Quantum Circuit Analysis with Model Counting (#SAT)

Santiago $\longleftrightarrow$ Leiden 2026 • Lecture 2 of 3

Alfons Laarman

Leiden University
Santiago 2026

Universiteit Leiden
The Netherlands

Santiago 2026 — a three-lecture series

1. Demystifying Quantum Computing — *quantum computing 101*
2. **Quantum Circuit Analysis with #SAT** — *model counting*
3. Knowledge Representation for Quantum Computing — *knowledge representation*

Classical SAT solvers already crack open problems in math

MATHEMATICS

Maths proof smashes size record

Supercomputer produces a 200-terabyte proof — but is it really mathematics?

BY EVELYN LAMB

Three computer scientists have announced the largest-ever mathematical proof: a file that comes in at a whopping 200 terabytes, equivalent to all the digitized text held by the US Library of Congress. The researchers have created[1] a 68-gigabyte compressed version of their solution — which would allow anyone with about 30,000 hours of spare processor time to download, reconstruct and verify it — but a human could never hope to read through it.

Computer-assisted proofs too large to be directly verifiable by humans have become common, as have computers that solve problems in combinatorics — the study of finite discrete structures — by checking through umpteen individual cases. Still, "200 terabytes is unbelievable", says Ronald Graham, a mathematician at the University of California, San Diego. The previous record-holder is thought to be a 13-gigabyte proof[2], published in 2014.

The puzzle that required the 200-terabyte proof, called the Boolean Pythagorean triples problem, has troubled mathematicians for decades. In the 1980s, Graham offered a prize of US$100 for anyone who could solve it. (He presented the cheque to one of the three computer scientists, Marijn Heule of the University of Texas at Austin, last month.) The problem asks whether it is possible to colour each positive integer either red or blue, so that no trio of integers a, b and c that satisfy Pythagoras' famous equation $a^2+b^2=c^2$ are all the same colour. For example, for the Pythagorean triple 3, 4 and 5, if 3 and 5 were blue, 4 would have to be red.

MORE ONLINE

VIDEO

Neanderthals built cave structures — and no one knows why

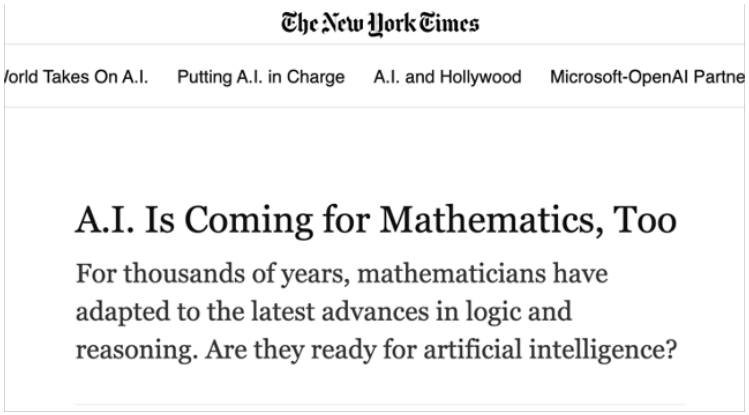
MORE NEWS

- Why the deal to fix US chemical laws matters go.nature.com/chemregs
- Cloud-seeding surprise could improve climate predictions go.nature.com/clouds

NATURE PODCAST

The genetics of moth evolution, Earth's core conundrum and Pluto's polygonal

Nature



The New York Times

World Takes On A.I. | Putting A.I. in Charge | A.I. and Hollywood | Microsoft-OpenAI Partne

A.I. Is Coming for Mathematics, Too

For thousands of years, mathematicians have adapted to the latest advances in logic and reasoning. Are they ready for artificial intelligence?

New York Times

[1] M. J. Heule and O. Kullmann, "The science of brute force," *Communications of the ACM*, vol. 60, 2017.

[2] M. J. Heule, O. Kullmann, and V. W. Marek, "Solving and verifying the boolean pythagorean triples problem via cube-and-conquer," in *International Conference on Theory and Applications of Satisfiability Testing*, Springer, 2016.

[3] M. Heule, "Schur number five," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

The *same* classical SAT / #SAT solvers
now simulate, verify, and synthesize quantum circuits.

How can that possibly work?

The *same* classical SAT / #SAT solvers
now simulate, verify, and synthesize quantum circuits.

How can that possibly work?

Spoiler: *one* Boolean encoding — going quantum only changes the weights.

SAT-based Quantum Circuit Simulation

Encoding Quantum Circuit Semantics in SAT

- SAT encoding of Clifford circuits [1] ✓scalable / ✗non-universal

[1] L. Berent, L. Burgholzer, and R. Wille, "Towards a SAT Encoding for Quantum Circuits," in *SAT*, vol. 236, 2022.

[2] F. Bauer-Marquart, S. Leue, and C. Schilling, "SymQV: Automated symbolic verification of quantum programs," in *Formal Methods*, 2023.

[3] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

SAT-based Quantum Circuit Simulation

Encoding Quantum Circuit Semantics in SAT

- SAT encoding of Clifford circuits [1] ✓scalable / ✗non-universal
- SMT encoding of universal circuits [2] ✗non-scalable / ✓universal

[1] L. Berent, L. Burgholzer, and R. Wille, "Towards a SAT Encoding for Quantum Circuits," in *SAT*, vol. 236, 2022.

[2] F. Bauer-Marquart, S. Leue, and C. Schilling, "SymQV: Automated symbolic verification of quantum programs," in *Formal Methods*, 2023.

[3] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

SAT-based Quantum Circuit Simulation

Encoding Quantum Circuit Semantics in SAT

- SAT encoding of Clifford circuits [1] ✓scalable / ✗non-universal
- SMT encoding of universal circuits [2] ✗non-scalable / ✓universal
- #SAT encoding of universal circuits [3] ✓scalable / ✓universal

[1] L. Berent, L. Burgholzer, and R. Wille, "Towards a SAT Encoding for Quantum Circuits," in *SAT*, vol. 236, 2022.

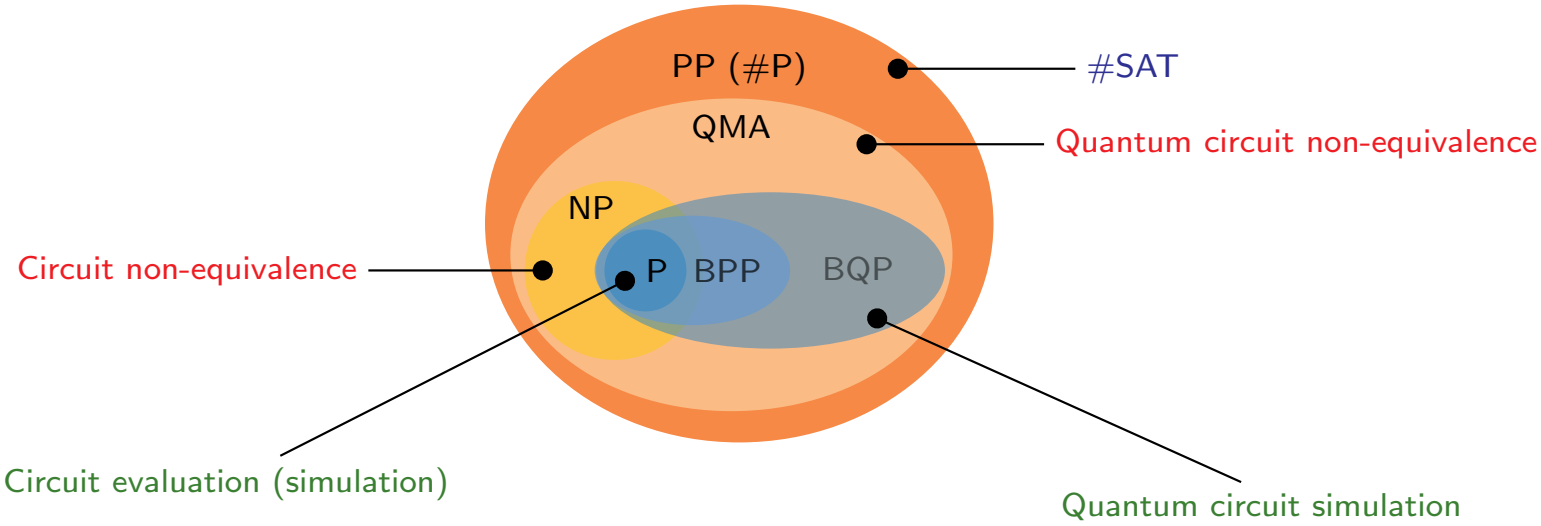
[2] F. Bauer-Marquart, S. Leue, and C. Schilling, "SymQV: Automated symbolic verification of quantum programs," in *Formal Methods*, 2023.

[3] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

SAT-based Quantum Circuit Simulation

Encoding Quantum Circuit Semantics in SAT

- SAT encoding of Clifford circuits [1] ✓scalable / ✗non-universal
- SMT encoding of universal circuits [2] ✗non-scalable / ✓universal
- #SAT encoding of universal circuits [3] ✓scalable / ✓universal



[1] L. Berent, L. Burgholzer, and R. Wille, "Towards a SAT Encoding for Quantum Circuits," in *SAT*, vol. 236, 2022.

[2] F. Bauer-Marquart, S. Leue, and C. Schilling, "SymQV: Automated symbolic verification of quantum programs," in *Formal Methods*, 2023.

[3] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

Roadmap

- SAT and #SAT (model counting)
- Encoding quantum circuits in logic
- Application: Simulation, Equivalence checking, Verification, Synthesis
- Different encodings and performance

SAT = DPLL + CDCL

```
1:  procedure DPLL(φ in CNF)
2:      UnitPropagation(φ)
3:      if φ is empty CNF then
4:          return SAT
5:      if empty clause in φ then
6:          return UNSAT
7:      Select x ∈ vars(φ)
8:      Select v ∈ {0, 1}
9:      if DPLL(φ[x := v]) then
10:         return SAT
11:     else
12:         return DPLL(φ[x := 1 − v])
```

[DPLL = Davis, Putnam, Logemann & Loveland]

Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1},\ \overbrace{1\bar{3}}^{c_2},\ \overbrace{234}^{c_3},\ \overbrace{\bar{4}\bar{5}}^{c_4},\ \overbrace{8\bar{4}\bar{6}}^{c_5},\ \overbrace{56}^{c_6},\ \cdots\}$$

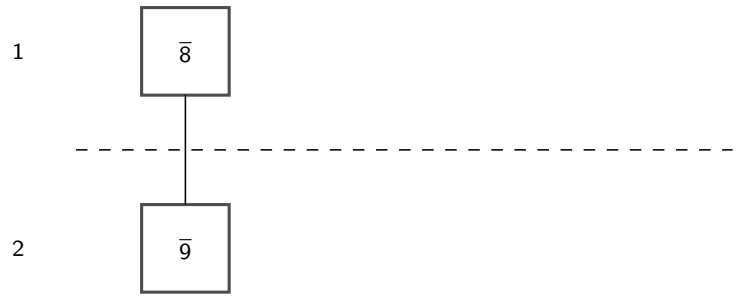
Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1},\ \overbrace{1\bar{3}}^{c_2},\ \overbrace{234}^{c_3},\ \overbrace{\bar{4}\bar{5}}^{c_4},\ \overbrace{8\bar{4}\bar{6}}^{c_5},\ \overbrace{56}^{c_6},\ \cdots\}$$

$$1 \qquad \boxed{\bar{8}}$$

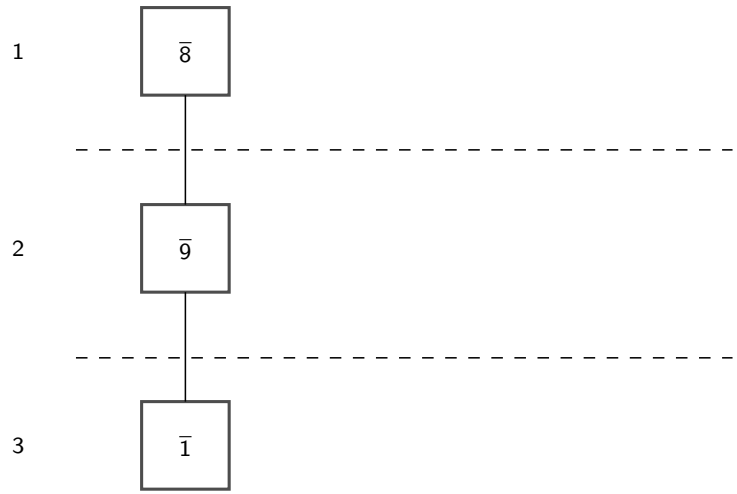
Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1}, \overbrace{1\bar{3}}^{c_2}, \overbrace{234}^{c_3}, \overbrace{\bar{4}\bar{5}}^{c_4}, \overbrace{8\bar{4}\bar{6}}^{c_5}, \overbrace{56}^{c_6}, \cdots\}$$



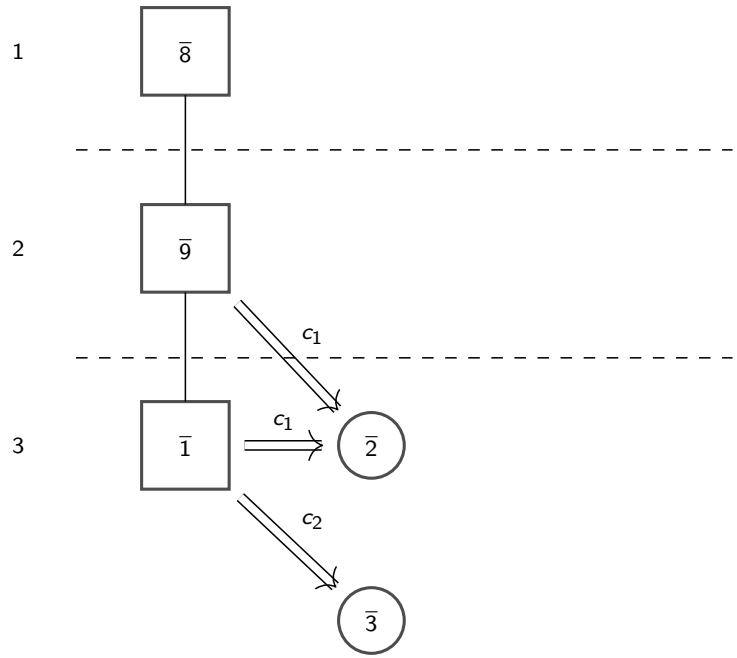
Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1}, \overbrace{1\bar{3}}^{c_2}, \overbrace{234}^{c_3}, \overbrace{\bar{4}\bar{5}}^{c_4}, \overbrace{8\bar{4}\bar{6}}^{c_5}, \overbrace{56}^{c_6}, \cdots\}$$



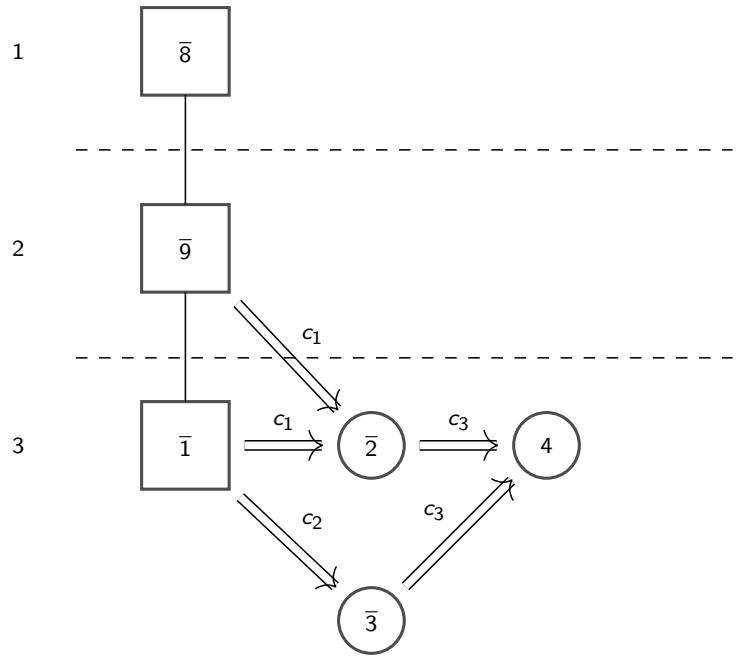
Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1},\ \overbrace{1\bar{3}}^{c_2},\ \overbrace{234}^{c_3},\ \overbrace{\bar{4}\bar{5}}^{c_4},\ \overbrace{8\bar{4}\bar{6}}^{c_5},\ \overbrace{56}^{c_6},\ \cdots\}$$



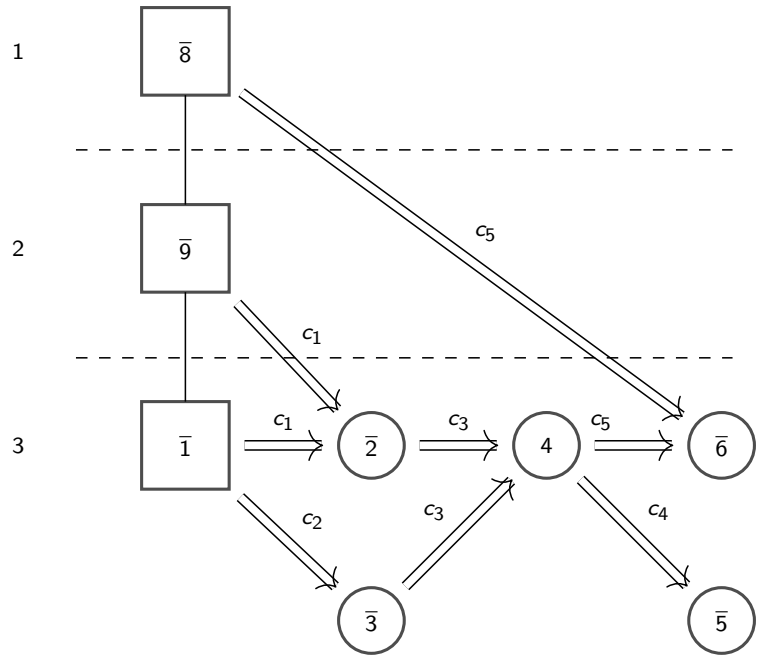
Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1},\ \overbrace{1\bar{3}}^{c_2},\ \overbrace{234}^{c_3},\ \overbrace{\bar{4}\bar{5}}^{c_4},\ \overbrace{8\bar{4}\bar{6}}^{c_5},\ \overbrace{56}^{c_6},\ \cdots\}$$

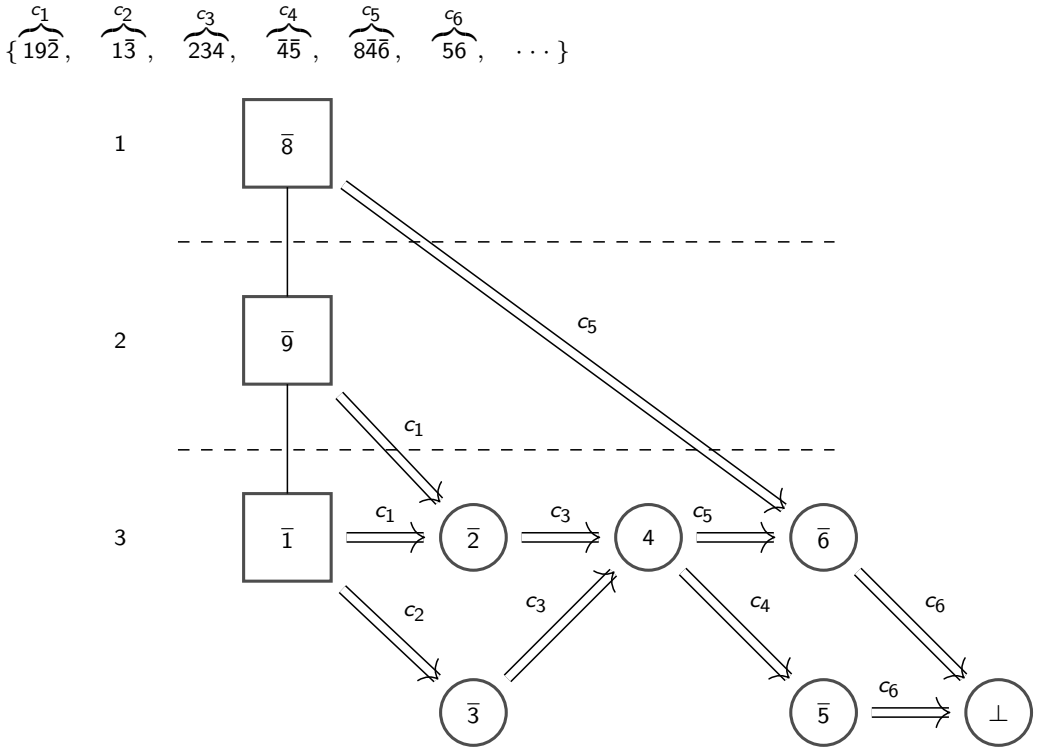


Implication Graph

$$\{\overbrace{19\bar{2}}^{c_1},\ \overbrace{1\bar{3}}^{c_2},\ \overbrace{234}^{c_3},\ \overbrace{\bar{4}\bar{5}}^{c_4},\ \overbrace{8\bar{4}\bar{6}}^{c_5},\ \overbrace{56}^{c_6},\ \cdots\}$$



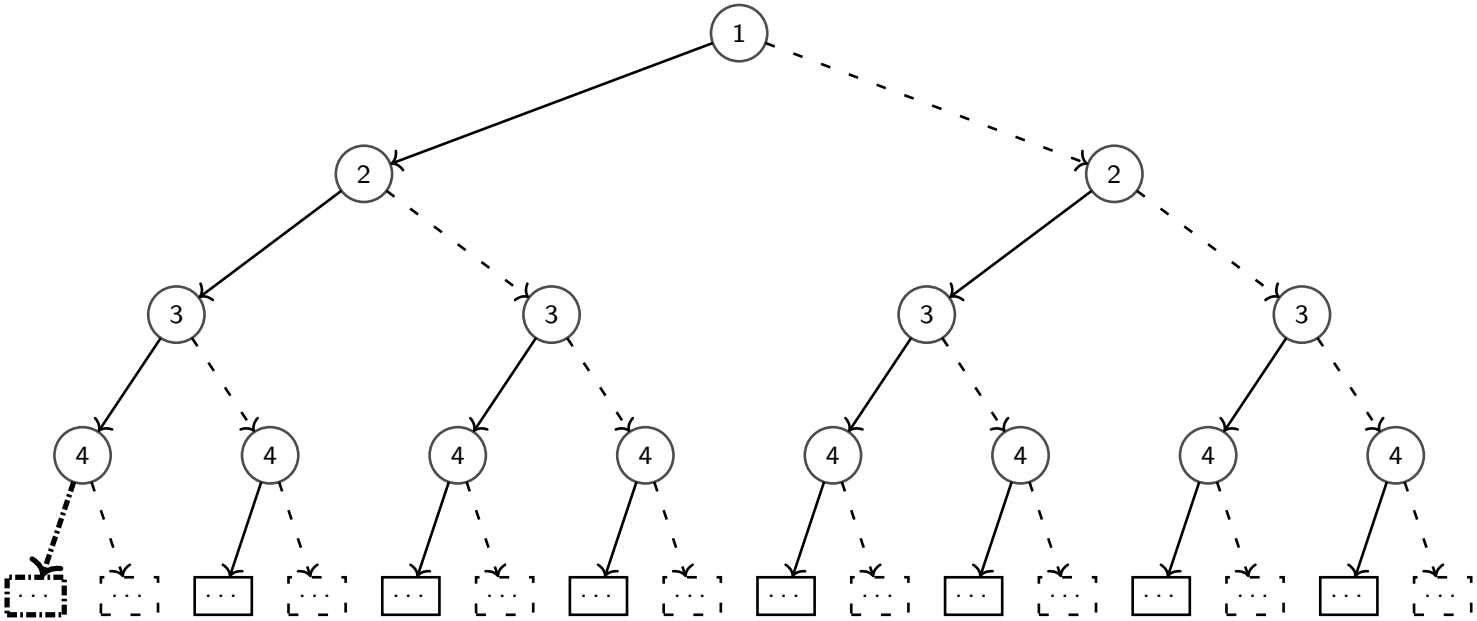
Implication Graph



We may 'learn' a clause $C = 5 \vee 6$ or $C = \overline{4} \vee 6$

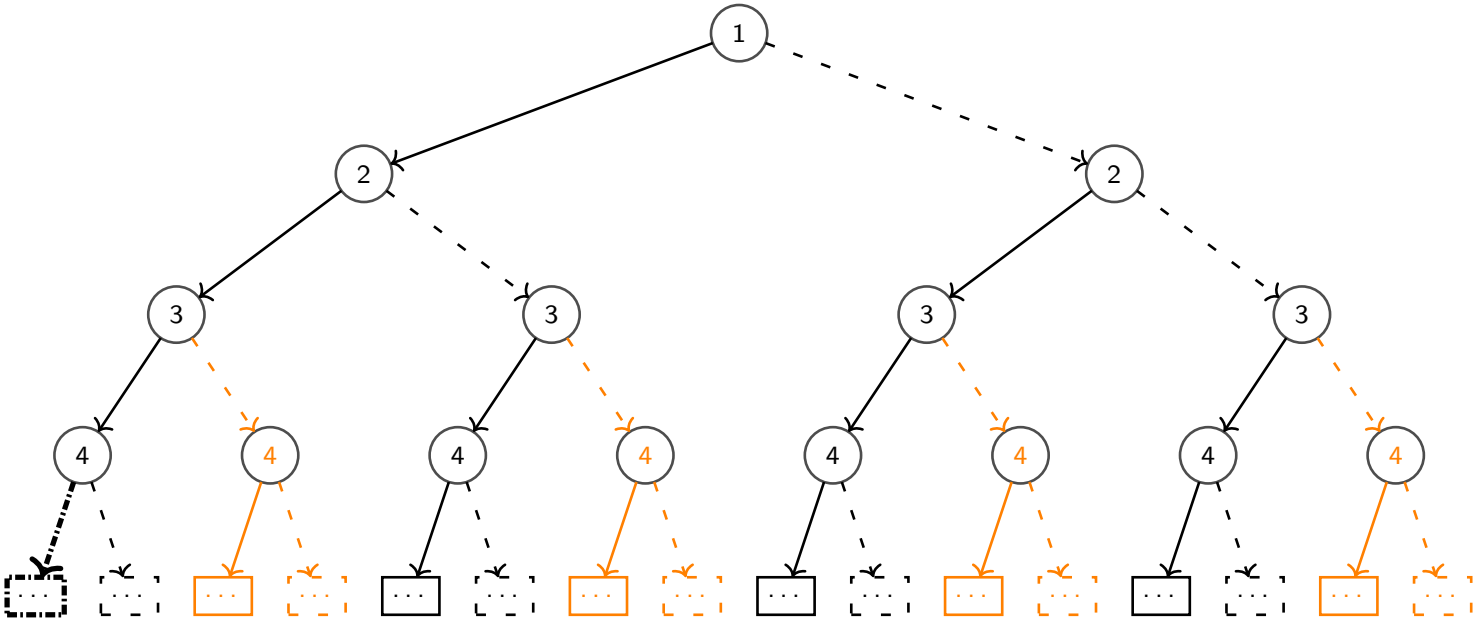
Constraint-Driven Clause Learning

$\varphi := \varphi \wedge c$ where c is a 'learned clause'



Constraint-Driven Clause Learning

$\varphi := \varphi \wedge c$ where c is a 'learned clause'



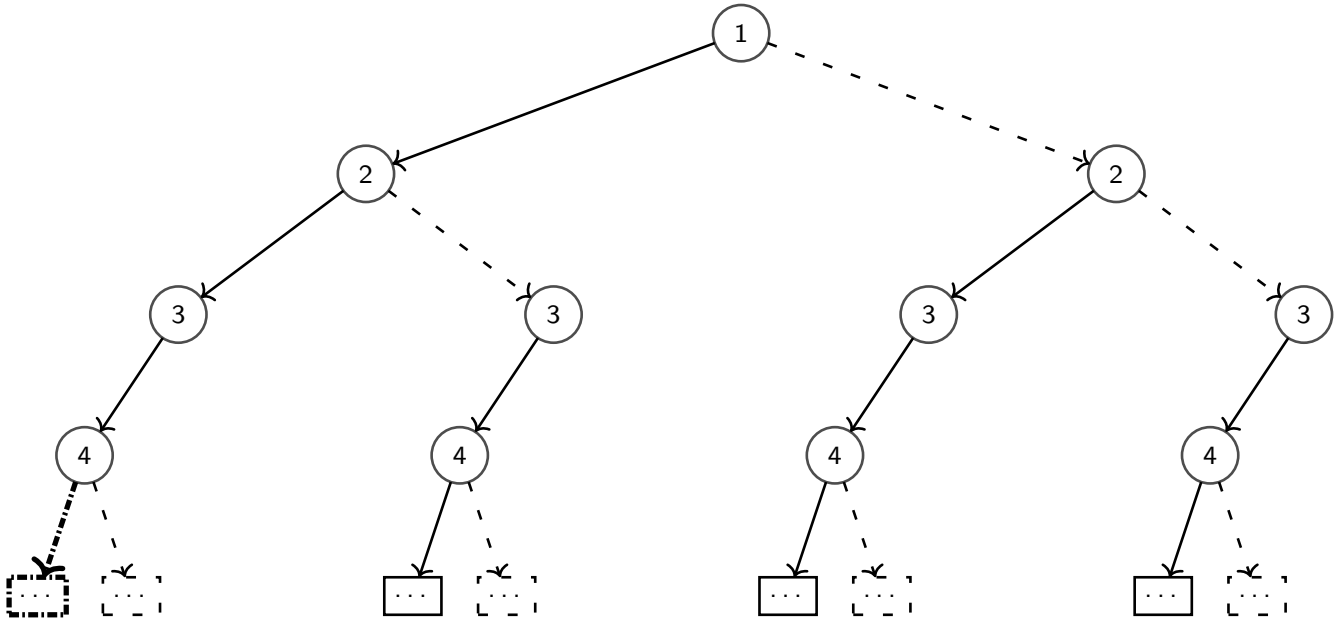
Proof theoretical results show exponential gains.

[1]

[1] J. M. Silva and K. A. Sakallah, "GRASP-a new search algorithm for satisfiability," in *Proceedings of International Conference on Computer Aided Design*, IEEE, 1996.

Constraint-Driven Clause Learning

$\varphi := \varphi \wedge c$ where c is a 'learned clause'

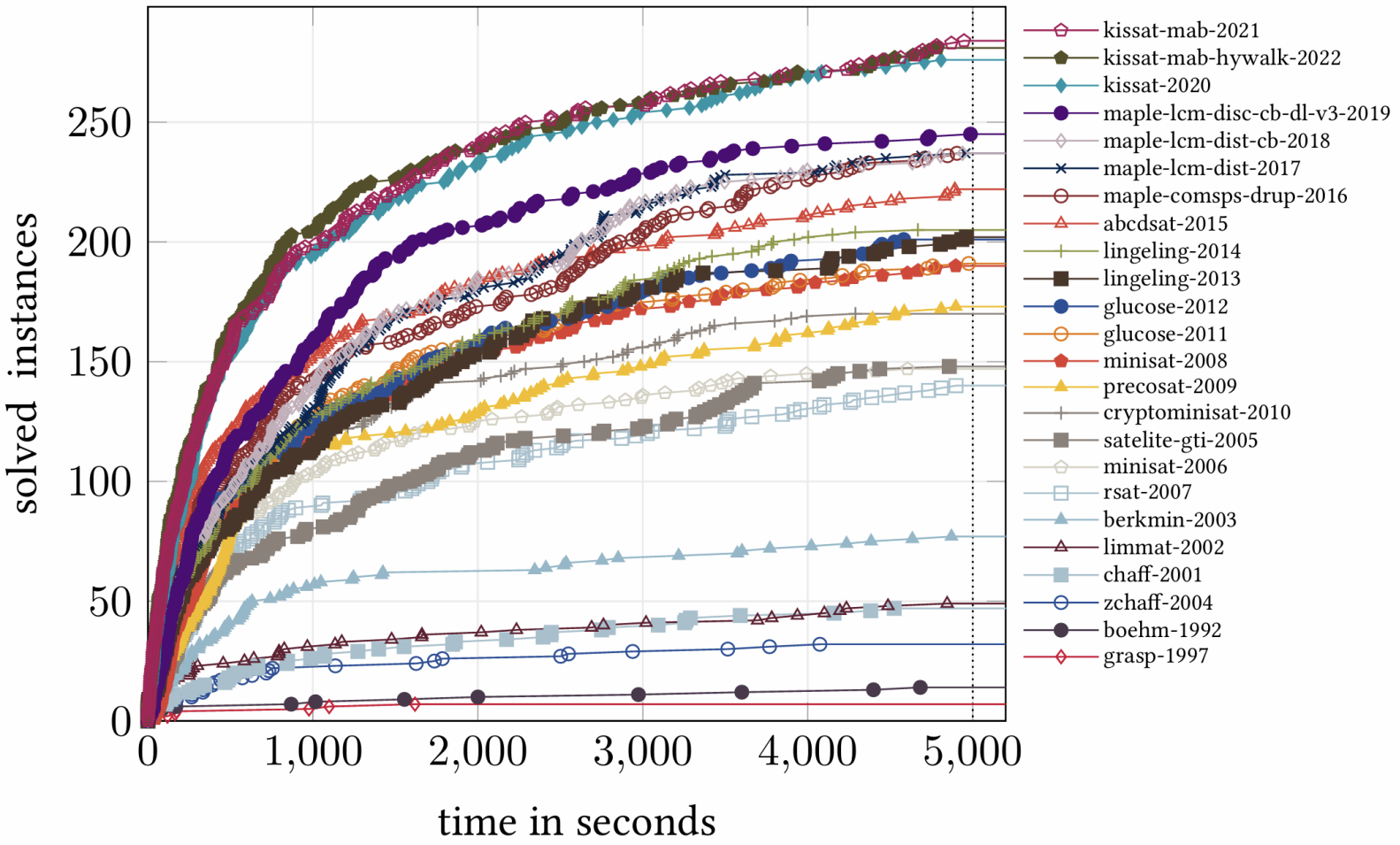


Proof theoretical results show exponential gains.

[1]

[1] J. M. Silva and K. A. Sakallah, "GRASP-a new search algorithm for satisfiability," in *Proceedings of International Conference on Computer Aided Design*, IEEE, 1996.

SAT Solvers Are Fast in Practice



Weighted Model Counting (#SAT)

Model counting (#SAT) is the counting version (#P) of satisfiability (SAT).

Weighted Model Counting (#SAT)

Model counting (#SAT) is the counting version (#P) of satisfiability (SAT).

Example (Weighted Model Counting)

Formula: $F = a \vee b$

Weights: $W(a) = \frac{1}{3}$,
$W(\bar{a}) = \frac{2}{3}$,
$W(b) = W(\bar{b}) = 1$

Weighted Model Counting (#SAT)

Model counting (#SAT) is the counting version (#P) of satisfiability (SAT).

Example (Weighted Model Counting)

Formula: $F = a \vee b$

Weights: $W(a) = \frac{1}{3}$,
$W(\bar{a}) = \frac{2}{3}$,
$W(b) = W(\bar{b}) = 1$

Satisfying assignments:
$SAT(F) = \{ \{ a \leftarrow 0, b \leftarrow 1 \}, \{ a \leftarrow 1, b \leftarrow 0 \}, \{ a \leftarrow 1, b \leftarrow 1 \} \} = \{ \bar{a}b, a\bar{b}, ab \}$

$\bar{a}\bar{b}$	—
$\bar{a}b$	$2/3 \cdot 1$
$a\bar{b}$	$1/3 \cdot 1$
ab	$1/3 \cdot 1$

Weighted Model Counting (#SAT)

Model counting (#SAT) is the counting version (#P) of satisfiability (SAT).

Example (Weighted Model Counting)

Formula: $F = a \vee b$

Weights: $W(a) = \frac{1}{3}$,
$W(\bar{a}) = \frac{2}{3}$,
$W(b) = W(\bar{b}) = 1$

Satisfying assignments:
$SAT(F) = \{ \{ a \leftarrow 0, b \leftarrow 1 \}, \{ a \leftarrow 1, b \leftarrow 0 \}, \{ a \leftarrow 1, b \leftarrow 1 \} \} = \{ \bar{a}b, a\bar{b}, ab \}$

$\bar{a}\bar{b}$	—
$\bar{a}b$	$2/3 \cdot 1$
$a\bar{b}$	$1/3 \cdot 1$
ab	$1/3 \cdot 1$

$$\text{ModelCount}_W(F) = \frac{4}{3}$$

Weighted Model Counting (#SAT)

Model counting (#SAT) is the counting version (#P) of satisfiability (SAT).

Example (Weighted Model Counting)

Formula: $F = a \vee b$

Weights: $W(a) = \frac{1}{3}$,
$W(\bar{a}) = \frac{2}{3}$,
$W(b) = W(\bar{b}) = 1$

Satisfying assignments:
$SAT(F) = \{ \{ a \leftarrow 0, b \leftarrow 1 \}, \{ a \leftarrow 1, b \leftarrow 0 \}, \{ a \leftarrow 1, b \leftarrow 1 \} \} = \{ \bar{a}b, a\bar{b}, ab \}$

$\bar{a}\bar{b}$	—
$\bar{a}b$	$2/3 \cdot 1$
$a\bar{b}$	$1/3 \cdot 1$
ab	$1/3 \cdot 1$

$$\text{ModelCount}_W(F) = \frac{4}{3} = \overbrace{\sum_{\vec{x} \in F^{-1}(1)}}^{\text{satisfying assignment}} \overbrace{\prod_{x_i \in \vec{x}} W(x_i)}^{\text{weight of assignment}}$$

Component caching

```
1:  procedure #DPLL(φ)
2:      φ ← UnitPropagation(φ)
3:      if φ = ∅ then
4:          return 1
5:      if ∅ ∈ φ then
6:          return 0
7:      if Cached(φ) then
8:          return Lookup(φ)
9:      if φ = φ1 ∧ φ2 with disjoint variables then
10:         return #DPLL(φ1) · #DPLL(φ2)
11:     x ← SelectVariable(φ)
12:     c ← #DPLL(φ[x := 0]) + #DPLL(φ[x := 1])
13:     Store(φ, c)
14:     return c
```

[1]

[1] M. Thurley, "Sharpsat–counting models with advanced component caching and implicit bcp," in *International Conference on Theory and Applications of Satisfiability Testing*, Springer, 2006.

Roadmap

- SAT and #SAT (model counting)
- Encoding quantum circuits in logic
- Application: Simulation, Equivalence checking, Verification, Synthesis
- Different encodings and performance

Encoding quantum states as Boolean formula

$|0\rangle$

$|1\rangle$

$|10\rangle$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables

$|0\rangle$

$|1\rangle$

$|10\rangle$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$
$\|0\rangle +$ $\|1\rangle$	$\{\{x \leftarrow 0\}, \{x \leftarrow 1\}\}$	

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbr
$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\left\{\{x \leftarrow 0, \text{“weight } \frac{1}{\sqrt{2}}\text{”}\}, \{x \leftarrow 1, \text{“weight } \frac{1}{\sqrt{2}}\text{”}\}\right\}$	

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.
- Step 3: Incorporate weight by adding additional variable h with

$$W(z) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } z = \bar{h} \\ -\frac{1}{\sqrt{2}} & \text{if } z = h \\ 1 & \text{if } z \text{ represents any other literal} \end{cases}$$

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$
$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 0\}\}$	

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.
- Step 3: Incorporate weight by adding additional variable h with

$$W(z) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } z = \bar{h} \\ -\frac{1}{\sqrt{2}} & \text{if } z = h \\ 1 & \text{if } z \text{ represents any other literal} \end{cases}$$

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$
$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 0\}\}$	$F(x,h) \triangleq \bar{h}$

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.
- Step 3: Incorporate weight by adding additional variable h with

$$W(z) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } z = \bar{h} \\ -\frac{1}{\sqrt{2}} & \text{if } z = h \\ 1 & \text{if } z \text{ represents any other literal} \end{cases}$$

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x, y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$
$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 0\}\}$	$F(x, h) \triangleq \bar{h}$
$\frac{1}{\sqrt{2}}\|0\rangle - \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 1\}\}$	

Encoding quantum states as Boolean formula

- Step 1: Encode each term as an assignment to a set of variables
- Step 2: Encode assignment as formula F.
- Step 3: Incorporate weight by adding additional variable h with

$$W(z) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } z = \bar{h} \\ -\frac{1}{\sqrt{2}} & \text{if } z = h \\ 1 & \text{if } z \text{ represents any other literal} \end{cases}$$

$\|0\rangle$	$\{\{x \leftarrow 0\}\}$	$F(x) \triangleq \bar{x}$
$\|1\rangle$	$\{\{x \leftarrow 1\}\}$	$F(x) \triangleq x$
$\|10\rangle$	$\{\{x \leftarrow 1, y \leftarrow 0\}\}$	$F(x,y) \triangleq x \wedge \bar{y}$, abbreviated $x\bar{y}$
$\frac{1}{\sqrt{2}}\|0\rangle + \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 0\}\}$	$F(x,h) \triangleq \bar{h}$
$\frac{1}{\sqrt{2}}\|0\rangle - \frac{1}{\sqrt{2}}\|1\rangle$	$\{\{x \leftarrow 0, h \leftarrow 0\}, \{x \leftarrow 1, h \leftarrow 1\}\}$	$F(x,h) \triangleq h \Leftrightarrow x$

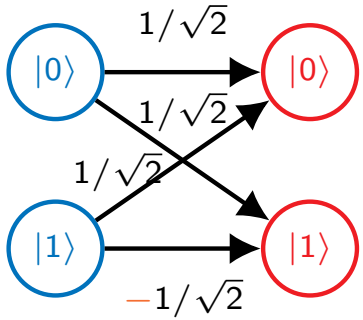
Encoding quantum gates

Goal: find formula $F_H(x, x', \quad)$ for the gate $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

Encoding quantum gates

Goal: find formula $F_H(x, x', \underbrace{h_1, h_2, \ldots}_{\text{auxiliary variables}})$ for the gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

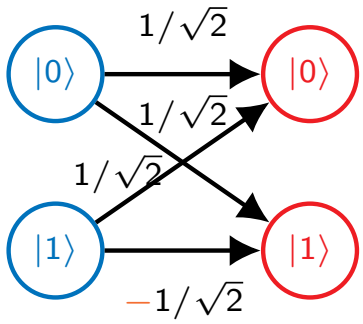
$H(|x\rangle, |x'\rangle) =$



Encoding quantum gates

Goal: find formula $F_H(x, x', \underbrace{h_1, h_2, \ldots}_{\text{auxiliary variables}})$ for the gate $H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

$H(|x\rangle, |x'\rangle) =$

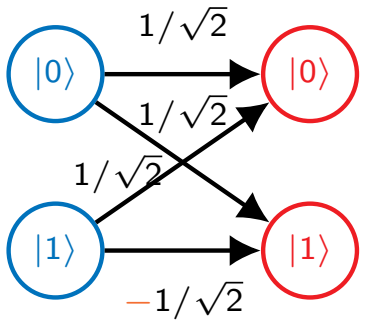


Solution: $F_H(x, x', h) \triangleq h \Leftrightarrow (x \wedge x')$ with $W(h) = -1/\sqrt{2}, W(\overline{h}) = 1/\sqrt{2}$

Encoding quantum gates

Goal: find formula $F_H(x, x', \underbrace{h_1, h_2, \ldots}_{\text{auxiliary variables}})$ for the gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

$H(|x\rangle, |x'\rangle) =$



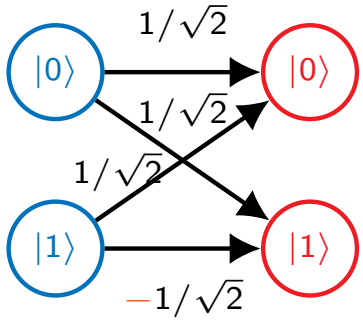
Solution: $F_H(x, x', h) \triangleq h \Leftrightarrow (x \wedge x')$ with $W(h) = -1/\sqrt{2}, W(\overline{h}) = 1/\sqrt{2}$

Similarly, we can find formulae for other gates (including fixed rotations).

Encoding quantum gates

Goal: find formula $F_H(x, x', \underbrace{h_1, h_2, \ldots}_{\text{auxiliary variables}})$ for the gate $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

$H(|x\rangle, |x'\rangle) =$



Solution: $F_H(x, x', h) \triangleq h \Leftrightarrow (x \wedge x')$ with $W(h) = -1/\sqrt{2}, W(\overline{h}) = 1/\sqrt{2}$

Similarly, we can find formulae for other gates (including fixed rotations).

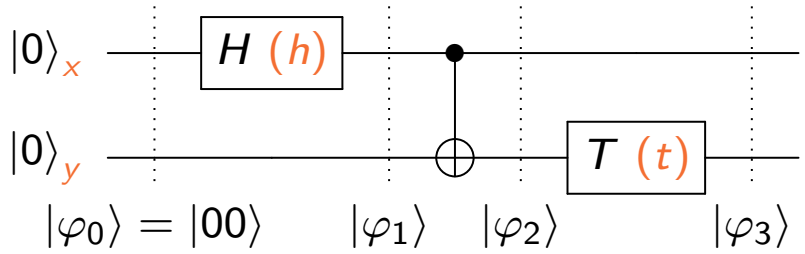
(For $H, CNOT$ and T is sufficient as these are universal; Solovey-Kitaev Thm)

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

Sequential and parallel composition of gates

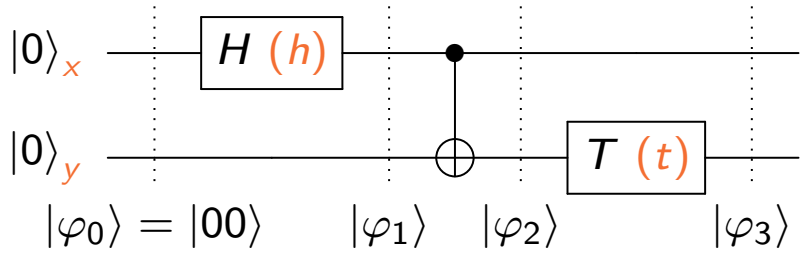
Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:



where x, y, h and t are Boolean variables.

Sequential and parallel composition of gates

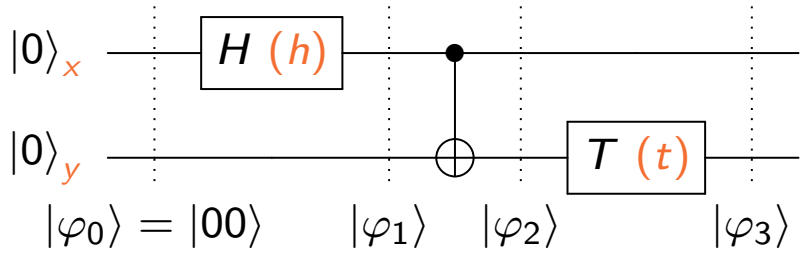
Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:



where x, y, h and t are Boolean variables. We copy x, y for every time step:

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

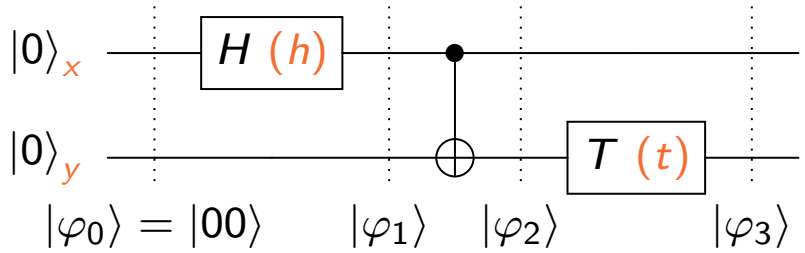


where x, y, h and t are Boolean variables. We copy x, y for every time step:

$$F_C = \underbrace{\bar{x}_0 \bar{y}_0}_{|\varphi_0\rangle} \wedge$$

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

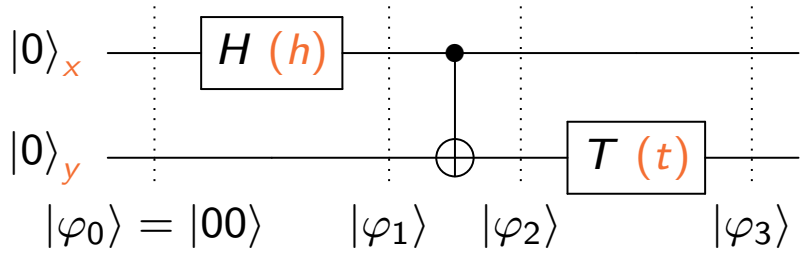


where x, y, h and t are Boolean variables. We copy x, y for every time step:

$$F_C = \underbrace{\underbrace{\bar{x}_0\bar{y}_0}_{|\varphi_0\rangle} \wedge F_H(x_0, x_1, h) \quad \wedge y_1 \Leftrightarrow y_0 \quad \wedge}_{|\varphi_1\rangle}$$

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

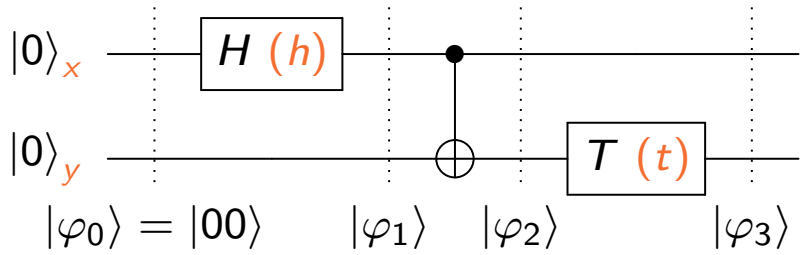


where x, y, h and t are Boolean variables. We copy x, y for every time step:

$$F_C = \underbrace{\underbrace{\bar{x}_0\bar{y}_0}_{|\varphi_0\rangle} \wedge F_H(x_0, x_1, h) \quad \wedge y_1 \Leftrightarrow y_0}_{|\varphi_1\rangle} \quad \wedge \quad F_{CNOT}(x_1, y_1, x_2, y_2) \quad \wedge}_{|\varphi_2\rangle}$$

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

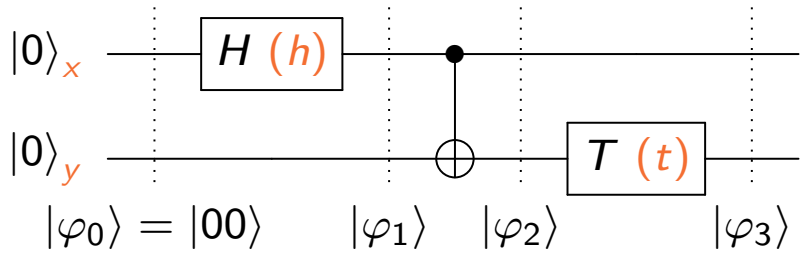


where x, y, h and t are Boolean variables. We copy x, y for every time step:

$$F_C = \underbrace{\underbrace{\bar{x}_0\bar{y}_0}_{|\varphi_0\rangle} \wedge F_H(x_0, x_1, h) \quad \wedge y_1 \Leftrightarrow y_0}_{|\varphi_1\rangle} \quad \wedge \quad \underbrace{F_{CNOT}(x_1, y_1, x_2, y_2)}_{|\varphi_2\rangle} \quad \wedge \quad \underbrace{F_T(y_2, y_3, t) \quad \wedge x_3 \Leftrightarrow x_2}_{|\varphi_3\rangle}$$

Sequential and parallel composition of gates

Using F_H, F_{CNOT}, F_T for encoding the output state of a circuit C:

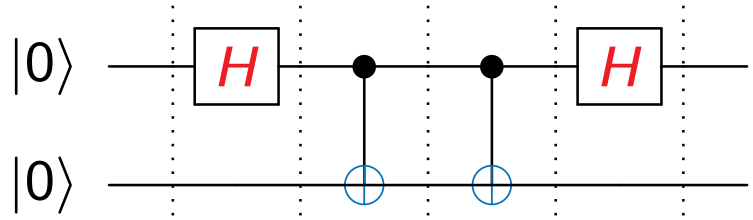


where x, y, h and t are Boolean variables. We copy x, y for every time step:

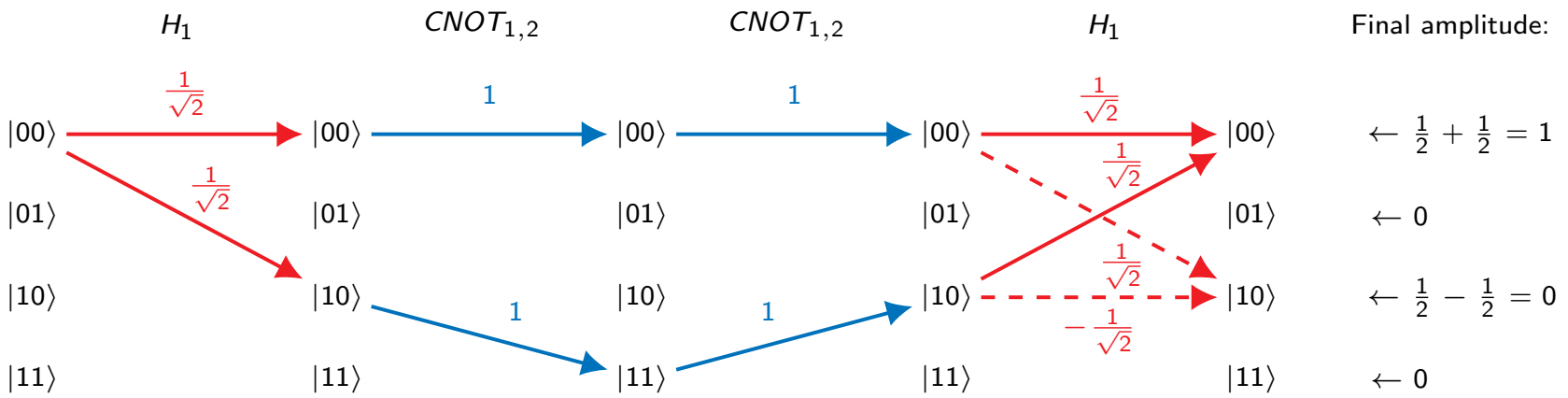
$$F_C = \underbrace{\underbrace{\bar{x}_0\bar{y}_0}_{|\varphi_0\rangle} \wedge F_H(x_0, x_1, h) \quad \wedge y_1 \Leftrightarrow y_0}_{|\varphi_1\rangle} \quad \wedge \quad \underbrace{F_{CNOT}(x_1, y_1, x_2, y_2)}_{|\varphi_2\rangle} \quad \wedge \quad \underbrace{F_T(y_2, y_3, t) \quad \wedge x_3 \Leftrightarrow x_2}_{|\varphi_3\rangle}$$

$$\#SAT_W(F_C \quad \wedge \quad x_3 = a \ \wedge \ y_3 = b) \quad = \quad \langle ab|\varphi_3\rangle$$

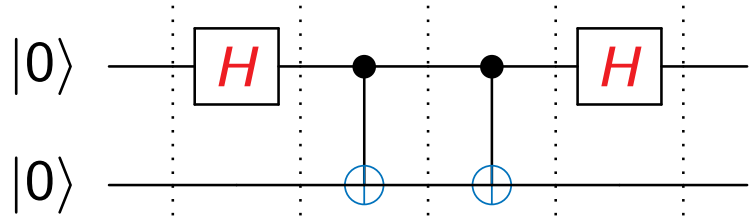
The Model Counter Computer Path Sums



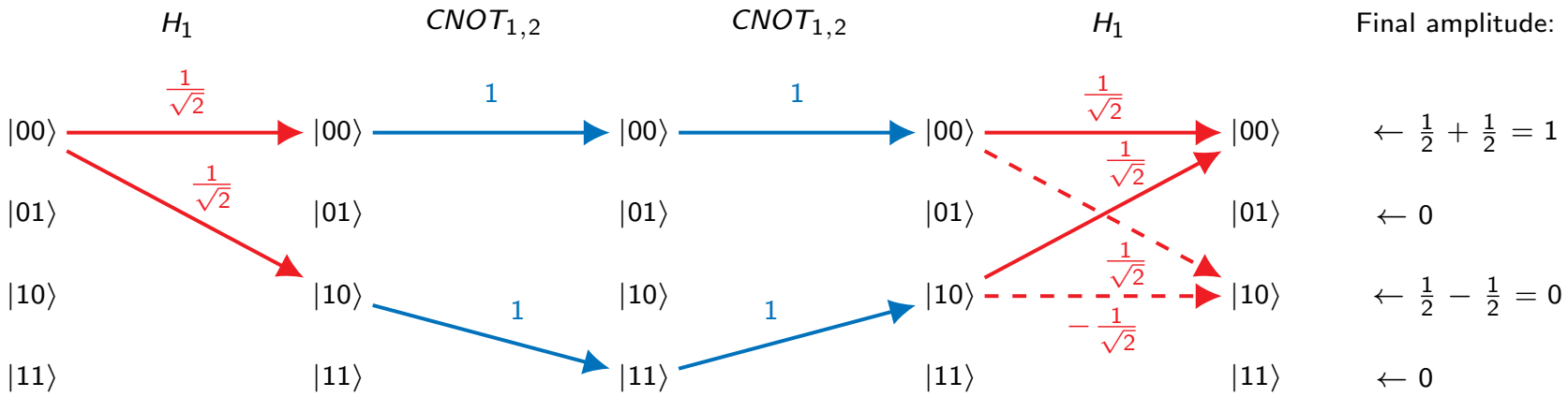
$|\varphi_0\rangle$ $|\varphi_1\rangle$ $|\varphi_2\rangle$ $|\varphi_3\rangle$ $|\varphi_4\rangle$



The Model Counter Computer Path Sums



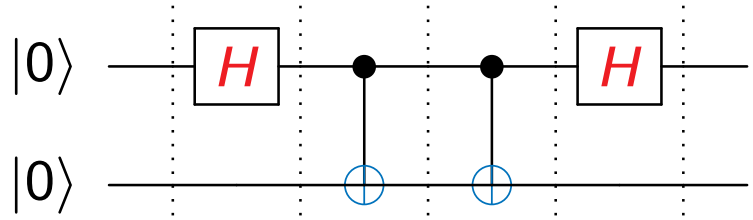
$$|\varphi_0\rangle \; |\varphi_1\rangle \; |\varphi_2\rangle \; |\varphi_3\rangle \; |\varphi_4\rangle$$

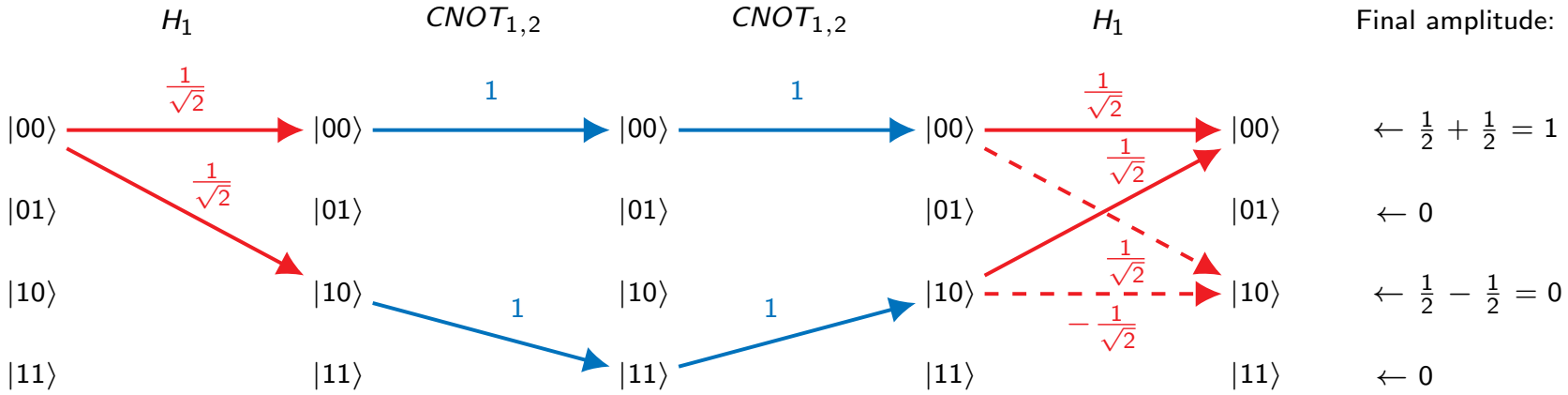


In WMC, the circuit above has four satisfying assignments:

$$\{h'\overline{h}x_1\overline{y}_1x_2y_2x_3\overline{y}_3x_4\overline{y}_4 \equiv -\tfrac{1}{2}\,|10\rangle\,, \quad \overline{h}'\overline{h}x_1\overline{y}_1x_2y_2x_3\overline{y}_3\overline{x}_4\overline{y}_4 \equiv \tfrac{1}{2}\,|00\rangle\,,$$
$$\overline{h}'\overline{h}\overline{x}_1\overline{y}_1\overline{x}_2\overline{y}_2\overline{x}_3\overline{y}_3x_4\overline{y}_4 \equiv \tfrac{1}{2}\,|10\rangle\,, \quad \overline{h}'\overline{h}\overline{x}_1\overline{y}_1\overline{x}_2\overline{y}_2\overline{x}_3\overline{y}_3\overline{x}_4\overline{y}_4 \equiv \tfrac{1}{2}\,|00\rangle\},$$

The Model Counter Computer Path Sums





In WMC, the circuit above has four satisfying assignments:

$$\{h'\overline{h}x_1\overline{y}_1x_2y_2x_3\overline{y}_3x_4\overline{y}_4 \equiv -\tfrac{1}{2}\,|10\rangle\,, \quad \overline{h}'\overline{h}x_1\overline{y}_1x_2y_2x_3\overline{y}_3\overline{x}_4\overline{y}_4 \equiv \tfrac{1}{2}\,|00\rangle\,,$$
$$\overline{h}'\overline{h}\overline{x}_1\overline{y}_1\overline{x}_2\overline{y}_2\overline{x}_3\overline{y}_3x_4\overline{y}_4 \equiv \tfrac{1}{2}\,|10\rangle\,, \quad \overline{h}'\overline{h}\overline{x}_1\overline{y}_1\overline{x}_2\overline{y}_2\overline{x}_3\overline{y}_3\overline{x}_4\overline{y}_4 \equiv \tfrac{1}{2}\,|00\rangle\},$$

Each satisfying assignment corresponds to a path! So WMC computes the path sum.

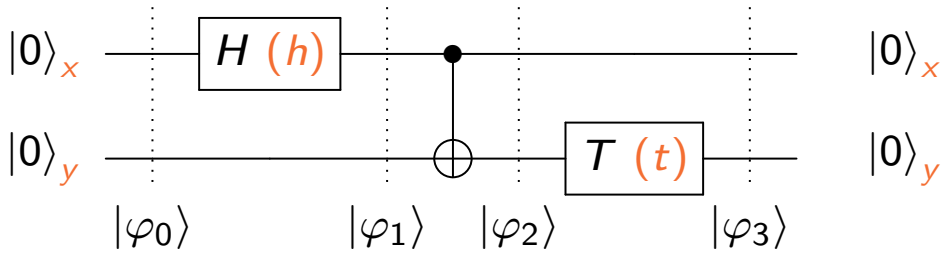
Roadmap

- SAT and #SAT (model counting)
- Encoding quantum circuits in logic
- Application: Simulation, Equivalence checking, Verification, Synthesis
- Different encodings and performance

Simulation

Simulation

Given an n-qubit quantum circuit C and a computational-basis measurement specification, which is described by a state $|\vec{b}\rangle$ on n qubits, compute the probability of measuring $|\vec{b}\rangle$ in state $C\,|0\ldots0\rangle$, i.e. $\langle\vec{b}|\,C\,|0\ldots0\rangle$.



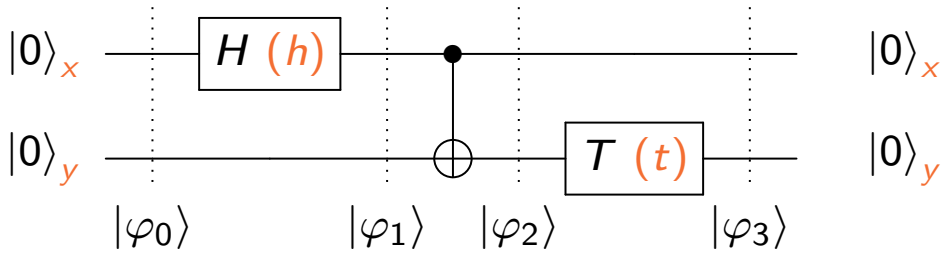
$$\bar{x}_0\bar{y}_0 \wedge F_C(x_0, y_0, \ldots, x_3, y_3) \wedge \bar{x}_3\bar{y}_3$$

[1] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

Simulation

Simulation

Given an n-qubit quantum circuit C and a computational-basis measurement specification, which is described by a state $|\vec{b}\rangle$ on n qubits, compute the probability of measuring $|\vec{b}\rangle$ in state $C\,|0\ldots0\rangle$, i.e. $\langle\vec{b}|\,C\,|0\ldots0\rangle$.



$$\bar{x}_0\bar{y}_0 \wedge F_C(x_0, y_0, \ldots, x_3, y_3) \wedge \bar{x}_3\bar{y}_3$$

We support general observables O by encoding $\langle 0|\,C^\dagger OC\,|0\rangle$.

[1] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

Verification

Quantum Hoare Logic [1]

Given a circuit C, projectors P and Q, the Hoare triple $\{P\}\ C\ \{Q\}$ is true if and only if for any input state $|\psi\rangle$, we have

$$|\psi\rangle \models P \Rightarrow C\,|\psi\rangle \models Q,$$

where $|\psi\rangle \models P$ iff $P\,|\psi\rangle = |\psi\rangle$.

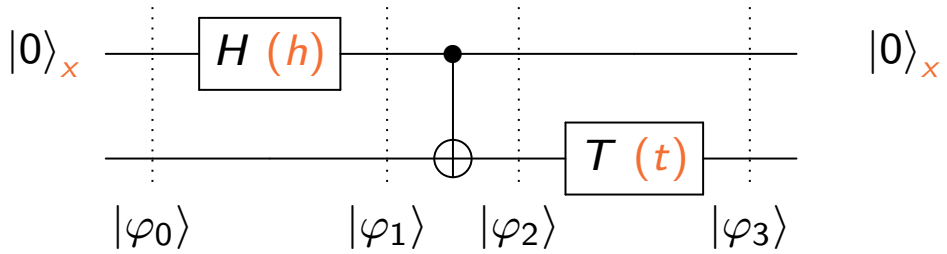
Verification

Quantum Hoare Logic [1]

Given a circuit C, projectors P and Q, the Hoare triple $\{P\}\ C\ \{Q\}$ is true if and only if for any input state $|\psi\rangle$, we have

$$|\psi\rangle \models P \Rightarrow C\,|\psi\rangle \models Q,$$

where $|\psi\rangle \models P$ iff $P\,|\psi\rangle = |\psi\rangle$.



$$\bar{x}_0 \wedge F_C(x_0, y_0, \ldots, x_3, y_3) \wedge \bar{x}_3.$$

[1] M. Ying, "Floyd–Hoare logic for quantum programs," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 33, 2012.

(Approximate) Quantum Circuit Equivalence Checking

Definition (Approximate equivalence checking)

Let U and V be two n-qubit quantum circuits, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the **Jamiolkowski fidelity** between U and V is not less than $1 - \epsilon$ ($\epsilon \in [0, 1]$).

[1] J. Mei, T. Coopmans, M. Bonsangue, and A. Laarman, "Equivalence checking of quantum circuits by model counting," in *IJCAR*, 2024.

[2] J. Mei, J. Martens, and A. Laarman, "Disentangling the Gap Between Quantum and #SAT," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2024.

(Approximate) Quantum Circuit Equivalence Checking

Definition (Approximate equivalence checking)

Let U and V be two n-qubit quantum circuits, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the **Jamiolkowski fidelity** between U and V is not less than $1 - \epsilon$ ($\epsilon \in [0, 1]$).

Exploit reversibility: $U \equiv V$ if and only if $UV^\dagger \equiv I$.

[1] J. Mei, T. Coopmans, M. Bonsangue, and A. Laarman, "Equivalence checking of quantum circuits by model counting," in *IJCAR*, 2024.

[2] J. Mei, J. Martens, and A. Laarman, "Disentangling the Gap Between Quantum and #SAT," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2024.

(Approximate) Quantum Circuit Equivalence Checking

Definition (Approximate equivalence checking)

Let U and V be two n-qubit quantum circuits, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the **Jamiolkowski fidelity** between U and V is not less than $1 - \epsilon$ ($\epsilon \in [0, 1]$).

Exploit reversibility: $U \equiv V$ if and only if $UV^\dagger \equiv I$.

The **Jamiołkowski fidelity** between U and V can be computed by

$$\frac{1}{2^n} \cdot |\#SAT_W(F_U(\vec{q}_{\text{in}}, \ldots, \vec{q}_{\text{mid}}) \wedge F_{V^\dagger}(\vec{q}_{\text{mid}}, \ldots, \vec{q}_{\text{out}}) \wedge F_I(\vec{q}_{\text{in}}, \vec{q}_{\text{out}}))|^2$$

[1] J. Mei, T. Coopmans, M. Bonsangue, and A. Laarman, "Equivalence checking of quantum circuits by model counting," in *IJCAR*, 2024.

[2] J. Mei, J. Martens, and A. Laarman, "Disentangling the Gap Between Quantum and #SAT," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2024.

(Approximate) Quantum Circuit Equivalence Checking

Definition (Approximate equivalence checking)

Let U and V be two n-qubit quantum circuits, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the **Jamiolkowski fidelity** between U and V is not less than $1 - \epsilon$ ($\epsilon \in [0, 1]$).

Exploit reversibility: $U \equiv V$ if and only if $UV^\dagger \equiv I$.

The **Jamiołkowski fidelity** between U and V can be computed by

$$\frac{1}{2^n} \cdot |\#SAT_W(F_U(\vec{q}_{\text{in}}, \ldots, \vec{q}_{\text{mid}}) \wedge F_{V^\dagger}(\vec{q}_{\text{mid}}, \ldots, \vec{q}_{\text{out}}) \wedge F_I(\vec{q}_{\text{in}}, \vec{q}_{\text{out}}))|^2$$

(Equals 1 if and only if $U \equiv V$.)

[1] J. Mei, T. Coopmans, M. Bonsangue, and A. Laarman, "Equivalence checking of quantum circuits by model counting," in *IJCAR*, 2024.

[2] J. Mei, J. Martens, and A. Laarman, "Disentangling the Gap Between Quantum and #SAT," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2024.

(Approximate) Quantum Circuit Equivalence Checking

Definition (Approximate equivalence checking)

Let U and V be two n-qubit quantum circuits, Then U and V are ϵ-equivalent, written as $U \simeq_\epsilon V$, if and only if the **Jamiolkowski fidelity** between U and V is not less than $1 - \epsilon$ ($\epsilon \in [0, 1]$).

Exploit reversibility: $U \equiv V$ if and only if $UV^\dagger \equiv I$.

The **Jamiołkowski fidelity** between U and V can be computed by

$$\frac{1}{2^n} \cdot |\#SAT_W(F_U(\vec{q}_{\text{in}}, \ldots, \vec{q}_{\text{mid}}) \wedge F_{V^\dagger}(\vec{q}_{\text{mid}}, \ldots, \vec{q}_{\text{out}}) \wedge F_I(\vec{q}_{\text{in}}, \vec{q}_{\text{out}}))|^2$$

(Equals 1 if and only if $U \equiv V$.)

We call this the "cyclic encoding" since it equals $F_{UV^\dagger}(\vec{q}_0, \ldots, \vec{q}_0)$

[1] J. Mei, T. Coopmans, M. Bonsangue, and A. Laarman, "Equivalence checking of quantum circuits by model counting," in *IJCAR*, 2024.

[2] J. Mei, J. Martens, and A. Laarman, "Disentangling the Gap Between Quantum and #SAT," in *International Colloquium on Theoretical Aspects of Computing*, Springer, 2024.

Quantum circuit synthesis

The synthesis problem

Inputs:

- A circuit C_1 is gateset $\mathcal{G}_1$
- A gate set $\mathcal{G}_2$

Output:

- A minimum-depth circuit C_2 in gateset $\mathcal{G}_2$ that is equivalent to C_1

Quantum circuit synthesis

The approximate synthesis problem

Inputs:

- A circuit C_1 is gateset $\mathcal{G}_1$
- A gate set $\mathcal{G}_2$
- An approximation parameter ϵ

Output:

- A minimum-depth circuit C_2 in gateset $\mathcal{G}_2$ that is ϵ-equivalent to C_1

Quantum circuit synthesis

The approximate synthesis problem

Inputs:

- A circuit C_1 is gateset $\mathcal{G}_1$
- A gate set $\mathcal{G}_2$
- An approximation parameter ϵ

Output:

- A minimum-depth circuit C_2 in gateset $\mathcal{G}_2$ that is ϵ-equivalent to C_1

Approach

quantum circuit synthesis $\longrightarrow$ maximum (weighted) model counting [1]

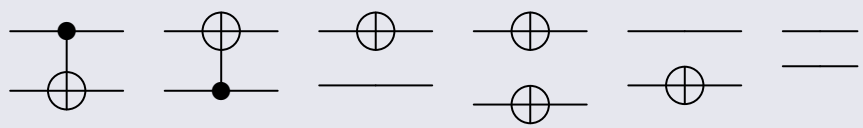
[1] D. Fremont, M. Rabe, and S. Seshia, "Maximum model counting," in *AAAI*, vol. 31, 2017.

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:

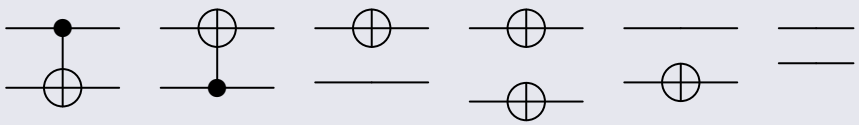


Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



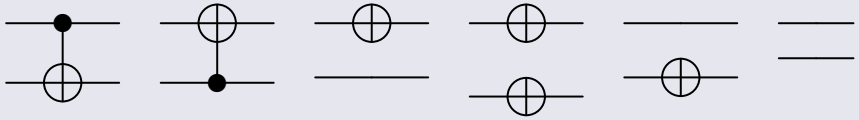
2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

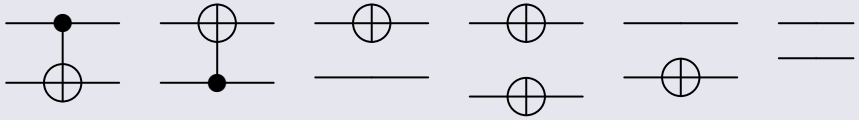
$$\exists \vec{p} \in [P]^d : \#\mathrm{SAT}\Big(\underbrace{F_{\mathcal{C}_1^{-1}}(\vec{x}_0, \vec{x}_1) \wedge F_{\mathcal{C}_{2,d}}(\vec{x}_1, \vec{x}_2, \vec{p})}_{\mathcal{C}_{2,d} \cdot \mathcal{C}_1^{-1}} \wedge (\vec{x}_0 \Leftrightarrow \vec{x}_2) \Big) \overset{?}{\geq} \epsilon \cdot 2^n$$

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

$$\exists \vec{p} \in [P]^d : \#\mathrm{SAT}\Big(\underbrace{F_{\mathcal{C}_1^{-1}}(\vec{x}_0, \vec{x}_1) \wedge F_{\mathcal{C}_{2,d}}(\vec{x}_1, \vec{x}_2, \vec{p})}_{\mathcal{C}_{2,d} \cdot \mathcal{C}_1^{-1}} \wedge (\vec{x}_0 \Leftrightarrow \vec{x}_2) \Big) \overset{?}{\geq} \epsilon \cdot 2^n$$

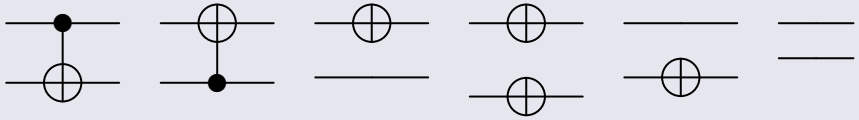
1. Increment $d = 1, 2, \ldots$ for depth-optimal

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

$$\exists \vec{p} \in [P]^d : \#\mathrm{SAT}\Big(\underbrace{F_{\mathcal{C}_1^{-1}}(\vec{x}_0, \vec{x}_1) \wedge F_{\mathcal{C}_{2,d}}(\vec{x}_1, \vec{x}_2, \vec{p})}_{\mathcal{C}_{2,d} \cdot \mathcal{C}_1^{-1}} \wedge (\vec{x}_0 \Leftrightarrow \vec{x}_2) \Big) \overset{?}{\geq} \epsilon \cdot 2^n$$

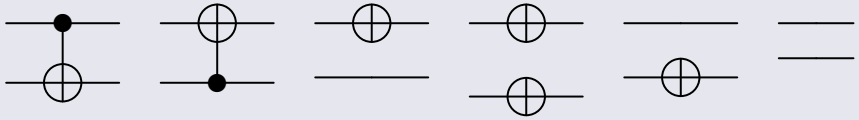
1. Increment $d = 1, 2, \ldots$ for depth-optimal
2. Using $\epsilon = 1$, we have exact synthesis.

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

$$\exists \vec{p} \in [P]^d : \#\mathrm{SAT}\Big(\underbrace{F_{\mathcal{C}_1^{-1}}(\vec{x}_0, \vec{x}_1) \wedge F_{\mathcal{C}_{2,d}}(\vec{x}_1, \vec{x}_2, \vec{p})}_{\mathcal{C}_{2,d} \cdot \mathcal{C}_1^{-1}} \wedge (\vec{x}_0 \Leftrightarrow \vec{x}_2) \Big) \overset{?}{\geq} \epsilon \cdot 2^n$$

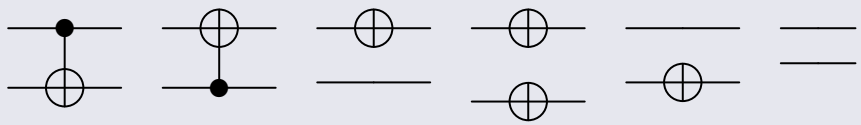
1. Increment $d = 1, 2, \ldots$ for depth-optimal
2. Using $\epsilon = 1$, we have exact synthesis.
3. $\{$ *Toffoli*, CX, X $\}$ is universal $\Rightarrow$ some d always works for exact synthesis.

Synthesis = Max#SAT over "which gate where"

Input $\mathcal{C}_1$ is given as $F_{\mathcal{C}_1}(\vec{x}, \vec{x}')$.

1. Encode one generic layer $F_{\mathcal{L}}(\vec{x}, \vec{x}', p)$ — selectors p *choose* the gate

e.g., for $p \in [P]$, the circuit $\mathcal{L}_p$ is:



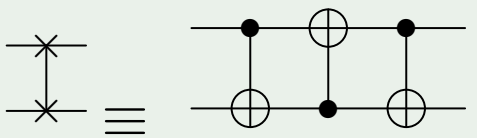
2. Stack d layers $\mathcal{C}_{2,d} = \underbrace{\mathcal{L} \cdots \mathcal{L}}_{d\times}$, check $\mathcal{C}_{2,d} \equiv_\epsilon \mathcal{C}_1$, and maximise

$$\exists \vec{p} \in [P]^d : \#\mathrm{SAT}\Big(\underbrace{F_{\mathcal{C}_1^{-1}}(\vec{x}_0, \vec{x}_1) \wedge F_{\mathcal{C}_{2,d}}(\vec{x}_1, \vec{x}_2, \vec{p})}_{\mathcal{C}_{2,d} \cdot \mathcal{C}_1^{-1}} \wedge (\vec{x}_0 \Leftrightarrow \vec{x}_2) \Big) \overset{?}{\geq} \epsilon \cdot 2^n$$

1. Increment $d = 1, 2, \ldots$ for depth-optimal
2. Using $\epsilon = 1$, we have exact synthesis.
3. $\{$ *Toffoli*, CX, X $\}$ is universal $\Rightarrow$ some d always works for exact synthesis.

Exactly Synthesise `SWAP` from $\{CX\}$

d=3: count $= 2^2 = 4$,
$\mathcal{C}_2 = (CX_{01}, CX_{10}, CX_{01})$.



Roadmap

- SAT and #SAT (model counting)
- Encoding quantum circuits in logic
- Application: Simulation, Equivalence checking, Verification, Synthesis
- Different encodings and performance

All the other gates: negative & complex weights

Gate	Relation F_G	Weights
X, CX, *Toffoli*	Boolean	none
H	$h \Leftrightarrow (x \wedge x')$	$W(h) = -\frac{1}{\sqrt{2}}$
T	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = \omega = \frac{1+i}{\sqrt{2}}$
S, $R_z(\theta)$	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = i,\ e^{i\theta}$

[1] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Information and Computation*, vol. 6, 2006.

[2] R. E. Bryant, "Numerical considerations in weighted model counting," *arXiv preprint arXiv:2508.06264*, 2025.

[3] G. Audemard, J.-M. Lagniez, and M. Miceli, "A new exact solver for (weighted) Max#SAT," in *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, ser. LIPIcs, vol. 236, 2022.

All the other gates: negative & complex weights

Gate	Relation F_G	Weights
X, CX, *Toffoli*	Boolean	none
H	$h \Leftrightarrow (x \wedge x')$	$W(h) = -\frac{1}{\sqrt{2}}$
T	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = \omega = \frac{1+i}{\sqrt{2}}$
S, $R_z(\theta)$	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = i$, $e^{i\theta}$

H,Toffoli suffice (universal, Solovay–Kitaev [1]).
Irrational H needs approximation [2].

[1] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Information and Computation*, vol. 6, 2006.

[2] R. E. Bryant, "Numerical considerations in weighted model counting," *arXiv preprint arXiv:2508.06264*, 2025.

[3] G. Audemard, J.-M. Lagniez, and M. Miceli, "A new exact solver for (weighted) Max#SAT," in *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, ser. LIPIcs, vol. 236, 2022.

All the other gates: negative & complex weights

Gate	Relation F_G	Weights
X, CX, *Toffoli*	Boolean	none
H	$h \Leftrightarrow (x \wedge x')$	$W(h) = -\frac{1}{\sqrt{2}}$
T	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = \omega = \frac{1+i}{\sqrt{2}}$
S, $R_z(\theta)$	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = i$, $e^{i\theta}$

H,Toffoli suffice (universal, Solovay–Kitaev [1]).
Irrational H needs approximation [2].

Tooling

No solver handled negative/complex weights — we extended `d4Max` [3].

[1] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Information and Computation*, vol. 6, 2006.

[2] R. E. Bryant, "Numerical considerations in weighted model counting," *arXiv preprint arXiv:2508.06264*, 2025.

[3] G. Audemard, J.-M. Lagniez, and M. Miceli, "A new exact solver for (weighted) Max#SAT," in *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, ser. LIPIcs, vol. 236, 2022.

All the other gates: negative & complex weights

Gate	Relation F_G	Weights
X, CX, *Toffoli*	Boolean	none
H	$h \Leftrightarrow (x \wedge x')$	$W(h) = -\frac{1}{\sqrt{2}}$
T	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = \omega = \frac{1+i}{\sqrt{2}}$
S, $R_z(\theta)$	$(x \Leftrightarrow x') \wedge (w \Leftrightarrow x)$	$W(w) = i$, $e^{i\theta}$

H,Toffoli suffice (universal, Solovay–Kitaev [1]).
Irrational H needs approximation [2].

Tooling

No solver handled negative/complex weights — we extended `d4Max` [3].

Different encoding in the Pauli basis (entanglement spectrum).

[1] C. M. Dawson and M. A. Nielsen, "The Solovay-Kitaev algorithm," *Quantum Information and Computation*, vol. 6, 2006.

[2] R. E. Bryant, "Numerical considerations in weighted model counting," *arXiv preprint arXiv:2508.06264*, 2025.

[3] G. Audemard, J.-M. Lagniez, and M. Miceli, "A new exact solver for (weighted) Max#SAT," in *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, ser. LIPIcs, vol. 236, 2022.

Different encodings

In the Pauli basis: $\rho = \sum_j \beta_j \cdot P_j$ for $P_j \in \{ I, X, Y, Z \}^{\otimes n}$ for $\beta_j \in \mathbb{R}$,

We reserve two Boolean variables (x, z) for a Pauli, so e.g.:
$Z_1, Z_2 \equiv \overline{x}_1 \wedge \overline{x}_2 \wedge (z_1 \oplus z_2)$

Quokka#: `https://github.com/System-Verification-Lab/Quokka-Sharp`

Different encodings

In the Pauli basis: $\rho = \sum_j \beta_j \cdot P_j$ for $P_j \in \{ I, X, Y, Z \}^{\otimes n}$ for $\beta_j \in \mathbb{R}$,

We reserve two Boolean variables (x, z) for a Pauli, so e.g.:
$Z_1, Z_2 \equiv \overline{x}_1 \wedge \overline{x}_2 \wedge (z_1 \oplus z_2)$

	Nr of qubit variables	Gate encoding length	weights
Pauli basis	$2n$	m	real (+negative)
Computational basis	n	$2m$	complex

Quokka#: `https://github.com/System-Verification-Lab/Quokka-Sharp`

Different encodings

In the Pauli basis: $\rho = \sum_j \beta_j \cdot P_j$ for $P_j \in \{ I, X, Y, Z \}^{\otimes n}$ for $\beta_j \in \mathbb{R}$,

We reserve two Boolean variables (x, z) for a Pauli, so e.g.:
$Z_1, Z_2 \equiv \overline{x}_1 \wedge \overline{x}_2 \wedge (z_1 \oplus z_2)$

	Nr of qubit variables	Gate encoding length	weights
Pauli basis	$2n$	m	real (+negative)
Computational basis	n	$2m$	complex

Encoding equivalence checking for n qubits:

Encoding Basis	Eq. Encoding	Performance
Pauli basis	Cyclic	+/-
Computational basis	Cyclic	++
Pauli basis	Choi iso. trick	++

Quokka#: `https://github.com/System-Verification-Lab/Quokka-Sharp`

Practical performance

We can use any state-of-the-art weighted model counting which supports negative weights, such as Ganak [1]!

Random-circuit simulation comparison [2]:

- solid line: WMC
- dashed line: ZX graphical-calculus based simulator

[1] S. Sharma, S. Roy, M. Soos, and K. S. Meel, "Ganak: ...," in *IJCAI*, vol. 19, 2019.

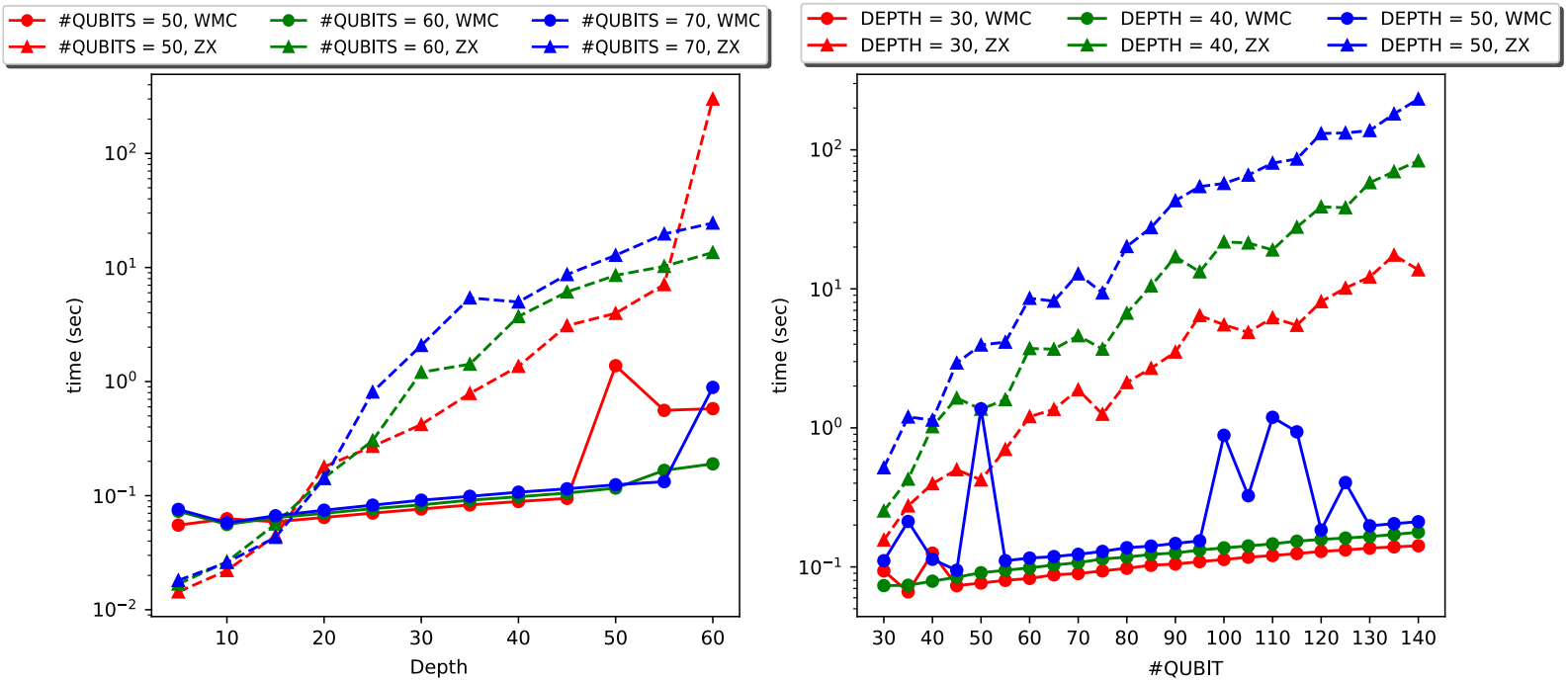
[2] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

Practical performance

We can use any state-of-the-art weighted model counting which supports negative weights, such as Ganak [1]!

Random-circuit simulation comparison [2]:

- solid line: WMC
- dashed line: ZX graphical-calculus based simulator



[1] S. Sharma, S. Roy, M. Soos, and K. S. Meel, "Ganak: ...," in *IJCAI*, vol. 19, 2019.

[2] J. Mei, M. Bonsangue, and A. Laarman, "Simulating quantum circuits by model counting," in *CAV*, 2024.

Does synthesis work?

Depth-optimal TOFFOLI synthesis into $\{H, CX, C\sqrt{X}\}$

$$\text{Toffoli} = C\sqrt{X}_{0,2}\, C\sqrt{X}_{1,2}\, CX_{0,1}\, C\sqrt{X^\dagger}_{0,2}\, CX_{0,1}$$

iteration = depth	1	2	3	4	5	6
Clauses	136	992	1886	2780	3674	4568
Literals	524	3760	7126	10492	13858	17224
runtime (s)	0.03	0.04	0.16	3.39	141.86	577.78

converges to the optimal 6-layer circuit.

[1] Z. Dekel, J. Mei, J.-M. Lagniez, and A. Laarman, "Reducing Quantum Circuit Synthesis to #SAT," in *Constraint Programming*, Springer, 2025.

[2] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and*

Does synthesis work?

Depth-optimal TOFFOLI synthesis into $\{H, CX, C\sqrt{X}\}$

$$\text{Toffoli} = C\sqrt{X}_{0,2}\, C\sqrt{X}_{1,2}\, CX_{0,1}\, C\sqrt{X^\dagger}_{0,2}\, CX_{0,1}$$

iteration = depth	1	2	3	4	5	6
Clauses	136	992	1886	2780	3674	4568
Literals	524	3760	7126	10492	13858	17224
runtime (s)	0.03	0.04	0.16	3.39	141.86	577.78

converges to the optimal 6-layer circuit.

Comparison with state-of-the-art `mitms`

$$CS = CX_{1,0}\, T_0^\dagger\, CX_{1,0}\, T_0\, T_1$$

depth d	3	4	5	6	7	8
`mitms` **[2]**	0.19	1.60	12.4	84.6	TO	TO
`Quokka#`	0.05	0.27	2.34	6.47	197	2354

reaches depths 7–8 where `mitms` times out.

[1] Z. Dekel, J. Mei, J.-M. Lagniez, and A. Laarman, "Reducing Quantum Circuit Synthesis to #SAT," in *Constraint Programming*, Springer, 2025.

[2] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and*

Takeaway

Quantum circuit synthesis $\longrightarrow$ Maximum weighted model counting

- New SAT-based methods for QC $\leftrightarrow$ new applications for solvers.
- Approximate equivance (QMA) and synthesis with exact symbolic tool (#SAT).

Analyze your quantum circuits with Quokka#:

`github.com/System-Verification-Lab/Quokka-Sharp` [1]

Takeaway

Quantum circuit synthesis $\longrightarrow$ Maximum weighted model counting

- New SAT-based methods for QC $\leftrightarrow$ new applications for solvers.
- Approximate equivance (QMA) and synthesis with exact symbolic tool (#SAT).

Open problems

- Approximation algorithms for #SAT with negative weights.
- Support for sampling and noise.
- Push synthesis well beyond small qubit count and depth.
- Push open problems in Q. foundations: contextuality, MUB, non-locality, ...

Analyze your quantum circuits with Quokka#:

`github.com/System-Verification-Lab/Quokka-Sharp` [1]