量子計算の基礎

連載の第4回目となる今回は、量子計算の基礎概念を紐解き、その能力に迫ります。

そもそも計算とは

そもそも「計算」とは何で、機械はどのように計算を行っているのでしょうか？現在世の中にあふれているパソコンやスマートフォンは絶えず計算をする機械です。 しかしその中身で何が動いているのかについて考えることはあまりないかもしれません。

連載第一回でも見たように、今存在しているコンピュータは通常、「0」と「1」の2つの状態を持つビットを情報の基本単位として計算を行っています。 例えば「足し算」という計算を行う場合、コンピュータは入力された2つの数値 A、 B をビット列（２進数）として受け取り、それらに対応した答え C=A+B をまたビット列として出力します。 私たち人間は2進数の足し算のルール（入出力関係）を知っているので、A、B の各ビットから C の各ビットを計算する際のルールを書き下すことができます。例えば C の最下位ビット c は、A の最下位ビット a と B の最下位ビット b が (a,b) = (0,0) のとき 0、(0, 1) または (1, 0) のとき 1、(1,1) のとき 0 になる、といえます。 このようなルールはすべて論理ゲートと呼ばれる素子で表現できることが知られています。 論理ゲートはトランジスタなどの素子で実装され、それらを組み合わせることで適切な計算を行うのがコンピュータです。

論理ゲートには様々な種類があり、極論するとビットを入力するとビットを出力する操作はすべて論理ゲートであると言えます。 しかし、ハードウェアで効率的に実装できる論理ゲートは限られており、重要なものには名前がついています。 例えば NAND という論理ゲートは特に重要で、すべての「ルール」をその組み合わせで表現できることが知られています。 NAND ゲートは 2 ビットを入力すると 1 ビットを出力する論理ゲートで、入力が 11 のときだけ 0 を出力し、それ以外の入力に対しては 1 を出力します。 例えば上の例における c を計算するルールは、(a NAND (a NAND b)) NAND ((a NAND b) NAND b) とかけます。図に対応する論理回路を示しました。 その他の任意の「計算」、例えば割り算や掛け算についても同様に、NAND ゲートの組み合わせで表現できます。

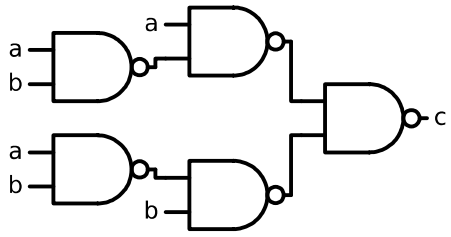


図1 NANDゲートだけで構成された、aとbの和の最下位ビットを求めるための論理回路。各直線が1ビットの情報線を表しており、左から右に信号が流れます。四角に半円を組み合わせたような記号がNANDゲートです。例えば一番左では a と b を左から入力として受け取り、a NAND b を右へ出力する NAND ゲートが二つ描かれています。その後 3 つの NAND ゲートを経て、最終的に所望の出力である c が得られます。

量子計算

量子ビット

古典コンピュータにおけるビットと論理ゲートの代わりに、量子ビットと量子ゲートを用いて計算を行うのが量子計算です。 量子ビットは、0 と 1 の状態のほかに、それらの重ね合わせ状態を取ることができる素子です。 ある一つの量子ビットの状態は、次のように表現されるのでした（連載第一回）。

$$c_0|0\rangle + c_1|1\rangle$$

ここで、c_0 と c_1 は複素数です。 この状態は二つの複素数によって完全に決定されるので、$|0\rangle, |1\rangle$ のような表記をやめて、$[c_0\ c_1]^{\mathrm{T}}$ のように、列ベクトルで表現することもあります。 量子計算は、このような性質を備えた量子ビットを多数用いて行う計算です。 n個の量子ビットがあるとき、これらの量子ビットは一般に 00...0 から 11...1 までの 2^n 個の状態を重ね合わせた状態を取ることができます。 数学的には次のように表されます。

$$c_{00\ldots0}|00\ldots0\rangle + c_{00\ldots1}|00\ldots1\rangle + \cdots + c_{11\ldots1}|11\ldots1\rangle$$

$c_{00\ldots0}$ などは複素数です。 ひとつの量子ビットの場合と同様に、2^n 個の複素数によって状態が完全に決定されるので、$[c_{00\ldots0}\ c_{00\ldots1}\ \cdots\ c_{11\ldots1}]^{\mathrm{T}}$ のように列ベクトルで表現することもできます。 **このことは、十分なメモリさえ確保すれば、古典コンピュータで量子ビットの状態を完全にシミュレートできることも意味します。** ある重ね合わせ状態に対応する 2^n 個の複素数を保存しておけば、それは量子状態と全く同じ情報を持っているからです。

さて、重ね合わせ状態にある量子ビットを、人間はそのまま認知できません。 人間が認知できる情報を得るためには、「観測」という操作を行う必要があります。 観測すると、各量子ビットの状態は 0 か 1 のどちらかに確定します。 このとき、どちらが得られるかは真に確率的に決まり、 上の式で表される状態を観測すると、$|c_{00\ldots0}|^2$ などの確率で、それぞれのビット列が得られます。 このことを反映して、量子ビットの状態としては、係数の絶対値の 2 乗の合計が必ず 1 になるような状態のみ許されます。 物理的には、例えば超伝導量子ビットの場合、適当な周波数の電磁波を打ち込んで、その反射波の振幅や位相を測定することで行われます。

量子ゲート

量子ゲートは古典コンピュータにおける論理ゲートに対応するものです。 古典コンピュータにおける論理ゲートと同様に、量子ビットに対する操作はすべて量子ゲートと呼ぶことができますが、ハードウェアで効率的に実装できるものは限られています。 重要な量子ゲートには名前がついており、例えばアダマール (H) ゲートや 制御 NOT (controlled-NOT, CNOT) ゲート、トフォリゲートがあります。 これらのゲートの回路記号とその動作について図2,3にまとめました。 H ゲートは均等な重ね合わせ状態を作り出す1量子ビットゲートです。 CNOT ゲートは制御ビットが 1 のときに対象ビットを反転するゲートです。 最後にトフォリゲートは、図3のように3つの量子ビットを入力として取り、最初の2つの量子ビットが 1 のときのみに対象ビットを反転するゲートです。

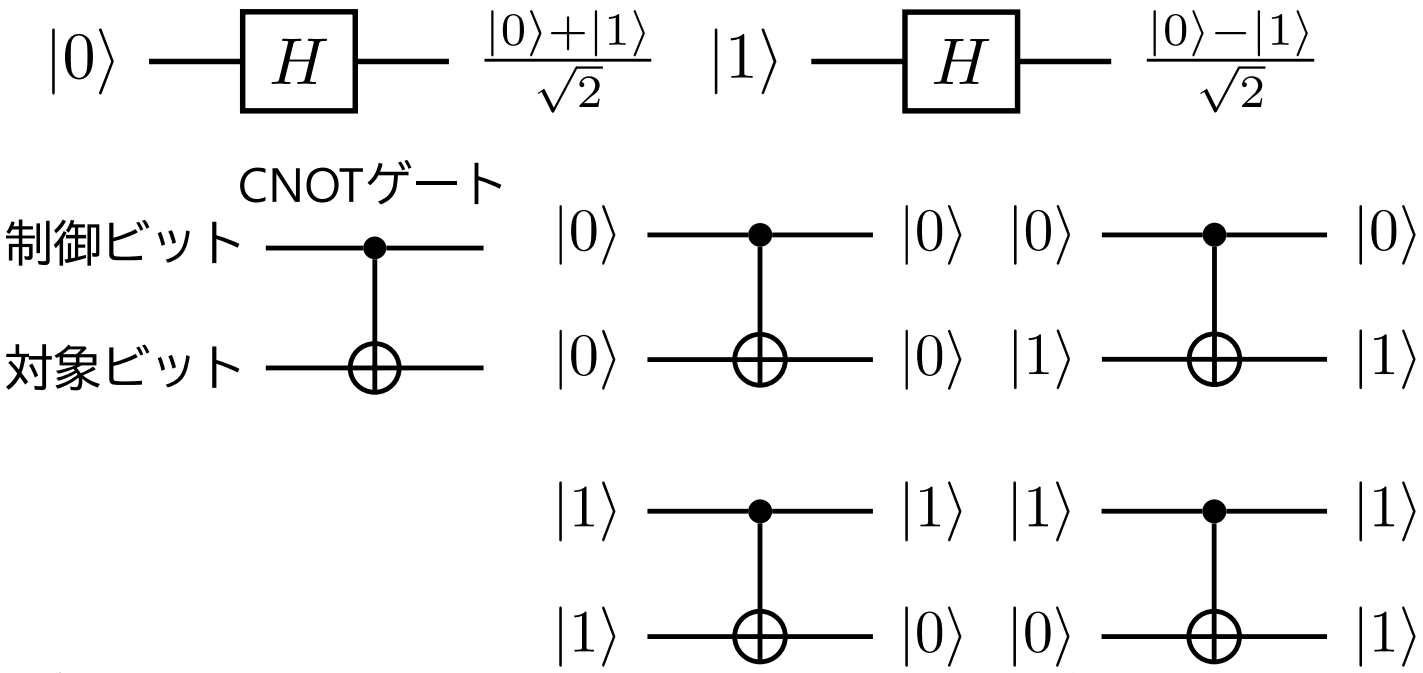
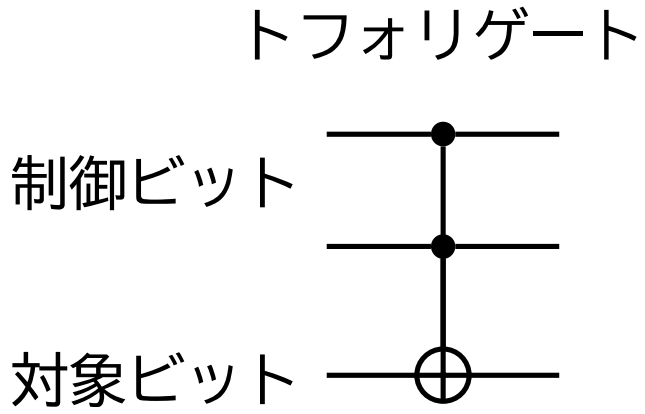


図2 アダマールゲートとCNOTゲート。古典の論理回路図（図１）と同様に、直線が信号（この場合は量子ビット）を表し、情報は左から右に流れていきます。上に示したアダマールゲートは、1ビットの情報を受け取り、均等な重ね合わせ状態を作り出します。下に示したCNOTゲートは、2ビットの情報を受け取り、制御ビットが1のときだけ対象ビットを反転するゲートです。



入力	出力
000	000
001	001
010	010
011	011
100	100
101	101
110	111
111	110

図3 トフォリゲートとその入出力関係。回路図は図２と同様に、直線が情報を表し、左から右に情報が流れます。トフォリゲートは3ビットの情報を受け取り、最初の2ビット（制御ビット）が両方とも1のときだけ対象ビットを反転します。

量子計算と古典計算

量子力学的な重ね合わせ状態を用いるなど、従来の古典計算とは全く異なる計算手法に思える量子計算ですが、実は量子計算は古典計算を完全に包含しています。 つまり、古典計算でできることはすべて量子計算でもできるということです。 トフォリゲートを見てみましょう。よく見ると、トフォリゲートの対象ビットに 1 を入力することで、制御ビットの2つのビットを入力とし、対象ビットを出力とした NAND ゲートを構成することができますね。 NAND ゲートはすべての古典計算を実行できるのでしたから、トフォリゲートを使えば、どんな古典計算も完全に模倣できることがわかります。

このことを見ると、量子コンピュータが完成すれば、従来の古典コンピュータは必要なくなるのではないか、と思うかもしれません。 しかしそれは正しくなく、多くの研究者は--少なくとも筆者は--量子コンピュータが古典コンピュータを完全に置き換えることはないと考えています。 それは、量子コンピュータはその動作速度（単位時間当たりに実行できる演算回数）が、古典コンピュータよりもほぼ間違いなく遅くなるだろうと考えられるからです。 連載第２回でも見たように、量子コンピュータの制御はマイクロ波などを量子ビットに照射することで行われますが、そのマイクロ波の制御は基本的に古典コンピュータや集積回路によって行われます。図4にその概念を示しました。 このような状況では、量子コンピュータの動作速度は古典コンピュータに律速され、動作速度が古典コンピュータを超えることはありません。

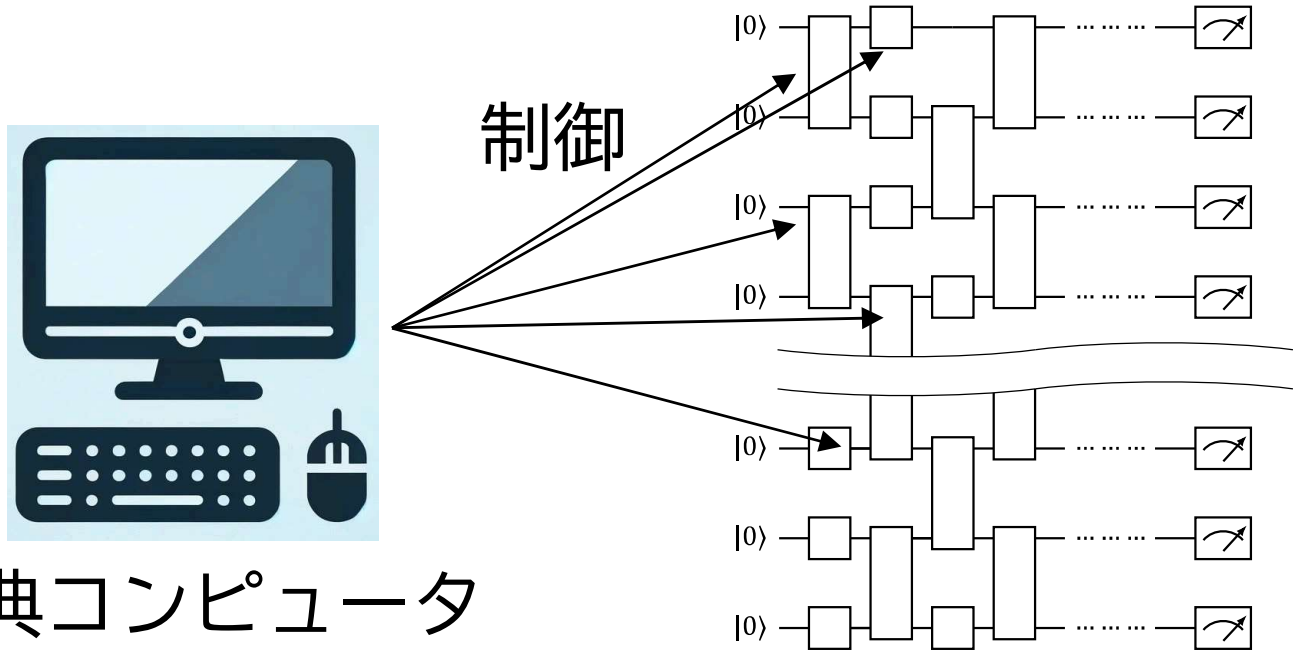


図4 量子コンピュータは古典コンピュータによって制御されます。右図は量子回路で、量子ゲートの列を図的に表したものです。量子計算は0に初期化した量子ビットに対して量子ゲートを作用させ、最後に量子ビットを観測することで行われます。

具体的な数字を見てみましょう。 例えば、[J. Lee, et al., PRXQuantum 2, 030305 (2021)] では、誤り訂正のためのオーバーヘッドも含めると、1 秒間に 25,000 回のトフォリゲートしか実行できないと試算されています。 例えば足し算を行うためにはトフォリゲートを数十個組み合わせる必要がありますから、量子コンピュータは、足し算を多くとも 1 秒間に 1000 回程度しか行えない計算になります。 現在の一般的なコンピュータでは、少なくとも 1 秒間に 10 億回の足し算が行えるので、量子コンピュータは古典コンピュータに比べて非常に「遅い」ことは明らかです。

量子計算の適用範囲

上のような事情から、量子計算の意味のある適用先は限られています。古典コンピュータと比べて数桁のオーダーで動作速度が遅いので、量子の力によって、それを補ってあまりある加速が得られる問題にのみ適用すべきです。

理論的な保証つきで、量子コンピュータによってそのような高速化が可能な問題は、

- 問題固有の数学的性質によって、なぜか解けてしまう問題
- そもそも特定の量子力学重ね合わせ状態を作ることが目的となるような問題

の２つに大別されます。1つ目の例としては、大きな整数の素因数分解があげられます。2つ目の例としては、量子力学に従う物理系のシミュレーションがあげられます。化学を専門とする読者に最も関連する問題であり、かつ量子コンピュータが最も得意とする問題です。

一方、近年では量子コンピュータのハードウェアの発展も相まって、理論保証のないヒューリスティックな量子アルゴリズムの研究も盛んに行われています。その応用範囲は多岐に及び、量子化学計算はもちろんのこと、機械学習や組み合わせ最適化問題などへの適用を目指し、研究が進められています。

今後の連載では、理論保証のあるアルゴリズムから、このようなヒューリスティックなアルゴリズムまで、量子計算の応用範囲について幅広く紹介する予定です。それでは、次回もお楽しみに！