양자 프로그램 자동 합성

오학주
고려대학교 컴퓨터학과

KOREA UNIVERSITY 1905

02/06/2023@컴퓨터시스템소사이어티 동계학술대회

소프트웨어 분석 연구실@Korea Univ.

- Members: 10 PhD and 5 MS students
- Research areas: programming languages (PL), software engineering (SE), software security
  - program analysis and testing
  - program synthesis and repair

- Publication: top-venues in PL, SE, and Security:
  - PL: POPL('22),PLDI('20,'14,'12),OOPSLA('15,'17a,'17b,'18a,'18b,'19,'20,'23)
  - SE: ICSE('17,'18,'19,'20,'21'22a,'22b,'23a,'23b,'23c), FSE('18,'19,'20,'21),ASE('18)
  - Security: IEEE S&P('17,'20), USENIX Security('21)

http://prl.korea.ac.kr

연구 목표

- SW 오류 = 사회 모든 영역에서 발생


Knight Capital Says Trading Glitch Cost It $440 Million

BY NATHANIEL POPPER AUGUST 2, 2012 9:07 AM

Runaway Trades Spread Turmoil Across Wall St.

금융거래SW(2012)


Tesla in fatal California crash was on Autopilot

31 March 2018

Share

자율주행SW(2017)


Up to 270 women may have died after breast cancer screening IT error

Health secretary says computer errors led to 450,000 missed screenings in England since 2009

Have you been affected?

의료SW(2018)


$25 million in BTC, ETH stolen in dForce hacking

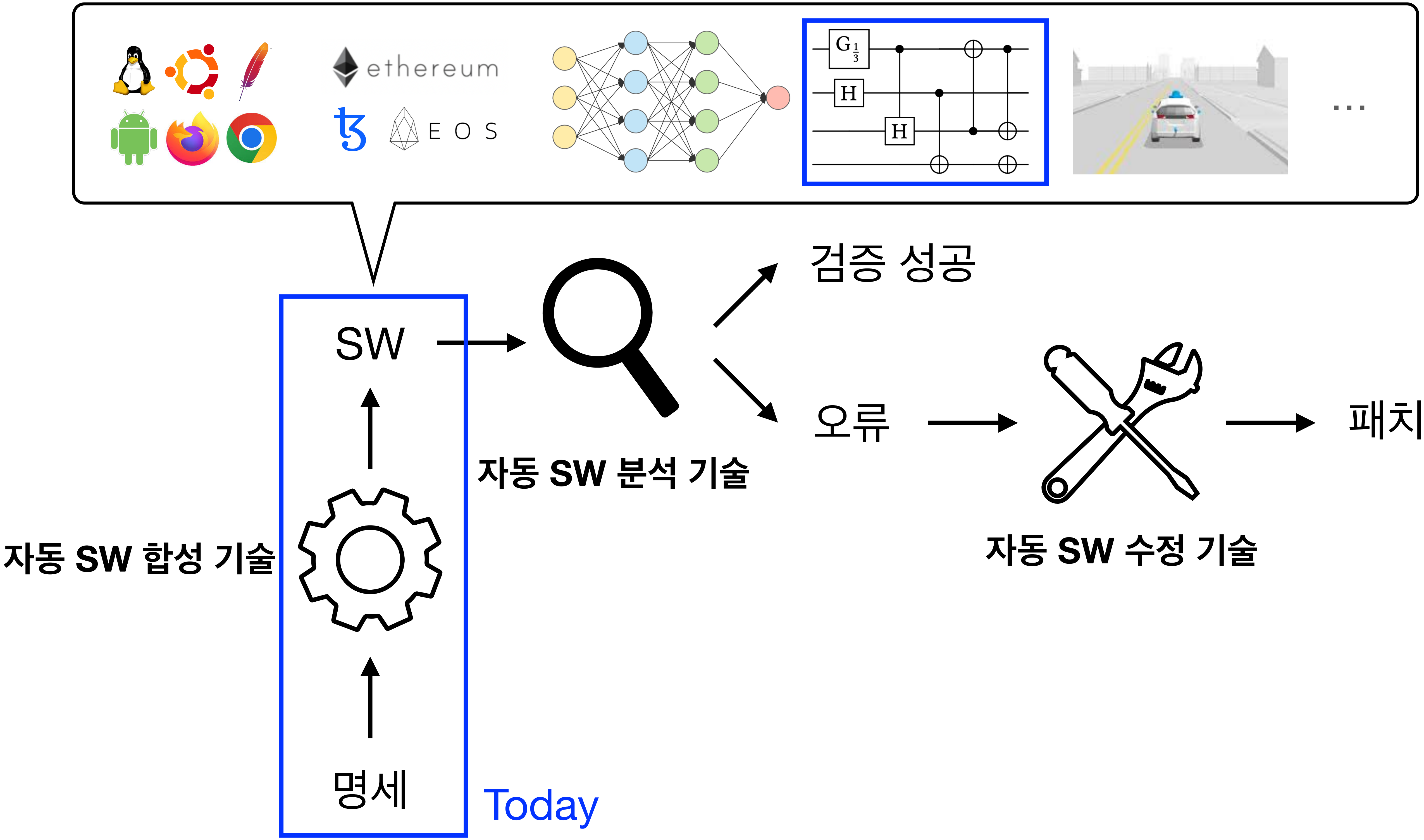
The decentralized finance (DeFi) community has been

블록체인SW(2020)

- SW 오류 = 사회경제적 비용 1.7조 달러/년

Software fail watch (5th edition). 2017

Our Research



프로그램 합성 (Program Synthesis)

- 명세로부터 프로그램 코드를 자동 생성

$$\text{reverse}(12) = 21,\ \text{reverse}(123) = 321$$

```
reverse(n){
  r := 0;
  while (          ){



  };
  return r;
}
```

2.5s

```
reverse(n){
  r := 0;
  while ( n > 0 ){
    x := n % 10;
    r := r * 10;
    r := r + x;
    n := n / 10;
  };
  return r;
}
```

프로그램 합성 (Program Synthesis)

- 명세로부터 프로그램 코드를 자동 생성

```python
# Write a python function to toggle all even bits of a given number.
# Your code should pass these tests:
#
# assert even_bit_toggle_number(10) == 0
# assert even_bit_toggle_number(20) == 30
# assert even_bit_toggle_number(30) == 20

def even_bit_toggle_number(n):
    count = 0
    res = 0
    temp = n
    while temp > 0:
        if count % 2 == 1:
            res |= 1 << count
        count += 1
        temp >>= 1
    return n ^ res
```

양자 회로 자동 합성

입출력 명세

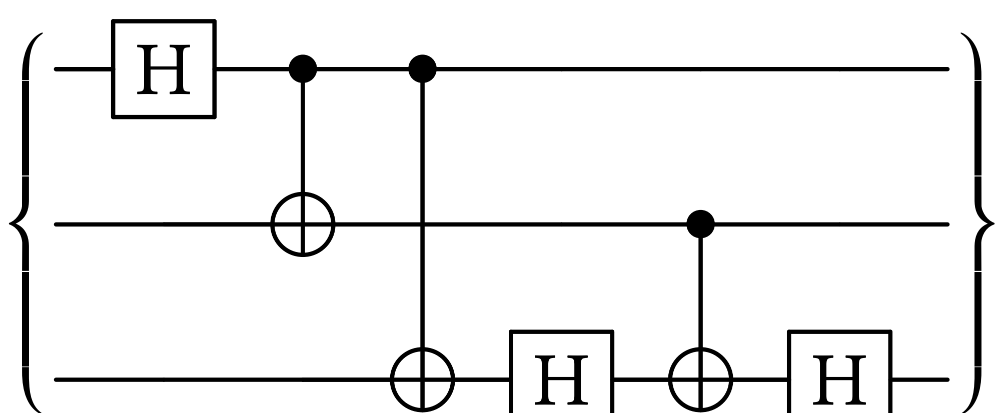
$$|100\rangle \mapsto \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

컴포넌트 게이트

$$H, CNOT$$

양자 회로 합성기

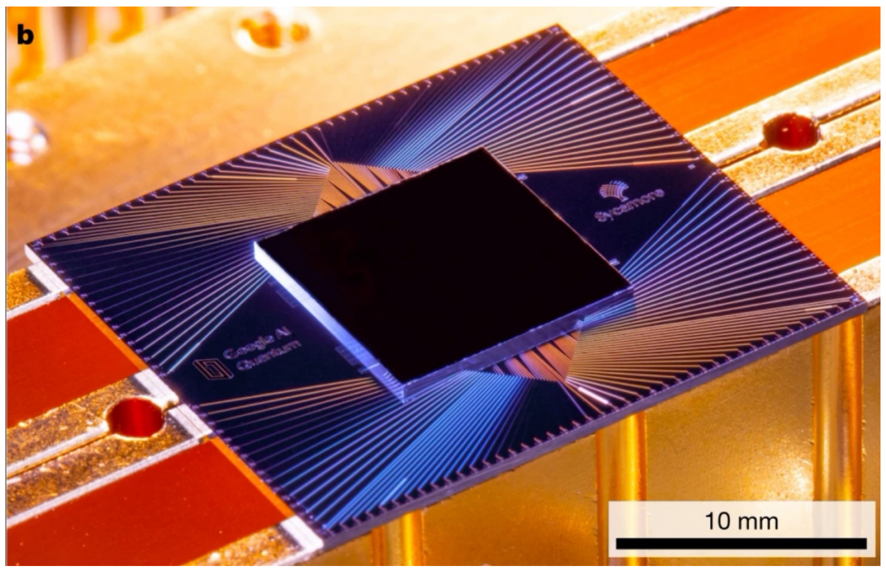
$|100\rangle$



$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

양자 컴퓨터

- 하드웨어



Google Sycamore (2019)

IBM Hummingbird, Eagle, Osprey (2020-2022)

- 소프트웨어

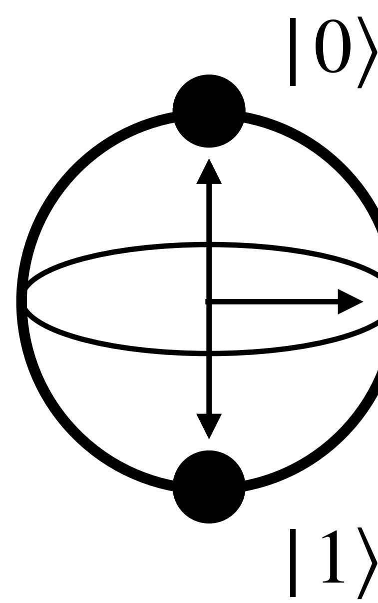
Qiskit Cirq Q# SILQ

양자 컴퓨터

- 고전 컴퓨터의 일반화

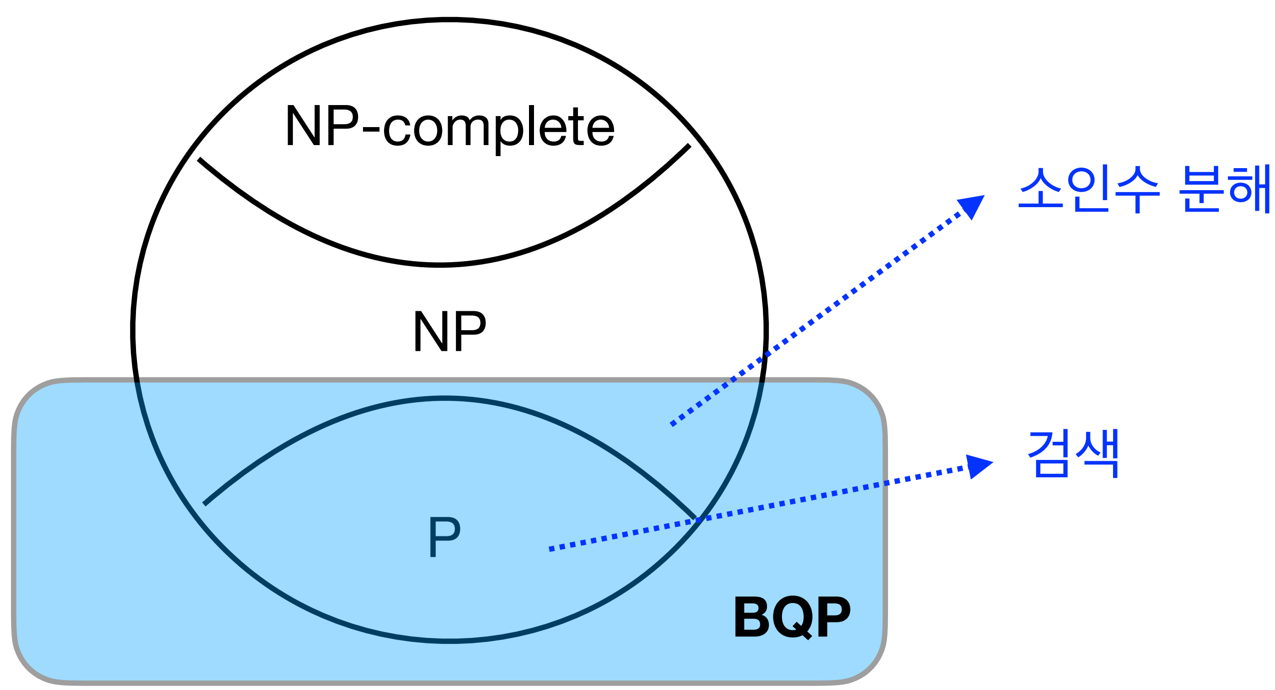
고전 비트

양자 비트
(큐비트)



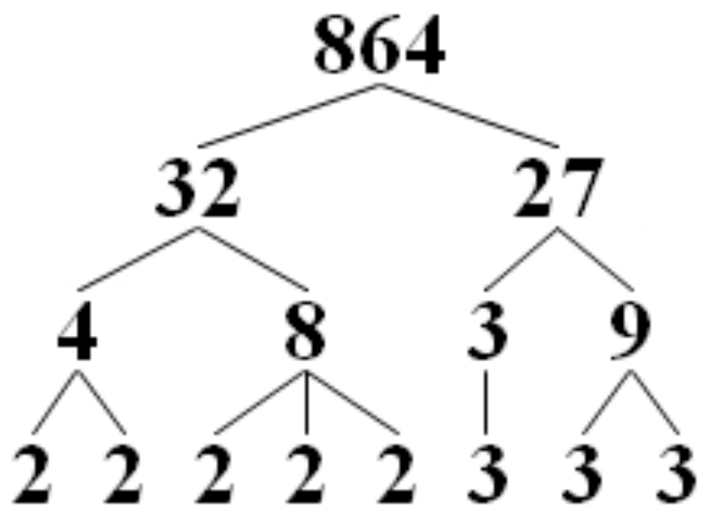
$$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

- 가능성 & 한계



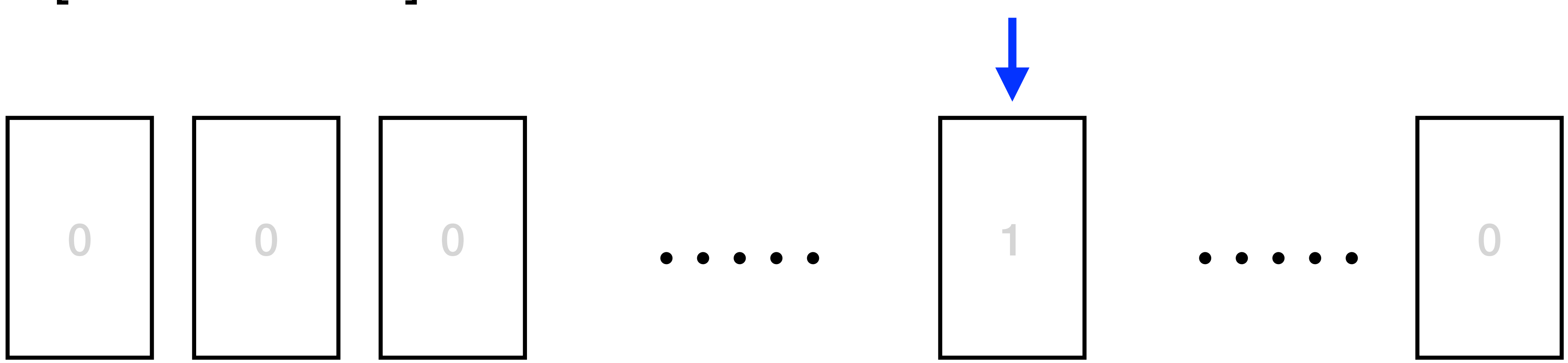
적용 사례

(1) 소인수분해 [Shor 1995]

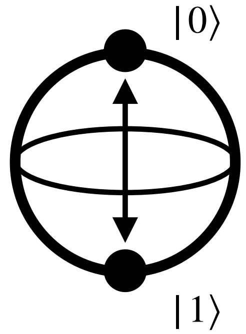


$O(2^N)$ vs $O(N^3)$

(2) 검색 [Grover 1996]



$O(N)$ vs $O(\sqrt{N})$

큐비트 

- 비트의 일반화 (0과 1의 중첩—선형 결합)

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$$

  - $\alpha_1, \alpha_2 \in \mathbb{C}$: 확률 진폭 (probability amplitude)

$$|\alpha_1|^2 + |\alpha_2|^2 = 1$$

  - $|0\rangle$
  - $|1\rangle$
  - $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
  - $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$
  - $\sqrt{\frac{3}{4}}|0\rangle + \frac{i}{\sqrt{4}}|1\rangle$

N개 큐비트

- 고전 비트 2개로 만들어지는 상태

00,　01,　10,　11

- 큐비트 2개 = 고전 2-bit 상태들의 중첩

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle \quad \Big(\sum_i |\alpha_i|^2 = 1\Big)$$

- 큐비트 N개 = 고전 N-bit 상태들의 중첩

$$|\psi\rangle = \sum_{x\in\{0,1\}^N} \alpha_x|x\rangle$$

2^N개 확률 진폭을 “저장”

병렬처리: 고전 컴퓨터 vs. 양자 컴퓨터

- $f : \{0,1\} \rightarrow \{0,1\}$

$$f(0)$$

$$f(1)$$

vs.

$$f\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)$$

- $f : \{0,1\}^N \rightarrow \{0,1\}$

$$f(0)$$

$$f(1)$$

$$\vdots$$

$$f(2^N - 1)$$

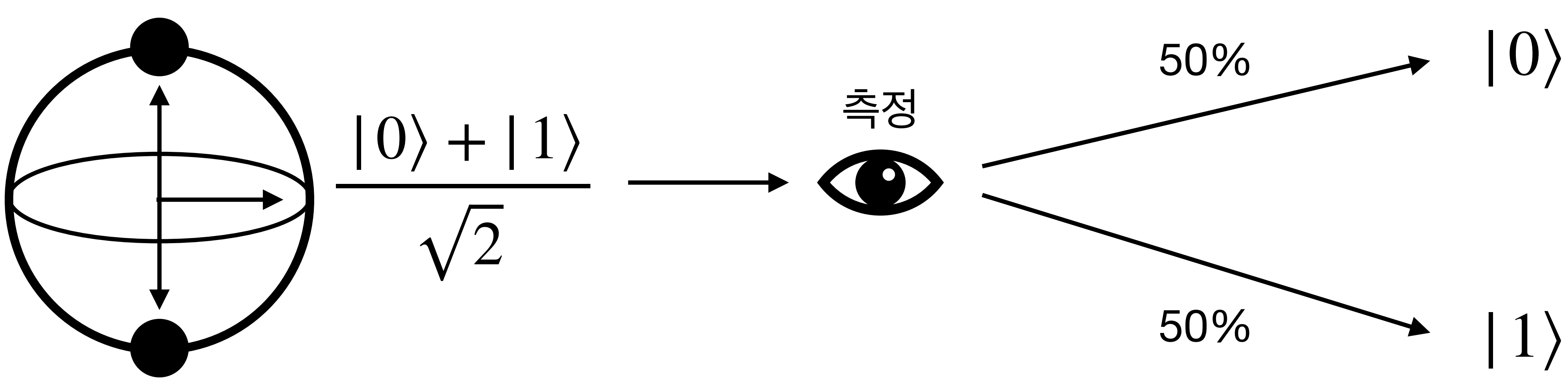
vs.

$$f\left(\frac{|0\rangle + |1\rangle + \ldots + |2^N - 1\rangle}{\sqrt{2^N}}\right)$$

측정과 붕괴

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

- 큐비트의 내부 상태(α_1, α_2)는 알 수 없고, 고전 정보만 관측 가능
- 큐비트 $|\psi\rangle$를 측정하면,
  - $|\alpha_0|^2$의 확률로 0이 관측되고 $|\psi\rangle = |0\rangle$ 로 붕괴
  - $|\alpha_1|^2$의 확률로 1이 관측되고 $|\psi\rangle = |1\rangle$ 로 붕괴



N개 큐비트 측정

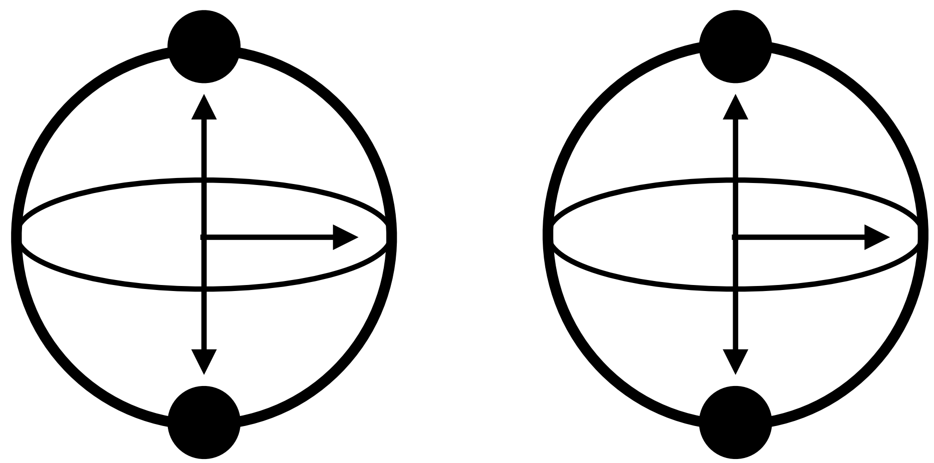
$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$$

- 상태 $|\psi\rangle$를 측정하면,
    - $|\alpha_0|^2$의 확률로 00이 관측되고 $|\psi\rangle = |00\rangle$ 로 붕괴
    - $|\alpha_1|^2$의 확률로 01이 관측되고 $|\psi\rangle = |01\rangle$ 로 붕괴
    - $|\alpha_2|^2$의 확률로 10이 관측되고 $|\psi\rangle = |10\rangle$ 로 붕괴
    - $|\alpha_2|^2$의 확률로 11이 관측되고 $|\psi\rangle = |11\rangle$ 로 붕괴
- 예:

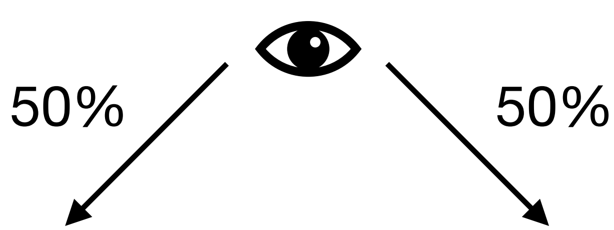
$$|\psi\rangle = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$$

얽힘 (Entanglement)

- 얽히지 않은 상태 = 개별 큐비트들이 단순 결합된 상태



$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad \text{x} \quad \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad = \quad \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$$



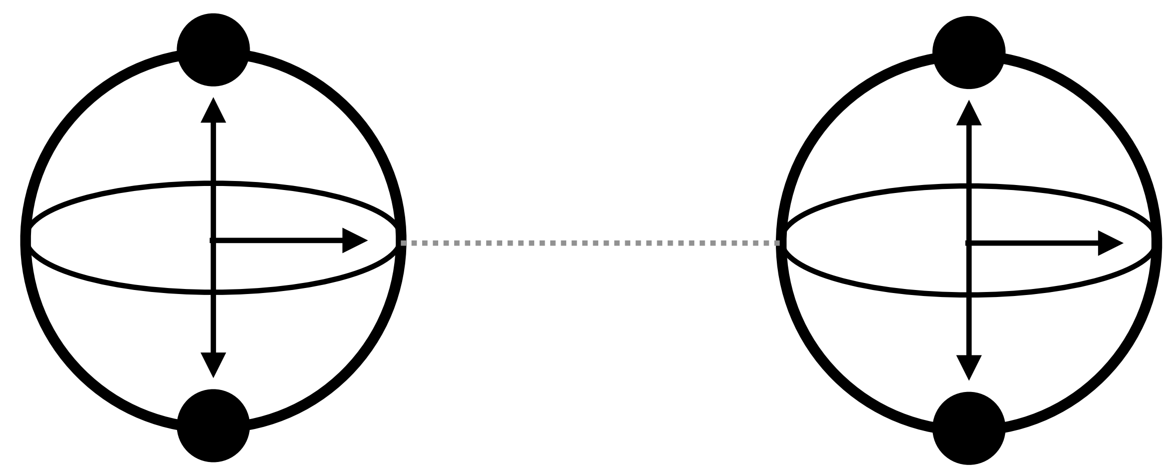
$|0\rangle$ $|1\rangle$

- 하나의 큐비트 관찰 결과가 다른 큐비트 상태에 영향을 주지 않음

얽힘 (Entanglement)

- 얽힌 상태 = 개별 큐비트 상태들로 분리 불가능

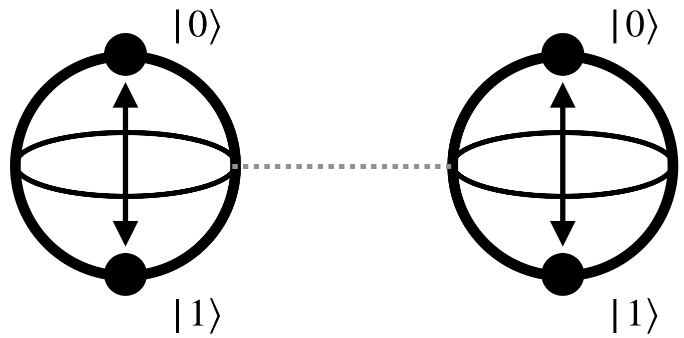
$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}$$



$$\neq (\alpha_0|0\rangle + \alpha_1|1\rangle) \times (\beta_0|0\rangle + \beta_1|1\rangle)$$

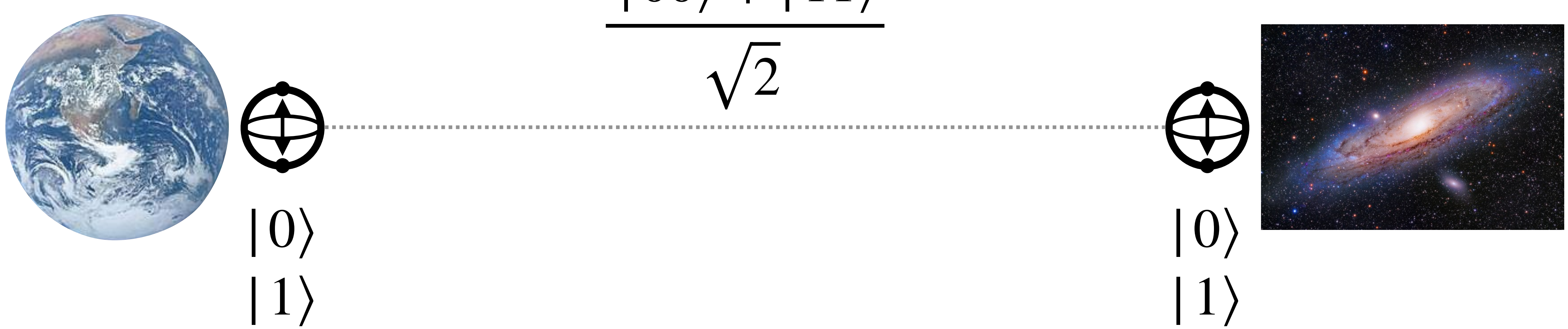
- 하나의 큐비트 관찰이 다른 큐비트 상태에 영향을 줌
    - 하나의 큐비트에서 0을 관측하면 다른 큐비트 상태는 $|0\rangle$로 결정
    - 하나의 큐비트에서 1을 관측하면 다른 큐비트 상태는 $|1\rangle$로 결정

EPR 패러독스

1. 두 큐비트  를 $\dfrac{|00\rangle + |11\rangle}{\sqrt{2}}$ 상태로 준비

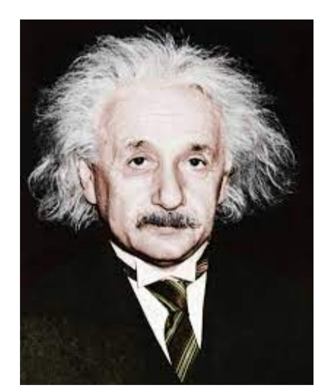
2. 두 큐비트를 서로 멀리 떨어지도록 이동

$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}$$



3. 측정

"Spooky action at a distance"



양자 게이트

$$\alpha_0|0\rangle + \alpha_1|1\rangle \quad \text{———}\boxed{X}\text{———} \quad \alpha_1|0\rangle + \alpha_0|1\rangle$$

- 큐비트 상태 = 벡터

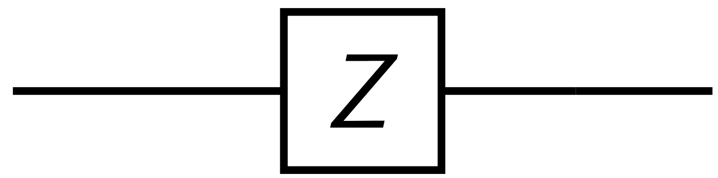
$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle = \alpha_0\begin{bmatrix}1\\0\end{bmatrix} + \alpha_1\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha_0\\\alpha_1\end{bmatrix}$$

- 게이트 연산 = 행렬 곱

$$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix} \times \begin{bmatrix}\alpha_0\\\alpha_1\end{bmatrix} = \begin{bmatrix}\alpha_1\\\alpha_0\end{bmatrix}$$

양자 게이트

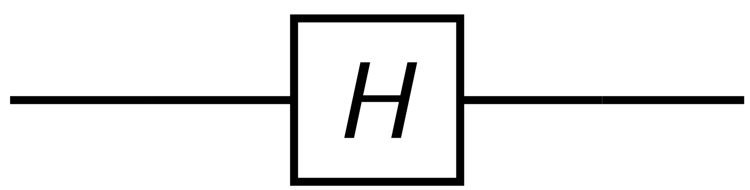
$$\alpha_0|0\rangle + \alpha_1|1\rangle$$



$$\alpha_0|0\rangle - \alpha_1|1\rangle$$

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\alpha_0|0\rangle + \alpha_1|1\rangle$$

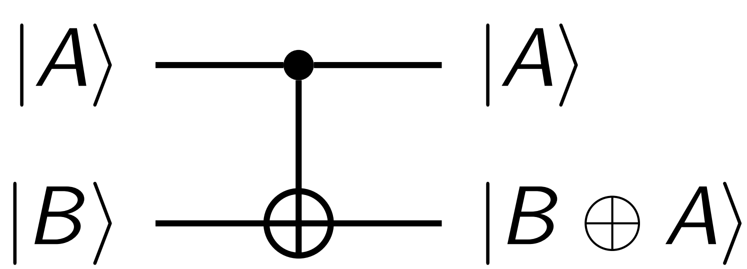


$$\alpha_0\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) + \alpha_1\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

양자 게이트

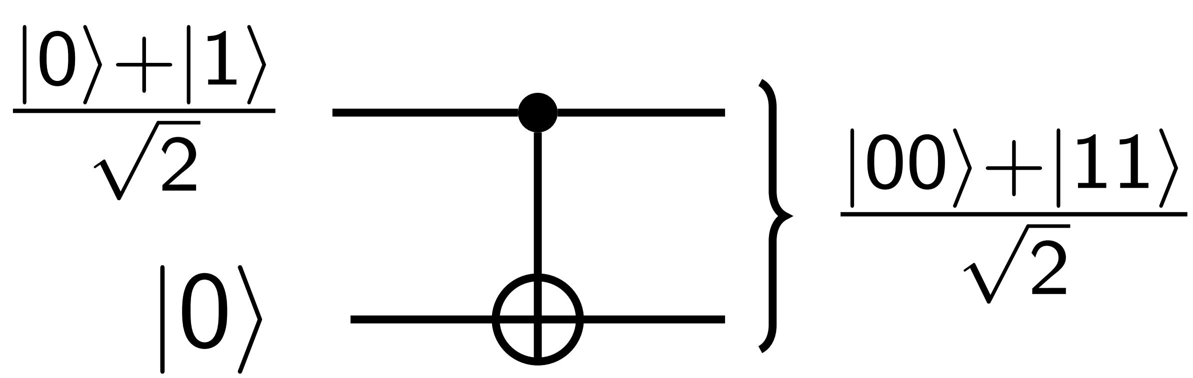
- CNOT (Controlled-Not) 게이트



$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{aligned} &CNOT(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle) \\ &= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle \end{aligned}$$

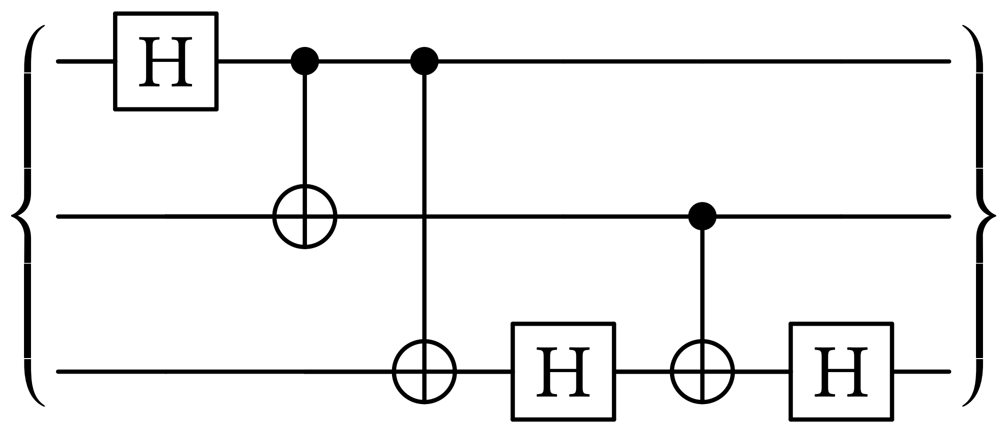
- CNOT: 얽힌 상태를 만드는데 사용



양자 회로

- 양자 게이트들의 조합

$$|100\rangle$$

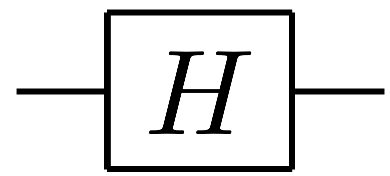


$$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

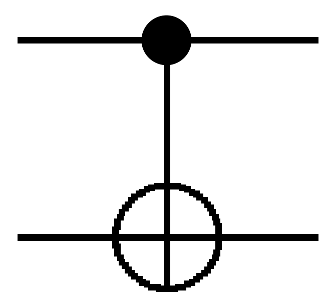
```
circ = QuantumCircuit(3)
circ.h(0)
circ.cx(0, 1)
circ.cx(0, 2)
circ.h(2)
circ.cx(1, 2)
circ.h(2)
```

양자 프로그래밍의 어려움

- 고전 프로그래밍(e.g. Python, C, Java, …)과는 전혀 다른 방식
- 양자 프로그래밍은 비직관적
    - 데이터 값 : 벡터
    - 프로그램 연산자 : 행렬



$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$



$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

양자 회로 자동 합성 (OOPSLA 2023)

입출력 명세

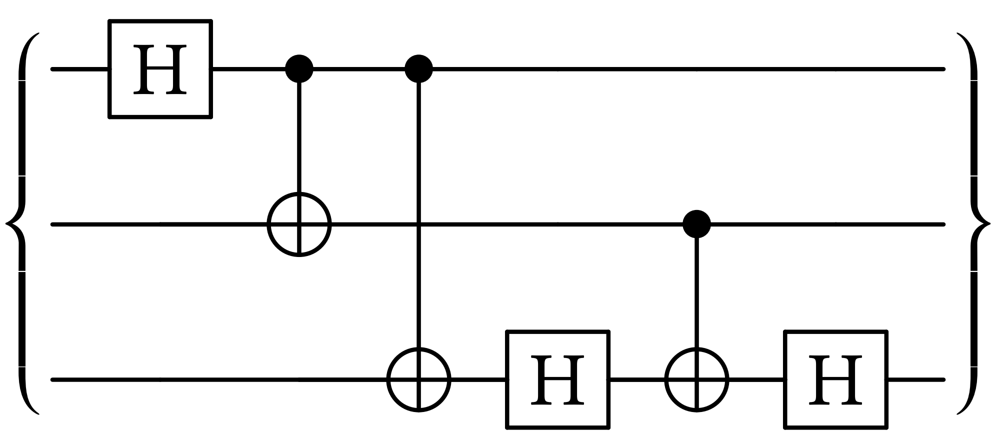
$$|100\rangle \mapsto \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

컴포넌트 게이트

$$H, CNOT$$

양자 회로 합성기

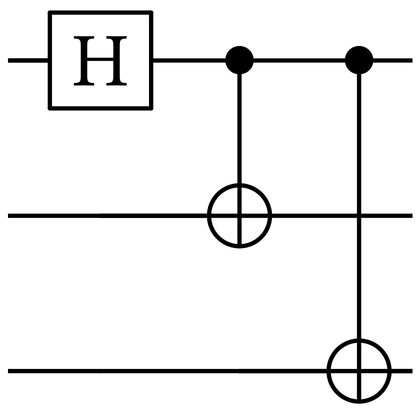
$|100\rangle$

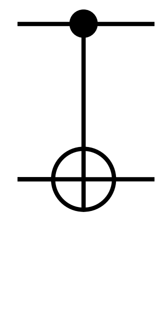


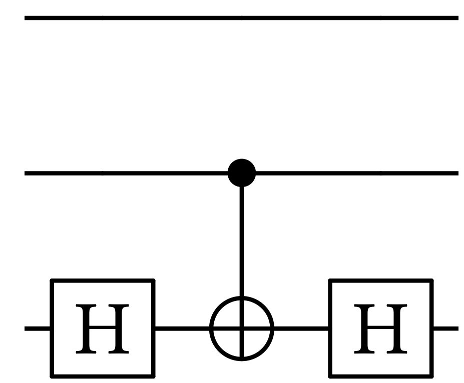
$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$

아이디어: 모듈 레벨 양자 회로 합성

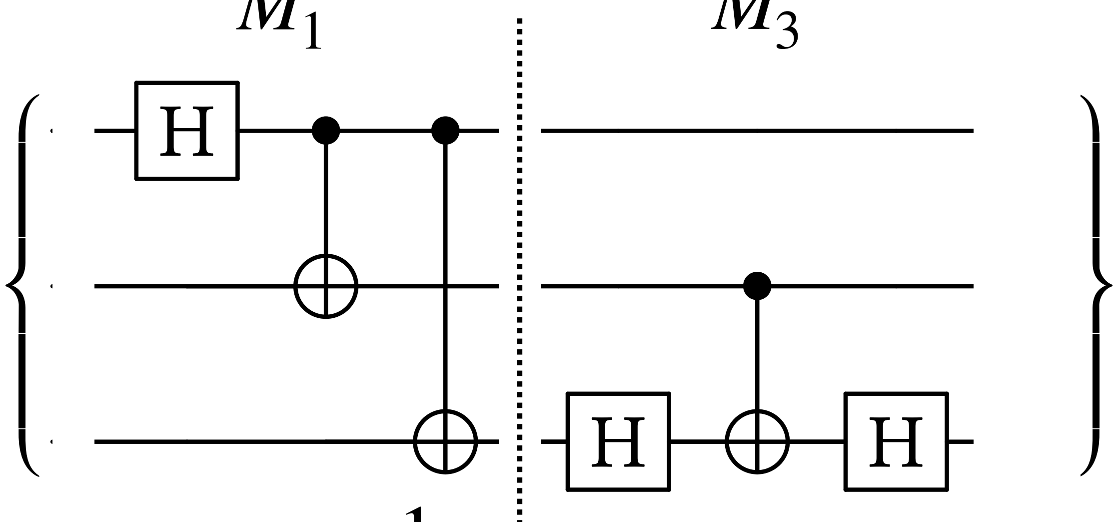
1. 모듈 = 게이트 시퀀스 (up to some fixed length)

$M_1 =$ 

$M_2 =$ 

$M_3 =$ 

2. 명세와의 특성 차이를 줄이는 방향으로 모듈 쌓기

$|100\rangle$ 

$|GHZ\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$

$$\frac{1}{\sqrt{2}}(|100\rangle - |111\rangle) \quad \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$$

$|100\rangle \longleftrightarrow$ '얽힘' 차이 $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \longleftrightarrow$ '위상' 차이 $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$

합성 알고리즘의 안전성 (Soundness)

“특정 가정하에서 모듈 기반 합성 알고리즘이 항상 정답을 찾아냄”

Theorem 4.17. *Let* $E = \{(|in\rangle, |out\rangle)\}$ *be an example and* $C^* = M_1; \cdots; M_k$ $(M_i \in \mathcal{M}$ *and attribute of each* M_i *is not* Identity*) be the solution circuit to be synthesized such that* $C^*(|in\rangle) = |out\rangle$. *Suppose* C^* *is monotonically decreasing (by input* $|in\rangle$*). Then, for any prefix* $C = M_1; \cdots; M_{l-1}$ $(l \leq k)$ *of* C^*,

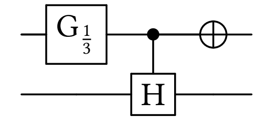
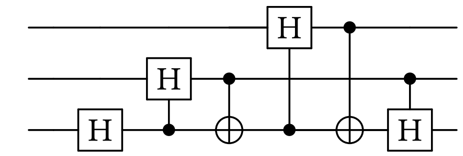
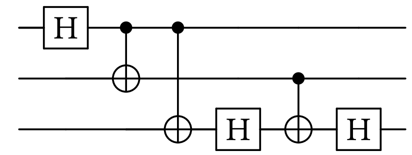
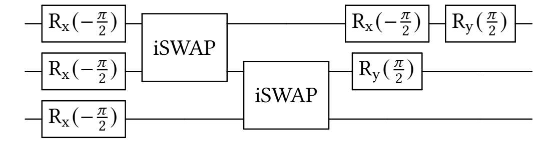
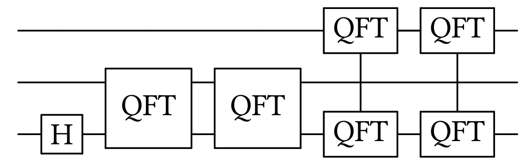
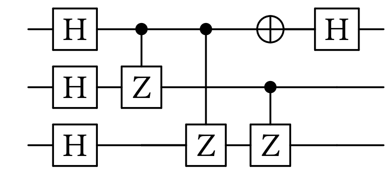
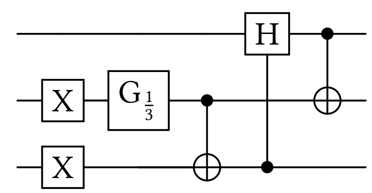
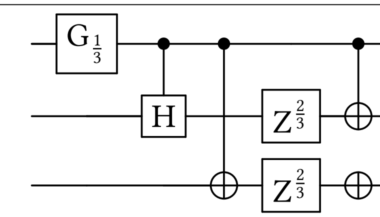
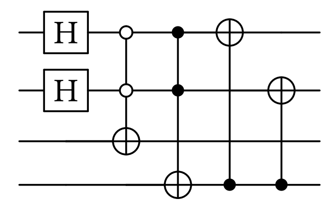
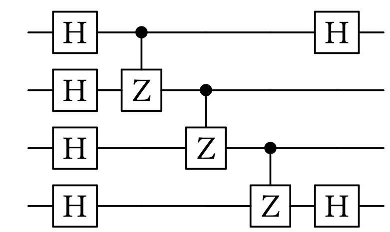
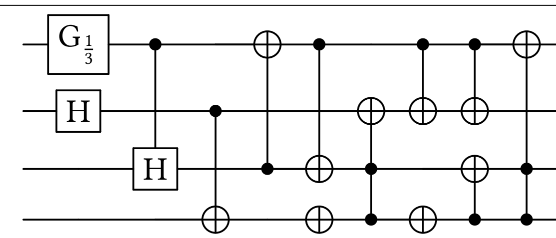
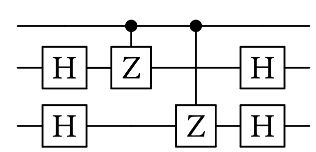
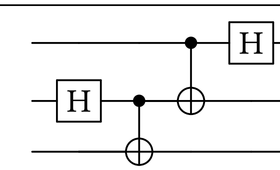
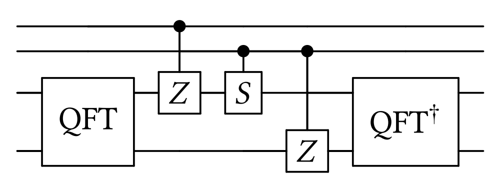
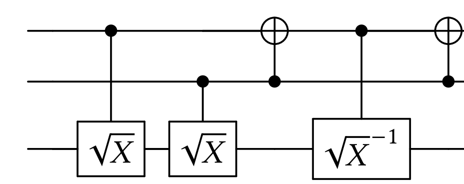
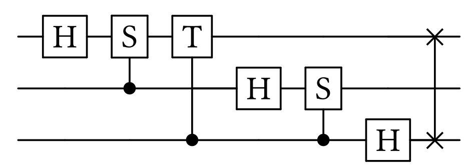
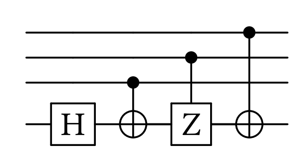
$$\text{is\_gap\_filled}(C, M_l, (|in\rangle, |out\rangle)) = \mathit{True}.$$

Proof. Let $|\psi\rangle = C\,|in\rangle = M_{l-1} \cdots M_1\,|in\rangle$ be the output vector of C. By definition $M_k \cdots M_l\,|\psi\rangle = |out\rangle$ and thus

$$Att_{|\psi\rangle}(M_k M_{k-1} \cdots M_l) = |out\rangle \ominus |\psi\rangle = \text{gap\_att}_{|in\rangle,|out\rangle}(C).$$

Since C^* (and so C) is decreasing, by Lemma 4.16 $Att_{|\psi\rangle}(M_k M_{k-1} \cdots M_l) = Att_{|\psi\rangle}(M_l)$. Therefore, $\text{gap\_att}_{|in\rangle,|out\rangle}(C) = Att_{|\psi\rangle}(M_l)$, which is satisfying the criterion. □

벤치마크

Type	ID	Circuit	ID	Circuit
State Preparation	three_superpose		M_valued	
	GHZ_from_100		GHZ_by_iSWAP	
	GHZ_by_QFT		GHZ_Game	
	W_orthog		W_phased	
	W_four		cluster	
	bit_measure			
Multi IO	flip		teleportation	
	draper		toffoli_by_$\sqrt{X}$	
	QFT		indexed_bell	

게이트 레벨 합성 알고리즘

모듈 레벨 합성 알고리즘

ID	$Base_{no\_prune}$	Base	$Ours_{no\_prune}$	Ours	Spd-up
three_superpose	0.14	0.12	0.12	0.09	1x
M_valued	1764.75	1126.79	666.25	3.89	290x
GHZ_from_100	106.81	48.16	⊥	0.47	102x
GHZ_by_iSWAP	⊥	⊥	690.67	2.19	-
GHZ_by_QFT	116.90	101.65	101.17	39.26	3x
GHZ_Game	⊥	2305.71	4.51	0.57	4058x
W_orthog	2927.20	2075.06	248.23	2.43	854x
W_phased	⊥	⊥	258.56	5.43	-
W_four	⊥	⊥	2851.10	254.88	-
cluster	⊥	⊥	3560.38	8.91	-
bit_measure	⊥	⊥	⊥	⊥	-
flip	18.56	3.95	0.76	0.83	5x
teleportation	2.02	1.30	1.35	1.35	1x
indexed_bell	14.67	11.66	1.60	1.52	8x
toffoli_by_$\sqrt{X}$	956.29	716.28	306.10	264.66	3x
QFT	⊥	⊥	⊥	220.87	-
draper	⊥	⊥	933.47	737.99	-
Avg. (excluding ⊥/-)	656.37	639.07	687.45	96.58	20x

Summary

- 양자 컴퓨팅 및 프로그래밍 소개
- 양자 회로 자동 합성 소개
- 양자 SW + X
  - 양자 SW 합성, 최적화, 분석, 검증, 수정, 언어 디자인

감사합니다!