第 3 章

Introduction to computer science

3.2 The analysis of computational problems

3.2.3 Decision problems and the complexity classes P and NP

多くの計算問題はたいてい「はい」か「いいえ」で答えられる決定問題としてはっきり定式化されている。計算複雑性の中心的なアイデアは簡単にそして頻繁に決定問題の言葉へ言い換えられる。これには 2 つの理由がある。1 つはそこでの定理がとても簡単かつシンプルな形になること、もう 1 は歴史的には計算複雑性というのは決定問題についての研究から生じた概念であるからである。

決定問題は「形式言語」というところから話が始まる。

形式言語

アルファベット Σ 上の言語 L というのは Σ の元からなるすべての有限列 (これを語という) を含む集合 Σ^* の部分集合のことである。

言語の例として、アルファベットを $\Sigma = \{0, 1\}$ をとると、語全体の集合は

$$\Sigma^* = \{\emptyset, 0, 1, 00, 01, 10, 11, 000, \cdots\}$$

というようになる。ここで重要なのは長さが 0 の空単語というのも含むことである。これは非決定性オートマトンといったものを考えるときに必要になる。この中の言語 L というのは

$$L = \{0, 10, 100, 110, \cdots\}$$

というようになる。ちょうど自然言語もすべてのアルファベットを組み合わせた語を取り入れてないのと同じである。

もともと、決定問題というのはチューリングマシンにある入力をしてそれが「はい」か「いいえ」で回答させるものである。ただ、現代的には入力を語としてそれが言語に含まれているかどうかというのを考えるものになる。入力を語として受け取ったオートマトンがちゃんとした終状態をとるか (受理するか) どうかで考えられる。

ある決定問題が与えられたとき、これの判別にかかる時間を知りたいという問題を考える。それは最速で判別するチューリングマシンはどのようなものかというように置き換えれる。

決定問題の分類

長さ n の入力された語 x が言語として受理するのに判別する時間が $\mathcal{O}(f(n))$ かかるようなチューリングマシンが存在するとき、そのような問題を $TIME(f(n))$ に属するという。また、チューリングマシンが存在するするということは判別するための言語というのがあるということを意味する。つまり、問題の集合 $TIME(f(n))$ の元は言語である。

正直この本文だけだと、判別するのに必要な時間というのがどのように決まるかというのは与えられていない。おそらく、オートマトンのステップ数で決まるような気はする。

この定義を使って決定問題を分類していこう

P 問題

$TIME(n^k)$ のように入力の語の長さに対し、多項式時間で決まる問題の集まり (複雑性クラス) を P 問題という。

P 問題に入ることを言うことは実際にそういったオートマトン等を作ればよいが、P 問題に入らないことを言うのは結構難しい。というのも、予想自体はオートマトンを試しにいろいろ作ってみて計算量のオーダー自体はつかめるが、思いつかないようなものがるかもしれないということだ。存在命題なので否定するには背理法使って証明するしかないというのが厳しいのだろう。そのような問題で単純かつ面白い決定問題が、素数であるかどうかを判別する問題である。具体的に因数を求めるのは大変であるがいったん求めると、すぐに確認 (witness) できる。このように答えがあっているかすぐ確認できるかどうかという基準は複雑性クラスを分類するのに使える。

NP 問題

与えられた入力を表す語 x が言語 L にとして受理される場合、確認に使える語が存在する。も与えたときに多項式時間で判別できる問題を NP 問題という。もっと専門用語を使うと非決定性チューリングマシンを用いて多項式時間で判別できる問題を NP 問題という。

この定義を見ると、P 問題は NP 問題に含まれているのがわかる。しかしその逆というのは証明するのは大変難しい。(P≠NP 問題)

2 つ目の定義は非決定性チューリングマシンは本文では扱っていないので 1 つ目の定義のみになっている。ここでは確認という単語は入ってないが、オートマトンの構造として非決定性をいれるというのが確認する語に相当するものになっている。

本文では入力された語 x が受理されるかされないかで条件を分けてもある。これは非決定オートマトンにおける終状態とそうでない状態の非対称性に来ているとも言い換えられる。

つまり、NP 問題における入力された語 x を言語が受理するかどうかというのを入れ替えた問題というのも作ることができる。

coNP 問題

ある決定問題 S の補問題 がクラス NP に属する場合、 S はクラス co-NP に属するという。

これは先ほど挙げた素数の判別の例でいうと、ある数が合成数であるかどうかというのは coNP 問題となる。この例だけ見ると NP も coNP も同じ複雑性クラスになっているように見えるが、これを証明するのは P≠NP を証明するのと同等以上に難しい。

Excercise 3.18

もし coNP≠NP なら P≠NP を示せ。

3.2.5 Energy and computation

今まで扱ってきた計算複雑性の問題は計算問題を解くのに必要な時間とメモリの大きさについて研究するものであった。そしてほかにも重要な計算資源としてエネルギーがある。この節では計算に必要なエネルギーについて学んでいく。驚くことに原理的には古典、量子ともにエネルギー消費なしに計算をすることがないことがわかる。

ここに関わってくるのは計算の可逆性と情報の消去である。これらは等価である (と言ってるがたぶん違う)。計算の前後で情報が落ちた場合、逆の計算をすることができない。例えば NOT ゲートはアルファベットを取り換えているだけなので情報は落ちていない。これはもとに戻すことができる。一方 NAND ゲートは 2 bits を 1 bit にしていて情報が落ちていているため不可逆である。逆に計算が不可逆であるならば情報が落ちるというのは示せない。待遇として情報が落ちていないなら可逆であるというのを考えてみる。計算を単なる物理的な過程だと思う。流体の拡散について、流体を構成するすべての粒子の運動の情報をすべて知っているとする。そのまま時間発展して拡散させていったときに、これの逆過程を行うことができるかどうかというとできない。そのような逆過程を実現できる確率 (測度) は 0 であるので実行できない[*1]。

情報の消去とエネルギー消費量を結びつけるものとして、Landauer の原理がある。

Landauer の原理

メモリが 1 bit 分の情報を蓄えているとする。確率 p で 0, $(1-p)$ で 1 になるとする。このときのシャノンエントロピーを

$$H(p) = -p\ln p - (1-p)\ln(1-p)$$

とする。これを確率 1 で 0 にするのを情報消去という。このとき、計算機の周囲の環境系へ散逸するエネルギーの量は 少なくとも

$$E = k_B T H(p)$$

である。また、エントロピーで表すと 少なくとも

$$S = k_B H(p)$$

だけ環境系に流出する。

これはシラードエンジンというメモリを熱力学的に扱うモデルによって説明される。[*2]確率 p で左に理想気体粒子が 1 つある 0 の状態にあり、確率 $1-p$ で右に理想気体粒子が 1 つある 1 の状態にあるものとする。初期状態では体積 V の真ん中で区切られているものとする。

まず初めに左に粒子があるとき、準静的な圧縮で体積を pV にするのに必要な仕事は

$$W_2^{(0)} = \int_{V/2}^{pV} -pdV = -k_B T \int_{V/2}^{pV} \frac{dV}{V} = -k_B T \ln(2p)$$

その後仕切りを外して右側の壁を左に寄せていってまた半分にすると、

$$W_4 = \int_{V}^{V/2} -pdV = k_B T \ln 2$$

一方、初期状態として右に粒子があるときに (ii) の過程で必要な仕事は

$$W_2^{(1)} = \int_{V/2}^{(1-p)V} -pdV = -k_B T \ln(2(1-p))$$

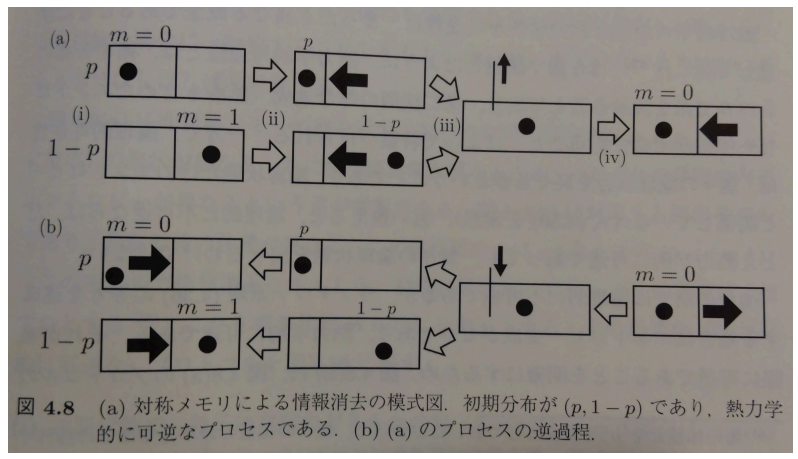

図 4.8 (a) 対称メモリによる情報消去の模式図．初期分布が $(p, 1-p)$ であり，熱力学的に可逆なプロセスである．(b) (a) のプロセスの逆過程．

図 3.1: シラードエンジンの模式図。沙川先生の非平衡統計力学から引用。黒い点は理想気体粒子を表していて、これがある領域では圧力があると考える。

[*1] ここが微妙なところなんだと思う。

[*2] 相当抽象化してるモデルになってるのでこれで一般的な場合をちゃんと扱えてるかどうかは私にはわからない。

なので消去に必要な仕事は

$$W = pW_2^{(0)} + (1-p)W_2^{(1)} + W_4 = k_B T\{-p\ln p - (1-p)\ln(1-p)\} = k_B T H(p)$$

というように示せる。エントロピーへの変換は準静的過程における熱力学第一法則

$$dU = -dW + TdS$$

を考えるとサイクルになっているので内部エネルギー変化 dU はないことからできる。これが下限になっているのは準静的過程での変化を使っているのに由来する。

進んだ補足をいくつか載せる。体積を $p : 1-p$ にしないとこの操作により、粒子の位置の確率分布は変化してしまい、熱力学的に不可逆になってしまうからしい。また、このサイクルでは初めにどちらに粒子があるかがわかった状態で話をしている。どちらに入っているかわからないときには観測をする必要がある。すると観測にもエネルギーが必要となり、観測して情報を消去するときにはシャノンエントロピーではなく、相互情報量が下限になるそうである。

では実際の計算機ではどれくらいになっているかというと、2000 年の計算機で大体 $500k_BT$ 程度である。

以下では情報の消去と計算の可逆性は同じものとして議論する。

この議論はいろいろと疑わしいが本文の流れに沿うため乗せる。

> 計算の不可逆性と情報の消去が等価であることと Landauer の原理を組み合わせると面白いことが言える。物理の理論は根本的に時間反転をすることができるためすべての過程は可逆である。もし、可逆過程のみで計算を行うことができるとすると、それは情報の消去をすることなしに計算をすることとなる。するとエネルギー消費の下限がなくなるため、エネルギー消費なしで計算することができるといえる。

まず計算の不可逆性と情報の消去は等価であるか疑わしい。次に物理の理論は時間反転をすることができるというように言っているが、時間反転対称性の破れた系は多くある。電子相関系でいうと交代磁性がその 1 つである。当然熱力学が使える系はボルツマンのH定理といったように時間反転はできない。

いずれにせよ可逆計算というのは注目されていて、どのようなモデルがあるかというのはいろいろと考えられてた。その 1 つとして Fredkin のビリヤードコンピュータがある。ビリヤードの玉の有無で情報ビットを表し、壁や玉同士の衝突によって計算を行うといったモデルである。完璧にビリヤードの玉を突いて、様々な擾乱が無いという古典力学が完璧に成り立つという非現実的な仮定をしているものの、セルオートマトンを用いたシミュレーションはできるのでそんなに悪くないモデルなのかもしれない。ビリヤードコンピュータは回路モデルと等価となるように設計することができる。その際に重要になるゲートが Fredkin (Controled-Swap) ゲートである。このゲートは制御ビットが 1 のとき、上 2 のビットを入れ替えるというゲートである。入れ替えの操作自体は 2 回連続で行うと元に戻るため、Fredkin ゲートを連続させるともとの状態に戻ることから確かに可逆ゲートである (Excercise 3.29)。教科書の図 3.14 は Fredkin ゲートになっている (Excercise 3.30)。

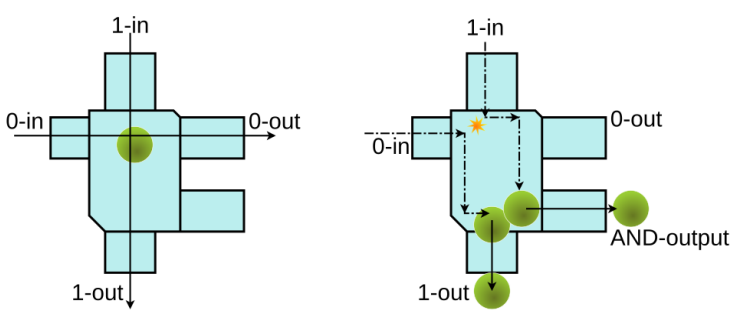

図 3.2: ビリヤードコンピュータによる AND ゲート。余計な出力のための穴があって情報が失われないようになっている。

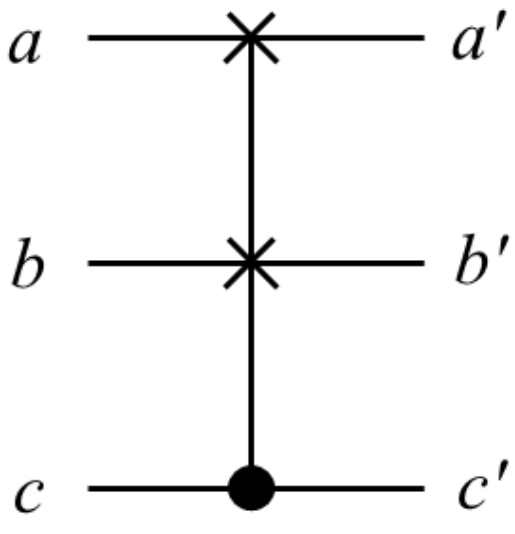

図 3.3: Fredkin (Controled-Swap) ゲート

また、この Fredkin ゲートは AND, NOT, CROSSOVER そして FANOUT 関数をシミュレーションできる (図 3.16)。この Fredkin ゲートを見ていったときにわかる特徴として、入力 x に加え、補助ビット a を用意して可逆ゲートに通すと望みの計算 $f(x)$ と副産物 (garbage)$g(x)$ を出力することになるというのがわかる。

この副産物は適切に可逆ゲートを使うとこで計算時間をあまり変えずに取り除くことができる。可逆ゲートとして計算結果に $f(x)$ と副産物 $g(x)$ を出力するようなゲート G と CNOT ゲートを考える。4 つのレジスタ $(x, 0, 0, y)$ を舞台に計算をしていく。古典計算においても CNOT ゲートは $(x, c) \to (x\oplus, c)$ というように働く。CNOT ゲートは $(0, c) \to (0 \oplus c, c)$ というよう

に FANOUT ゲートとして使える。なので次のような操作ができる。

$$(x,\,0,\,0,\,y) \overset{\text{FANOUT}}{\longrightarrow} (x,\,x,\,0,\,y) \overset{G}{\longrightarrow} (x,\,f(x),\,g(x),\,y) \overset{\text{CNOT}}{\longrightarrow} (x,\,x,\,g(x),\,y \oplus f(x)) \overset{G^{-1}}{\longrightarrow} (x,\,0,\,0,\,y \oplus f(x))$$

1. 第一レジスタを第二レジスタへとコピーする。
2. 第二レジスタと第三レジスタを可逆ゲート G に通す。
3. 第二レジスタを制御ビット、第四レジスタを目標ビットとして CNOT ゲートに通す。
4. 第二レジスタと第三レジスタを可逆ゲート G^{-1} に通す。