Tensor Network Rewriting Strategies for Satisfiability and Counting

Niel de Beaudrap [1], Aleks Kissinger [1], Konstantinos Meichanetzidis [1,2]
[1] Quantum Group, Department of Computer Science, University of Oxford
[2] Cambridge Quantum Computing Ltd.
{firstname.lastname}@cs.ox.ac.uk

We provide a graphical treatment of SAT and #SAT on equal footing. Instances of #SAT can be represented as tensor networks in a standard way. These tensor networks are interpreted by diagrams of the ZH-calculus: a system to reason about tensors over $\mathbb{C}$ in terms of diagrams built from simple generators, in which computation may be carried out by *transformations of diagrams alone*. In general, nodes of ZH diagrams take parameters over $\mathbb{C}$ which determine the tensor coefficients; for the standard representation of #SAT instances, the coefficients take the value 0 or 1. Then, by choosing the coefficients of a diagram to range over $\mathbb{B}$, we represent the corresponding instance of SAT. Thus, by interpreting a diagram either over the boolean semiring or the complex numbers, we instantiate either the *decision* or *counting* version of the problem. We find that for classes known to be in P, such as 2SAT and #XORSAT, the existence of appropriate rewrite rules allows for efficient simplification of the diagram, producing the solution in polynomial time. In contrast, for classes known to be NP-complete, such as 3SAT, or #P-complete, such as #2SAT, the corresponding rewrite rules introduce hyperedges to the diagrams, in numbers which are not easily bounded above by a polynomial. This diagrammatic approach unifies the diagnosis of the complexity of CSPs and #CSPs and shows promise in aiding tensor network contraction-based algorithms.

The solution lies within the problem
The answer is in every question
- Funkadelic

1 Introduction

The Boolean satisfiability problem, or SAT, and its variants, is central to theoretical computer science and complexity theory, with many practical real-world applications. The formula defining the problem can always be given in conjunctive normal form (CNF) comprising of an AND of OR-*constraints*. In a boolean formula on variables $(x_1, x_2, \ldots, x_n)$, a *literal* ℓ is either some variable x_a or its negation $\neg x_a$, for some $1 \leq a \leq n$.

Problem 1.1. CNFSAT (or simply SAT)

Input: $\phi(x) = \bigwedge_{i=1}^{m} C_i(x)$, where $C_i(x) = \bigvee_{j=1}^{k_i} \ell_{i,j}$.

Output: $\exists x \in \{0,1\}^n : \phi(x)$.

The formula consists of a conjunction of clauses $C_i(x)$ for $1 \leq i \leq m$, each being a disjunction of a $k_i \in \mathbb{N}$ number of literals. SAT is the *decision* problem of determining whether there *exists* an assignment x of the variables satisfying *all* clauses. Given a formula ϕ in CNF form, we write $\exists\phi$ for the *instance of* SAT *to evaluate* $\exists x : \phi(x)$, and $[\exists\phi]$ for its answer.

B. Valiron, S. Mansfield, P. Arrighi and P. Panangaden (Eds.): Quantum Physics and Logic 2020 (QPL'20)
EPTCS 340, 2021, pp. 46–59, doi:10.4204/EPTCS.340.3

Famously, SAT is NP-complete. At the heart of this work is the family of special cases kSAT, where $k_i \leq k$ for every clause C_i. For $k = 2$ we have that 2SAT $\in$ P. However, 3SAT $\in$ NP-complete, and so are all cases for $k > 3$ [11, 4, 14]. An important special case of SAT (in the more general formulation of determining whether a boolean formula of *some* form is satisfiable) is XORSAT, where the OR clauses are replaced with XORs, *i.e.* replacing $\vee \to \oplus$ in Def. 1.1. This special case is significantly easier, in that XORSAT $\in$ P.

For each decision problem, X, the corresponding *counting* problem #X asks for the *number* of satisfying assignments to a Boolean formula. In particular, #SAT is given as follows:

Problem 1.2. #CNFSAT (or simply #SAT)

Input: $\phi(x) = \bigwedge_{i=1}^{m} C_i(x)$, where $C_i(x) = \bigvee_{j=1}^{k_i} \ell_{i,j}$.

Output: $\#\{x \in \{0,1\}^n : \phi(x)\}$.

In general, #SAT $\in$ #P-complete [15]. The counting versions of the special cases defined above are denoted similarly #XORSAT, #kSAT. Given a formula ϕ in CNF form, we write $\#\phi$ for the *instance of* #SAT described by ϕ, and $[\#\phi] := \#\{x : \phi(x)\}$ for its solution. Note that #XORSAT $\in$ P and that #kSAT $\in$ #P-complete for $k \geq 2$. In this sense, #2SAT is significantly harder than its decision version.

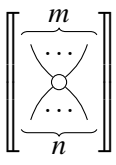
This work is motivated by the will to understand the structural origin of the easiness or hardness of these problems. What is it that makes counting the solutions to XORSAT, and consequently deciding if there is at least one, easy? Why is 2SAT easy to decide but hard to count? What changes when we turn to 3SAT and both deciding and counting are hard? In the following, we investigate these questions by employing graphical methods.

2 Counting with ZH rewrites

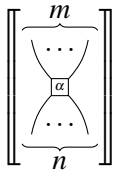
Instances of #SAT may be represented as tensor networks [9, 6] and the solution is returned by full tensor contraction, a #P-hard problem in general [5]. Here, we describe problem instances in terms of a *specific variety* of tensor network, i.e. diagrams of the ZH-calculus [1, 17] — a formal system to analyse complex-valued tensor networks of bond-dimension 2 (all indices ranging over $\{0,1\}$) by transformations of diagrams. This provides a means of considering the complexity of these problems by tensor-based techniques, while nevertheless avoiding explicit tensor contractions.

2.1 Tensor networks from the ZH calculus

The ZH calculus is a diagrammatic notation for tensor networks, which allows computations to be done with these diagrams *in lieu of matrix computations*. These diagrams consist of graphs with different kinds of nodes: *Z-spiders*, which are represented as *white dots* with a specific number of wires, and *H-boxes*, which are represented as white boxes labelled with a complex number $\alpha \in \mathbb{C}$. These generators are interpreted as follows, where $[\![\cdot]\!]$ denotes the map from diagrams to matrices:



$:= |0\rangle^{\otimes n}\langle 0|^{\otimes m} + |1\rangle^{\otimes n}\langle 1|^{\otimes m}$,



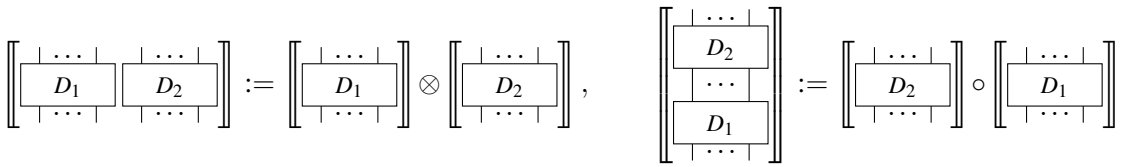
$:= \sum_{\substack{x\in\{0,1\}^n \\ y\in\{0,1\}^m}} \alpha^{x_1\ldots x_n y_1\ldots y_m} |x\rangle\langle y|$

The Z-spider is a high-dimensional Kronecker delta-function whose matrix elements are all zeros

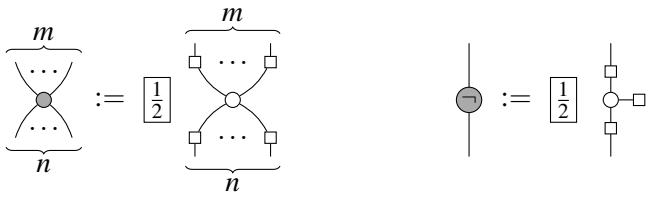
except for the two entries indexed by all zeros or all ones which are 1. An $\mathrm{H}(\alpha)$-box represents an all-ones matrix with the exception of the entry indexed by all ones which is set to α. If not denoted, by convention $\alpha = -1$. Straight and curved wires have the interpretations:

$$[\![\ |\ ]\!] := |0\rangle\langle 0| + |1\rangle\langle 1|\,, \qquad [\![\ \smile\ ]\!] := |00\rangle + |11\rangle\,, \qquad [\![\ \frown\ ]\!] := \langle 00| + \langle 11|\,.$$

Diagram juxtaposition corresponds to tensor product and composition to matrix product:



Expressions in the ZH calculus also use two derived generators, called *X-spiders*, represented by a gray dot with a number of wires, and *NOT*.

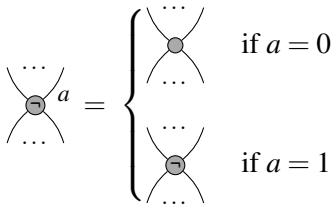


(1)

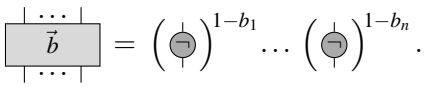
With these definitions,  acts on computational basis states as XOR, i.e. one wire caries the parity of the other wires, and  acts as NOT (or Pauli X). We will write a X-spider labelled by a '$\neg$' to represent an X-spider with a NOT applied to any of its legs. Concretely, the correspond to the even- and odd-parity tensors:

$$\left[\!\!\left[\ \right]\!\!\right] = \sum_{\substack{x_1,\ldots,x_m,y_1,\ldots,y_n \\ \bigoplus x_i \oplus \bigoplus y_j = 0}} |y_1 \ldots y_n\rangle\langle x_1 \ldots x_m| \qquad \left[\!\!\left[\ \right]\!\!\right] = \sum_{\substack{x_1,\ldots,x_m,y_1,\ldots,y_n \\ \bigoplus x_i \oplus \bigoplus y_j = 1}} |y_1 \ldots y_n\rangle\langle x_1 \ldots x_m|$$

and in the following we will also use the compact notation

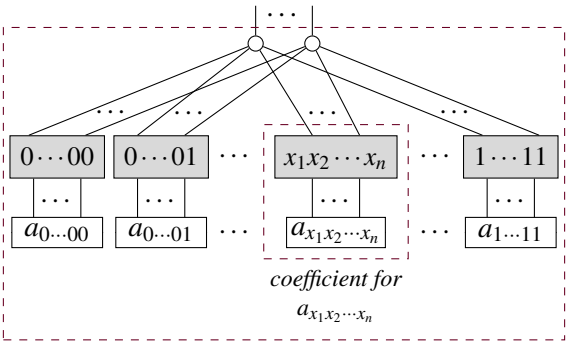


These nodes suffice to express any tensor of any rank with bond-dimension 2 over $\mathbb{C}$, as we now show. For a vector $\vec{b} \in \{0,1\}^n$, we define the short-hand notation



Then, given such a tensor A, with n indices, we may construct the ZH diagram representing it, called

the *ZH normal form for A*, using one H box for each of its 2^n coefficients $a_{x_1x_2\cdots x_n}$:



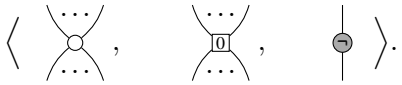
(2)

For any coefficient $a_{x_1x_2\cdots x_n} = 1$, the corresponding coefficient gadget may be omitted — we call a ZH diagram with these omitted coefficient gadgets a *condensed normal form*. These normal forms will in many cases not be the most efficient way to represent a tensor by ZH diagrams; for our purposes it suffices that they exist as a means to represent *any particular* diagram. Thus, the ZH calculus forms a way to represent tensors with diagrams assembled from simple generators.

2.2 Representing #SAT instances as ZH diagrams

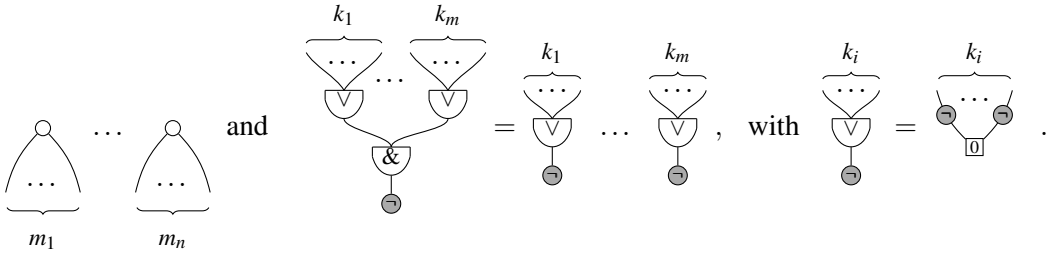
Instances $\#\phi \in \#\mathrm{SAT}$ may be represented by ZH tensor networks in a straightforward way.

Theorem 2.1. Any instance $\#\phi \in \#\mathrm{SAT}$ can be represented as a closed ZH diagram, i.e. with no open wires, which evaluates to $[\#\phi]$, composed of the following nodes:



Proof. We construct a bijection between CNF formulae and ZH diagrams, keeping consistent with the notation in Defs. 1.1, 1.2. The construction is of the form of a circuit (preparation, process, measurement) with postselection.

Variables are initialised as "states" to all their possible assignments. Let m_j be the number of clauses in which x_j or its negation participates. Each x_j is represented by a m_j-ary Z-spider so that it is copied enough times to participate in the clauses, after possibly going through a negation, as dictated by ϕ. The states then enter the OR-gates, which play the role of the "processes". Let k_i be the number of variables that are involved in clause C_i. Finally, an m-ary AND-gate accepts accepting as inputs all outputs of the OR-gates. Post-selecting this AND-gate on 1, implements an "effect" and represents the satisfiability requirement, which is equivalent to postselecting all outputs of the OR-gates to 1, by the property of the AND-gate. A postselected-on-1 OR-gate is an OR-constraint and is a tensor with entries the OR truth table. This tensor is represented by an all-negated H(0)-box. Diagrammatically we have:



Thus, any ϕ in CNF is expressed as a bipartite ZH tensor network by connecting Z-spiders with all-negated H(0)-boxes. We can capture the fact that a variable appears negated in a given clause by removing a negation from that wire, since two negations cancel out.

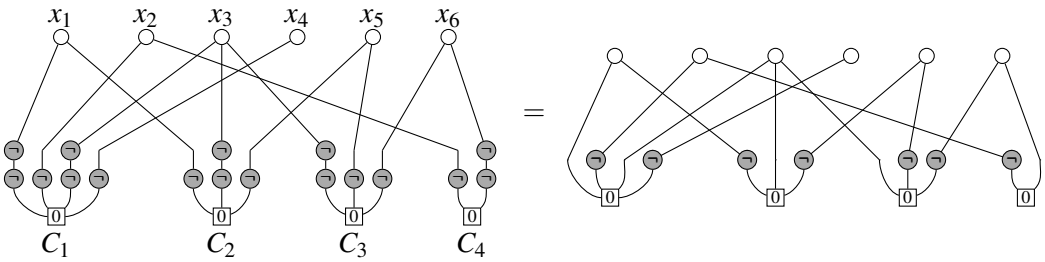
In this way, we can exactly capture the instance ϕ as a closed ZH-diagram, with one Z-spider for each variable, one H(0) box for each clause, and wires (possibly with NOTs) connecting them. □

Note that the H(0)-box can be viewed as an NAND-constraint, in the sense that its tensor elements are given by $(\mathrm{H}(0))_{x_1\ldots x_k} = \mathrm{NAND}(x_1,\ldots,x_k)$. Also, the postselected-circuit construction above returns the tensor network representations of CNFSAT of [9].

Example 2.2. Consider the following CNF formula:

$$\phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4 = (\neg x_1 \vee x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_3 \vee x_5) \wedge (\neg x_3 \vee x_5 \vee x_6) \wedge (x_2 \vee \neg x_6) \quad (3)$$

Its corresponding ZH tensor network is:



2.3 Evaluating tensor networks with the ZH calculus

The ZH calculus is not just a notation for tensor networks, but a *sound* system of diagrammatic transformations that preserve *semantics*, i.e. the tensor represented by the network.

The most basic diagram transformations, or "rewrites", allowed in the ZH calculus are ways that sub-networks of ZH generators may be transformed while preserving semantics. For Mat[$\mathbb{C}$], the ZH rewrite rules are not only *sound*, but also *complete* — meaning that any two networks using ZH generators which are equivalent, may be proven to be equivalent using *only* diagram transformations. In particular, any closed tensor network (e.g. the ZH-diagram representing $\#\phi$) expressed as a ZH diagram corresponds to a scalar (e.g. $[\#\phi]$). By completeness, it is always possible to transform a closed ZH-diagram into an efficient representation of that scalar, such as a disconnected collection of arity-0 generators. We call *full simplification* of a closed diagram a sequence of rewrites that removes all wires from the diagram.

Thus, as an alternative to tensor network contraction in the conventional way, we may in principle evaluate $[\#\phi]$ using transformations of ZH diagrams. Taking this approach, the complexity of the computation is governed by the size and complexity of the network as it is transformed as well as the number of transformation steps taken. This is in contrast to full tensor contraction, in which the network becomes simpler (in terms of edge count) throughout the evaluation. In this case, the complexity is in terms of the cost of storing the entries of the nodes which increases exponentially with their rank, which in turn is upper bounded by the number of incident wires. One may elaborate on this by performing SVDs on the tensor nodes to limit the bond-dimension as much as possible [10, 7, 16]. Naturally, the different approach of the ZH calculus does not ensure that contraction can be performed efficiently — the number of rewrites required to perform a full simplification may be exponential in the number of nodes of the input tensor, and the diagrams constructed in doing so may

also in principle become exponentially complex — but because the nodes range over a simple set, it is possible in some cases to use relationships between the generators and their types to identify special cases which may be evaluated efficiently.

In either case — for full tensor contraction, or full simplification by ZH rewrites — one generally expects to avoid mounting complexity in special cases, either to avoid mounting complexity of the tensor nodes (for tensor contraction) or mounting complexity of networks or difficulty in determining how to transform the networks (for ZH diagrams). In the following sections, we consider the complexity of ZH simplification strategies for #SAT, #XORSAT, as well as their corresponding decision problems.

3 Satisfiability with ZH rewrites

One may ask whether or not the *decision* problem SAT may similarly be represented by ZH diagrams. The decision problem corresponds to determining whether or not the answer to a counting problem is non-zero.

3.1 Matrices over semirings

The complex numbers $\mathbb{C}$, natural numbers $\mathbb{N}$ and the booleans $\mathbb{B}$ are all commutative semirings: sets S equipped with two commutative and associative binary operations $+ : S \times S \to S$, with identity element 0_S, and $* : S \times S \to S$, with identity element 1_S. Operation $*$ distributes over $+$ and 0_S is absorbing for $*$ so that $\alpha * 0_S = 0_S * \alpha = 0_S, \forall a \in S$. In $\mathbb{N}$, the operations $+$ and $*$ are just the usual addition and multiplication. For $\mathbb{B}$, we take $(+, 0_{\mathbb{B}}) = (\vee, 0)$ and $(*, 1_{\mathbb{B}}) = (\wedge, 1)$, treating disjunction as addition and conjunction as multiplication. We may also write $a \wedge b$ as a product ab, when $a, b \in \mathbb{B}$.

A homomorphism of semi-rings is a function $S \to T$ which preserves $0, 1, +$ and $*$. The most important semi-ring homomorphisms for our purposes are the inclusion $e : \mathbb{N} \to \mathbb{C}$ and the projection $p : \mathbb{N} \to \mathbb{B}$. In both cases, these maps are uniquely defined, due to the property of being homomorphisms. Notably, we can see $\mathbb{B}$ as a quotient of $\mathbb{N}$, where we additionally impose the (seemingly nonsensical) equation $1 = 2$. In this case, p is the quotient map.

For any semi-ring, we can define a monoidal category $\mathrm{Mat}[S]$ of matrices of S, whose objects are natural numbers, morphisms $\psi : m \to n$ are $n \times m$ matrices, composition is given by matrix multiplication, and $\otimes$ by tensor product. That is, for $\psi : m \to n$ and $\phi : m' \to n'$, $\psi \otimes \phi : mm' \to nn'$ is a matrix whose elements are $(\psi \otimes \phi)^{n'j+j'}_{m'i+i'} = \psi^j_i * \phi^{j'}_{i'}$. In particular, we can regard a morphism $\psi : m_1 \otimes \ldots \otimes m_k \to n_1 \otimes \ldots \otimes n_l$ as a (k, l)-tensor.

We can lift a semiring homomorphism $h : S \to T$ to a functor $H : \mathrm{Mat}[S] \to \mathrm{Mat}[T]$ simply by applying h to each of the elements of a matrix. In this way, we can obtain a faithful functor $E : \mathrm{Mat}[\mathbb{N}] \to \mathrm{Mat}[\mathbb{C}]$ from e and a full functor $P : \mathrm{Mat}[\mathbb{N}] \to \mathrm{Mat}[\mathbb{B}]$ from p.

3.2 From counting to deciding by change of semiring

Counting and deciding may be interpreted as asking essentially the same question of "counting", but relative to different semirings: either in $\mathbb{N}$ for the counting problem, or in $\mathbb{B}$ for the decision problem. Therefore, a closed ZH diagram representing a ϕ in CNF evaluates to $[\#\phi]$ by interpreting the diagram as a tensor network over $\mathbb{N}$ and to $[\phi]$ when the diagram is viewed as a tensor with coefficients in $\mathbb{B}$. However, graph-partitioning based optimisers for the tensor contraction path are expected to exhibit similar performance regardless of the choice of semiring over which the tensor network's coefficients take values [9]. This is the case even for easy problem families such as XORSAT or 2SAT. We

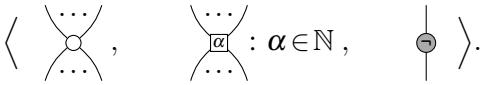
present an approach to reasoning about boolean tensor networks using the ZH calculus which allows us to recover the efficient solvability of said easy problems.

3.3 Boolean tensor rewrites from the ZH calculus

Consider approaching the idea of rewriting boolean tensor diagrams, from the direction of taking an existing diagrammatic calculus, which is well suited to describe matrices over $\mathbb{N}$, such as the ZH calculus, and "projecting" it down to the boolean semiring $\mathbb{B}$. Subtly, this cannot be simply done by changing the semiring from which the node parameters are chosen. This is because there is no corresponding element to -1 in the booleans (1 has no additive inverse) whereas $\mathrm{H}(-1)$-boxes play a prominent role in the ZH rewrites. Put another way, there is no semiring homomorphism $\mu : \mathbb{C} \to \mathbb{B}$ (or even $\nu : \mathbb{Z} \to \mathbb{B}$). Despite this, we can still reason soundly about boolean tensor networks with the ZH calculus. However, while ZH is defined for generators taking parameters in $\mathbb{C}$, among these generators are ones taking parameters in $\mathbb{N}$. Diagrams which are composed exclusively of such generators will represent tensors over $\mathbb{N}$. Furthermore, some gadgets of the ZH calculus — in particular, the X-spider and NOT-dot of Eqn. (1) — represent matrices over $\mathbb{N}$.

Any tensor network involving the $\mathbb{N}$-valued generators together with these gadgets will also represent tensors over $\mathbb{N}$. We may then consider ways to reason about boolean tensors through the use of these diagrams. Note that $\mathbb{N}$-valued ZH generators, together with NOT-dots, suffice to represent the ZH normal forms (and condensed normal forms) of any $\mathbb{N}$-valued tensor with indices of dimension 2, as described in Eqn. (2). This motivates the following definition:

Definition 3.1. A *NatZH diagram* is a ZH diagram generated from



Some of the basic ZH rewrites may be interpreted as rewrites of NatZH diagrams, but not all. Notably, any rule involving an $\mathrm{H}(-1)$-box cannot be realised on NatZH diagrams, and as the definitions of gray nodes in Eqn. (1) involve these, rewrites involving them also pose problems. However, just as it is possible to reason about real polynomials by making use of the complex numbers, any theorem of ZH which describes the equivalence of NatZH diagrams is still usable to reason about NatZH diagrams, regardless of whether the *intermediate* steps preserve the set of NatZH diagrams. This is essentially re-stating the fact that $\mathrm{Mat}[\mathbb{N}]$ embeds faithfully in the larger category $\mathrm{Mat}[\mathbb{C}]$.

Having defined a sub-theory of ZH which maps onto $\mathrm{Mat}[\mathbb{N}]$, we can approach the subject of boolean tensors via the functor $P : \mathrm{Mat}[\mathbb{N}] \to \mathrm{Mat}[\mathbb{B}]$ which projects $\mathbb{N}$-matrices down to $\mathbb{B}$-matrices. This projection will preserve any equation we proved between $\mathbb{N}$-matrices, e.g. those proven with the ZH-calculus. But also, more rules become true. Notably, since in booleans $1 = 2$, $\mathrm{H}(2)$-boxes are the same as $\mathrm{H}(1)$-boxes, which will have a dramatic effect in Section 4.3.

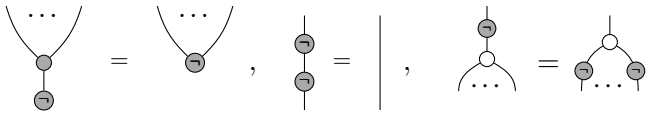
4 Identifying Efficiently Simplifiable Instances

In this section we study the above introduced paradigmatic counting and decision problem families whose complexity is well studied in the literature. We will observe that application of rewrites that result in *variable elimination*, or killing Z-spiders, is an efficient diagrammatic technique for solving the problem. In contrast, (potentially) hard instances are those whose analogous rewrites either block further simplification or exponentially grow the diagram. We will illustrate precisely what we mean by this using the examples in this section.

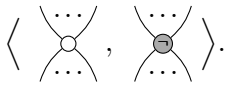
4.1 #XORSAT and XORSAT

We begin with the easy case of parity constraints. Since #XORSAT $\in$P, then it trivially follows that XORSAT $\in$P. Interestingly however, this problem shows rich behaviour. When one tries to anneal to a solution, one encounters a glassy landscape which makes this approach inefficient [13, 12].

As in Proof. 2.2, an XOR-constraint is a postselected-on-1 XOR-gate. Since the XOR-gate is the X-spider, a XOR-constraint is a NOT-spider. A gray dot can absorb a number of NOT-dots and by *double negation elimination* it is an X-spider if it absorbs an even number of negations and a NOT-spider if odd. Also, a NOT-dot gets copied by a white dot:

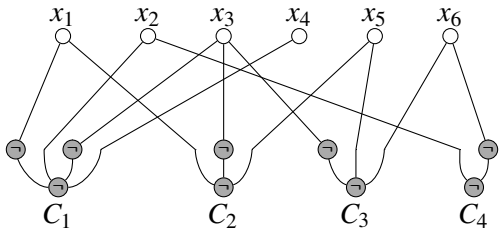


Therefore, any #XORSAT instance diagram is represented in terms of:

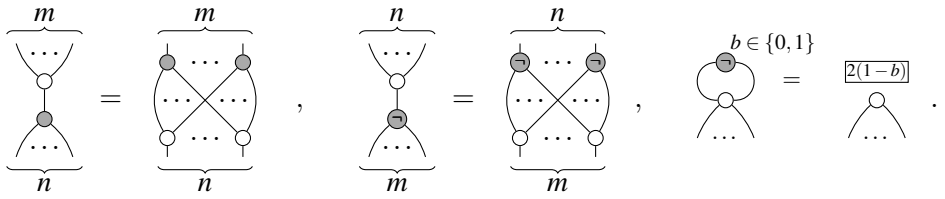


Consider Example 2.2, but with the OR-constraints replaced by XOR-constraints. Then one gets an example of a #XORSAT formula and its representative ZH tensor network:

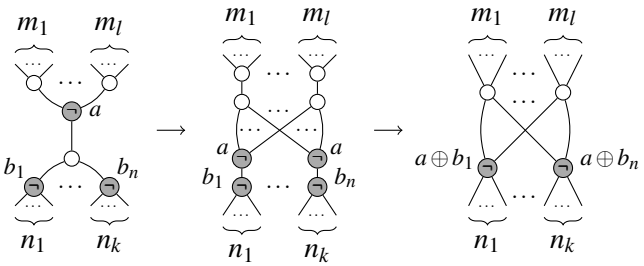
$$\phi = (\neg x_1 \oplus x_2 \oplus \neg x_3 \oplus x_4) \wedge (x_1 \oplus \neg x_3 \oplus x_5) \wedge (\neg x_3 \oplus x_5 \oplus x_6) \wedge (x_2 \oplus \neg x_6)$$



The X-spider, and by consequence the NOT-spider satisfy the ZH bialgebra law [17]. Also, a looping wire evaluates to 0 if the wire is negated and to 2 if the wire is naked. In the boolean case a naked looping wire can be trivially removed since $p(2) = 1$:



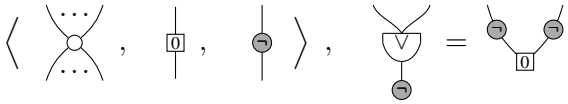
The graph defining the problem is bipartite; every wire connects a white dot with a gray dot (either X-spider or NOT-spider). After one application of the bialgebra rewrite rule, we can always fuse dots of the same colour:



Iteratively using the bialgebra rewrite, we can eliminate variables by introducing only polynomially many edges. In fact, the number of wires introduced during this dance between gray and white spiders is quadratically upper-bounded by the number of nodes in the network. In the case when the solution is not zero, full simplification returns $[\#\phi] = 2^c$, resulting from the number c of disconnected components of the network after the iterated application of the bialgebra rule. For the decision problem, exactly the same procedure returns $[\phi] = p(2^c) = 1$

4.2 #2SAT

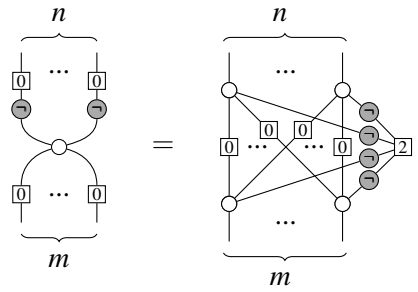
Now that we've seen that #XORSAT can be efficiently solved by eliminating variables, we turn to the seemingly simple but actually hard #2SAT problem and attempt a similar strategy. An instance $\#\phi \in$ #2SAT is represented by a NatZH diagram generated by variable tensors, *binary* H(0)-boxes, and negations. The latter two generators compose the binary OR-constraint:



(4)

We locally arrange the wires around a variable so to separate those that go through a negation from the naked wires. To eliminate a variable, we use the following theorem.

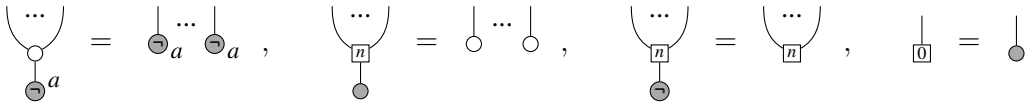
Theorem 4.1. The following rule holds in the ZH-calculus, for all $m, n \geq 0$:



(5)

Proof. By completeness of the ZH-calculus, it suffices to show the matrices of the LHS and RHS are equal. In each case, the matrix has a single 2 in the top-left corner, 1's in the rest of the first row and column, and 0's elsewhere.

We can evaluate matrix elements by pre- and post-composing with computational basis states $\{ = |0\rangle,\ = |1\rangle\}$ and their adjoints and simplifying using the rules of the ZH-calculus. In particular, we use the following rules:



(6)

and split into 3 cases: (i) If all the inputs are plugged with and outputs with , the LHS simplifies to a single Z-spider with no inputs and outputs and the RHS to a single H-box labelled by 2. In both cases, this equals the scalar 2. (ii) If at least 1 input is and 1 output is , the LHS simplifies to a diagram containing at least 1 copy of $\circ$ and RHS contains an $H(0)$ box with no inputs and outputs. Hence both sides go to 0. (iii) Otherwise the LHS and the RHS both simplify to many copies of $\circ$ or $\circ$, which goes to 1. □

In addition to the matrix derivation given in the proof above, equation (5) has a logical interpretation as well, in light of the representation of CNFs as ZH-diagrams from Theorem 2.1. *Propositional resolution*, i.e. reducing a pair of clauses of the form $(P \vee x)$ and $(\neg x \vee Q)$ to a clause $(P \vee Q)$, is a form of modus ponens which is particularly well-suited to CNFs. If we just focus on the H(0)-boxes in equation (5), we see that this transformation replaces all of the clauses containing a single variable x with all of the possible resolutions not containing x:

$$\bigwedge_i (x \vee y_i) \wedge \bigwedge_j (\neg x \vee z_j) \implies \bigwedge_{ij} (y_i \vee z_j) \tag{7}$$

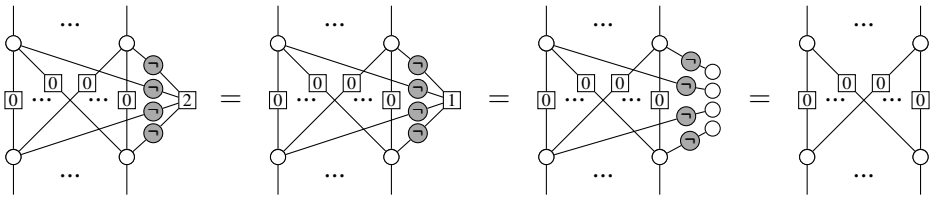
Indeed the conclusion of this implication is satisfiable if and only if its premise is.

However, a lingering question is: what is going on with the extra H(2) on the RHS of (5)? This captures precisely the fact that, for every satisfying assignment to the conclusion of (7) where all the y_i, z_j are true, the premise has 2 satisfying assignments, one where $x = 0$ and one where $x = 1$.

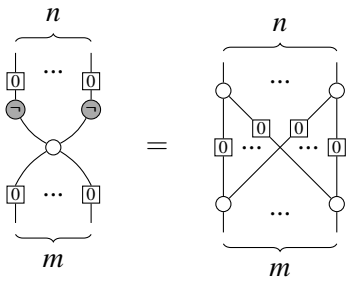
The extra H(2)-box in the RHS of (5) prevents us from being able to iteratively apply this rule to a CNF instance to obtain a solution in polynomial time. Indeed we should expect to find some obstruction to any polynomial-time strategy, since #2SAT $\in$ #P-complete. However, we will now see what happens when we switch to a category where $2 = 1$.

4.3 2SAT

We now look at the decision version of the above problem which is in P. The ZH-diagram of an instance $\phi \in$ 2SAT is generated just as in Theorem 2.1, but it is evaluated as a tensor network over the Booleans, rather than the natural numbers. In particular, the parameter $2 = 1 + 1$ on the RHS of equation (5) becomes $1 \vee 1 = 1$ in the case of Booleans. So, the extra H(2)-box on RHS vanishes:



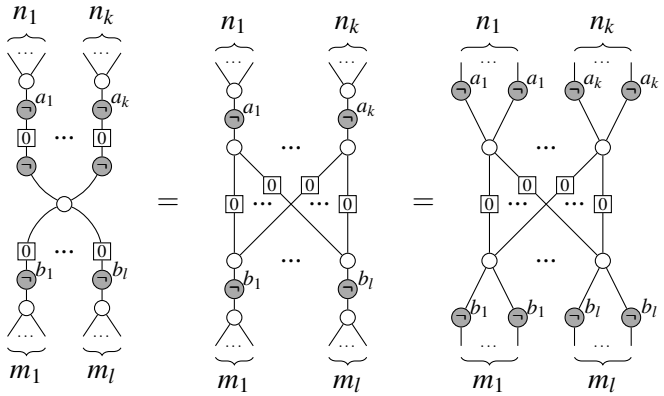
Hence, for the booleans, we get a new, simpler rule:



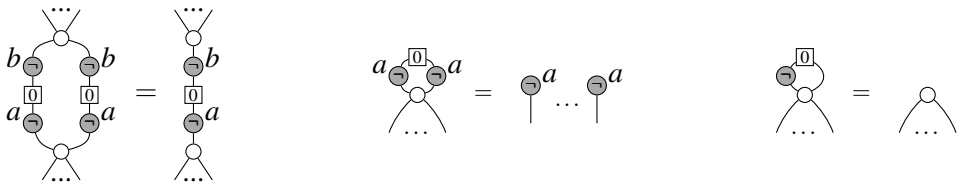
(8)

This rule no longer introduces H-boxes that would obstruct further application of the rule to its neighbours. Hence, any 2SAT instance can be fully simplified as follows. Every application of (8) eliminates one variable, and after every variable elimination we perform white-dot fusions, as in the case of #XORSAT, since again the network is bipartite, every wire, either naked or negated, connects

a H(0)-box with a white-dot.



We observe that all variables can be eliminated by iterative application of the above boolean-spider killing rule (8), followed by a small amount of 'tidying up'. That is, we can remove 'duplicate' H(0) boxes, which correspond to repeated clauses, and 'self-loops', which correspond to a variable occurring twice in a clause, using the following rules:

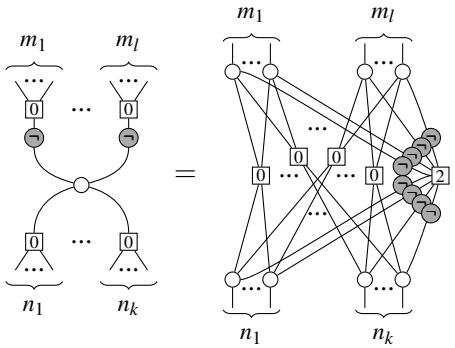


In the case of the first 2 kinds of loops, we end up with some basis states, so we are not a CNF-like ZH-diagram. However, applying the rules (6) from the proof of Theorem 4.1 will either get back to CNF or produce a 0 (i.e. 'unsatisfiable').

With every white dot removed, we introduce a number of wires going through H(0)-boxes which is quadratic in the arity of that white dot. Since we can remove repeated H(0)-boxes at every step, the arity of a given white dot is at most linear in the number of variables remaining. We thus recover the polynomial cost of full simplification of 2SAT instances, and we witness this purely in terms of diagram rewriting.

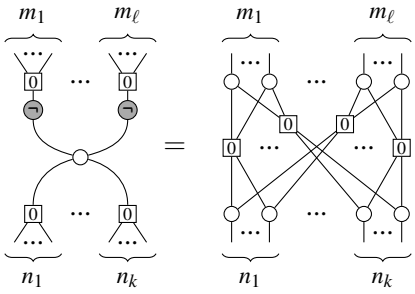
4.4 #SAT and SAT

For the generic #SAT case, which is #P-complete, and the corresponding SAT case, which is NP-complete, the variable elimination rewrites introduced above for #2SAT and SAT have analogues that introduce H(0) boxes connected to more than just pairs of white dots. In particular, for #SAT we have that killing a white spider results in:



This rule is straightforward to prove from the #2SAT version (8) and the basic ZH-calculus rules.

Again, passing to SAT, we can use $2 = 1$ to obtain a simpler rule:



(9)

which we can indeed apply iteratively to solve a SAT instance. However, comparing the 2SAT simplification rule to the SAT version, we notice one crucial difference. In equation (8), the arity of the $\mathrm{H}(0)$-boxes remains unchanged (at 2). Hence, we can argue that there are *at most polynomially many* of them at each step. However, if the CNF contains clauses of 3 or more variables, iterating (9) will cause the arities of $\mathrm{H}(0)$-boxes to grow arbitrarily large. Hence, in the worst case, we will obtain exponentially many distinct $\mathrm{H}(0)$-boxes, and the efficiency of our procedure is destroyed.

5 Conclusions and Future Work

We have imported rewrite rules from the sound and complete ZH graphical calculus in order to reason about tensor networks that represent instances of #SAT and SAT. In this way, we have identified that tensors of instances of problem families known to be in P, such as #XORSAT and 2SAT, can be efficiently rewritten to a scalar encoding the solution to the problem.

In particular, in order to be able to reason about boolean tensor networks encoding instances of SAT, we observed that tensors encoding instances of #SAT take values over the natural numbers. The ZH calculus is complete for $\mathrm{Mat}[\mathbb{N}]$ since it is complete for $\mathrm{Mat}[\mathbb{C}]$, and so for counting problems we simple used the available rewrites of ZH. In order to switch from counting to decision, we interpreted decision as counting but over the boolean semiring. Interestingly, projecting from $\mathrm{Mat}[\mathbb{N}]$ to $\mathrm{Mat}[\mathbb{B}]$ gave rise to rewrite rules which made apparent the difference in complexity between #2SAT and 2SAT.

For further work, we aim to complete ongoing work with preliminary results on defining a sound and complete graphical calculus for *boolean* tensor networks. Such a calculus is motivated by the boolean projection P we have used here but is an *independent* calculus from ZH. Notably, there exists highly relevant work on the ZX& calculus [3], which provides a sound and complete framework for natural-number tensors as well as for boolean tensors. It is inspired by the ZX calculus [2] and adds the AND and NOT gates as primitive generators.

Finally, we note that the Python library PyZX [8], developed to support automated rewriting for ZX-diagrams supports the ZH-calculus as well. Building on this library, we aim to develop a useful tool both for classical and quantum many-body problems. It would be interesting to benchmark such a library against existing techniques for counting and decision problems, as well as aid existing tensor contraction methods by equipping them with rewrite strategies and create hybrid algorithms for counting problems.

6 Acknowledgments

N.dB. is supported by a Fellowship funded by a gift from Tencent Holdings (`tencent.com`). A.K. would like to acknowledge support from AFOSR grant FA2386-18-1-4028. K.M. is supported by an 1851 Research Fellowship (`royalcommission1851.org`) and by Cambridge Quantum Computing Ltd.. K.M. acknowledges Stefanos Kourtis for inspiring discussions.

A ZH rewrite rules

Rewrite rules for the complete ZH-calculus in Fig. 1.

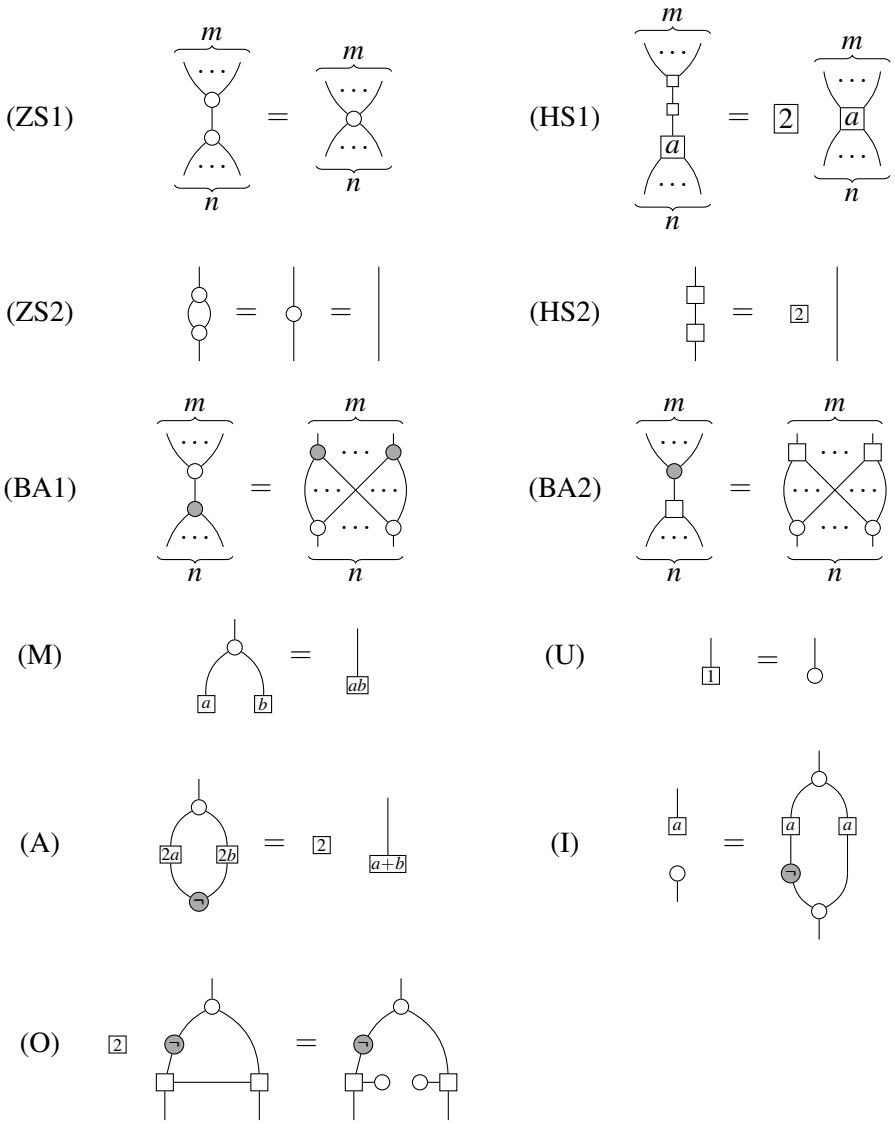


Figure 1: The rules of the ZH-calculus. m,n are nonnegative integers and a,b are arbitrary complex numbers. The right-hand sides of both *bialgebra* rules (BA1) and (BA2) are complete bipartite graphs on $(m+n)$ vertices, with an additional input or output for each vertex.

References

[1] Miriam Backens & Aleks Kissinger (2018): *ZH: A Complete Graphical Calculus for Quantum Computations Involving Classical Non-linearity*.

[2] 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.

[3] Cole Comfort (2020): *The ZX& calculus: A complete graphical calculus for classical circuits using spiders*.

[4] Stephen A. Cook (1971): *The Complexity of Theorem-Proving Procedures*. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, Association for Computing Machinery, New York, NY, USA, p. 151–158, doi:10.1145/800157.805047.

[5] Carsten Damm, Markus Holzer & Pierre McKenzie (2002): *The complexity of tensor calculus*. *computational complexity* 11(1), pp. 54–89, doi:10.1007/s00037-000-0170-4.

[6] Artur García-Sáez & José I. Latorre (2012): *An Exact Tensor Network for the 3SAT Problem*. *Quantum Info. Comput.* 12(3–4), p. 283–292.

[7] Johnnie Gray & Stefanos Kourtis (2020): *Hyper-optimized tensor network contraction*.

[8] Aleks Kissinger & John van de Wetering (2019): *PyZX: Large Scale Automated Diagrammatic Reasoning*.

[9] Stefanos Kourtis, Claudio Chamon, Eduardo R. Mucciolo & Andrei E. Ruckenstein (2019): *Fast counting with tensor networks*. *SciPost Phys.* 7, p. 60, doi:10.21468/SciPostPhys.7.5.060. Available at https://scipost.org/10.21468/SciPostPhys.7.5.060.

[10] M. Levin & C. P. Nave (2007): *Tensor Renormalization Group Approach to Two-Dimensional Classical Lattice Models*. *Phys. Rev. Lett.* 99(12), p. 120601, doi:10.1103/PhysRevLett.99.120601.

[11] Cristopher Moore & Stephan Mertens (2011): *The Nature of Computation*. Oxford University Press, Inc., USA, doi:10.1093/acprof:oso/9780199233212.001.0001.

[12] Pranay Patil, Stefanos Kourtis, Claudio Chamon, Eduardo R. Mucciolo & Andrei E. Ruckenstein (2019): *Obstacles to quantum annealing in a planar embedding of XORSAT*.

[13] Federico Ricci-Tersenghi (2010): *Being Glassy Without Being Hard to Solve*. *Science* 330(6011), pp. 1639–1640, doi:10.1126/science.1189804. Available at https://science.sciencemag.org/content/330/6011/1639.

[14] B. A. Trakhtenbrot (1984): *A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms*. *Annals of the History of Computing* 6(4), pp. 384–400, doi:10.1109/MAHC.1984.10036.

[15] Leslie G Valiant (1979): *The complexity of enumeration and reliability problems*. *SIAM Journal on Computing* 8(3), pp. 410–421, doi:10.1137/0208032.

[16] F. Verstraete & J. I. Cirac (2004): *Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions*. *arXiv:cond-mat/0407066*. Available at http://arxiv.org/abs/cond-mat/0407066.

[17] John van de Wetering & Sal Wolffs (2019): *Completeness of the Phase-free ZH-calculus*.