9 (where we had erroneously claimed that there exists an i , $\Pi_i\text{-SkewCC} = \mathsf{NLOG}$). The reviewer also had outlined an argument the details of which we have incorporated in this version as the proof of Theorem 9.

References

- [1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.
- [2] Stephen A. Cook, Yuval Filmus, and Dai Tri Man Lê. The complexity of the comparator circuit value problem. *ACM Trans. Comput. Theory*, 6(4):15:1–15:44, August 2014.
- [3] Brian A. Davey and Hilary A. Priestley. *Introduction to lattices and order*. Cambridge University Press, Cambridge, 1990.
- [4] Bernhard Ganter and Sergei O. Kuznetsov. Stepwise construction of the Dedekind-MacNeille completion. In *Conceptual structures: theory, tools and applications. 6th international conference, ICCS '98, Montpellier, France, August 10–12, 1998. Proceedings*, pages 295–302. Berlin: Springer, 1998.
- [5] Ernst W. Mayr and Ashok Subramanian. The complexity of circuit value and network stability. *J. Comput. Syst. Sci.*, 44(2):302–323, 1992.
- [6] P.M.Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of 4th Hawaii Symp. on System Sciences*, pages 525–527, 1971.
- [7] Pavel Pudlák and Jiří Tůma. Every finite lattice can be embedded in a finite partition lattice. *algebra universalis*, 10(1):74–95, 1980.

- [8] Ashok Subramanian. *The Computational Complexity of the Circuit Value and Network Stability Problems*. PhD thesis, Stanford, CA, USA, 1990. AAI9102356.

A Comparator Circuits over Growing Lattices

We can generalize the comparator circuit model even further by allowing it to compute over lattices that grow with the size of the input. If the size of the lattice is polynomial in the size of the input and if the lattice can be computed by the uniformity machine, then the languages computed by these circuits are in the class P . However, since we have the freedom to change the lattice according to the size of the input, we may be able to capture the class P using structurally simpler lattices. It is conceivable that the class P could be captured by a family of distributive lattices, while no finite lattice capturing P can be distributive.

In this section, we present a formal definition of comparator circuits over growing posets and then present a lattice family that captures the class P . Then we will show that, even for this simpler lattice, an embedding to a family of distributive lattices is not possible (Similar to Theorem 6).

Definition 4 (Comparator Circuits over Growing Bounded Posets). *A comparator circuit family over a growing bounded poset family $P = \{P_n\}$ with accepting set $A = \{A_n\}$ where $A_n \subseteq P_n$ is a family of circuits $C = \{C_n\}_{n \geq 0}$ where $C_n = (W, G, f)$ where $f : W \mapsto (P_n \cup \{(i, g) : 1 \leq i \leq n \text{ and } g : \Sigma \mapsto P_n\})$ is a comparator circuit. Here $W = \{w_1, \dots, w_m\}$ is a set of lines and G is an ordered list of gates (w_i, w_j) .*

On input $x \in \Sigma^n$, we define the output of the comparator circuit C_n as follows. Each line is initially assigned a value according to f as follows. We denote the value of the line w_i by $\text{val}(w_i)$. If $f(w) \in P_n$, then the value is the element $f(w)$. Otherwise $f(w) = (i, g)$ and the initial value is given by $g(x_i)$. A gate (w_i, w_j) (non-deterministically) updates the value of the line w_i into $\text{val}(w_i) \wedge \text{val}(w_j)$ and the value of the line w_j into $\text{val}(w_i) \vee \text{val}(w_j)$. The values of lines are updated by each gate in G in order and the circuit accepts x iff $\text{val}(w) = a \in A_n$ at the end of the computation for some sequence of non-deterministic choices.

Let Σ be any finite alphabet. A comparator circuit family C over a growing bounded poset family P_n with an accepting $A_n \subseteq P_n$ decides $\mathsf{L} \subseteq \Sigma^$ iff $C_{|x|}$ correctly decides whether $x \in \mathsf{L}$ for all $x \in \Sigma^*$.*

The circuit family is called P -uniform if there exists a TM that given 1^n as input runs in $\text{poly}(n)$ time and outputs P_n, A_n and C_n .

First, we show a lattice family that captures P .

Theorem 11. *The comparator circuit family over DM completions for the poset family in Figure 11 captures the class P .*

Proof Sketch. We construct a comparator circuit over the poset family in Figure 11 from a layered circuit with NOT gates only at the input level. The elements 0^i and 1^i in the poset correspond to the logical values 0 and 1 at the i^{th} level of the circuit. As in the proof of Lemma 1, there is a sequence of lubs and glbs that creates two copies of the logical value at the i^{th} level and then converts them to the corresponding value in the $(i + 1)^{\text{th}}$ level.

Define $m = |P_n|$. The elements of the DM completion of P_n consists of ordered pairs (A, B) where $A, B \subseteq P_n$ and $A = UP(B)$ and $B = DOWN(A)$. Here $UP(A)$ ($DOWN(A)$) is the set of all elements in the poset that are greater (less) than or equal to all elements in A . Note that in the poset P_n , if $|A| > 11$, then we have $DOWN(A) = \phi = B$ and then we have $A = P_n$. We claim that the DM completion has at most $O(m^{24})$ elements. Consider an element (A, B) in the DM completion such that $|A| > 11$ or $|B| > 11$. If $|A| > 11$, then we have $B = \phi$ and therefore $A = P_n$. Similarly, if $|B| > 11$, then we have $A = \phi$ and $B = P_n$. Therefore, all elements (A, B)

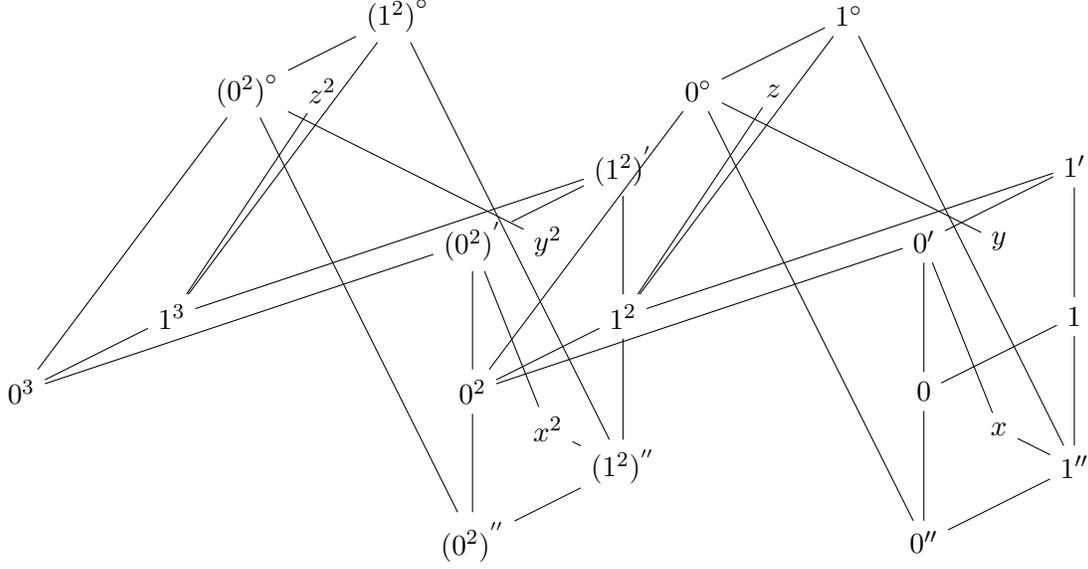


Figure 11: A growing poset family for simulating P

except (ϕ, P_n) and (P_n, ϕ) in the DM completion has $|A| \leq 11$ and $|B| \leq 11$. This implies that the DM completion has at most $O(m^{24})$ elements. To prove the P -uniformity of the comparator circuit family, we have to prove that the DM completion can be computed in polynomial time. There exists an algorithm that can compute the DM completion of a poset in time polynomial in the number of elements in the DM completion [4]. Since, the number of elements in the DM completion of P_n is polynomial in n , the P -uniformity of the comparator circuit family follows. \square

Now we prove that even this growing lattice family cannot be embedded into any distributive lattice.

Theorem 12. *The poset in Figure 11 cannot be embedded in any distributive lattice.*

Proof Sketch. The proof is similar to the proof of Theorem 6. We use the same labelling used in the proof of Theorem 6.

We have $A^2 = (A'' \cup Y) \cap Z = A^\circ \cap Z$ and $B^2 = (B'' \cup Y) \cap Z$. Since $B^2 \supset A^2$, we have $x \in B^2 \setminus A^2$. So $x \in Z$ and $x \in (B'' \cup Y) \setminus A'$. Now if $x \in Y$, then $x \in A'' \cup Y$ and so $x \in A^2$. But if $x \notin Y$, then $x \in B''$ which implies $x \in A'$ which in turn implies $x \in A^2$. A contradiction. \square