Constructing $\mathrm{NP}^{\#\mathrm{P}}$-complete problems and $\#\mathrm{P}$-hardness of circuit extraction in phase-free ZH calculus

Piotr Mitosek

School of Computer Science
University of Birmingham

QPL 2023, Paris

Overview

1. Background
2. Boolean formulae in ZH
3. Circuit extraction
4. $\mathrm{NP}^{\#\mathrm{P}}$-complete problems
5. Summary

Background

ZH and computational complexity

Some problems arising in ZH calculus are believed to be hard.

- Given a phase-free ZH diagram, can we find an equivalent circuit?
- Given two diagrams, are they equal?

This talk:

- Circuit extraction is $\#\mathrm{P}$-hard
- Two problems related to comparing diagrams are $\mathrm{NP}^{\#\mathrm{P}}$-complete

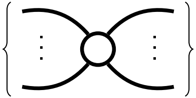
ZH and computational complexity

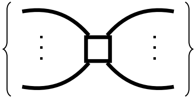
Some problems arising in ZH calculus are believed to be hard.

- Given a phase-free ZH diagram, can we find an equivalent circuit?
- Given two diagrams, are they equal?

This talk:

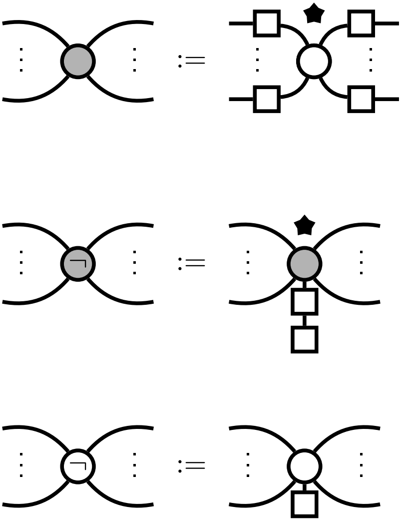
- Circuit extraction is $\#\mathrm{P}$-hard
- Two problems related to comparing diagrams are $\mathrm{NP}^{\#\mathrm{P}}$-complete

$[\![$ n  m $]\!]$ $= \begin{pmatrix} 1 & & & \\ & 0 & & \mathbf{0} \\ & & \ddots & \\ & \mathbf{0} & & 0 & \\ & & & & 1 \end{pmatrix}$ $(2^m \times 2^n$ matrix$)$

$[\![$ n  m $]\!]$ $= \begin{pmatrix} 1 & & & \\ & 1 & & \mathbf{1} \\ & & \ddots & \\ & \mathbf{1} & & 1 & \\ & & & & -1 \end{pmatrix}$ $(2^m \times 2^n$ matrix$)$

$[\![$  $]\!] = \frac{1}{2}$ (scalar)

Phase-free ZH calculus



NP – problems solvable by a polynomial time non-deterministic Turing Machine (NDTM).

SAT

Input: *Variables* $x_1, \ldots, x_n$ *and a boolean formula* ϕ *on (some of)* $x_1, \ldots, x_n$
Output: $True$ *when* ϕ *is satisfiable,* $False$ *otherwise.*

For example, $\mathrm{SAT}\left((x_1 \wedge x_2) \wedge (x_1 \wedge \neg x_3)\right) = True$.

Computational Complexity – $\#\mathrm{P}$

$\#\mathrm{P}$ – problems of the form: given a polynomial time NDTM and an input a, compute how many runs accept a.

$\#\mathrm{SAT}$

Input: *Variables* $x_1, \ldots, x_n$ *and a boolean formula* ϕ *on (some of)* $x_1, \ldots, x_n$
Output: *Number of satisfying assignments of* ϕ

For example, $\#\mathrm{SAT}\left((x_1 \wedge x_2) \wedge (x_1 \wedge \neg x_3)\right) = 1$.

Oracles

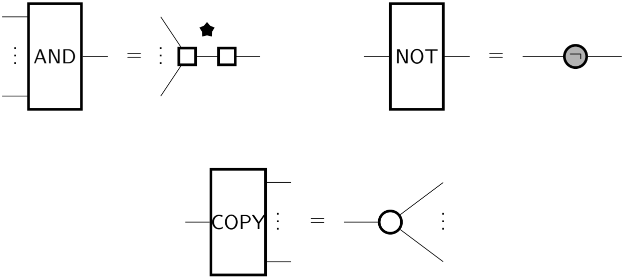
$\mathrm{NP}^{\#\mathrm{SAT}}$ – problems solvable by a polytime NDTM with access to oracle for $\#\mathrm{SAT}$. By completeness, this class equals $\mathrm{NP}^{\#\mathrm{P}}$.
An oracle call counts as a single step of computation.

Boolean formulae in ZH

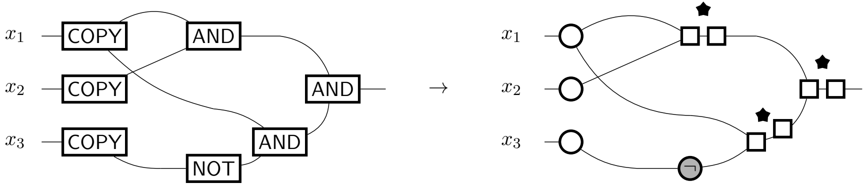
Logic in ZH

$[\![$  $]\!] = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \rightarrow True$

$[\![$  $]\!] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \rightarrow False$



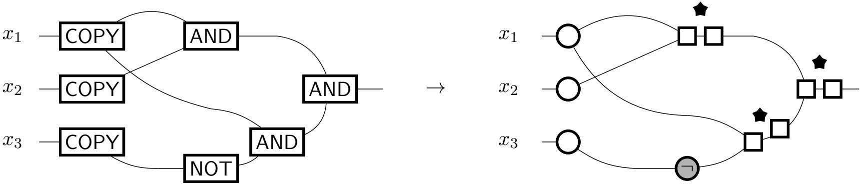
$(x_1 \wedge x_2) \wedge (x_1 \wedge \neg x_3)$ in ZH:



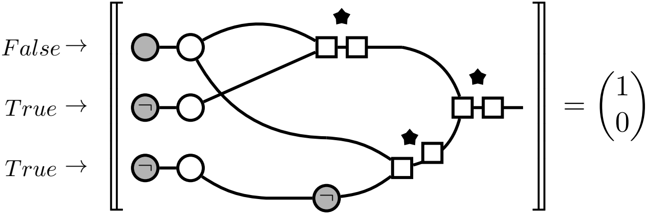
One assignment:

Boolean formulae in ZH

$(x_1 \wedge x_2) \wedge (x_1 \wedge \neg x_3)$ in ZH:

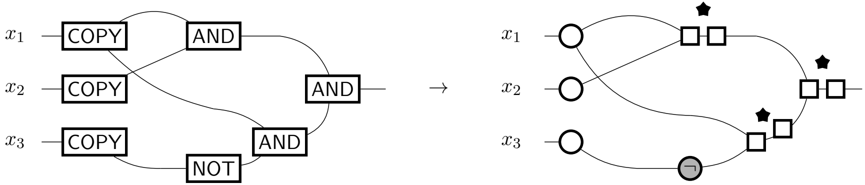


One assignment:

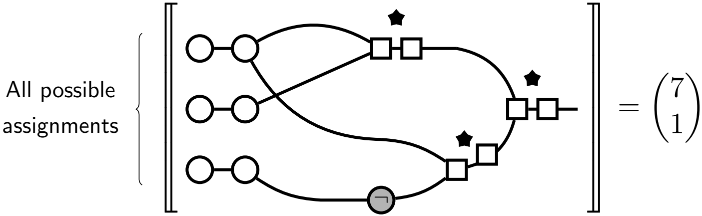


Boolean formulae in ZH

$(x_1 \wedge x_2) \wedge (x_1 \wedge \neg x_3)$ in ZH:



All assignments:



Circuit extraction

Circuit extraction hardness

Circuit Extraction

Input: *A phase-free ZH diagram D proportional to a unitary and set of unitaries $\mathcal{G}$ acting on $O(1)$ qubits.*

Output: *A polynomial (in size of D) circuit C, constructed from $\mathcal{G}$, expressing unitary proportional to D, or a message that no such circuit exists.*

Theorem

Circuit Extraction *is* $\#\mathrm{P}$*-hard.*

Proof idea: reduce from $\#\mathrm{SAT}$, i.e. show $\#\mathrm{SAT} \in \mathrm{FP}^{\mathrm{Circuit\ Extraction}}$.

Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): Circuit Extraction for ZX-diagrams Can Be #P-hard. 10.4230/LIPIcs.ICALP.2022.119

Circuit extraction hardness

Circuit Extraction

Input: *A phase-free ZH diagram* D *proportional to a unitary and set of unitaries* $\mathcal{G}$ *acting on* $O(1)$ *qubits.*

Output: *A polynomial (in size of* D*) circuit* C*, constructed from* $\mathcal{G}$*, expressing unitary proportional to* D*, or a message that no such circuit exists.*

Theorem

Circuit Extraction *is* #P*-hard.*

Proof idea: reduce from #SAT, i.e. show $\#\mathrm{SAT} \in \mathrm{FP}^{\text{Circuit Extraction}}$.

Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): Circuit Extraction for ZX-diagrams Can Be #P-hard. 10.4230/LIPIcs.ICALP.2022.119

Circuit extraction hardness

Circuit Extraction

Input: *A phase-free ZH diagram D proportional to a unitary and set of unitaries $\mathcal{G}$ acting on $O(1)$ qubits.*

Output: *A polynomial (in size of D) circuit C, constructed from $\mathcal{G}$, expressing unitary proportional to D, or a message that no such circuit exists.*

Theorem

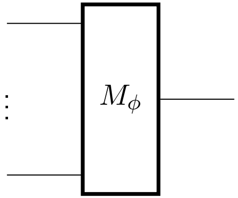
Circuit Extraction *is* $\#\mathrm{P}$*-hard.*

Proof idea: reduce from $\#\mathrm{SAT}$, i.e. show $\#\mathrm{SAT} \in \mathrm{FP}^{\text{Circuit Extraction}}$.

Niel de Beaudrap, Aleks Kissinger & John van de Wetering (2022): Circuit Extraction for ZX-diagrams Can Be #P-hard. 10.4230/LIPIcs.ICALP.2022.119

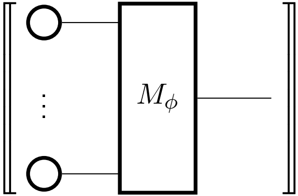
Circuit extraction hardness – proof

- Given a boolean formula ϕ, we encode it in phase-free ZH



Circuit extraction hardness – proof

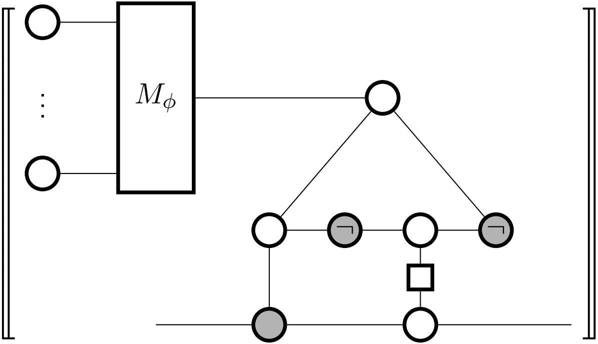
- Given a boolean formula ϕ, we encode it in phase-free ZH
- Let a_0, a_1 be the numbers of unsatisfying and satisfying assignments of ϕ. Then, we have:



$$= \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$$

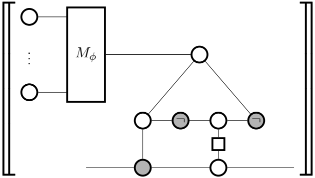
Circuit extraction hardness – proof

- Given a boolean formula ϕ, we encode it in phase-free ZH
- Let a_0, a_1 be the numbers of unsatisfying and satisfying assignments of ϕ. Then, we have:



$$= \begin{pmatrix} a_0 & a_1 \\ a_1 & -a_0 \end{pmatrix}$$

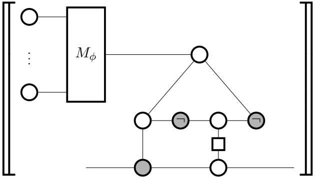
Circuit extraction hardness – proof finish



$$= \begin{pmatrix} a_0 & a_1 \\ a_1 & -a_0 \end{pmatrix}$$

- Suppose, we have a circuit C proportional to the above diagram
- We can approximate the matrix representation of C
- Then, we can find a_1, i.e. solve initial instance of $\#\mathrm{SAT}$
- Therefore $\#\mathrm{SAT} \in \mathrm{FP}^{\mathrm{Circuit\ Extraction}}$.

Circuit extraction hardness – proof finish



$$= \begin{pmatrix} a_0 & a_1 \\ a_1 & -a_0 \end{pmatrix}$$

- Suppose, we have a circuit C proportional to the above diagram
- We can approximate the matrix representation of C
- Then, we can find a_1, i.e. solve initial instance of $\#\mathrm{SAT}$
- Therefore $\#\mathrm{SAT} \in \mathrm{FP}^{\mathrm{Circuit\ Extraction}}$.

$\mathrm{NP}^{\#\mathrm{P}}$-complete problems

More dangling edges

- A diagram with k dangling edges has a matrix representation of the size 2^k.
- Given a diagram D, finding a matrix entry in $[\![D]\!]$ on some given position is $\#\mathrm{P}$-hard and within $\mathrm{FP}^{\#\mathrm{P}}$.
- Informally: given D, checking some property of all entries of $[\![D]\!]$ could be $\mathrm{NP}^{\#\mathrm{P}}$-hard (or $\mathrm{coNP}^{\#\mathrm{P}}$-hard).

More dangling edges

- A diagram with k dangling edges has a matrix representation of the size 2^k.
- Given a diagram D, finding a matrix entry in $[\![D]\!]$ on some given position is $\#\mathrm{P}$-hard and within $\mathrm{FP}^{\#\mathrm{P}}$.
- Informally: given D, checking some property of all entries of $[\![D]\!]$ could be $\mathrm{NP}^{\#\mathrm{P}}$-hard (or $\mathrm{coNP}^{\#\mathrm{P}}$-hard).

Comparing diagrams

Comparing Diagrams

Input: *two diagrams D_1, D_2 with matching dangling edges.*
Output: $True$ *if* $[\![D_1]\!] = [\![D_2]\!]$ *and* $False$ *otherwise.*

Upper bound $\mathrm{coNP}^{\#\mathrm{P}}$ idea:
Non-deterministically choose a position in matrix representations of D_1 and D_2. Using the oracle, compute entries on such position e_1 and e_2. Reject if $e_1 \neq e_2$ and accept otherwise.

Comparing diagrams

Comparing Diagrams

Input: *two diagrams D_1, D_2 with matching dangling edges.*

Output: $True$ *if* $[\![D_1]\!] = [\![D_2]\!]$ *and* $False$ *otherwise.*

Upper bound $\mathrm{coNP}^{\#\mathrm{P}}$ idea:

Non-deterministically choose a position in matrix representations of D_1 and D_2. Using the oracle, compute entries on such position e_1 and e_2. Reject if $e_1 \neq e_2$ and accept otherwise.

$\mathrm{NP}^{\#\mathrm{P}}$-complete problems

State Equality

Input: *Two phase-free ZH diagrams D_1 and D_2 with n dangling edges each.*
Output: *$True$ if there exists a state $|V\rangle$ in n qubits computational basis such that D_1 and D_2 applied to $|V\rangle$ result in the same scalar, and $False$ otherwise.*

Comparing Diagrams:
Do matrix representations agree on **all** positions?

State Equality:
Do matrix representations agree on **some** position?

Theorem

State Equality *is* $\mathrm{NP}^{\#\mathrm{P}}$*-complete.*

$\mathrm{NP}^{\#\mathrm{P}}$-complete problems

State Equality

Input: *Two phase-free ZH diagrams D_1 and D_2 with n dangling edges each.*

Output: *$True$ if there exists a state $|V\rangle$ in n qubits computational basis such that D_1 and D_2 applied to $|V\rangle$ result in the same scalar, and $False$ otherwise.*

Comparing Diagrams:
Do matrix representations agree on **all** positions?

State Equality:
Do matrix representations agree on **some** position?

Theorem

State Equality *is* $\mathrm{NP}^{\#\mathrm{P}}$*-complete.*

$\mathrm{NP}^{\#\mathrm{P}}$-complete problems

State Equality

Input: *Two phase-free ZH diagrams D_1 and D_2 with n dangling edges each.*

Output: *$True$ if there exists a state $|V\rangle$ in n qubits computational basis such that D_1 and D_2 applied to $|V\rangle$ result in the same scalar, and $False$ otherwise.*

Comparing Diagrams:
Do matrix representations agree on **all** positions?

State Equality:
Do matrix representations agree on **some** position?

Theorem

State Equality *is* $\mathrm{NP}^{\#\mathrm{P}}$*-complete.*

$\mathrm{NP}^{\#\mathrm{P}}$-hardness proof overview

- Idea: reduce any problem A in $\mathrm{NP}^{\#\mathrm{P}}$ to $\mathrm{State\ Equality}$.
- Take polytime NDTM $\mathcal{M}$ with $\#\mathrm{SAT}$ oracle that recognizes A
- Express run of $\mathcal{M}$ on an input a as a boolean formula with extra conditions checking oracle uses
- Reduce boolean formula with oracle conditions to a pair of diagrams forming $\mathrm{State\ Equality}$ instance

$\mathrm{NP}^{\#\mathrm{P}}$-hardness proof overview

- Idea: reduce any problem A in $\mathrm{NP}^{\#\mathrm{P}}$ to $\mathrm{State\ Equality}$.
- Take polytime NDTM $\mathcal{M}$ with $\#\mathrm{SAT}$ oracle that recognizes A
- Express run of $\mathcal{M}$ on an input a as a boolean formula with extra conditions checking oracle uses
- Reduce boolean formula with oracle conditions to a pair of diagrams forming $\mathrm{State\ Equality}$ instance

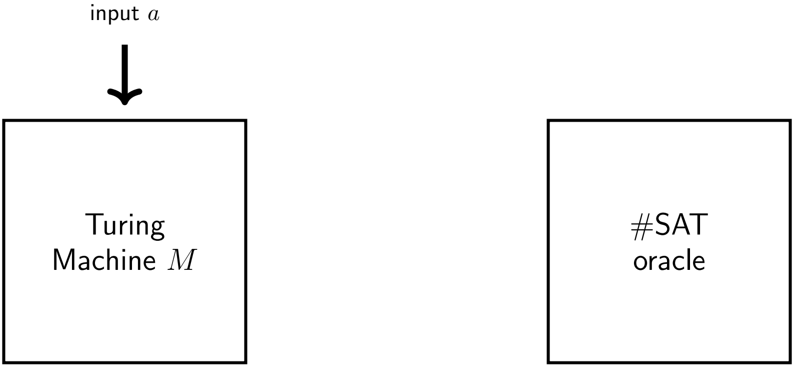
$\mathrm{NP}^{\#\mathrm{P}}$-hardness proof overview

- Idea: reduce any problem A in $\mathrm{NP}^{\#\mathrm{P}}$ to $\mathrm{State\ Equality}$.
- Take polytime NDTM $\mathcal{M}$ with $\#\mathrm{SAT}$ oracle that recognizes A
- Express run of $\mathcal{M}$ on an input a as a boolean formula with extra conditions checking oracle uses
- Reduce boolean formula with oracle conditions to a pair of diagrams forming $\mathrm{State\ Equality}$ instance

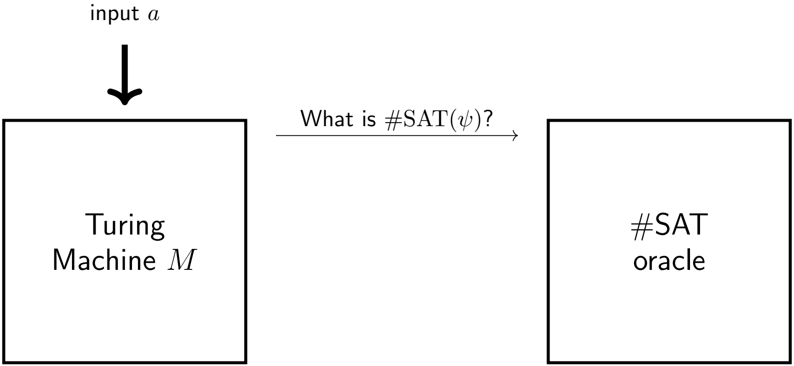
$\mathrm{NP}^{\#\mathrm{P}}$-hardness proof overview

- Idea: reduce any problem A in $\mathrm{NP}^{\#\mathrm{P}}$ to $\mathrm{State\ Equality}$.
- Take polytime NDTM $\mathcal{M}$ with $\#\mathrm{SAT}$ oracle that recognizes A
- Express run of $\mathcal{M}$ on an input a as a boolean formula with extra conditions checking oracle uses
- Reduce boolean formula with oracle conditions to a pair of diagrams forming $\mathrm{State\ Equality}$ instance

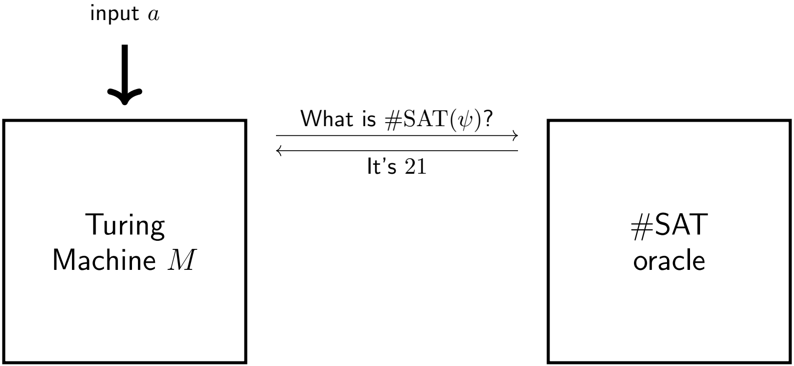
How does $\mathcal{M}$ work?



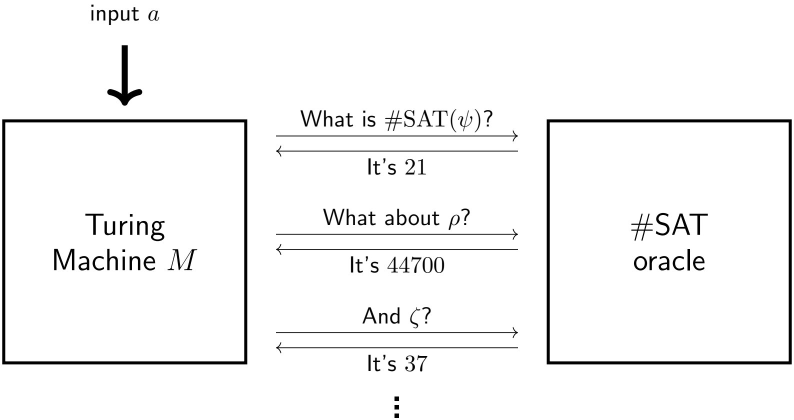
How does $\mathcal{M}$ work?



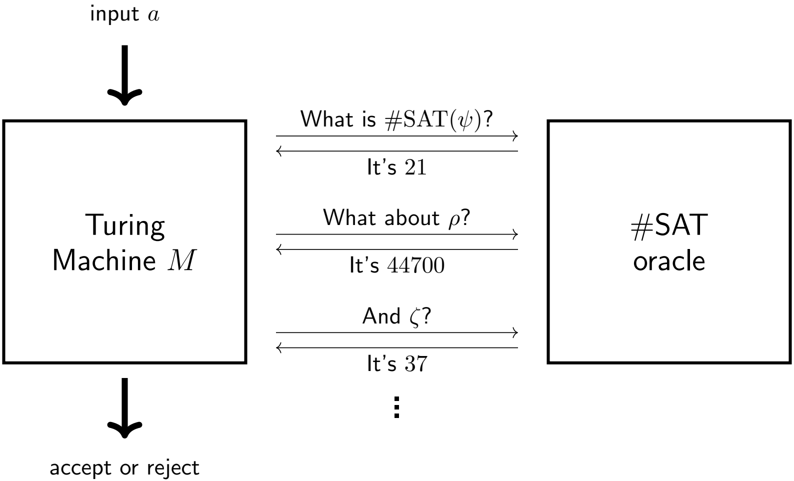
How does $\mathcal{M}$ work?



How does $\mathcal{M}$ work?



How does $\mathcal{M}$ work?



Oracle calls

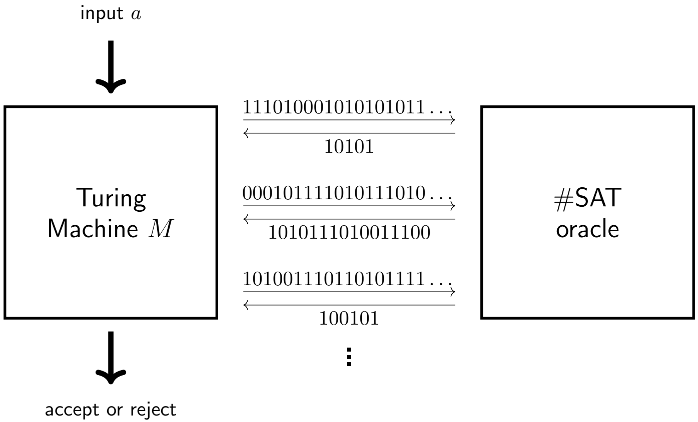
- TM $\mathcal{M}$ must communicate with its $\#\mathrm{SAT}$ oracle.
- This can be done by passing a sequence of 0s and 1s that encode a boolean formula.

Oracle calls

- TM $\mathcal{M}$ must communicate with its $\#\mathrm{SAT}$ oracle.
- This can be done by passing a sequence of 0s and 1s that encode a boolean formula.

Oracle calls

- TM $\mathcal{M}$ must communicate with its $\#\mathrm{SAT}$ oracle.
- This can be done by passing a sequence of 0s and 1s that encode a boolean formula.



Boolean formula and oracle conditions

Given input a, we can express run of $\mathcal{M}$ on a as a boolean formula, without verifying oracle uses

Boolean formula and oracle conditions

Given input a, we can express run of $\mathcal{M}$ on a as a boolean formula, without verifying oracle uses:

- Introduce variables like $T_{h,i,j,k}$ – $True$ iff i^{th} cell of h^{th} tape contains symbol j at k^{th} step of computation

Boolean formula and oracle conditions

Given input a, we can express run of $\mathcal{M}$ on a as a boolean formula, without verifying oracle uses:

- Introduce variables like $T_{h,i,j,k}$ – $True$ iff i^{th} cell of h^{th} tape contains symbol j at k^{th} step of computation
- Same for head position, state etc.

Boolean formula and oracle conditions

Given input a, we can express run of $\mathcal{M}$ on a as a boolean formula, without verifying oracle uses:

- Introduce variables like $T_{h,i,j,k}$ – $True$ iff i^{th} cell of h^{th} tape contains symbol j at k^{th} step of computation
- Same for head position, state etc.
- Combine into one formula ϕ_a, similar to the proof of the Cook-Levin theorem

Boolean formula and oracle conditions

Given input a, we can express run of $\mathcal{M}$ on a as a boolean formula, without verifying oracle uses:

- Introduce variables like $T_{h,i,j,k}$ – $True$ iff i^{th} cell of h^{th} tape contains symbol j at k^{th} step of computation
- Same for head position, state etc.
- Combine into one formula ϕ_a, similar to the proof of the Cook-Levin theorem
- The meaning behind ϕ_a is as follows:

  If ϕ_a is satisfied by some assignment V, then V encodes a path from initial configuration of $\mathcal{M}$ on input a, to an accepting configuration, without checking that oracle returned correct data.

Oracle conditions

To verify oracle uses, we add conditions $C_{a,1}, C_{a,2} \ldots$, where $C_{a,k}$ stands for:

> *If in k^{th} step an oracle is called on some input w, then in the $k+1^{th}$ step, $\mathcal{M}$ contains the result of running oracle on w.*

Oracle conditions can be combined into a single condition $\mathcal{C}_a$.

Theorem

ϕ_a and $\mathcal{C}_a$ can be simultaneously satisfied iff $\mathcal{M}$ accepts a.

Oracle conditions

To verify oracle uses, we add conditions $C_{a,1}, C_{a,2} \ldots$, where $C_{a,k}$ stands for:

> *If in k^{th} step an oracle is called on some input w, then in the $k+1^{th}$ step, $\mathcal{M}$ contains the result of running oracle on w.*

Oracle conditions can be combined into a single condition $\mathcal{C}_a$.

Theorem

ϕ_a and $\mathcal{C}_a$ can be simultaneously satisfied iff $\mathcal{M}$ accepts a.

Oracle conditions

To verify oracle uses, we add conditions $C_{a,1}, C_{a,2} \ldots$, where $C_{a,k}$ stands for:

> *If in k^{th} step an oracle is called on some input w, then in the $k+1^{th}$ step, $\mathcal{M}$ contains the result of running oracle on w.*

Oracle conditions can be combined into a single condition $\mathcal{C}_a$.

Theorem

ϕ_a and $\mathcal{C}_a$ can be simultaneously satisfied iff $\mathcal{M}$ accepts a.

Informal description

ϕ_a under valuation V means:

Does the run of $\mathcal{M}$ on a given by V result in an accepting configuration, ignoring the oracle?

$\mathcal{C}_a$ under valuation V means:

In run given by V, $\mathcal{M}$ asked for $\#\mathrm{SAT}(\psi), \#\mathrm{SAT}(\rho), \ldots$ and oracle returned ans. On its tapes, $\mathcal{M}$ wrote number num.
Does $ans = num$?

Informal description

ϕ_a under valuation V means:

Does the run of $\mathcal{M}$ on a given by V result in an accepting configuration, ignoring the oracle?

$\mathcal{C}_a$ under valuation V means:

In run given by V, $\mathcal{M}$ asked for $\#\mathrm{SAT}(\psi), \#\mathrm{SAT}(\rho), \dots$ and oracle returned ans. On its tapes, $\mathcal{M}$ wrote number num.
Does $ans = num$?

ZH encoding

- Given ϕ_a and $\mathcal{C}_a$ we construct $\mathrm{State\ Equality}$ instance, i.e. two diagrams D_1 and D_2.
- We already know how two encode ϕ_a.
- To encode $\mathcal{C}_a$ we construct to gadgets.

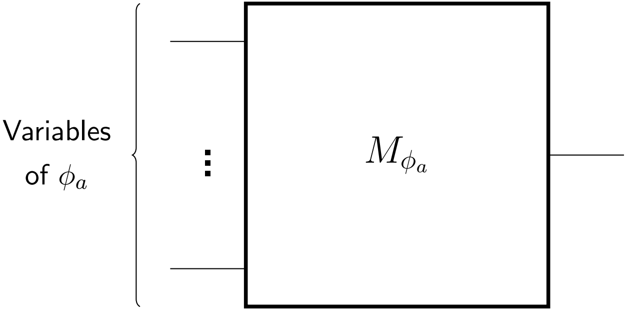
ZH encoding

- Given ϕ_a and $\mathcal{C}_a$ we construct $\mathrm{State\ Equality}$ instance, i.e. two diagrams D_1 and D_2.
- We already know how two encode ϕ_a.
- To encode $\mathcal{C}_a$ we construct to gadgets.

ZH encoding

- Given ϕ_a and $\mathcal{C}_a$ we construct $\mathrm{State\ Equality}$ instance, i.e. two diagrams D_1 and D_2.
- We already know how two encode ϕ_a.
- To encode $\mathcal{C}_a$ we construct to gadgets.

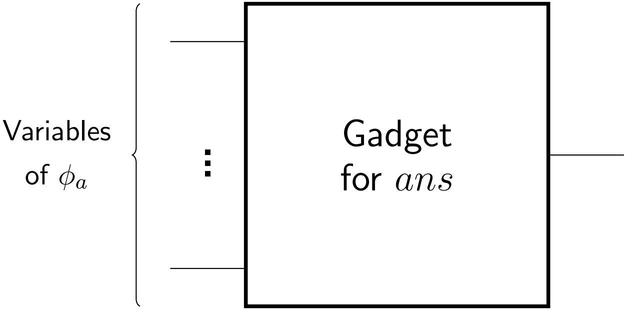
Formula ϕ_a



For $|V\rangle$ from computational basis:

$$[\![M_{\phi_a}]\!] |V\rangle = \begin{cases} |1\rangle, & \phi_a \text{ is satisfied under } V \\ |0\rangle, & \text{otherwise} \end{cases}$$

Oracle answers

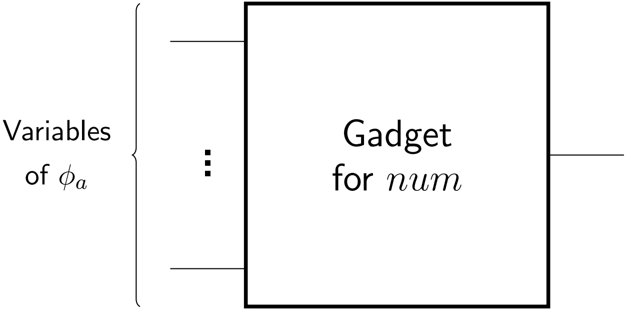


For $|V\rangle$ from computational basis:

$$[\![\text{Gadget for } ans]\!] \, |V\rangle = \left(2^P - ans\right) |0\rangle + ans \, |1\rangle$$

where ans is a concatenation of answers to the oracle queries.

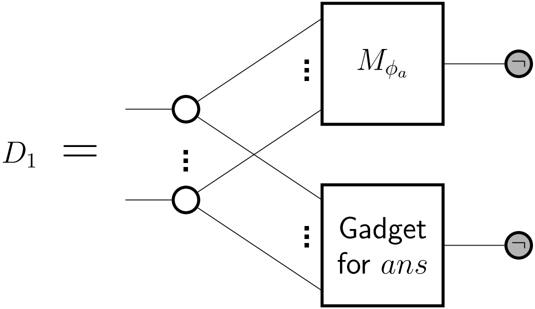
Numbers written as oracle answers

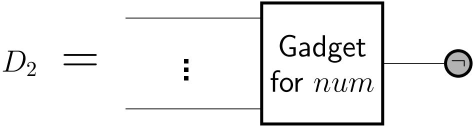


For $|V\rangle$ from computational basis:

$$[\![\text{Gadget for } num]\!] |V\rangle = \left(2^P - num\right)|0\rangle + num\,|1\rangle$$

where num is a concatenation of numbers written as oracle answers.





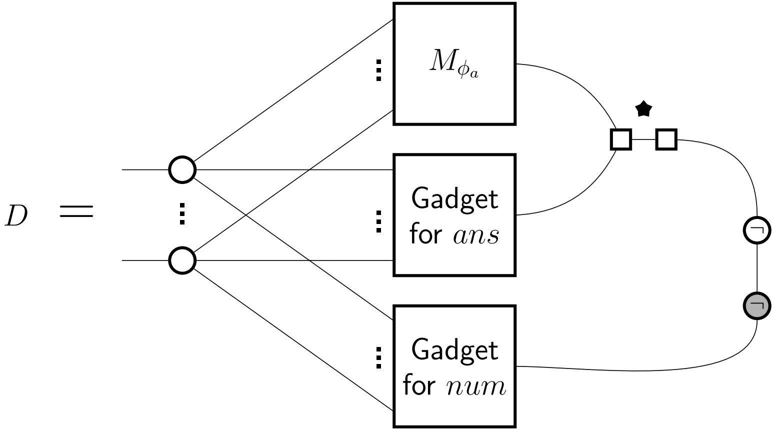
$$[\![D_1]\!]\,|V\rangle = \begin{cases} 0, & \phi_a \text{ unsatisfied under } V \\ ans, & \text{otherwise} \end{cases}$$

$$[\![D_2]\!]\,|V\rangle = num$$

Related problem

Matrix Entry

Input: *A phase-free ZH diagram* D *with* n *dangling edges and a number* $l \in \mathbb{Z}[\frac{1}{2}]$.
Output: $True$ *if matrix interpretation of* D *contains an entry equal to* l, *and* $False$ *otherwise.*



Summary

Summary and further work

- Connections of ZH and computational complexity
- Circuit Extraction is #P-hard
- State Equality and Matrix Entry are $\mathrm{NP}^{\#\mathrm{P}}$-complete

- Can we improve circuit extraction bounds?
- Does the Turing Machine approach work for other problems?

Summary and further work

- Connections of ZH and computational complexity
- Circuit Extraction is #P-hard
- State Equality and Matrix Entry are $NP^{\#P}$-complete

- Can we improve circuit extraction bounds?
- Does the Turing Machine approach work for other problems?

Summary and further work

- Connections of ZH and computational complexity
- Circuit Extraction is #P-hard
- State Equality and Matrix Entry are $\mathrm{NP}^{\#\mathrm{P}}$-complete

- Can we improve circuit extraction bounds?
- Does the Turing Machine approach work for other problems?

Summary and further work

- Connections of ZH and computational complexity
- $\text{Circuit Extraction}$ is $\#\text{P}$-hard
- State Equality and Matrix Entry are $\text{NP}^{\#\text{P}}$-complete

- Can we improve circuit extraction bounds?
- Does the Turing Machine approach work for other problems?

Thank you!