A Graphical #SAT Algorithm for Formulae with Small Clause Density

Tuomas Laakkonen[1,a], Konstantinos Meichanetzidis[1,b], John van de Wetering [2,c]
[1] Quantinuum, 17 Beaumont Street, Oxford OX1 2NA, United Kingdom
[2] Informatics Institute, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
[a,b] {tuomas.laakkonen, k.mei}@quantinuum.com, [c] john@vdwetering.name

We study the counting version of the Boolean satisfiability problem **#SAT** using the ZH-calculus, a graphical language originally introduced to reason about quantum circuits. Using this, we generalize **#SAT** to a weighted variant we call $\#\mathbf{SAT}_{\pm}$, which is complete for the class GapP. We show there is an efficient linear-time reduction from **#SAT** to $\#\mathbf{2SAT}_{\pm}$, unlike previous reductions from **#SAT** to **#2SAT** which blow up the size of the formula by a polynomial factor. Our main conceptual contribution is that introducing weights to **#SAT** allows for more efficient translations, and we use this to remove the dependence on clause width k in this case. We observe that DPLL-style algorithms for **#2SAT** can be adapted to $\#\mathbf{2SAT}_{\pm}$ directly and hence the best-known upper bounds for **#2SAT** apply. Applying an upper bound for **#2SAT** in terms of variables gives us upper bounds for **#SAT** in terms of clauses and variables that are better than $O^*(2^n)$ for small clause densities of $\frac{m}{n} < 2.25$, and improve on previous average-case and worst-case bounds for $k \geq 6$ and $k \geq 4$, respectively. Applying a similar bound in terms of clauses produces a bound of $O^*(1.1740^L)$ in terms of the length of the formula. These are, to our knowledge, the first non-trivial upper bounds for **#SAT** that is independent of clause size, and in terms of formula length, respectively. Based on a result of Kutzkov, we find an improved bound on **#3SAT** for $1.2577 < \frac{m}{n} \leq \frac{7}{3}$. Finally, we use this technique to find an upper bound on the complexity of calculating amplitudes of quantum circuits in terms of the total number of gates. Our results demonstrate that graphical reasoning can lead to new algorithmic insights, even outside the domain of quantum computing that the calculus was intended for.

1 Introduction

A graphical calculus is a language consisting of diagrams that can be transformed according to specific graphical rewrite rules. Usually these diagrams correspond to some underlying mathematical object that would be hard to reason about directly—like a matrix, tensor, relation or some combinatorial object—and the rewrite rules preserve the semantics of this interpretation. There are for instance graphical calculi for linear algebra [9, 11, 13, 57], for studying concurrency [10, 12], and for finite-state automata [41].

Most relevant for this paper are the graphical calculi developed for studying quantum computing. The *ZX-calculus* [15, 16] can represent arbitrary linear maps between any number of qubits, and has different versions of rewrite rules that are *complete* (meaning the rules can prove any true equality) for various relevant fragments of quantum computing [2, 29, 31, 47]. It has seen use in a variety of areas like optimizing quantum computations [5, 6, 21, 30, 32], more effectively classically simulating quantum computations [33, 34], and several others like [14, 26, 45]; see [54] for a review.

There are a number of variations on the ZX-calculus that include different or additional generators [28, 48, 53]. The one we will use is the *ZH-calculus* [3, 4]. The ZH-calculus has turned out to be useful in a variety of areas [22,49,50], but in particular it has been shown to naturally encode Boolean satisfiability and counting problems [19]. We will build on this representation to show that this perspective leads to better algorithms for formulae that have a low number of clauses.

A. Díaz-Caro and V. Zamdzhiev (Eds.):
Quantum Physics and Logic 2024 (QPL 2024)
EPTCS 406, 2024, pp. 137–152, doi:10.4204/EPTCS.406.7

The Boolean satisfiability problem (**SAT**) is to determine whether a given Boolean formula has a satisfying assignment of variables, and is a canonical example of an **NP**-complete problem. Worst-case upper bounds for solving **SAT** instances can be phrased in terms of some relevant parameters for a Boolean formula $f : \{0,1\}^n \to \{0,1\}$ in conjunctive normal form: its number of variables n, the maximum clause size k, the number of clauses m, the clause density $\delta := \frac{m}{n}$, the number of literals L, and the maximal number of clauses a variable participates in. For instance, for fixed k, there are bounds $O^*(c_k^n)$ where $c_k < 2$ [20]. Yamamoto [56] showed that **SAT** can be solved in $O^*(2^{0.3033m})$ time (independent of k) . This hence implies a better than $O^*(2^n)$ runtime for clause densities $\delta < 3.297$, regardless of k. While for large k and large δ all known bounds converge to $O^*(2^n)$, for small k and arbitrary δ or vice-versa, better bounds are possible.

In this paper we will study the problem **#SAT**, which asks *how many* solutions a Boolean formula has. Hence, this is not a decision problem, but a counting problem. It is complete for the complexity class **#P**, which is the 'counting version' of **NP**. **#SAT** is believed to be a significantly harder problem than **SAT**. For instance, the entire polynomial hierarchy is contained in $\mathbf{P}^{\mathbf{\#SAT}}$ [44]. As it is **#P**-complete, it has applications in a variety of areas. For instance, tensor-network contraction (when suitably formalized) is in **#P** [17]. **#SAT** also has applications in the field of artificial intelligence, where it is usually referred to as *model counting* [7, Chapter 20]. Note that while for **SAT** the problem only becomes hard for $k \geq 3$, for **#SAT**, the problem is already hard for $k = 2$, as **#2SAT** is **#P**-complete [46]. In fact, previously in [37], we used the ZH-calculus to provide a proof of this and other reductions between related counting problems. In this work, we make use of similar techniques in a different direction to give improved upper bounds for **#SAT**.

We can phrase the known upper bounds to **#SAT** in terms of n, k, m and δ, although for **#SAT** much less is known. Similarly to **SAT**, good bounds are known for small k — $O^*(1.2377^n)$ for $k = 2$ [51], and $O^*(1.6423^n)$ for $k = 3$ [35]. There are also bounds known for arbitrary (but fixed) k in the worst-case setting [20] and the average-case setting [55]. Unlike **SAT**, bounds in terms of m are not known for **#SAT** independent of k, but only for $k = 2$, $O^*(1.1740^m)$ [52], and $k = 3$, $O^*(1.4142^m)$ [58].

In this paper we establish, to the best of our knowledge, the first algorithm for **#SAT** that is better than brute-force for low clause density, independent of the clause size k. Specifically, we prove the following theorem:

Theorem 1. *Given a CNF formula $\phi : \{0,1\}^n \to \{0,1\}$, we can count the number of satisfying assignments*

$$\#(\phi) = \#\{\vec{x} \in \{0,1\}^n \mid \phi(\vec{x}) = 1\}$$

in time $O^*(1.2377^{n+m_{\geq 3}})$, where $m_{\geq 3}$ is the number of clauses of width at least three. In particular, for clause density $\delta < 2.2503$, this gives a better than $O^*(2^n)$ bound, independent of maximal clause size k.

The worst-case bound for our algorithm improves the best-known bound for a variety of different parameters. We summarize this in Tables 1 and 2. Note that these tables also contain our results based on a more fine-grained analysis of **#3SAT** that is presented in Section 4.2, as well as a bound on **#SAT** in terms of literals L that is presented in Section 4.1. Furthermore, assuming the strong exponential time hypothesis, our results indicate that the 'hardest' density of **#SAT** must be some $\delta > 2.2503$. As far as we aware this is the first known bound on where the hardest clause density of **#SAT** lies.

While the bound of Theorem 1 is only effective at low densities, this is sufficient for many real-world use cases – in particular, for the unweighted **#SAT** instances from the Model Counting Competition 2020 [24], we found that we improve on the previous best worst-case bound for 88% of instances, and the average-case bound for 45% of instances, assuming the constant factors of both algorithms are equal (we take k to be the 90th percentile of clause widths to avoid few large clauses biasing the result in our

Problem	Previous Best	New Bound	Relevant Region	Algorithm
#kSAT for $k > 3$	$O^*(c_k^n)$, $c_k \to 2$ [20]	$O^*(1.2377^{n+m})$	$\delta < 2.2503$ (as $k \to \infty$)	[51]
#kSAT for $k > 3$	$O^*(c_k^n)$, $c_k \to 2$ [20]	$O^*(1.1740^L)$	$\frac{L}{n} < 4.3209$ (as $k \to \infty$)	[52]
#3SAT	$O^*(1.6423^n)$ [35]	$O^*(1.6350^n)$	$1.2577 < \delta \leq \frac{7}{3}$	[51] [35]

Table 1: The different bounds obtained in this paper, along with the corresponding best previous bounds, and the underlying algorithm on which they are based. The given relevant regions are the intervals of formula parameters where our bounds are valid and improve on the previous bound.

Improvement on	k	2	3	4	5	6	7	8	9
Average-Case	δ	–	–	–	–	–	< 0.968	< 1.106	< 1.207
Worst-Case	δ	–	$< 2.333^*$	< 2.077	< 2.170	< 2.212	< 2.231	< 2.241	< 2.246

Table 2: The maximum densities δ at which our upper bound improves on other bounds, as dependent on k. We include average [55] and worst-case bounds [20], both of which converge to $\delta = 2.2503$ as k increases. Entries marked with '–' are where our bound is always worse. The entry marked '$*$' uses the alternate bound presented in Section 4.2 and only improves on the previous best when also $\delta > 1.2577$.

favor). However, it is important to note that practical **#SAT** solvers perform much better than predicted by any of these upper bounds.

Our results are based on two observations. The first observation is that the standard CDP (Counting Davis-Putnam) algorithm [8] for solving **#SAT** can actually solve a more general problem that we dub **#SAT**$_\pm$. In this problem, variables x_i are labeled by a $\phi_i = \pm 1$ phase that determines whether a solution to f should be added or subtracted to the total, as determined by $\prod_{\vec{x}} \phi_i^{x_i}$. The second observation is that we can translate an arbitrary **#SAT** instance into a **#2SAT**$_\pm$ instance. This removes the dependence on maximal clause size k from our problem, and means we can use the known upper bounds to **#2SAT** that apply directly to the problem **#2SAT**$_\pm$ as well.

We found this last observation by writing a **#SAT** instance as a ZH-diagram as described by de Beaudrap *et al.* [19]. We extend their methods by relaxing the conditions on the types of diagrams we consider, which shows that ZH-diagrams also naturally represent **#SAT**$_\pm$ instances. The translation from a **#SAT** instance into a **#2SAT**$_\pm$ instance then follows from a known rewrite rule of the related ZXΔ-calculus [48]. We also find that **#SAT**$_\pm$ is in fact complete for the complexity class **GapP** [23], which is **#P** closed under negation.

In Section 2 we recall the definition of the ZH-calculus, how to encode **#SAT** instances as ZH-diagrams, and how the CDP algorithm for solving **#SAT** works. Then in Section 3 we present our main results: we show how to interpret CDP graphically inside the ZH-calculus, and we find a graphical reduction from **#SAT** to **#2SAT**$_\pm$. We end with a complexity analysis of combining this reduction with the CDP algorithm, modified to work for **#2SAT**$_\pm$. In Section 4 we study some variations on our algorithm: Section 4.1 presents a new bound of $O^*(1.1740^L)$ for **#SAT** that is in terms of number of literals; Section 4.2 gives a modified algorithm for **#3SAT** that is better for certain densities; and Section 4.3 presents the problem of **#SAT** where variables are labeled by arbitrary complex numbers, which we conjecture might be helpful for future improvements. In particular, in Appendix B we apply this technique to calculating amplitudes of quantum circuits, and show an upper bound in terms of the number of gates. We

end with some concluding remarks in Section 5.

2 Preliminaries

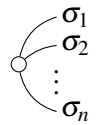
We say that a Boolean formula $\phi : \mathbb{B}^n \to \mathbb{B}$ is in conjunctive normal form if we have

$$\phi(x_1,\ldots,x_n) = \bigwedge_{i=1}^{m} (c_{i1} \vee c_{i2} \vee \cdots \vee c_{ik_i}) \tag{1}$$

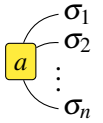
where each c_{ij} is x_l or $\neg x_l$ for some l. We say that ϕ has n variables, m clauses, maximum clause width $k = \max_i\{k_i\}$, density $\delta = \frac{m}{n}$ and number of literals $L = \sum_i k_i$.

2.1 The ZH-calculus

We use the ZH-calculus, a graphical language designed for reasoning about quantum computations [3]. Here we only recall the definition of the generators; see [54, Section 8] for a more detailed overview. The calculus is defined by rewrites on ZH-diagram, which are composed of two generators, called the Z-spider and the H-box. These are given by



$$= \quad \delta_{\sigma_1\sigma_2\ldots\sigma_n} \quad = \quad \begin{cases} 1 & \sigma_1 = \sigma_2 = \cdots = \sigma_n \\ 0 & \text{otherwise} \end{cases}$$



$$= \quad 1 + (a-1)\delta_{1\sigma_1\sigma_2\ldots\sigma_n} \quad = \quad \begin{cases} a & 1 = \sigma_1 = \sigma_2 = \cdots = \sigma_n \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

along with their interpretation as tensors. Each H-box is labeled by a constant $a \in \mathbb{C}$, with unlabeled H-boxes corresponding to $a = -1$. Diagrams composed from these can be interpreted as tensor networks: see [39, Section 4.1] for an introduction. In particular, parallel composition of generators corresponds to composing the tensors via tensor product, and connecting wires between generators corresponds to contracting the shared index. The tensors are symmetric over their indices, which implies that *only connectivity matters* – diagrams with the same topology represent the same tensor network.

We also have the following derived generators, defined as

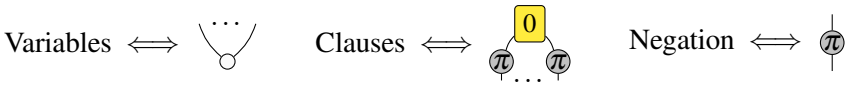
(3)

where the first is an extension of the Z-spider with $\alpha \in \mathbb{C}$ a phase, and the second is the X-spider. In this work, we consider any ZH-diagram to represent its underlying tensor network - hence, if there are n open wires, this represents a tensor with n indices. In particular, if there are no open wires we call this a scalar diagram, since this represents a scalar.

The ZH-calculus is equipped with a set of rewrite rules, which are shown in Appendix A. They are sound with respect to the tensor representation of ZH-diagrams, and also complete: for any pair of diagrams with identical tensor representations, there is a proof that they are equal using these rules.

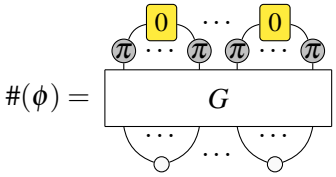
2.2 #SAT instances as ZH-diagrams

In previous work, de Beaudrap *et al.* [19] gave a translation from **#SAT** instances into ZH-diagrams, which we adopt here. In particular, a CNF Boolean formula ϕ is mapped into a ZH-diagram by translating clauses to zero-labeled H-boxes, variables to Z-spiders and negation to X-spiders:



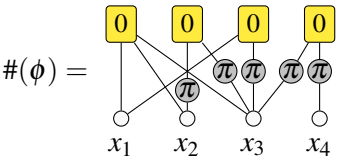
(4)

These are combined by connecting the variables to the clauses that they occur in. For unnegated literals, they are connected with a wire. For negated literals, they are connected via an X-spider. This yields the following



(5)

where G defines the connections. The scalar value of this diagram is exactly equal to the number of satisfying assignments to ϕ. This can be simplified by canceling all adjacent X-spiders, leaving an X-spider connected between a variable and a clause if and only if that variable is contained in that clause *unnegated*. For example, for $\phi(x_1,x_2,x_3,x_4) = (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee x_3) \wedge (x_3 \vee x_4)$, we have:



(6)

Therefore, a diagram derived this way has m H-boxes, n Z-spiders, at most L X-spiders, and the maximum clause size k corresponds to the maximum degree of any H-box. Note that a complete graphical calculus for **SAT** was introduced recently in [27]. However, the semantics of their diagrams directly correspond to a matrix of True or False values, and hence cannot represent **#SAT** instances directly. In this sense it is similar to the modified ZH-diagrams of [19] where they set $2 = 1$ and consider diagrams over the Boolean semi-ring.

2.3 The CDP algorithm for #SAT

The Counting David-Putnam (CDP) algorithm is an algorithm for solving **#SAT** that was first introduced in 1999 [8], as an extension of the DPLL algorithm [18] for **SAT** solving. It is effectively an optimized depth-first search over all possible assignments of variables in a Boolean formula. The algorithm is based on the following two rules:

1. *Unit Propagation:* The following rewrite holds for any clauses A_i and B_i not containing some literal x:

$$x \wedge (A_1 \vee x) \wedge \cdots \wedge (A_n \vee x) \wedge (B_1 \vee \neg x) \wedge \cdots \wedge (B_m \vee \neg x) = B_1 \wedge \cdots \wedge B_m \tag{7}$$

2. *Variable Branching:* For any variable x appearing in a formula f, the number of satisfying assignments of f is the sum of the numbers of satisfying assignments for

$$f_1 = f \wedge x \qquad \text{and} \qquad f_2 = f \wedge \neg x \tag{8}$$

Algorithm 1: The CDP [8] algorithm for solving **#SAT**.

Input: A CNF formula f with n variables and m clauses.
Output: The value of $\#\{\vec{x} \in \{0,1\}^n \mid f(\vec{x}) = 1\}$.

if *f contains the clauses x and $\neg x$ for some variable x* **then**
  return 0
end
if *f has no clauses remaining* **then**
  return 2^n
end
Apply unit propagation to f until it is no longer possible.
Pick a variable x that occurs in f and generate $f_1 = f \wedge x$ and $f_2 = f \wedge \neg x$.
return $\mathrm{CDP}(f_1) + \mathrm{CDP}(f_2)$

since in each such assignment, either x or $\neg x$.

The CDP algorithm, given in Algorithm 1, applies these two rules recursively until either a contradiction occurs and so the formula is unsatisfiable, or the formula has no clauses remaining, in which case the number of satisfying assignments is 2^n. Clearly, at each recursive step, the formulas to be considered have fewer variables than at the previous steps, so this procedure terminates.

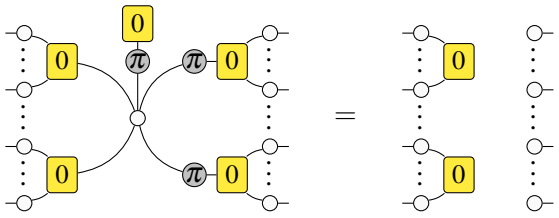
This works better in practice than naively checking every assignment because unit propagation can eliminate many variables, thus removing whole branches from the computation tree. Additionally, the substitution of the assignment into the formula is done incrementally via unit propagation, rather than repeated for every assignment.

3 Results

3.1 Interpreting CDP diagrammatically

To interpret CDP diagrammatically we will first see how the two rules apply to the ZH-diagrams for **#SAT** instances detailed in Section 2.2. The proofs of these lemmas and all the following results are postponed to Appendix E.

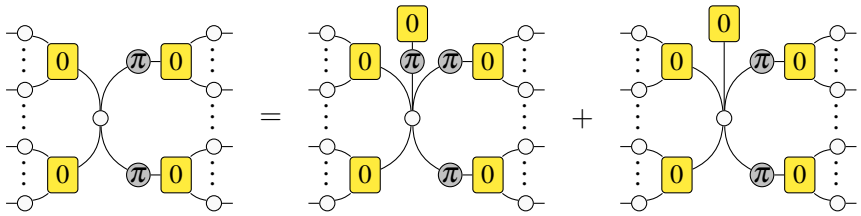
Lemma 1. *The following diagrammatic equivalent to the unit propagation rule holds (without loss of generality, we assume the literal to be propagated is not negated):*



(9)

It can be read as follows - on the left-hand side, the zero H-box with one leg is the clause with a single non-negated literal x, the H-boxes on the left represent the clauses $B_i \vee \neg x$ while the H-boxes on the right represent the clauses $A_i \vee x$. On the right-hand side, we see that the clauses B_i remain, while the clauses $A_i \vee x$ have been removed entirely. Because the variable x is now no longer mentioned, the Z-spider representing it is also removed.

Lemma 2. *The following diagrammatic equivalent to the variable branching rule holds:*



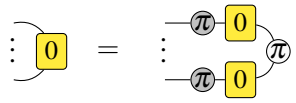
On the left-hand side, we have a variable connected to arbitrary clauses, whereas on the right-hand side we have two terms, each with a clause of one literal introduced onto the variable.

Modifications of the CDP algorithm are used extensively in practice, with some of the best solvers like sharpSAT [43] and Cachet [42] making use of this technique. All the best-known theoretical upper bounds on runtime are based on careful analysis of this algorithm. Note that we left the choice of variable to branch on unspecified - choosing this wisely is crucial to obtaining good runtimes, as we will see in the next section.

3.2 Reduction from #SAT to #2SAT$_{\pm}$

Valiant [46] showed that **#2SAT** is **#P**-complete, which implies that there is a polynomial-time Turing reduction from **#SAT** to **#2SAT**. Since good upper bounds are known for **#2SAT**, reducing **#SAT** to **#2SAT** is one strategy to obtain better than brute-force bounds that are independent of k. However, the polynomial-time reduction guaranteed by **#P**-completeness maps instances with n variables and m clauses to instances with $O(nm)$ variables, which destroys any advantage we could have gained from faster **#2SAT** algorithms. Instead we present the following linear-time reduction to a weighted variant of **#SAT**:

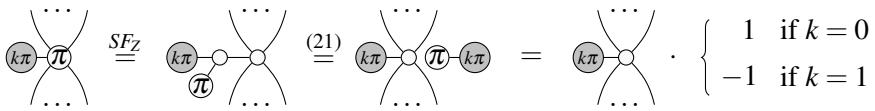
Lemma 3. *The following diagrammatic equivalence holds:*



(10)

This directly generalizes the BW *axiom of the* ΔZX*-calculus [48].*

This lemma allows us to remove clauses with degree greater than two, at the cost of introducing extra Z-spiders. If we applied this to every clause in a **#SAT** diagram with n variables and m clauses, we would have a diagram with $n+m$ spiders and furthermore, with the exception of π-phases on m of these spiders, this diagram would represent a **#2SAT** instance. Therefore, we will relax our definition of **#SAT** to permit π-phases appearing on the Z-spiders (i.e. on the variables). Note that we have the following



(11)

which implies that these diagrams are the same as **#SAT** instances, but the sign is flipped whenever a variable corresponding to a Z-spider with a π-phase is assigned to be true. The overall sign of the diagram for a particular assignment of variables is given by the parity of the assignment of such variables. We can extend the **#SAT** problem as follows to handle this natively.

Definition 1. *The problem* ***#SAT***$_\pm$ *is defined as follows. Given a CNF formula* $f(x_1,\ldots,x_n)$ *with* n *variables and* m *clauses, and a set* $N \subseteq \{1,\ldots,n\}$*, compute the quantity*

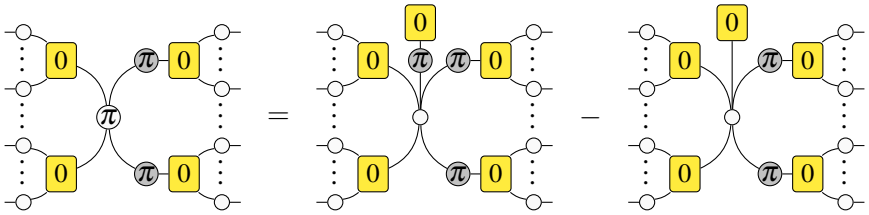
$$\textbf{\#SAT}_\pm(f,N) = \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f(\vec{x}) \;-\; \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f(\vec{x}) \tag{12}$$

where a vector $\vec{x}\in\mathbb{B}^n$ *has even or odd* N*-parity if* $\bigoplus_{i\in N} x_i = 0$ *or* 1*, respectively.*

Theorem 2. ***#SAT***$_\pm$ *is GapP-complete*

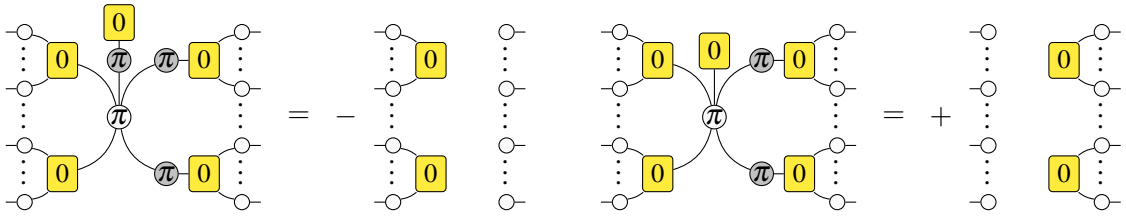
Since **#SAT**$_\pm$ is in GapP, which is strictly harder than #P (for instance, GapP is the closure of #P under subtraction [23]), it may seem that an upper bound for this problem is guaranteed to be worse. However, this is not the case, as the following diagrammatic arguments show that the DPLL algorithm can be easily adapted to handle the sign change as Algorithm 2.

Lemma 4. *The following diagrammatic equivalent to the variable branching rule holds for variables with* π*-phases:*



(13)

Lemma 5. *The following diagrammatic equivalent to the unit propagation rule holds for variables with* π*-phases:*



(14)

With a smart choice of the variables to branch on, the worst-case runtime of Algorithm 2 can be bounded in exactly the same way as the regular CDP algorithm and its variants. This is because the bounds we consider here (e.g [25, 35, 51, 52]) are obtained from two principles:

- Exploiting the fact that the conjunction of unrelated **#SAT** instances combine multiplicatively. Diagrammatically, this corresponds to the parallel composition of scalar diagrams being defined as scalar multiplication. This remains true for the case when phases are present on Z-spiders.
- By analyzing which classes of sub-formulas can occur in any instance, and how they are affected by the unit propagation and branching rules. Diagrammatically, this corresponds to a case analysis on the possible subgraphs of the diagram, ignoring scalar factors. Since unit propagation and branching are unaffected (except for scalar factors) by the presence of phases on the Z-spiders (see Lemmas 4 and 5), this also applies directly in this case.

Algorithm 2: The $\mathrm{CDP}_\pm$ algorithm solving **#SAT**$_\pm$.

Input: A CNF formula f with n variables and m clauses, and a set of variables $N \subseteq \{1,\dots,n\}$.
Output: The value of **#SAT**$_\pm(f,N)$.

if *f contains an empty clause* **then**
  return 0
else if *f has no clauses* **then**
  return 2^n
else
  Pick $i \in \{1,\dots,n\}$ according to some strategy.
  $f_1 \leftarrow$ Unit-Propagate$(f \wedge \neg x_i)$
  $f_2 \leftarrow$ Unit-Propagate$(f \wedge x_i)$
  if $i \in N$ **then**
    return $\mathrm{CDP}_\pm(f_1,N) - \mathrm{CDP}_\pm(f_2,N)$
  else
    return $\mathrm{CDP}_\pm(f_1,N) + \mathrm{CDP}_\pm(f_2,N)$
  end
end

Algorithm 3: The $\mathrm{CDP}^{\to 2}_\pm$ algorithm for **#SAT**.

Input: A CNF formula f with n variables and m clauses.
Output: The value of **#SAT**(f).

Generate f' by applying Lemma 3 to every clause in f of width at least three.
Set N to be all the variables labeled with π-phases.
return $\mathrm{CDP}_\pm(f',N)$

In particular, these bounds do not depend on the specific scalar factors of each diagram, or the way that separate diagrams generated by branching are recombined. This means that the $O^*(1.2377^{\text{variables}})$ bound of Wahlström [51] can be adapted directly. By applying Lemma 3 to any **#SAT** diagram and then applying $\mathrm{CDP}_\pm$ to the resulting diagram directly, we can evaluate **#SAT** instances in time $O^*(1.2377^{n+m}) = O^*(2^{0.3068n+0.3068m})$, which is certainly better than the bound given by decomposing into a sum of diagrams. We will refer to this method, given in Algorithm 3, as $\mathrm{CDP}^{\to 2}_\pm$.

Theorem 3 (Restatement of Theorem 1)**.** *Given a CNF formula $f : \{0,1\}^n \to \{0,1\}$, we can count the number of satisfying assignments $\#\{\vec{x} \in \{0,1\}^n \mid f(\vec{x}) = 1\}$ in time $O^*(1.2377^{n+m_{\geq 3}})$, where $m_{\geq 3}$ is the number of clauses of width at least three.*

Proof. Apply the algorithm $\mathrm{CDP}^{\to 2}_\pm$ to f using Wahlström's [51] $O^*(1.2377^{\text{variables}})$ algorithm for solving **#2SAT**. Then $m_{\geq 3}$ new (negative) variables will be created by applying Lemma 3, so the overall runtime is given by $O^*(1.2377^{n+m_{\geq 3}})$. □

It remains to ask, when is $\mathrm{CDP}^{\to 2}_\pm$ actually useful? First, note that if only positive variables are picked for branching, the action of $\mathrm{CDP}_\pm$ on a translated **#SAT** diagram is *exactly* the same as the action of regular CDP on the original diagram. Therefore, we would only expect gains when decomposing some of the negative variables (i.e clauses), and thus it is natural to suspect that this bound will only be useful for instances with few clauses.

3.3 Complexity analysis

For instances with a fixed maximum density δ_{max}, and assuming the worst-case of $m_{\geq 3} = m$, we have that $m \leq n\delta_{max}$, and so the runtime of $\text{CDP}^{\to 2}_{\pm}$ is bounded by $O^*(2^{0.3068(1+\delta_{max})n})$. Firstly, we can see that this is better than the naive $O^*(2^n)$ whenever $\delta_{max} < 2.2503$. Since this is independent of k, it means that for any $\delta_{max} < 2.2503$ and sufficiently large k, this beats both the worst-case bound of Dubois [20] and the average-case bound of Williams [55].

Concretely, $\text{CDP}^{\to 2}_{\pm}$ is better than the average-case bounds of Williams [55] whenever $\delta_{max} < 1.217$ and $k \geq 6$, and better than the worst-case bounds of Dubois [20] whenever $k \geq 3$ and $\delta_{max} < 1.858$. The exact bounds for each k are given in Table 2. Clearly, $\text{CDP}^{\to 2}_{\pm}$ offers no improvement on **#2SAT**, but it is also not directly applicable to **#3SAT** - when $\delta < 1.6$, the $O^*(1.4142^m)$ bound of Zhou et al [58] is sharper, and when $\delta \geq 1.6$ the $O^*(1.6423^n)$ bound of Kutzkov [35] is sharper.

For **SAT**, it has been shown that there is a phase transition in the satisfiability of a random formula as δ passes some threshold. Instances with densities near this threshold are known to be hard to solve, and it is known that this threshold scales exponentially with k [1]. However, no bound is known for the equivalent 'hardest' density in **#SAT**. Assuming SETH, our result indicates that the 'hardest' density of **#SAT** must be some $\delta > 2.2503$, since otherwise **#SAT** (and hence **SAT**) could be solved in time better than $O^*(2^n)$.

3.4 A non-diagrammatic argument for $\text{CDP}^{\to 2}_{\pm}$

While we originally found this reduction and algorithm using the ZH-calculus and prefer its diagrammatic presentation, in Appendix C we present a self-contained argument for the reduction from **#SAT** to $\textbf{#2SAT}_{\pm}$ without any diagrams that may be more intuitive to readers unfamiliar with graphical calculi.

4 Variations on the main result

4.1 Bounding #SAT in terms of literals

Note that $\text{CDP}^{\to 2}_{\pm}$ maps a **#SAT** instance of m clauses to a **#2SAT** instance of at most L clauses (with negative variables), where L is the number of literals in the original instance. Therefore, applying the upper bound of $O^*(1.1740^m)$ on **#2SAT** found by Wang and Gu [52], we can bound the runtime of $\text{CDP}^{\to 2}_{\pm}$ in terms of literals as $O^*(1.1740^L)$. This implies a better than $O^*(2^n)$ runtime whenever the average degree $\hat{d} = \frac{L}{n}$ of variables in an instance satisfies $\hat{d} < 4.3209$, or the maximal number of clauses d a variable participates in is $d \leq 4$. As far as we are aware this is the first bound of this type for **#SAT** with unrestricted clause width, but similar bounds are known for **SAT** - e.g the $O^*(1.0646^L)$ of Peng and Xiao [40]. If k is small, then better bounds for **#SAT** follow trivially from bounds in terms of m - e.g for $k = 2$, $m \leq \frac{L}{2}$, so [52] implies $O^*(1.0835^L)$.

4.2 An algorithm for low-density #3SAT instances

In the previous section, we noted that $\text{CDP}^{\to 2}_{\pm}$ can't beat existing bounds for **#3SAT** on its own. This is because we have to assume that $m = m_{\geq 3}$ in the worst-case. However, we can use a technique introduced by Kutzkov [35] to take advantage of the extra structure afforded by **#3SAT** and introduce extra branching steps which allow us to assume that $m_{\geq 3} < m$.

Assume we have some **#3SAT** instance f. Every time we branch on a variable x that occurs in d 3-clauses in f, in both branches at least d 3-clauses are eliminated (since each clause will be totally

Algorithm 4: The algorithm $\mathrm{CDP}_{\pm}^{3\to 2}$ for solving **#3SAT**.

Input: A CNF formula f with n variables and m clauses.
Output: The value of $\#\mathbf{SAT}(f)$.

```
if δ3 > 2/3 then
    Pick x in f with maximal 3-degree.
    f1 ← Unit-Propagate(f ∧ ¬x_i)
    f2 ← Unit-Propagate(f ∧ x_i)
    return CDP^{3→2}_±(f1) + CDP^{3→2}_±(f2)
else
    return CDP^{→2}_±(f)
end
```

removed in one branch and become a 2-clause in the other). If we know that the density of 3-clauses in f is δ_3, then the average number of 3-clauses a variable is connected to (its average 3-degree) is $3\delta_3$. Therefore, whenever $d-1 < 3\delta_3 \leq d$, there must exist a variable with 3-degree at least d, and so by branching on the variable with the highest 3-degree, we remove at least d 3-clauses.

Following Kutzkov [35], suppose x is the number of variables needed to reduce δ_3 to at most $\frac{d-1}{3}$ by repeatedly branching on the variable with the highest 3-degree. Then we need

$$\frac{d}{3}n - dx \leq \frac{d-1}{3}(n-x) \tag{15}$$

thus $x \leq \frac{n}{2d+1}$, so in the limit of large n, we have $x = \frac{n}{2d+1}$ in the worst case, and $n - \frac{n}{2d+1}$ variables remain unassigned. Therefore, the number of variables we need to branch on to reduce δ_3 to at most $\frac{2}{3}$ is:

$$n_{2/3} = n - n\prod_{i=3}^{d}\left(1 - \frac{1}{2i+1}\right) \tag{16}$$

Since $m_{\geq 3} = \delta_3 n$, after performing this branching on $n_{2/3}$ variables, we will have $2^{n_{2/3}}$ instances, each with $m_{\geq 3} \leq \frac{2}{3}(n - n_{2/3})$, so these can be evaluated with $\mathrm{CDP}_{\pm}^{\to 2}$. This strategy, formalized as Algorithm 4, therefore has an overall time bound of:

$$O^*(2^{n_{2/3}})O^*(1.2377^{(n-n_{2/3})+m_{\geq 3}}) \leq O^*(2^{n_{2/3}}1.2377^{(n-n_{2/3})(1+\frac{2}{3})}) = O^*(2^{0.5128n+0.4872n_{2/3}}) \tag{17}$$

In order to calculate the running-time bound for a given maximum δ_3, we can plug $d = \lceil 3\delta_3 \rceil$ into Equation (16) to calculate $n_{2/3}$ as a fraction of n. For example, if $\delta_3 < \frac{5}{3}$ then $d = 5$ and $n_{2/3} = 0.3074n$, yielding a time of

$$O^*(2^{(0.5128+0.4872\cdot 0.3074)n}) = O^*(1.5829^n).$$

Suppose then that $\delta = \delta_3$ (i.e the worst-case), then this bound is better than the bound of Zhou [58] whenever $\delta > 1.2577$ (i.e $d \geq 4$ but not $d = 3$, comparing the $O^*(1.5463^n)$ complexity for $d = 4$ to Zhou's $O^*(1.4142^m)$ to find the exact cutoff point) and the $O^*(1.6423^n)$ bound of Kutzkov [35] whenever $\delta \leq \frac{7}{3}$ (i.e for $d \leq 7$), yielding complexities of $O^*(1.5463^n)$ ($d = 4$) to $O^*(1.6350^n)$ ($d = 7$) respectively. It is possible this bound could be extended to a yield an improved bound on general **#3SAT** using a case analysis similar to Kutzkov's, but this is quite complicated so we postpone exploring this to future work.

4.3 Solving #SAT with arbitrary phases

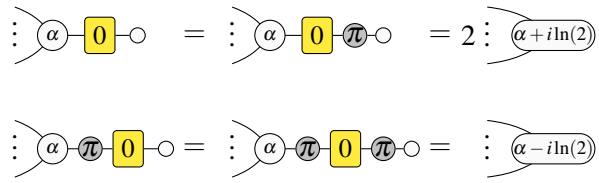
While allowing π-phases on variables has allowed us to find a simple reduction from **#SAT** to **#2SAT**$_\pm$ which can be solved with CDP$_\pm$, the CDP algorithm easily extends to arbitrary phases in the same way. Indeed let us define a generalization of the **#SAT** problem, **#SAT**$_{\mathscr{A}}$ - this is exactly the weighted model counting problem where the weights are restricted to $\mathscr{A}$.

Definition 2. *The problem **#SAT**$_{\mathscr{A}}$ for $\mathscr{A} \subseteq \mathbb{C}\setminus\{0\}$ is defined as follows. Given a CNF formula $f(x_1,\dots)$ with n variables and m clauses and a vector $A \in \mathscr{A}^n$, compute the quantity:*

$$\textbf{\#SAT}_{\mathscr{A}}(f,A) = \sum_{\vec{x}\in\mathbb{B}^n} \left(\prod_{i=1}^{n} A_i^{x_i}\right) f(\vec{x}) \tag{18}$$

It is easy to see then that **#SAT** = **#SAT**$_{\{1\}}$, and **#SAT**$_\pm$ = **#SAT**$_{\{1,-1\}}$: let $A_j = -1$ if $j \in N$ and $A_j = 1$ otherwise. A straightforward adaptation of the CDP algorithm can solve **#SAT**$_{\mathscr{A}}$; see Algorithm 5.

The complex numbers in $\mathscr{A}$ on the variables of an instance can be easily represented in the ZH-calculus, by generalizing Eq. (3). The variable branching and unit propagation rules then generalize from Eq. (22). The advantage of working with **#SAT**$_{\mathscr{A}}$ is that by expanding $\mathscr{A}$, we are afforded additional rewriting rules on the corresponding ZH-diagrams. Moving from $\{1\}$ to $\{1,-1\}$ allowed us to rewrite arbitrary arity zero-labeled H-boxes into arity two H-boxes. Further expanding this to $\{\frac{k}{2} \mid k \in \mathbb{Z}\}$ allows us the following rule, removing (up to a scalar) any variables that only occur once in a formula:



(19)

This is a limited form of pure-literal elimination, a rewrite rule that is usually only valid in **SAT** and not **#SAT**. Applying this simplification to the formula recursively (after unit propagation in the CDP algorithm, i.e between lines 7 and 8 of Algorithm 1), we may assume that every variable has degree at least two. This would allow improvement on early bounds such as [20], while being much simpler. Therefore, an interesting avenue for further research would be investigating how this approach of weighting variables could be used to simplify **#SAT** instances or find upper bounds on runtime. Another example is given in Appendix B where we show how this leads to an algorithm for simulating quantum circuits with runtime in terms of the total number of gates.

5 Conclusion

In this paper, we used the ZH-calculus to study the **#SAT** problem and produced an upper bound which does not depend on the clause width k. We believe bounds of this kind were previously only known for the decision variant **SAT** [56]. The bound is less than $O^*(2^n)$ whenever the clause density $\delta = \frac{n}{m}$ is smaller than 2.2503, suggesting that the 'hardest' density of **#SAT** problems must be some $\delta > 2.2503$, assuming the strong exponential time hypothesis. We found these bounds by rephrasing the **#SAT** problem in terms of ZH-diagrams, and generalizing known rewrite rules to give a reduction from **#SAT** to **#2SAT**$_\pm$, a weighted variant of **#SAT** that can be solved with Wahlström's [51] variant of the CDP algorithm [8]. Using a more involved analysis and algorithm we also improved on the upper bound for **#3SAT** for $1.2577 < \delta < \frac{7}{3}$. In addition, using a result of Wang and Gu [52], we produced an explicit bound of

Algorithm 5: The $\mathrm{CDP}_{\mathscr{A}}$ algorithm for solving $\#\mathbf{SAT}_{\mathscr{A}}$.

Input: A CNF formula f with n variables and m clauses, and $A \in \mathscr{A}^n$.
Output: The value of $\#\mathbf{SAT}_{\mathscr{A}}(f,A)$.

```
if f contains an empty clause then
    return 0
else if f has no clauses then
    return 2^n
else
    Pick i ∈ {1,...,n} according to some strategy.
    f_1 ← Unit-Propagate(f ∧ ¬x_i)
    f_2 ← Unit-Propagate(f ∧ x_i)
    return CDP_A(f_1,A) + A_i CDP_A(f_2,A)
end
```

$O^*(1.1740^L)$ for **#SAT** in terms of the number of literals L, to our knowledge the first such non-trivial bound for **#SAT**. A summary of all the bounds obtained in this paper is presented in Table 1. We suggest extending this technique of reducing **#SAT** to weighted **#SAT** as an avenue of future research.

Our results show that graphical calculi can lead to concrete algorithmic improvements in areas where significant research has already been done, even when originally intended for a different domain like quantum computing. An interesting question then is in which other domains we can make improvements by framing the problem using graphical reasoning.

Acknowledgments

We thank Matty Hoban and the anonymous QPL reviewers for helpful feedback. The majority of this work was done while TL was a student at the University of Oxford, and the main result is also presented in an MSc thesis [36]. JvdW acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101018390.

References

[1] Dimitris Achlioptas, Assaf Naor & Yuval Peres (2005): *Rigorous Location of Phase Transitions in Hard Optimization Problems*. Nature 435(7043), pp. 759–764, doi:10.1038/nature03602.

[2] Miriam Backens (2014): *The ZX-calculus is complete for stabilizer quantum mechanics*. New Journal of Physics 16(9), p. 093021, doi:10.1088/1367-2630/16/9/093021.

[3] Miriam Backens & Aleks Kissinger (2019): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*. Electronic Proceedings in Theoretical Computer Science 287, pp. 23–42, doi:10.4204/EPTCS.287.2. arXiv:1805.02175.

[4] Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering & Sal Wolffs (2021): *Completeness of the ZH-calculus*, doi:10.48550/arXiv.2103.06610. arXiv:2103.06610.

[5] Miriam Backens, Hector Miller-Bakewell, Giovanni de Felice, Leo Lobski & John van de Wetering (2021): *There and back again: A circuit extraction tale*. Quantum 5, p. 421, doi:10.22331/q-2021-03-25-421.

[6] Niel de Beaudrap, Xiaoning Bian & Quanlong Wang (2020): *Techniques to Reduce $\pi/4$-Parity-Phase Circuits, Motivated by the ZX Calculus*. In Bob Coecke & Matthew Leifer, editors: Proceedings 16th International Conference on Quantum Physics and Logic, Chapman University, Orange, CA, USA., 10-14 June

2019, *Electronic Proceedings in Theoretical Computer Science* 318, Open Publishing Association, pp. 131–149, doi:10.4204/EPTCS.318.9.

[7] A. Biere, M. Heule & H. van Maaren (2009): *Handbook of Satisfiability*. IOS Press, Incorporated, doi:10.3233/FAIA336.

[8] E. Birnbaum & E. L. Lozinskii (1999): *The Good Old Davis-Putnam Procedure Helps Counting Models*. *Journal of Artificial Intelligence Research* 10, pp. 457–477, doi:10.1613/jair.601. arXiv:1106.0218.

[9] Guillaume Boisseau & Robin Piedeleu (2022): *Graphical Piecewise-Linear Algebra*. In Patricia Bouyer & Lutz Schröder, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 101–119, doi:10.1007/978-3-030-99253-8_6.

[10] Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński & Fabio Zanasi (2019): *Diagrammatic Algebra: From Linear to Concurrent Systems*. *Proc. ACM Program. Lang.* 3(POPL), doi:10.1145/3290338.

[11] Filippo Bonchi, Robin Piedeleu, Pawel Sobociński & Fabio Zanasi (2019): *Graphical Affine Algebra*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–12, doi:10.1109/LICS.2019.8785877.

[12] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2014): *A Categorical Semantics of Signal Flow Graphs*. In Paolo Baldan & Daniele Gorla, editors: *CONCUR 2014 – Concurrency Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 435–450, doi:10.1007/978-3-662-44584-6_30.

[13] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2017): *Interacting Hopf Algebras*. *Journal of Pure and Applied Algebra* 221(1), pp. 144–184, doi:10.1016/j.jpaa.2016.06.002.

[14] Enrique Cervero Martín, Kirill Plekhanov & Michael Lubasch (2022): *Barren plateaus in quantum tensor network optimization*. doi:10.48550/arXiv.2209.00292. arXiv:2209.00292.

[15] Bob Coecke & Ross Duncan (2008): *Interacting quantum observables*. In: *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, doi:10.1007/978-3-540-70583-3_25.

[16] Bob Coecke & Ross Duncan (2011): *Interacting Quantum Observables: Categorical Algebra and Diagrammatics*. *New Journal of Physics* 13(4), p. 043016, doi:10.1088/1367-2630/13/4/043016. arXiv:0906.4725.

[17] Carsten Damm, Markus Holzer & Pierre McKenzie (2002): *The Complexity of Tensor Calculus*. *Computational Complexity* 11(1/2), pp. 54–89, doi:10.1007/s00037-000-0170-4.

[18] Martin Davis, George Logemann & Donald Loveland (1962): *A Machine Program for Theorem-Proving*. *Communications of the ACM* 5(7), pp. 394–397, doi:10.1145/368273.368557.

[19] Niel de Beaudrap, Aleks Kissinger & Konstantinos Meichanetzidis (2021): *Tensor Network Rewriting Strategies for Satisfiability and Counting*. *Electronic Proceedings in Theoretical Computer Science* 340, pp. 46–59, doi:10.4204/EPTCS.340.3. arXiv:2004.06455.

[20] Olivier Dubois (1991): *Counting the Number of Solutions for Instances of Satisfiability*. *Theoretical Computer Science* 81(1), pp. 49–64, doi:10.1016/0304-3975(91)90315-S.

[21] Ross Duncan, Aleks Kissinger, Simon Perdrix & John van de Wetering (2020): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. *Quantum* 4, p. 279, doi:10.22331/q-2020-06-04-279.

[22] Richard D. P. East, Pierre Martin-Dussaud & John van de Wetering (2021): *Spin-networks in the ZX-calculus*. doi:10.48550/arXiv.2111.03114. arXiv:2111.03114.

[23] Stephen A Fenner, Lance J Fortnow & Stuart A Kurtz (1994): *Gap-Definable Counting Classes*. *Journal of Computer and System Sciences* 48(1), pp. 116–148, doi:10.1016/S0022-0000(05)80024-8.

[24] Johannes K. Fichte, Markus Hecher & Florim Hamiti (2020): *The Model Counting Competition 2020*, doi:10.48550/arXiv.2012.01323. arXiv:2012.01323.

[25] Martin Fürer & Shiva Prasad Kasiviswanathan (2007): *Algorithms for Counting 2-Sat Solutions and Colorings with Applications*. In Ming-Yang Kao & Xiang-Yang Li, editors: *Algorithmic Aspects in Information and Management*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 47–57, doi:10.1007/978-3-540-72870-2_5.

[26] Craig Gidney & Austin G. Fowler (2019): *Efficient magic state factories with a catalyzed $|CCZ\rangle$ to $2|T\rangle$ transformation*. Quantum 3, p. 135, doi:10.22331/q-2019-04-30-135.

[27] Tao Gu, Robin Piedeleu & Fabio Zanasi (2022): *A Complete Diagrammatic Calculus for Boolean Satisfiability*. doi:10.48550/arXiv.2211.12629. arXiv:2211.12629.

[28] Amar Hadzihasanovic (2015): *A diagrammatic axiomatisation for qubit entanglement*. In: *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*, IEEE, pp. 573–584, doi:10.1109/LICS.2015.59.

[29] Amar Hadzihasanovic, Kang Feng Ng & Quanlong Wang (2018): *Two Complete Axiomatisations of Pure-state Qubit Quantum Computing*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, ACM, New York, NY, USA, pp. 502–511, doi:10.1145/3209108.3209128.

[30] Michael Hanks, Marta P. Estarellas, William J. Munro & Kae Nemoto (2020): *Effective Compression of Quantum Braided Circuits Aided by ZX-Calculus*. Physical Review X 10, p. 041030, doi:10.1103/PhysRevX.10.041030.

[31] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A Complete Axiomatisation of the ZX-Calculus for Clifford+T Quantum Mechanics*. In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, ACM, New York, NY, USA, pp. 559–568, doi:10.1145/3209108.3209131.

[32] Aleks Kissinger & John van de Wetering (2020): *Reducing the number of non-Clifford gates in quantum circuits*. Physical Review A 102, p. 022406, doi:10.1103/PhysRevA.102.022406.

[33] Aleks Kissinger & John van de Wetering (2022): *Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions*. Quantum Science and Technology 7(4), p. 044001, doi:10.1088/2058-9565/ac5d20.

[34] Aleks Kissinger, John van de Wetering & Renaud Vilmart (2022): *Classical Simulation of Quantum Circuits with Partial and Graphical Stabiliser Decompositions*. In François Le Gall & Tomoyuki Morimae, editors: *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 232, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 5:1–5:13, doi:10.4230/LIPIcs.TQC.2022.5.

[35] Konstantin Kutzkov (2007): *New Upper Bound for the #3-SAT Problem*. Information Processing Letters 105(1), pp. 1–5, doi:10.1016/j.ipl.2007.06.017.

[36] Tuomas Laakkonen (2022): *Graphical Stabilizer Decompositions For Counting Problems*. Master's thesis, University of Oxford. Available at https://www.cs.ox.ac.uk/people/aleks.kissinger/theses/laakkonen-thesis.pdf.

[37] Tuomas Laakkonen, Konstantinos Meichanetzidis & John van de Wetering (2023): *Picturing Counting Reductions with the ZH-Calculus*. Electronic Proceedings in Theoretical Computer Science 384, p. 89–113, doi:10.4204/eptcs.384.6.

[38] Kang Feng Ng & Quanlong Wang (2018): *Completeness of the ZX-calculus for Pure Qubit Clifford+T Quantum Mechanics*, doi:10.48550/arXiv.1801.07993. arXiv:1801.07993.

[39] Román Orús (2014): *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*. Annals of Physics 349, pp. 117–158, doi:10.1016/j.aop.2014.06.013.

[40] Junqiang Peng & Mingyu Xiao (2021): *Further Improvements for SAT in Terms of Formula Length*, doi:10.48550/arXiv.2105.06131. arXiv:2105.06131.

[41] Robin Piedeleu & Fabio Zanasi (2021): *A String Diagrammatic Axiomatisation of Finite-State Automata*. In Stefan Kiefer & Christine Tasson, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 469–489, doi:10.1007/978-3-030-71995-1_24.

[42] Tian Sang, Fahiem Bacchus, Paul Beame, Henry A. Kautz & Toniann Pitassi (2004): *Combining Component Caching and Clause Learning for Effective Model Counting*. In: *SAT 2004 - The Seventh International Conference on Theory and Applications of Satisfiability Testing, 10-13 May 2004, Vancouver, BC, Canada, Online Proceedings*.

[43] Marc Thurley (2006): *sharpSAT – Counting Models with Advanced Component Caching and Implicit BCP*. In Armin Biere & Carla P. Gomes, editors: *Theory and Applications of Satisfiability Testing - SAT 2006*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 424–429, doi:10.1007/11814948_38.

[44] Seinosuke Toda (1991): *PP Is as Hard as the Polynomial-Time Hierarchy*. *SIAM Journal on Computing* 20(5), pp. 865–877, doi:10.1137/0220053.

[45] Alex Townsend-Teague & Konstantinos Meichanetzidis (2021): *Classifying Complexity with the ZX-Calculus: Jones Polynomials and Potts Partition Functions*. doi:10.48550/arXiv.2103.06914. arXiv:2103.06914.

[46] Leslie G. Valiant (1979): *The Complexity of Enumeration and Reliability Problems*. *SIAM Journal on Computing* 8(3), pp. 410–421, doi:10.1137/0208032.

[47] Renaud Vilmart (2019): *A Near-Minimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–10, doi:10.1109/LICS.2019.8785765.

[48] Renaud Vilmart (2019): *A ZX-Calculus with Triangles for Toffoli-Hadamard, Clifford+T, and Beyond*. *Electronic Proceedings in Theoretical Computer Science* 287, pp. 313–344, doi:10.4204/EPTCS.287.18. arXiv:1804.03084.

[49] Renaud Vilmart (2021): *Quantum Multiple-Valued Decision Diagrams in Graphical Calculi*. In Filippo Bonchi & Simon J. Puglisi, editors: *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 202, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 89:1–89:15, doi:10.4230/LIPIcs.MFCS.2021.89.

[50] Renaud Vilmart (2021): *The Structure of Sum-over-Paths, Its Consequences, and Completeness for Clifford*. In Stefan Kiefer & Christine Tasson, editors: *Foundations of Software Science and Computation Structures*, Springer International Publishing, Cham, pp. 531–550, doi:10.1007/978-3-030-71995-1_27.

[51] Magnus Wahlström (2008): *A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances*. In Martin Grohe & Rolf Niedermeier, editors: *Parameterized and Exact Computation*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 202–213, doi:10.1007/978-3-540-79723-4_19.

[52] Honglin Wang & Wenxiang Gu (2013): *The Worst Case Minimized Upper Bound in #2-SAT*. In Wei Lu, Guoqiang Cai, Weibin Liu & Weiwei Xing, editors: *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, Lecture Notes in Electrical Engineering, Springer, Berlin, Heidelberg, pp. 675–682, doi:10.1007/978-3-642-34522-7_72.

[53] Quanlong Wang (2021): *An Algebraic Axiomatisation of ZX-calculus*. In Benoît Valiron, Shane Mansfield, Pablo Arrighi & Prakash Panangaden, editors: *Proceedings 17th International Conference on Quantum Physics and Logic, Paris, France, June 2 - 6, 2020*, *Electronic Proceedings in Theoretical Computer Science* 340, Open Publishing Association, pp. 303–332, doi:10.4204/EPTCS.340.16.

[54] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*. doi:10.48550/arXiv.2012.13966. arXiv:2012.13966.

[55] Ryan Williams (2004): *On Computing K-CNF Formula Properties*. In Enrico Giunchiglia & Armando Tacchella, editors: *Theory and Applications of Satisfiability Testing*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 330–340, doi:10.1007/978-3-540-24605-3_25.

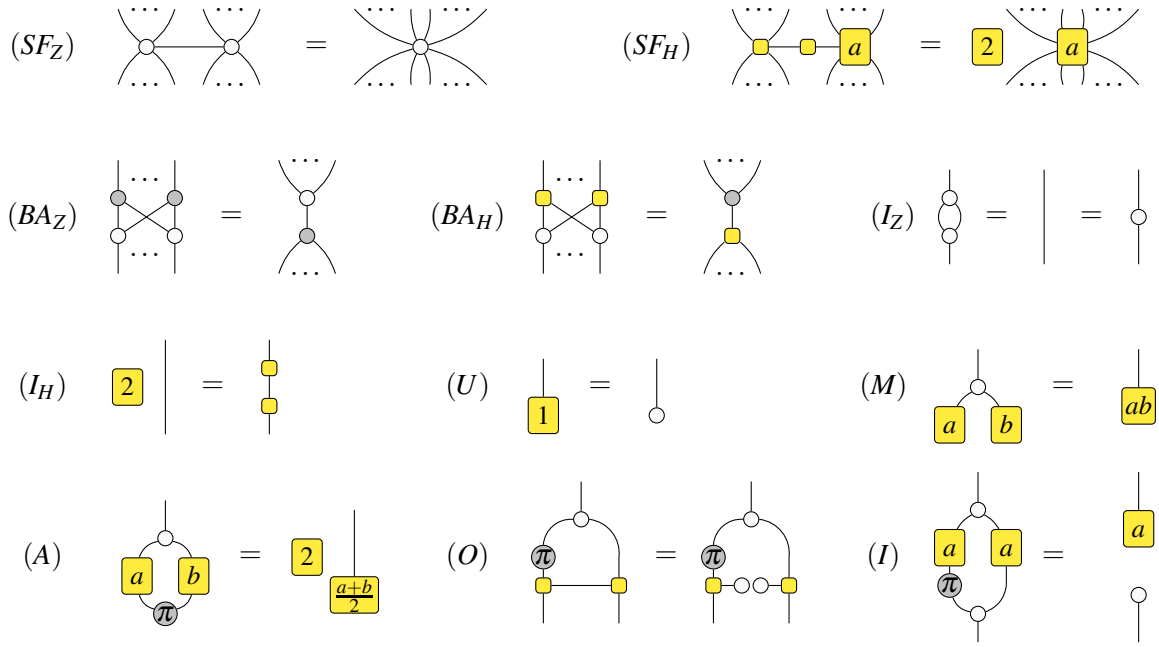
[56] Masaki Yamamoto (2005): *An Improved O(1.234$\hat{m}$)-Time Deterministic Algorithm for SAT*. In Xiaotie Deng & Ding-Zhu Du, editors: *Algorithms and Computation*, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 644–653, doi:10.1007/11602613_65.

[57] Fabio Zanasi (2015): *Interacting Hopf Algebras: the theory of linear systems*. Ph.D. thesis, Ecole Normale Superieure de Lyon, doi:10.48550/arXiv.1805.03032. Available at https://arxiv.org/abs/1805.03032.

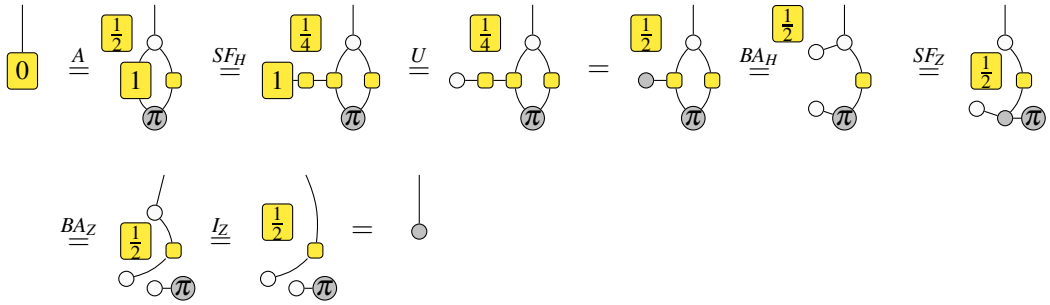
[58] Junping Zhou, Minghao Yin & Chunguang Zhou (2010): *New Worst-Case Upper Bound for #2-SAT and #3-SAT with the Number of Clauses as the Parameter*. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, AAAI Press, Atlanta, Georgia, pp. 217–222, doi:10.48550/arXiv.1006.1537.

A Rewriting Rules

The ZH-calculus [3] has the following rewriting rules:

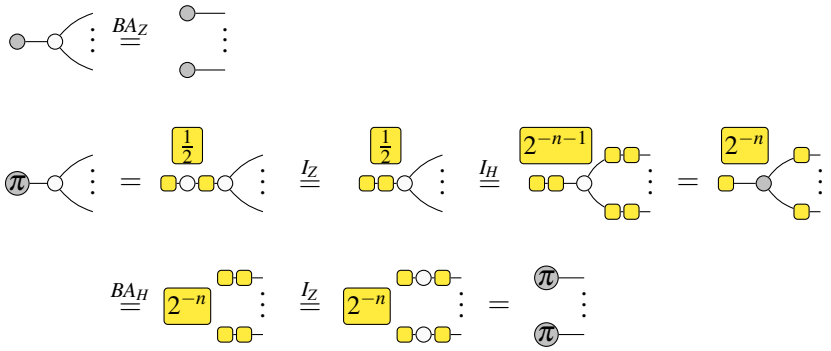


We will also used some derived rules. In particular, we have that



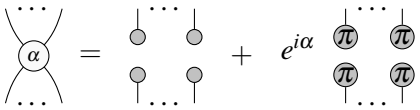
(20)

and also that:



(21)

Finally, we recall the presentation of the Z-spider as a sum over basis states:

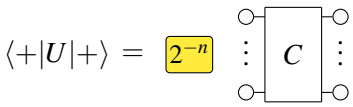


(22)

B Upper Bounds on Quantum Circuit Simulation

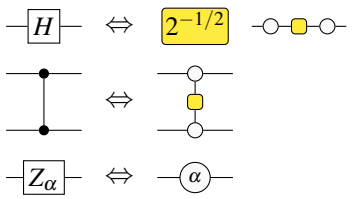
Extending the ideas from Section 4.3, we can produce an upper bound on the time required to exactly compute the amplitude of a quantum computation. This task is naturally represented in terms of tensor networks [39], which can be presented in terms of a graphical calculus, see [54, page 6] for an example in the ZX-calculus. Note that while this task is easily shown to be **#P**-hard, it is not obviously in **#P** and so it is hard to relate directly to **#SAT**. By relaxing our definition of **#SAT** to $\mathbf{\#SAT}_{\mathbb{C}}$, we can do this translation more straightforwardly, but still apply existing knowledge about **#SAT**.

Specifically, given a quantum circuit C with n qubits and G gates representing a unitary U over the CZ, Hadamard, and Z-phase gate set, the quantity $\langle +|U|+\rangle$ can be represented by the following ZH-diagram:



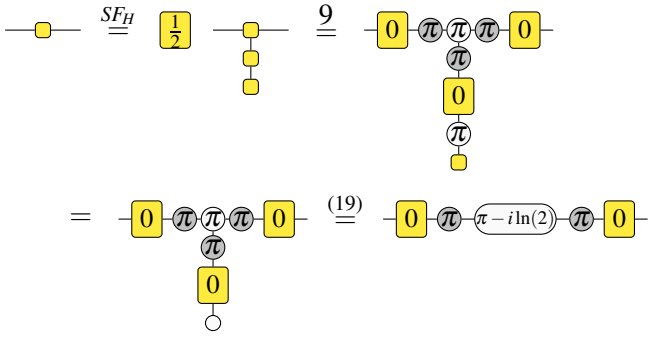
(23)

Where the portion marked C is assembled from the following components



(24)

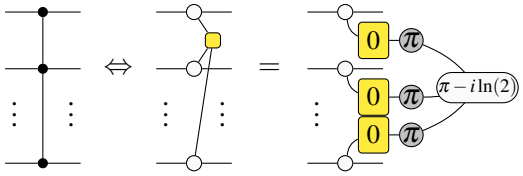
by placing them according to their connections in the quantum circuit. However, we can translate these to $\mathbf{\#2SAT}_{\mathbb{C}}$ diagrams, up to scalar factors, by applying the following rewrite:



(25)

To map $\langle +|U|+\rangle$ into this form, we apply this translation and then spider fusion wherever possible. This leaves a $\mathbf{\#2SAT}_{\mathbb{C}}$ diagram and a scalar factor. Therefore, to compute the value of $\langle +|U|+\rangle$, we can compute the model count of the corresponding $\mathbf{\#2SAT}_{\mathbb{C}}$ instance by $\mathrm{CDP}_{\mathbb{C}}$ and multiply it by the scalar factor. Since Hadamard gates and CZ gates each add two clauses, and Z-phase gates add no clauses, we have $m \leq 2G$. Hence, by applying the $O^*(1.1740^m)$ bound for **#2SAT** by Wang and Gu [52], we can calculate $\langle +|U|+\rangle$ in time $O^*(1.3783^G)$. This is better than statevector simulation whenever $G \leq 2.16n$.

It is also possible to work with circuits that contain $\mathbf{C}^k\mathbf{Z}$ gates directly, since we have the following translation, as before:



(26)

Then each gate adds at most $k+1$ clauses to the **#2SAT** instance, leading to an overall runtime of $O^*(1.1740^{(k+1)G})$. For instance, for $k=2$ this is $O^*(1.6181^G)$, which is substantially better than the corresponding runtime given by decomposing CCZ gates into single- and two-qubit gates (which would be $O^*(6.8552^G)$, as each CCZ would require twelve clauses). It may be possible to give better bounds by analyzing the spider fusion that occurs in this translation and applying bounds for **#2SAT** in terms of variables rather than clauses, but we postpone this to future work.

C A Non-diagrammatic Argument for $\mathrm{CDP}^{\to 2}_{\pm}$

Suppose we have a function $\phi : \mathbb{B}^n \to \mathbb{B}$ given by

$$\phi(x_1,\ldots,x_n) = \bigwedge_{i=1}^{m} (c_{i1} \vee c_{i2} \vee \cdots \vee c_{ik_i})$$

then we can define $\phi' : \mathbb{B}^{n+m} \to \{-1,0,1\}$ given by:

$$\phi'(x_1,\ldots,x_n,y_1\ldots,y_m) = (-1)^{\sum_i y_i}\left[\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij} \vee \neg y_i)\right] = (-1)^{\sum_i y_i}\left[\bigwedge_{i=1}^{m} \neg\left(y_i \wedge \bigvee_{j=1}^{k_i} c_{ij}\right)\right]$$

Lifting Boolean logic to integer arithmetic, we have that

$$\begin{aligned}\phi'(x_1,\ldots,x_n,y_1\ldots,y_m) &= (-1)^{\sum_i y_i}\prod_{i=1}^{m}\left(1 - y_i\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\right) = (-1)^{\sum_i y_i}\sum_{S\subseteq[1,m]}\prod_{i\in S}(-y_i)\left[\bigvee_{j=1}^{k_i} c_{ij}\right] \\ &= \sum_{S\subseteq[1,m]} (-1)^{|S|+\sum_i y_i}\left[\bigwedge_{i\in S} y_i\right]\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right] \\ &= \sum_{S\subseteq[1,m]} (-1)^{\sum_{i\notin S} y_i}\left[\bigwedge_{i\in S} y_i\right]\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\end{aligned}$$

and summing over all possibilities for y_i,

$$\begin{aligned}\sum_{y_1,\ldots,y_m\in\mathbb{B}} \phi'(x_1,\ldots,x_n,y_1\ldots,y_m) &= \sum_{S\subseteq[1,m]}\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\sum_{y_1,\ldots,y_m\in\mathbb{B}} (-1)^{\sum_{i\notin S} y_i}\left[\bigwedge_{i\in S} y_i\right] \\ &= \sum_{S\subseteq[1,m]}\prod_{i\in S}\left[\bigvee_{j=1}^{k_i} c_{ij}\right]\cdot\begin{cases}1 & |S|=m \\ 0 & |S|<m\end{cases} = \left[\bigwedge_{i=1}^{m}\bigvee_{j=1}^{k_i} c_{ij}\right] = \phi(x_1,\ldots,x_n)\end{aligned}$$

therefore, to compute $\#(\phi)$, it is sufficient to sum over all values of ϕ':

$$\#(\phi) = \sum_{x_1,\ldots,x_n\in\mathbb{B}} \phi(x_1,\ldots,x_n) = \sum_{x_1,\ldots,x_n,y_1,\ldots,y_m\in\mathbb{B}} \phi'(x_1,\ldots,x_n,y_1,\ldots,y_m)$$

Finally, we can see that

$$\begin{aligned}
\#(\phi) &= \sum_{x_1,\ldots,x_n,y_1,\ldots,y_m\in\mathbb{B}} (-1)^{\sum_i y_i}\left[\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij}\vee\neg y_i)\right] \\
&= \sum_{\substack{x_1,\ldots,x_n,y_1,\ldots,y_m\in\mathbb{B}\\ \sum_i y_i \text{ even}}}\left[\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij}\vee\neg y_i)\right] - \sum_{\substack{x_1,\ldots,x_n,y_1,\ldots,y_m\in\mathbb{B}\\ \sum_i y_i \text{ odd}}}\left[\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij}\vee\neg y_i)\right] \\
&= \mathbf{\#SAT}_{\pm}\left(\bigwedge_{i=1}^{m}\bigwedge_{j=1}^{k_i}(\neg c_{ij}\vee\neg y_i), \{y_1,\ldots,y_m\}\right)
\end{aligned}$$

by definition, and thus this defines a reduction from **#SAT** to $\mathbf{\#2SAT}_{\pm}$ which can be computed using Algorithm 2.

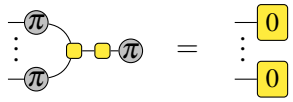
The argument that Wahlström's bound [51] can be applied to Algorithm 2 is the same as in Section 3.2: since the structure of the algorithm remains the same (regardless of scalar factors, unit propagation and branching still generate the same structures), and because unrelated instances combine multiplicatively in that

$$\mathbf{\#SAT}_{\pm}(f_1(x_1,\ldots,x_n)\wedge f_2(y_1,\ldots,y_m), N_1\cup N_2) = \mathbf{\#SAT}_{\pm}(f_1(x_1,\ldots,x_n),N_1)\mathbf{\#SAT}_{\pm}(f_2(y_1,\ldots,y_m),N_2)$$

then the bound doesn't distinguish between Algorithm 1 and 2, so it applies directly.

D Additional Lemmas

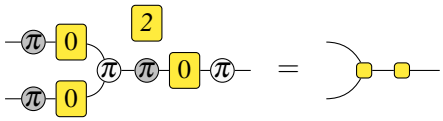
Lemma 6. *The following diagrammatic equivalence holds:*



(27)

Proof. This follows from Lemma 2.28 in [4] and Equation (20). □

Lemma 7. *The following diagram equivalence holds:*

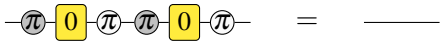


(28)

This is translated from a quantum circuit identity of Ng and Wang [38] and is a generalization of the rules HT *and* BW *from the* $\Delta\mathrm{ZX}_\pi$*-calculus [48], a system for describing tensor networks that is closely related to ZH-calculus.*

Proof. This can be verified by concrete calculation of the matrices. □

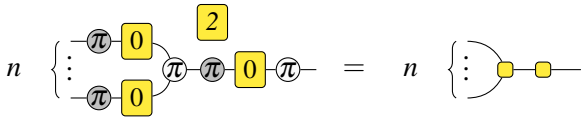
Lemma 8. *The following diagram equivalence holds:*



(29)

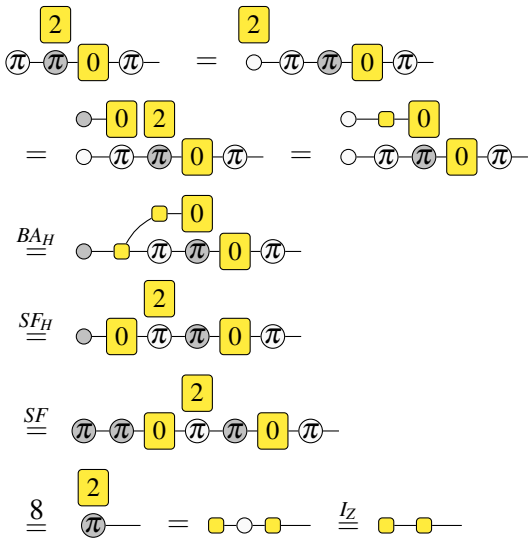
Proof. This can be verified by concrete calculation of the matrices. □

Lemma 9. *The following diagram equivalence holds for all $n \geq 0$:*



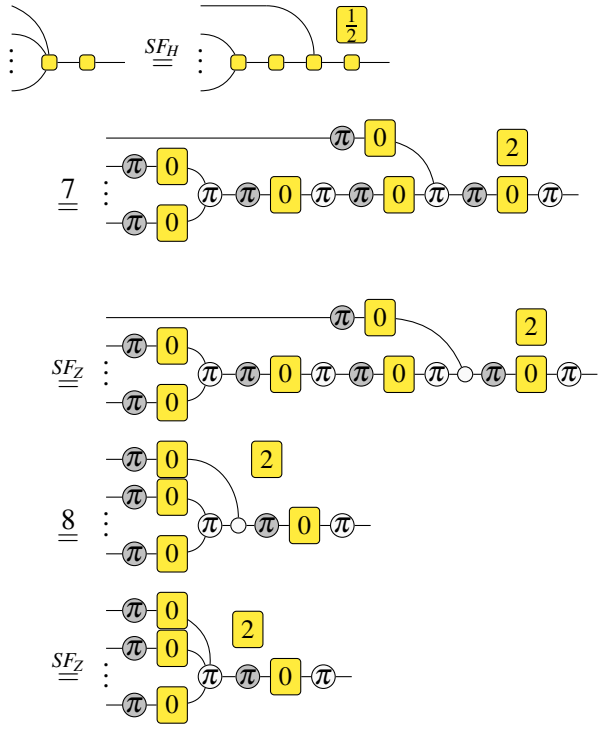
(30)

Proof. We proceed by induction on n. For the case of $n = 0$:



(31)

For the case of $n = k + 1$, assuming the result holds for k:

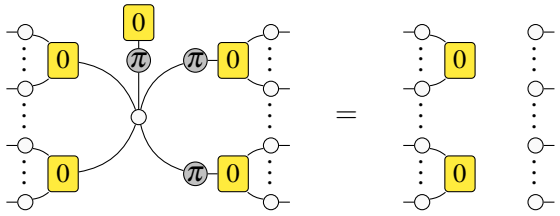


(32)

□

E Proofs of Theorems

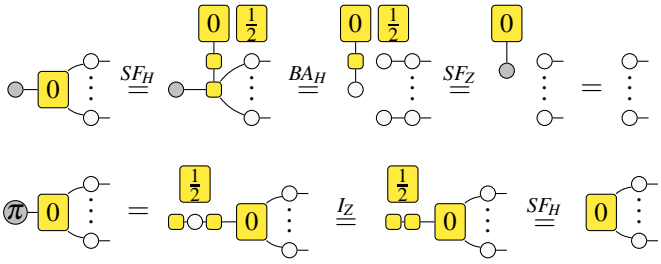
Lemma 1. *The following diagrammatic equivalent to the unit propagation rule holds (without loss of generality, we assume the literal to be propagated is not negated):*



(9)

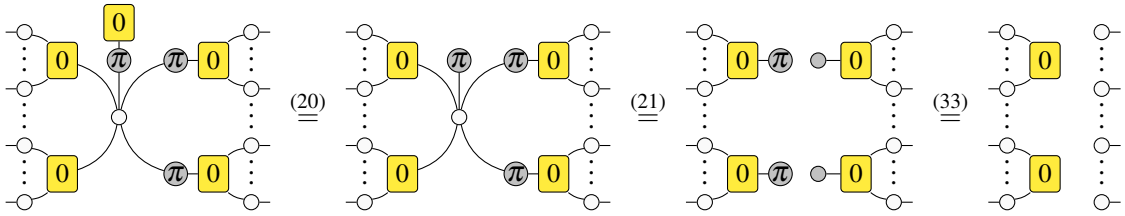
It can be read as follows - on the left-hand side, the zero H-box with one leg is the clause with a single non-negated literal x, the H-boxes on the left represent the clauses $B_i \vee \neg x$ while the H-boxes on the right represent the clauses $A_i \vee x$. On the right-hand side, we see that the clauses B_i remain, while the clauses $A_i \vee x$ have been removed entirely. Because the variable x is now no longer mentioned, the Z-spider representing it is also removed.

Proof. First, note that



(33)

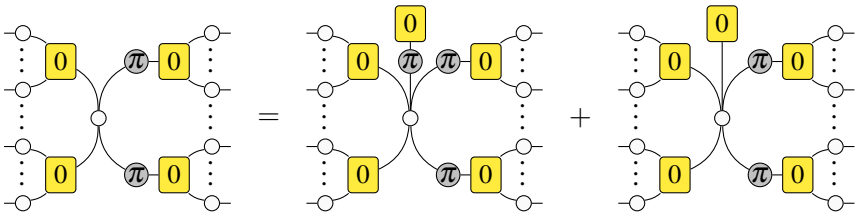
and so we have that



(34)

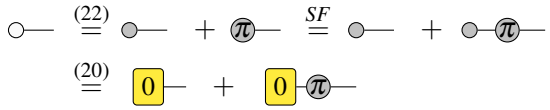
which completes the proof. □

Lemma 2. *The following diagrammatic equivalent to the variable branching rule holds:*



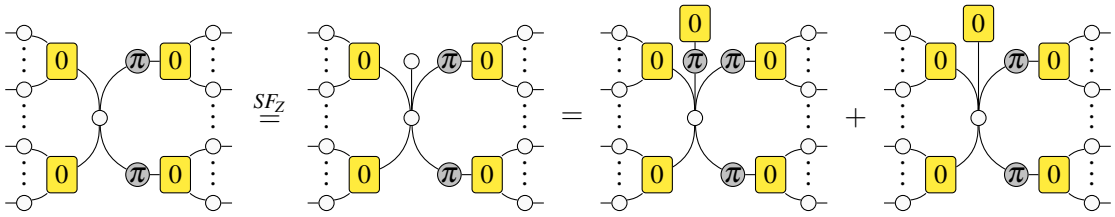
On the left-hand side, we have a variable connected to arbitrary clauses, whereas on the right-hand side we have two terms, each with a clause of one literal introduced onto the variable.

Proof. This follows from writing the Z-spider as a sum



(35)

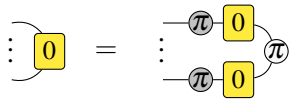
and applying the SF_Z rule to the central spider:



(36)

□

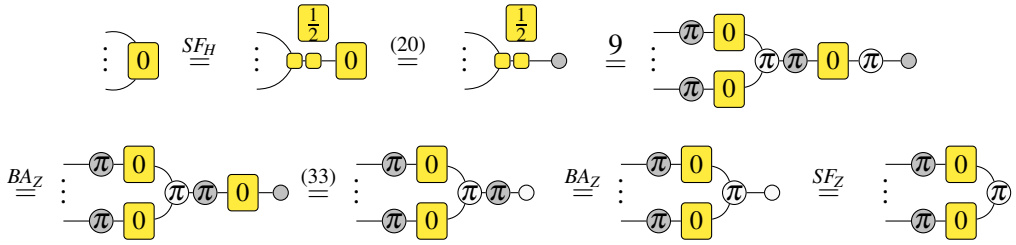
Lemma 3. *The following diagrammatic equivalence holds:*



(10)

This directly generalizes the BW *axiom of the* ΔZX*-calculus [48].*

Proof. We can see the following



which completes the proof. □

Theorem 2. ***#SAT***$_\pm$ *is GapP-complete*

Proof. Suppose there is a problem M in **GapP**, then the aim is to compute $\text{gap}_M = \#M - \#\overline{M}$ where $\#M$ ($\#\overline{M}$) is the number of accepting (rejecting) paths of a non-deterministic Turing machine M. Clearly $\#M$ and $\#\overline{M}$ are both in #P. Because the Cook-Levin reduction from NP to **SAT** is parsimonious, **#SAT** is #P-complete, and there exist CNF formulae f_M and $f_{\overline{M}}$ such that $\textbf{\#SAT}(f_M) = \#M$ and $\textbf{\#SAT}(f_{\overline{M}}) = \#\overline{M}$. Define the following formula:

$$f_{M-\overline{M}} = \left(\bigwedge_{i=1}^{m_M} C_M^i \vee z\right) \wedge \left(\bigwedge_{j=1}^{n_M} \neg x_M^j \vee \neg z\right) \wedge \left(\bigwedge_{i=1}^{m_{\overline{M}}} C_{\overline{M}}^i \vee \neg z\right) \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x_{\overline{M}}^j \vee z\right)$$

where C^i_M ($C^i_{\overline{M}}$) and x^j_M ($x^j_{\overline{M}}$) are the clauses and variables of f_M ($f_{\overline{M}}$) respectively, and z is a fresh variable. When z is false, $f_{M-\overline{M}}$ reduces to

$$f_{M-\overline{M}} \wedge \neg z = \left(\bigwedge_{i=1}^{m_M} C^i_M\right) \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x^j_{\overline{M}}\right) = f_M \wedge \left(\bigwedge_{j=1}^{n_{\overline{M}}} \neg x^j_{\overline{M}}\right)$$

which has exactly $\#\mathbf{SAT}(f_M) = \#M$ satisfying solutions, and likewise with z true, we have

$$f_{M-\overline{M}} \wedge z = f_{\overline{M}} \wedge \left(\bigwedge_{j=1}^{n_M} \neg x^j_M\right)$$

which has exactly $\#\mathbf{SAT}(f_{\overline{M}}) = \#\overline{M}$ satisfying solutions. Finally let $N = \{z\}$, then an assignment of $f_{M-\overline{M}}$ has odd or even N-parity exactly when z is true or false respectively. Therefore,

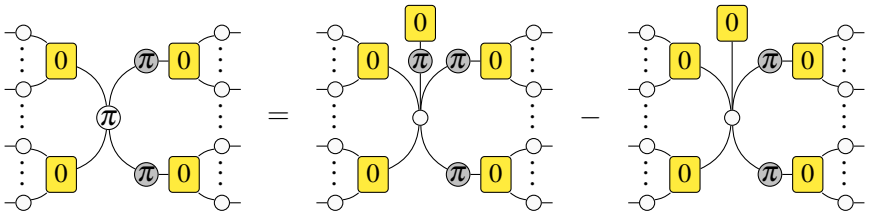
$$\begin{aligned}\#\mathbf{SAT}_{\pm}(f_{M-\overline{M}}, N) &= \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f_{M-\overline{M}}(\vec{x}) - \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f_{M-\overline{M}}(\vec{x}) = \#\mathbf{SAT}(f_{M-\overline{M}} \wedge \neg z) - \#\mathbf{SAT}(f_{M-\overline{M}} \wedge z) \\ &= \#\mathbf{SAT}(f_M) - \#\mathbf{SAT}(f_{\overline{M}}) = \#M - \#\overline{M} = \mathrm{gap}_M\end{aligned}$$

so there is a polynomial-time counting reduction from **GapP** to $\#\mathbf{SAT}_{\pm}$, and it must be **GapP**-hard. Furthermore, let f, N be an instance of $\#\mathbf{SAT}_{\pm}$ with variables x_i, then clearly we have that

$$\#\mathbf{SAT}_{\pm}(f,N) = \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{even } N\text{-parity}}} f(\vec{x}) - \sum_{\substack{\vec{x}\in\mathbb{B}^n \\ \text{odd } N\text{-parity}}} f(\vec{x}) = \#\mathbf{SAT}\left(f \wedge \neg\left(\bigoplus_{i\in N} x_i\right)\right) - \#\mathbf{SAT}\left(f \wedge \left(\bigoplus_{i\in N} x_i\right)\right)$$

since $g = 1 \iff g$ and $g = 0 \iff \neg g$, so $\neg(\bigoplus_{i\in N} x_i)$ is the same as asserting even N-parity (and likewise odd N-parity). But **GapP** is the closure of #P under subtraction, so $\#\mathbf{SAT}_{\pm}(f,N)$ is in **GapP**. Thus it follows that $\#\mathbf{SAT}_{\pm}$ is **GapP**-complete. □

Lemma 4. *The following diagrammatic equivalent to the variable branching rule holds for variables with π-phases:*



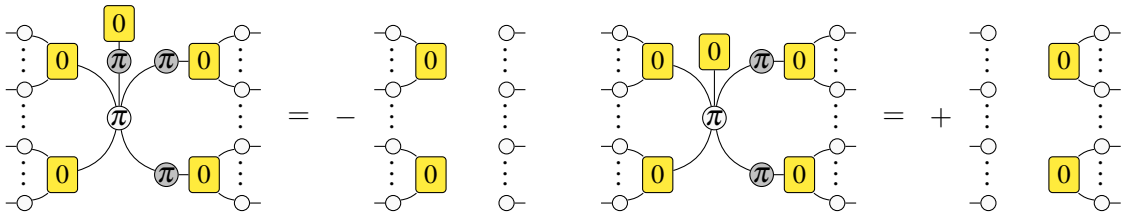
(13)

Proof. This follows from the sum

(37)

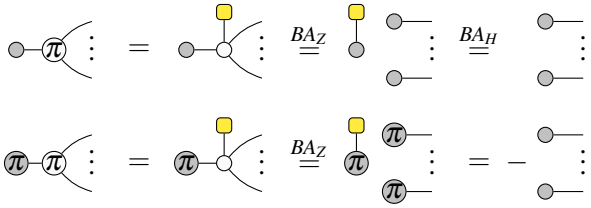
together with the proof of Lemma 2. □

Lemma 5. *The following diagrammatic equivalent to the unit propagation rule holds for variables with π-phases:*



(14)

Proof. This follows from the identities



(38)

together with the proof of Lemma 1. □