計算量理論　演習問題

平成 26 年 10 月作成　令和元年 11 月更新

目次

この演習問題について　情報系や数学系の学生が計算量理論を学ぶための演習問題である．東京大学理学部 (平成 24〜26 年度)，京都大学総合人間学部・大学院人間環境学研究科 (平成 25 年 12 月)，東京大学教養学部・大学院総合文化研究科 (平成 27〜28 年度)，九州大学理学部・大学院数理学府 (平成 29 年 7 月)，九州大学理学部 (平成 30〜31 年度) での講義に併用したものをもとに独習の便のため少し説明を加えた．通常の教科書よりは簡素だが，問に答えつつ読み進めることにより寧ろ短時間で要点を深く理解できることを目指した．

各問の左の小さい数字は学生が黒板で答を発表する際の制限時間 (分) としたものである．よく準備すればこの時間で述べ得る内容ということであり，解くにはその 5〜20 倍を要するであろう．特に 10 分以上の問は演習問題としてはかなり高度であり，意欲的な学生に研究分野としての計算理論の面白さを垣間見てもらうことを意図したものである．

第 1 章

計算量とは：上界と下界

単純な処理を組合せて何らかの目的を達する手順を定めたものを算法（アルゴリズム）という．一つの目的を達する（問題を解く）にも様々な算法があるが，費やす手間や資源の量は算法によって違う．計算量理論では，目的を達するために最低どれほどの手間を要するか考える．言い換えれば，一定の仕組や制約の下で行われる算法によって，どのような問題を解くことができ，どのような問題は解けないかを考える．これは，効率の良い有用な計算法を見出そうとする立場からだけでなく，計算によりなし得ることの限界はどこにあるのかという，理論的な興味からも重要な問といえるだろう．

勿論そのためには計算の手間というものをどう測るか決めねばならない．その尺度は 2〜3 章でも詳しく扱うが，ここでは取りあえず大まかに算法の中で行われる特定の演算・操作の回数のことであると考えて，幾つかの問題を例に計算の手間を調べてみよう．

1.1　例 1：整列

与えられた配列の要素を小さい順に並べ替える（整列する）という問題を考える．次の問ではこの問題に対する二つの算法をそれぞれ(1)と(2)で考えて計算量を測る．なおその量を大まかに表すために，(2)では附録 A.1 節で説明する O 記法を用いる．

1+8**問 1.1**　(1) 長さ n の整数配列を昇順に整列する次の方法を考える．まず配列の先頭 (1 番目) の要素を 2 番目，3 番目，…，n 番目の要素と順に比較し，前者が大きい場合には入れ替える．こうすれば配列中の最小の要素が先頭に来ることになる．次に 2 番目の要素を 3 番目，4 番目，…，n 番目の要素と順に比較し，前者が大きい場合には入れ替える．これで配列中で次に小さい要素が 2 番目の位置に来る．以下これを繰返すことで配列全体を整列する．この手順を選抜整列法(selection sort)という．選抜整列法を長さ n の配列に施すと，二つの要素の比較が何回行われるか．

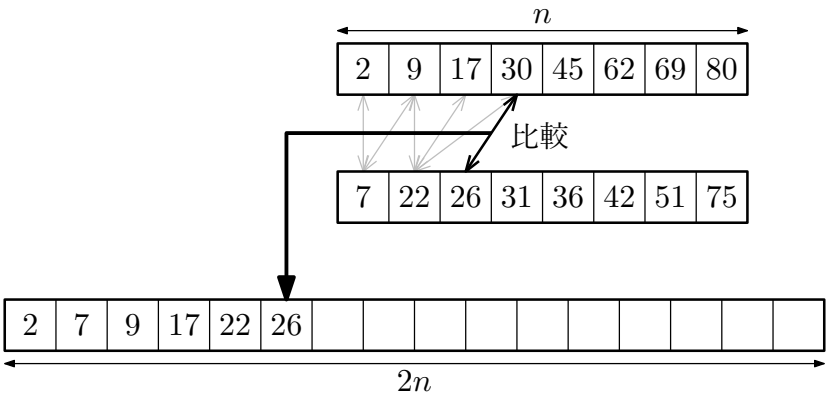


図 1.1　長さ n の既に整列された配列二つを併合して長さ $2n$ の配列にする．

数値のみ答えればよい．

(2) (1)よりも高速な整列算法を考えよう．

(a) 長さ n の整数配列が二つあり，それぞれは既に昇順に整列されている．両者を並行して先頭から読みながら比較することで，これら $2n$ 個の整数が昇順に整列された新たな配列を作ることができる．図 1.1 はその手順が行われている様子であり，30 と 26 を比較した結果，小さい方である 26 を新たな配列に書き写した所である．この次に行われるのはどの二数の比較であるか．

(b) (2)(a)の手順で行われる整数値の比較は $2n$ 回以下であることに注意せよ．さて，この手順を再帰的に用いることで配列を整列する算法が併合整列法(merge sort)である．どのような再帰を行うのか簡潔に述べよ．またこの算法で長さ n の整数配列を整列するときに行われる整数値の比較の最大回数 $T(n)$ について次が成立つことを説明せよ．

$$T(2n) = 2T(n) + \mathrm{O}(n) \qquad (\bigstar)$$

但し $\mathrm{O}(n)$ は n の定数倍以内であることを表す記法である (A.1 節)．

(c) ($\bigstar$) を満す任意の非減少函数 T について $T(n) = \mathrm{O}(n \log n)$ が成立つことを示せ．

注意　(2)(c)では O 記法の正確な定義に照して正しく論証すること．必要なら ($\bigstar$) をまず O 記法を使わない言い方に直してから議論せよ．

(1)より選抜整列法は $\mathrm{O}(n^2)$ 回，(2)より併合整列法は $\mathrm{O}(n \log n)$ 回の比較を行うから，この点では併合整列法の方が，大きな n については高速であるといえる．

すると更に少く済ます方法がないか知りたくなる．しかし次の問 1.2 からわかるように，併合整列法が最良 (のものの一つ) である．

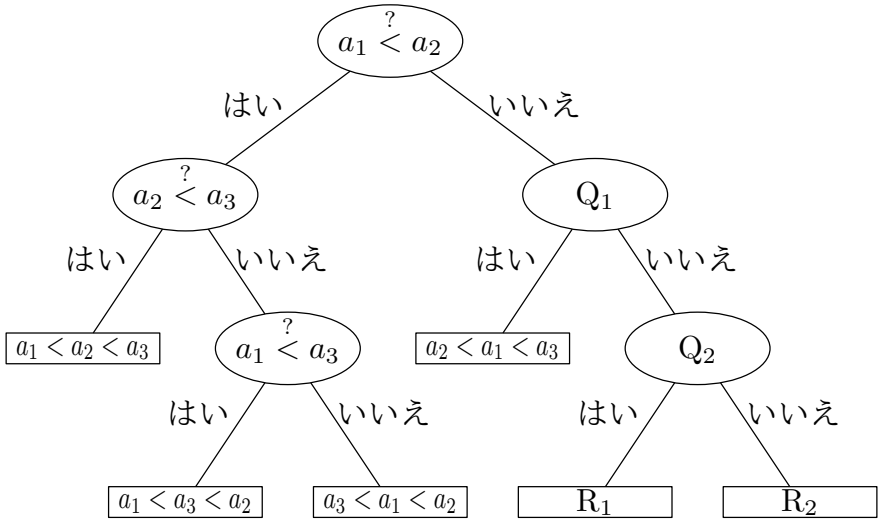


図 1.2 相異なる三数 a_1, a_2, a_3 の大小関係を二値の比較により決定する手順.

1+2+1+2**問 1.2** 与えられた n 個の相異なる整数 $a_1, \cdots, a_n$ を，二つの整数の比較のみを用いて整列したい．必要な比較の回数 k の下界を求めよう．

(1) $n = 3$ のとき図 1.2 の上部から始め，楕円の中に書かれた比較を順に行いながら結果に応じて進むと，終端では三数の大小関係が判る．図中の空欄 Q_1, Q_2 に入る比較と，空欄 R_1, R_2 に入る大小関係を答えよ．

(2) このように二整数の比較により $a_1, \cdots, a_n$ の大小関係を決定する手順は，根つき二分木を根から葉に向って辿ることで表される．この手順で行われる比較の回数の最大値 k は, すなわち木の深さ (葉から根までの最長径路の長さ) である．深さ k の二分木に葉は高々 **イ** 枚である．一方 n 個の相異なる整数の大小関係は **ロ** 通りあり得るから，葉は少くとも **ロ** 枚ある．これらより **イ** $\geq$ **ロ** が成立つ．空欄を埋めよ．

(3) $n! \geq (n/2)^{n/2}$ を示せ.

(4) $k = \Omega(n \log n)$ を示せ (Ω 記法については A.1 節).

問 1.1 (2)と同じように再帰的な算法の計算量を解析する例をもう一つ挙げておく．

3+3**問 1.3** n 行 n 列の行列(matrix) A, B の積 $C = AB$ を計算したい．A の (i, j) 成分を $a_{i,j}$ とし，B の (j, k) 成分を $b_{j,k}$ とすると，C の (i, k) 成分 $c_{i,k}$ は次で与えられる．

$$c_{i,k} = a_{i,1} \cdot b_{1,k} + a_{i,2} \cdot b_{2,k} + \cdots + a_{i,n} \cdot b_{n,k}$$

ここでは計算の手間を，数 (各成分) 二つの乗算を行う回数で測ることにする．上の定義に素朴に従うと一成分 $c_{i,k}$ を得るのに n 回の乗算を行うから，C 全体では n^3 回となる．

シュトラッセン(V. Strassen)は 1969 年，これよりも速い算法を得た．簡単のため以下 n を 2 の正整数乗とする．A, B をそれぞれ $n/2$ 行 $n/2$ 列の行列 4 つに分けて

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

と書き

$$\begin{aligned} M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) & M_2 &= (A_{21} + A_{22})B_{11} \\ M_3 &= A_{11}(B_{12} - B_{22}) & M_4 &= A_{22}(B_{21} - B_{11}) \\ M_5 &= (A_{11} + A_{12})B_{22} & M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

とすると次が成立つ．

$$AB = \begin{pmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{pmatrix}$$

これを再帰的に利用して AB を求めるのがシュトラッセンの算法である．この算法において数の乗算を行う回数を $T(n)$ とする．

(1) $T(n)$ と $T(n/2)$ の間の関係をいえ．

(2) (1)より $T(n)$ を求め，Θ 記法 (→ A.1 節) を用いて書け．

現在では問 1.3 よりも少い $\mathrm{O}(n^{2.373})$ 回程度の演算で済む算法が知られている[*1]．一方で積 AB には n^2 個の成分があり，それらは一般に別々の新しい値であるから，これを生み出すだけで少くとも $\Omega(n^2)$ 回の演算を要することは明らかである．この間のどこに真の限界があるかはわかっていない．

1.2　例 2：最大公約数

与えられた二つの正整数 a, b の最大公約数を求める問題を考えよう．$a > b$ とすると，この問題を解くには，各 $d = 1,\ 2,\ 3, \cdots,\ b$ について，d が a と b の両方を割り切るか調べ，割り切る d のうち最大のものを答えれば良い．この算法は「最大」の「公約数」を求めたいという目的からすると素直な考えだが，より速く計算する方法もある．

その一つが，古代ギリシアの数学者エウクレイデスによって紀元前三世紀頃に成ったとされる『原論』に記された**互除法**と呼ばれる算法である．図 1.3 はこれを再帰を使って記述したものであり，例えば $euclid(80, 36)$ を実行すると，初めの除算で $r = 8$ となるの

[*1] F. Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. 39th International Symposium on Symbolic and Algebraic Computation*, 2014.

手順 $euclid(a, b)$:
　a を b で割った余り r を求める
　もし $r = 0$ ならば
　　b を出力する
　さもなくば
　　$euclid(b, r)$ を出力する

図 1.3　エウクレイデスの互除法．正整数 a, b の最大公約数 $euclid(a, b)$ を出力する．

で最終行で改めて $euclid(36, 8)$ が実行され，これにより更に $euclid(8, 4)$ が実行される結果，4 を出力して終了する．このように入力を取り換えながら次々に手順 $euclid$ が呼び出されるので，何時までも計算が終らない心配もありそうだが，よく考えると実はその虞はない（つまり初めの二数が何であっても必ず有限回の繰返しの後に第 2 行で求めた余り r が 0 となる）．何故なら，i 回目の実行で $euclid(a_i, b_i)$ を求めようとした結果 $i + 1$ 回目の実行 $euclid(a_{i+1}, b_{i+1})$ が行われたとすると，この b_{i+1} は b_i による除算の余りとして生じたものなので，$b_{i+1} < b_i$ が成立つ．従って $b_1 > b_2 > b_3 > \cdots$ となり，これは正整数の下降列なので無限には続かない．

では具体的に，$euclid(a, b)$ を実行したとき手順 $euclid$ が（開始時を含め）何遍呼び出されるだろうか．これは図 1.3 第 2 行の除算が行われる回数でもある．上の議論，つまり b_1, b_2, b_3, …が必ず毎回 1 以上減少する数列であることからは，この回数が b 以下であると判るが，より精密に考えるともっと少い回数で済んでいることが次のように判る．

4+2**問 1.4**　(1) 上記のように $euclid$ の i 回目の実行における入力を (a_i, b_i) とするとき，（もし $i + 2$ 回以上の実行がなされたならば）$b_i > 2b_{i+2}$ が成立つことを示せ．
(2) $euclid(a, b)$ の実行中になされる除算の回数が $1 + 2\log_2 b$ 以下であることを示せ．

除算の回数が $\mathrm{O}(\log b)$ であることが判った．これは整数 b を十進法で表したときの長さ $1 + \lfloor \log_{10} b \rfloor$ と同程度である．入力である整数 a と b を十進法で書いたものの長さを n とすると，除算の回数が $\mathrm{O}(n)$ であるとも言える．冒頭に述べた全 d を調べ尽す方法では $2b$ 回の除算を行ったのに比べると，随分少い回数で済んだ．

1.3　例 3：独立集合

無向グラフ (V, E)（→ A.3 節）の頂点の集合 $S \subseteq V$ が**独立**(independent)であるとは，S のどの二点も隣接しないことをいう．例えば図 A.1.2 のグラフにおいて，$\{\mathrm{B}, \mathrm{C}\}$ や

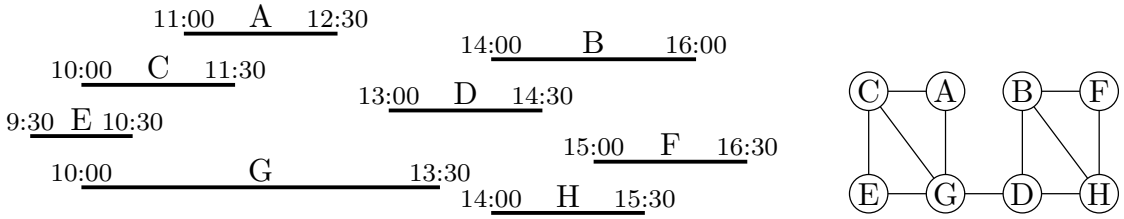


図 1.4　左図の利用申込の重なり方は，右図のグラフで表される．

$\{\mathrm{A}, \mathrm{D}, \mathrm{F}\}$ や $\{\mathrm{B}\}$ はみな独立集合である．当然，大きな集合ほど独立になり難い傾向がある．そこで，与えられたグラフの独立集合のうち頂点の個数が最大であるものを求めるという問題を考えよう．これを最大独立集合問題と呼ぶことにする．

これに当てはまる現実の状況は例えば次のものである．或る公共施設があり，利用したい団体は開始・終了時刻を指定して申込む．希望者は多いが一度に利用できるのは一団体だから，施設では時間が重ならないよう申込の一部を選んで受入れ，残りは断らざるを得ない．さて，今この施設には来季の分の申込書 n 枚が届いており，このうちなるべく多数を受入れたい．これは，申込書を n 個の頂点とし，利用希望時間帯が重なる二頂点の間に辺を設けたグラフにおいて，最大の独立集合を求めたい，と言い換えられる．例えば図 1.4 のように 8 団体からの申込があるとき，A, D, E, F の 4 団体を受入れるのが最良である．

最大独立集合問題に対する一応正しい算法としては，2^n 個ある集合 $S \subseteq V$ を逐一調べて独立か否か確かめ，独立なもののうち頂点数が最も大きいものを選ぶ，というものがある．だが，n が増えると 2^n はすぐ厖大な数になり，この方法は実行し難い．

だが実は，上述のように施設の利用申込から作られたグラフに対しては，次の問 1.5 に見る通り効率的な算法がある．

3+4**問 1.5**　頂点を一つ見る毎に即座に受入を決めてしまう方針だとどうなるか考えよう．つまり初めは S を空集合としておき，V の頂点を何らかの順で一つずつ見ながら，その頂点が既に S にある点に隣接していないならば S に加えてゆくのである．このように「後々のことを深く考えずに各時点でどんどん選択を行う」ような算法を総称して貪慾法 (greedy algorithm) と呼ぶことがある．

こうして得られるのは常に独立集合ではあるが，頂点を調べる順番が悪いと，一般に最大ではない．例えば図 1.4 のグラフには大きさ 4 の独立集合があったが，初めに G を選んでしまうと，後は B, F, H しか選べなくなり，大きさ 2 の集合しか得られない．

しかし実は，まず V の頂点 (申込書) を或る順に並べ替えておき，その順に見てゆくことにすれば，上述の貪慾法によって常に最大の独立集合が得られる．

(1) その順とは次のいずれか一つである．記号で選び，他の三つでは最大の独立集合が

得られない場合があることを示せ.

㋑ 開始時刻の早い順

㋺ 終了時刻の早い順

㋩ 利用時刻の短い順

㋥「自分の利用希望時間帯に属する時刻を開始時刻として希望している他の団体」の個数が少い順

(2) (1)で選んだ通りにすると正しく最大の独立集合 (の一つ) が得られることを示せ.

この方法に要する手間の量は——測り方にもよるので厳密には 3 章の計算時間の定義などを用いねばならないが——初めの並べ替えに問 1.1 (2)の方法を使うと，$O(n \log n)$ 程度である.

ここでは,「すべて調べ尽す」には 2^n という指数的な手間が掛るのを，良い方法を考え出すことで多項式程度に抑えた (問 1.5 の $O(n \log n)$ というのは無論 $O(n^2)$ 以下だから多項式的と言ってよい). このように愚直なやり方では指数的な手間を要する所を何とかして $O(n)$, $O(n^2)$, $O(n^3)$, …などの多項式的な手間で済ませたいという状況はよくある[*2]. 計算時間が多項式的であるか否かは，計算の効率についての最も重要な尺度であり，本演習でも大きく取扱う. 問 1.5 の算法はちょっとした思いつきという程度だが，もっと巧妙で高度な手法によりやっと多項式時間で解けるという場合も勿論ある.

さて，問 1.5 の算法が通用するのはあくまで施設の利用申込を図 1.4 のように変換することで作られたグラフに対してのみであることに注意しよう (現に問 1.5 (1)では，各頂点に対応する開始・終了時刻という情報を使って並べ替えを行った). このように時刻区間の重なり関係から得られるグラフを区間グラフ(interval graph)と呼ぶ.

2+4 **問 1.6** (1) 図 A.1.2 のグラフは区間グラフか.

(2) 長さ 4 の閉路 (→図 A.3.4) は区間グラフか.

一般のグラフの独立集合問題を多項式時間で解く算法は知られていない. 実は後に 5.2.2 節で見る理由により，恐らくそのような算法は無いと強く信ぜられている. つまり問 1.5 は，一般のグラフに対しては困難と判っている問題について，しかし区間グラフという特殊な状況では容易に解ける，と示したのである[*3]. 逆に言えば，現実の状況において本当は区間グラフを扱いたいだけなのに，一般のグラフに対して解ける方法を探すのは徒労である. 似た問題のうちどこまでが容易で，どこから困難であるかという境目を見極

[*2] この文のように「指数的」という言葉を，多項式的の対義語のごとく用いることがあるが，これはやや不正確な言い方である. $n^{\log n}$ のように，多項式的ではないが指数的でもない函数もある.

[*3] なお，区間グラフが与えられれば，その区間表現 (すなわちそのグラフに合うような各頂点の開始・終了時刻) を多項式時間で見つける算法があることは (自明ではないが) 知られている.

めることは，実用上大いに役立つ場合がある．

1.4　例 4：偶奇判定

ここまでは計算量を主に「計算にかかる時間」のことと考え，それを（大まかにでも）測るために演算の回数を数えた．ここでは少し変った計算量の尺度として，回路の大きさというものを考える．

回路（論理回路）の定義は附録 A.4 にある通りである．入力数 n，出力数 1 の回路は何らかの函数 $f\colon \{0,1\}^n \to \{0,1\}$ を実現する．一つの f を実現する回路にも色々あるが，そのうちなるべく小さなものを作りたい．但し回路の**大きさ**とは $\wedge$ か $\vee$ の置かれた頂点の個数をいう（ここでは $\neg$ は数に入れないことにする）．例えば図 A.5 の回路の大きさは 7 である．

2**問 1.7**　函数 $\oplus_n\colon \{0,1\}^n \to \{0,1\}$ は入力 n ビット中に 1 が奇数個ならば 1，偶数個ならば 0 を返すものとする．$\oplus_n$ を実現する大きさ $3n-3$ の回路を作れ．

15**問 1.8**　問 1.7 の函数 $\oplus_n$ を計算する大きさ $3n-3$ 未満の回路が存在しないことを示せ．

ヒント　「n に関する帰納法」「入力を一つ固定すると $\oplus_{n-1}$」「固定により不要になった素子を消す」

注意　これは，大きさ $3n-3$ 未満の如何なる回路も決して $\oplus_n$ を計算できないという主張である．問 1.7 では何らかの作り方に従って回路を構成したであろうけれども，そこで用いた特定の手法で作れないことを示しただけでは問 1.8 の証明にはならない．

1.5　計算量の上界と下界

計算量を測るということについて幾つかの例を見た．ここで議論を振り返ってみよう．

■上界と下界　同じ目的（配列を整列する，行列の積を求める，函数 $\oplus_n$ を計算する）を達するにも様々な方法がある．問 1.1, 1.3, 1.7 では，特定の方法を用いたときの計算量（演算の回数，回路の大きさ）を調べた．つまり目的とする問題を解くために要する手間について，「○○以内で済ます方法がある」という一つの**上界**を得た．

一方，問 1.2 や問 1.8 は，如何なる方法を用いても「必ず○○以上の手間がかかる」という**下界**を示している．どの算法を以てしてもこの壁を破れないというのだから，問題そのものに内在する本質的な難しさ，複雑さを表すものといえよう．これにより問 1.1, 1.7 の方法が最良であることがわかった．

しかしこのように上界と一致する下界が証明できることは稀である．下界というものは，あらゆる計算方法を漏れなく考え尽した上で，手間○○以内では絶対に不可能と言い

切らねばならぬわけで，中々そこまでは示せないことが多い．問 1.8 のごとき凡々たる下界を得るのさえ一苦労であったことにも，その難しさの一斑が窺われよう．そこで計算量理論では困難さを主張するにしても，絶対的な下界を直接示すのは諦め，後の章で見るように問題どうしの困難さを互に比べる相対的な形で述べることもある．

■問題と入力　これまでの議論のもう一つの特徴は，与えられた入力（配列，行列，真理値など）に応じて何かをなす問題を考え，計算の手間もその入力の大きさ n に応じた量として測ったということである．日常語では「$26+38$ は？」のような一つ限りの問いかけのことも「問題」と呼ぶが，計算量理論で問題といえば，考えられる入力すべてに対し答を定めたもの，例えば「与えられた二整数の和を求める」という課題全体を指す．個別の「$26+38$ は？」のごときはその問題に対する「入力」ないし「個例」と呼ぶ．一つの算法は，すべての入力を正しく処理できて初めて，その問題を解いたとされる．そもそも算法（計算手順）というものを作る目的は，それさえ用意しておけば如何なる入力が来ても自動で——つまり，人が入力を見て新たに考えを巡らすのではなく，計算機が勝手に——対処できるようにしておくことだからである．

無限個の入力すべてに正しく処するという要求を，有限に記述された算法によって満そうというのだから，おのずから限界がある．つまり，単純な問題ならば簡単な仕組で素早く解けるが，複雑であれば解くことが不可能であったり，可能であっても時間がかかったりするであろう．このように計算の困難さによって測られる問題の複雑さを理解するのが計算理論の目標である．

■計算量の尺度　本章では手間の尺度として「比較の回数」「回路の大きさ」などを扱ったが，より一般的で重要な尺度はやはり計算にかかる「時間」や「空間（記憶領域）」の量であろう．勿論上述の尺度も計算時間に近いものではあるが，第 3 章以降では改めて一般的な形で時間・空間を扱う．

第 2 章

問題と機械

本章では「問題」とそれを解く「算法」(機械) とを厳密に定義する．1.5 節で述べたように計算量理論の目標の一つは何らかの問題を解くのに「如何なる手順を使っても」○○以上の手間を要すると主張することであるから，計算手順といえるものすべてを漏れなく扱いたい．しかし計算機の機構の細部は製品によってまちまちであるし，算法に使われる考え方にも様々なものがあるから，そのすべてに及ぶ性質について考察するには，計算の本質をなす仕組をうまく取り出して調べる必要がある．つまり抽象化された単純な計算機械を定義し，その機械によって如何なる処理ができるかを論ずるのである．

この演習で扱う計算はすべて文字列に対する操作である (文字列に関する用語・記法については附録 A.2 節を見よ)．整数，グラフ，論理式などの意味をもつ対象を扱うときも，すべて文字列に符号化して表した上で処理すると考える．また目標とする課題は，入力された文字列に対して何かを答えるという，一度限りの受け答えである．何度も入出力を繰返したり，複数の計算機がやりとりをして一つの目的を達したりする計算も時には有用かもしれないが，本演習では扱わない．

与えられた文字列に対して何をしたら正解なのか定めたものが**問題**(problem)である．例えば十進法で整数を表す文字列 x が与えられたときに，それが素数であるか答えよ，というのは一つの問題である (図 2.1)．1.5 節で注意したように，個々の「1517」のごとき文字列は「問題」ではなく，**個例**(instance)或いは単に**入力**(input)と呼ぶ．問題を解いたというには，すべての個例に対して正しく答えなければならない．

以下 2.1，2.2 節ではまず，このように入力が或る条件を満すか答えることを目的とする問題を考える．この場合，問題とはすなわち言語 (附録 A.2) のことであると考えてよい．例えば今挙げた素数の問題は，与えられた文字列が $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ 上の言語

$$\text{PRIMES} = \{\, x \in \Sigma^* : x \text{ は或る素数を十進法で書いたものである} \,\}$$

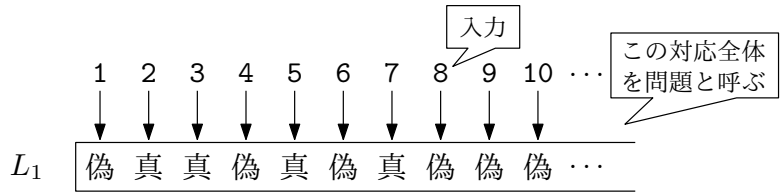


図 2.1　問題とは各個例（入力）に対する正答を定めたものである．例えば与えられた正整数が素数か否か答えよという問題は，入力が素数（を表す文字列）ならば真，さもなくば偽という答を定めたもの全体であり，言語 PRIMES によって表される．

に属するかどうか知りたい，という問題である[*1]．なお，このような真偽の判断でなく，適切な文字列を**出力**(output)することを正解として期待する問題の扱いについては 2.3.1 節で述べる．

まず 2.1 節では有限状態機械と呼ばれる単純な計算機構を考える．2.2 節で，機能を拡張したチューリング機械を考える．有限状態機械の計算能力は小さいが，チューリング機械はあらゆる算法と同等の能力をもつ．3 章以降で扱うのは，チューリング機械に時間や空間の制限を課した場合の計算能力である．

2.1　有限状態機械

まず単純な計算機構として，**有限状態機械**と呼ばれるものを考える．

2.1.1　定義と例

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ 上の文字列が与えられ，その中に連続する 3 文字として「aba」が現れるか否か知りたいとしよう．例えば与えられた文字列が abbaabab であるときは条件を満すと判断（**受理**）し，abbaabba であるときは満さないと判断（不受理）したい．つまり，言語

$$\text{CONTAINS-ABA} = \left\{ x \in \Sigma^* : x \text{ は } \mathtt{aba} \text{ という連続する 3 文字を含む} \right\}$$

に属するか否かを正しく区別したいのである．

このためには，文字列を先頭から読み進め，現時点で「aba」が何文字目まで出たところか常に注視していればよい．その手順は図 2.2 で表される．図に描かれた四つの場所 q_0，q_1，q_2，q_3 を**状態**という．どの状態からも a と書かれた矢印と b と書かれた矢印が出ているので（q_3 から出る「a, b」は，二本の矢印の行先が同じなので纏めて描いたものである），「始」と書かれた状態 q_0 から始め，与えられた文字列を読み進めながら，各時点

[*1] このように本稿では問題の名称にしばしば小型大文字を使う．

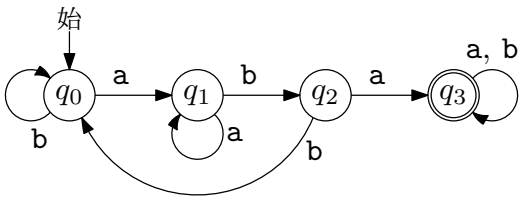


図 2.2　言語 CONTAINS-ABA を認識する有限状態機械.

の文字に従って矢印を辿ることにする．例えば文字列 abbaabab を読むと状態が順に q_0, q_1, q_2, q_0, q_1, q_1, q_2, q_3 となり，終に二重丸で示された状態 q_3 に至る．図をよく観察すると，この q_3 に至るか否かを以て，文字列が aba を含むか否かが判定できるようになっていることが解るだろう．

図 2.2 のような仕掛を有限状態機械という．きちんと定義すると，

定義 1　有限状態機械(finite state machine)は次のものにより指定される．

- 有限個の**状態**(state)の集合 Q．但し**始状態**と呼ばれる状態 $q_{\text{始}} \in Q$ と，**受理状態**と呼ばれる状態の集合 $Q_{\text{受理}} \subseteq Q$ が定まっている．
- 字母 (有限個の文字の集合，附録 A.2 節参照) Σ.
- **遷移規則**(transition function)と呼ばれる函数 $\delta\colon Q \times \Sigma \to Q$.

この機械に文字列 $x \in \Sigma^*$ を入力したときの動作を以下で定める．初め機械は始状態 $q_{\text{始}}$ にあり，x の左端の文字から始め，次のこと (**遷移**) を繰り返す.

> 現在の状態が q であり，読んだ文字が σ であるとき，状態を $\delta(q, \sigma)$ に変え，次の文字に進む.

右端まで終ったときの状態が $Q_{\text{受理}}$ に属すれば，機械は文字列 x を**受理**(accept)したという.

これで個々の入力 x を受理するか否かが定まるが，先に述べたように，機械の目標はすべての入力 (個例) を正しく処することである．すなわち，

> 機械 M が言語 A を**認識**(recognize)するとは，任意の個例 x に対して,
> - もし $x \in A$ ならば，M は x を受理し，かつ
> - もし $x \notin A$ ならば，M は x を受理しない
>
> ことをいう.

例えば図 2.2 の機械を定義 1 に従って表すと次のようになる.

- 文字集合 $\Sigma = \{\mathtt{a}, \mathtt{b}\}$
- 状態集合 $Q = \{q_0, q_1, q_2, q_3\}$ (始状態 $q_{\text{始}} = q_0$，受理状態集合 $Q_{\text{受理}} = \{q_3\}$)

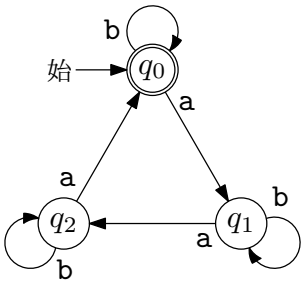


図 2.3　与えられた文字列中の a の個数が 3 の倍数か判定する有限状態機械.

- 遷移規則 δ は次で定める

$$\begin{array}{llll} \delta(q_0,\mathtt{a})=q_1 & \delta(q_1,\mathtt{a})=q_1 & \delta(q_2,\mathtt{a})=q_3 & \delta(q_3,\mathtt{a})=q_3 \\ \delta(q_0,\mathtt{b})=q_0 & \delta(q_1,\mathtt{b})=q_2 & \delta(q_2,\mathtt{b})=q_0 & \delta(q_3,\mathtt{b})=q_3 \end{array}$$

そしてこの機械は上の枠内の意味で言語 CONTAINS-ABA を認識している.

有限状態機械で認識できる言語の例をもう一つ挙げよう．字母 $\Sigma=\{\mathtt{a},\mathtt{b}\}$ 上の言語

$$\{\, x\in\Sigma^* : x \text{ に現れる } \mathtt{a} \text{ の個数は } 3 \text{ の倍数である} \,\}$$

を考える．この言語を認識するには今までに a を幾つ見たか数えながら進めばよいが，状態は有限個だから a の個数そのものは覚え切れない．そこで 3 で割った余りを覚えることにしよう．この方針で作った機械を図 2.3 に示す．状態 q_0，q_1，q_2 はそれぞれ，これまでの a の個数を 3 で割ると 0, 1, 2 余ることを表す.

1+1+1+1+2 **問 2.1**　$\Sigma=\{\mathtt{a},\mathtt{b}\}$ 上の次の言語をそれぞれ認識する有限状態機械を作れ.

(1) $\{\, x\in\Sigma^* : x$ に文字 a が 3 回以上現れる $\}$

(2) $\{\, x\in\Sigma^* : x$ に文字 a が現れる回数はちょうど 3 回である $\}$

(3) $\{\, x\in\Sigma^* : x$ に文字 a が連続して 3 回以上現れる $\}$

(4) $\{\, x\in\Sigma^* : x$ の長さは 3 以上であり，左端から 3 文字目は a である $\}$

(5) $\{\, x\in\Sigma^* : x$ の長さは 3 以上であり，右端から 3 文字目は a である $\}$

2.1.2　有限状態機械の限界

有限状態機械は眼前の文字に応じて状態を変えるだけの単純な計算機構であった．文字列のうち既に読んだ部分については，それを読んだ結果どの状態に至ったかのみを記憶に留め，経緯は忘れてしまう．例えば図 2.2 の機械の四つの状態は，現時点でそれぞれ図 2.4 の条件が成立つことを表している．如何に長い文字列を読まされても，この 4 通りに

q_0	まだ aba は現れておらず，現時点で最後が a でも ab でもない
q_1	まだ aba は現れていないが，現時点で最後の文字は a である
q_2	まだ aba は現れていないが，現時点で最後の二文字は ab である
q_3	既に aba が現れた

図 2.4　図 2.2 の有限状態機械の各状態の意味

分類してさえおけば以後の判断に困らないという，言語 CONTAINS-ABA の単純さを利用していたといえる．

そのような性質をもたない言語もあり，有限状態機械では認識できない．例えば幾つかの連続した a の後に同数の b が続くという形をした文字列の全体，すなわち

$$\text{ANBN} = \{\, x \in \Sigma^* : \text{或る } n \in \mathbb{N} \text{ が存在して } x = \mathtt{a}^n\mathtt{b}^n \,\}$$

がそれである[*2]．これを認識するには a の個数を完全に数えねばならず，それは幾らでも大きくなりうるので，状態が有限個では無理である．

このことをきちんと証明してみよう．ANBN を認識する有限状態機械 M が存在したとし，文字列 a，aa，aaa，…をそれぞれ与えたとき至る状態を考える．状態は有限個なので，相異なる m，n が存在し，$\mathtt{a}^m$ と $\mathtt{a}^n$ を読んだ後の状態が一致する．この状態から $\mathtt{b}^n$ を読んで至るのが

- 受理状態であれば，M は $\mathtt{a}^m\mathtt{b}^n$ を受理し，$\mathtt{a}^m\mathtt{b}^n \notin$ ANBN に反する．
- 受理状態でなければ，M は $\mathtt{a}^n\mathtt{b}^n$ を受理せず，$\mathtt{a}^n\mathtt{b}^n \in$ ANBN に反する．

いずれにせよ矛盾するので，ANBN を解く有限状態機械は存在しない．

問 2.2　同様に考えて，$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ 上の言語

$$\{\, x \in \Sigma^* : x \text{ に現れる } \mathtt{a} \text{ の個数は } \mathtt{b} \text{ よりも多い} \,\}$$

を認識する有限状態機械が存在しないことを証明せよ．

2.1.3　計算モデルの頑健性

計算機構の能力を強めれば，より多くの問題が解けるようになる可能性がある．本節では計算の性質を考えるという目標を掲げ，いきなり有限状態機械というものを定義して例

[*2] この Σ は $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ と考えてよい．以後一々断らずに Σ は文脈に応じてその場の議論に適当な文字を含んでいるものと仮定することがある．

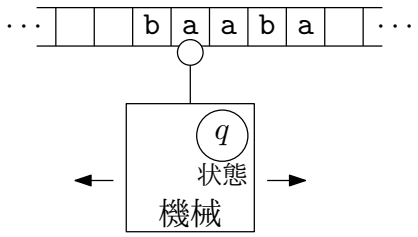


図 2.5　機械とテープ．現在は状態が q，読んでいる文字が a なので，次の動作は遷移規則の結果 $\delta(q, \mathtt{a})$ に従う．

を見てきたが，本当にこの定義で良かったのか，定義の細部を変えるとどうなるか，一度ここで反省してみるべきかもしれない．

例えば，定義 1 では機械は与えられた文字列を左端から読み始めてひたすら右へ進むのみとしたが，行ったり来たりすることを許したらどうなるであろうか．つまり図 2.5 のごとく枡目状に仕切られたテープに入力文字列が書いてあり，機械はテープ上の一区劃を見ているが，その位置を右ばかりでなく左にも動かせるとしてはどうか．形式的にいうと次のように定義を変えることになる．

定義 2　両方向有限状態機械は次のものにより指定される．

- 有限個の状態の集合 Q．ただし始状態と呼ばれる状態 $q_{\text{始}} \in Q$ と，受理状態と呼ばれる状態の集合 $Q_{\text{受理}} \subseteq Q$ が定まっている．
- 字母 Σ．これに**空白文字** ␣ を加えた集合を Γ と書くことにする（$\Gamma = \Sigma \cup \{\text{␣}\}$）．
- 遷移規則と呼ばれる函数 $\delta\colon Q \times \Gamma \to (Q \times \{ \text{㊧}, \text{㊨} \}) \cup \{ \textbf{止} \}$.

Σ 上の文字列 x をこの機械に入力したときの動作を以下で定める．初め機械は始状態 $q_{\text{始}}$ にあり，左右に無限なテープ上の一区劃を見ている．この位置から右に向って入力文字列 x が，連続する区劃に一字ずつ書かれている．他の各区劃には ␣(空白) が書かれていると考える．機械は次の遷移を繰返す．

現在の状態が q であり，読んだ文字が σ であるとき，もし $\delta(q, \sigma)$ が

- **止** ならば，機械は**停止**(halt)する[*3].
- $(q', d) \in Q \times \{ \text{㊧}, \text{㊨} \}$ という形ならば，機械は状態を q' に変え，d の向きに一区劃だけ進む．

機械が停止し，そのときの状態が $Q_{\text{受理}}$ に属すれば，機械は文字列 x を**受理**したという．

定義 1 に比べると，遷移規則 δ に $\{ \text{㊧}, \text{㊨} \}$ が現れ，左にも進めるようになった．定義 1 では文字列の右端に達したら終了すると決まっていたが，今度は空白 ␣ を読んだときの

[*3] この「停止」という言葉は，機械が壊れて動かなくなるということではなく，正常に計算を終了するという感じで使われている．

遷移も定めてあり，終了（停止）するのは遷移規則で **止** が出て来たときとした（停止せず永遠に動き続けてしまう場合は，受理しなかったと扱う）．定義 1 は，定義 2 の機械のうち，空白 ␣ を読んだら常に停止し，そうでないときは常に右に動くような遷移規則をもつ特殊な場合に当たる．

さて，この「左に移動する」という機能を機械に加えてみたけれども，実は次に示すように，認識できる言語は増えない（証明は問 2.4）．

> 言語 A を認識する両方向有限状態機械（定義 2）が存在するならば，A を認識する（両方向でない）有限状態機械（定義 1）が存在する．

このように定義を多少変えても概念そのものが変りにくいことを，その概念が**頑健**（robust）であると言うことがある．

2+2+2 **問 2.3** このように言語を認識する能力が変らないと証明するには，任意の言語 A に対し，一方の定義の下で A を認識する機械があるならば，他方の定義の下でも A を認識する機械が作れることを示せばよい．そのような証明を簡単な場合について考えてみよう．

(1) 定義 2 ではテープ上の位置を各時刻に左か右へ動かすものとしていた．その場に留まることを許しても変らないことを示せ．

(2) 定義 2 では機械は入力された文字列の先頭（左端）から動作を始めるとしていた．末尾（右端）から始めるとしても変らないことを示せ．

(3) 定義 2 では受理状態の集合 $Q_{\text{受理}}$ を指定できるものとしていた．$Q_{\text{受理}} = Q$ に限っても（つまり，機械が停止しさえすればどの状態でも受理したと扱うことにしても）変らないことを示せ．

なお，次に見るチューリング機械（定義 3）もこれらの変更に対して頑健である．

4+3+3 **問 2.4** 上で述べた「両方向有限状態機械」と元々の有限状態機械との等価性を示そう．両方向有限状態機械 M が言語 A を認識するとする．ただし前問(2)を使い，M は入力文字列の右端から動作を開始するものとしておく．定義 2 のように M の状態集合を Q，文字集合（空白以外）を Σ とする．Σ 上の文字列 x と状態 q とに対し，$[x](q)$ を次で定める．x をテープ上に置いてその右端の文字の位置で状態 q から M を動かし始めると，やがて x の右端の一つ右の区劃を

- 訪れる場合，その初めて訪れた時点での状態を $[x](q)$ とする．
- 訪れることなく受理する場合，$[x](q) = +$.
- 訪れることも受理することもない場合，$[x](q) = -$.

つまり $[x]$ は Q から $Q \cup \{+, -\}$ への函数である．

(1) Σ 上の文字列 x，x' がもし $[x] = [x']$ を満すならば，Σ の任意の文字 σ について，$[x\sigma] = [x'\sigma]$ が成り立つ．何故か．

(2) Σ 上の文字列 x，x' がもし $[x] = [x']$ を満すならば，M が x を受理するか否かと x' を受理するか否かは一致する．何故か．

(3) 定義 1 の有限状態機械であって A を認識するものを作れ．Q から $Q \cup \{+, -\}$ への函数一つ一つを状態とするとよい (そのような函数は有限通りしかない)．遷移規則や始状態，受理状態集合をどのように定めたらよいか．

言語 A を認識する有限状態機械が存在するとき，A は**正則**(regular)であるという．上で示したことより，「有限状態機械」に代えて「両方向有限状態機械」を使っても変らない概念である．言語が正則であることはさらに，**正則表現**(regular expression)と呼ばれる表し方で書けることや，或る形の論理式や代数系で記述できることなどとも等価であることが知られている．見かけの甚だしく異なる複数の定義から同じ概念が得られることは，その概念が自然で有用なものであることの傍証と言えるだろう．正則という性質はこのように，問題の単純さを捉える良い尺度の一つであり，実際上も言語処理系 (コンパイラ) の理論などにおいて重要な役割を果す．

とはいえ 2.1.2 節で見たように，随分たやすそうな問題の中にも正則でないものがある．現実の計算機の能力を説明するため，もっと強力な仕組を次に考える．

2.2　チューリング機械

2.1 節で扱った有限状態機械に，文字を書き込む能力を与えてみよう．1930 年代に**チューリング**(A. Turing)は，計算の限界を論ずるためにそのような機械を考案した[*4]．記憶が一定量しかないという限界 (2.1.2 節) を取り除き，凡そ算法として書ける処理を何でも行えるような仕組を考えようというのである．

2.2.1　定義と例

チューリング機械は，一定の遷移規則に従って (有限個の) 状態を刻々と変えながら動作する点では定義 2 の機械 (図 2.5) と同じだが，入力テープの他に幾本かの作業テープを有し (図 2.6)，計算中に文字を読むばかりでなく作業テープには書くことができる (入力テープは読取り専用とし，書き込まない)[*5]．遷移規則は，機械の状態と現在読んでいる (入力・作業テープ上の) 文字とから，次の状態，(作業テープに) 書き込む文字，(入

[*4] 但し以下で定義するチューリング機械は，前後の議論との繋がりの都合上チューリングの原論文での定義とは若干異なる．

[*5] このように新たにテープを設けるのではなく，入力と同じ一本のテープを書込にも使うとするチューリング機械の定義もあり，実は暫くはそれでも差支えないのだが，3.4 節以降で空間計算量を考える際には作業テープを分けておくと都合が良いためこのようにしておく．

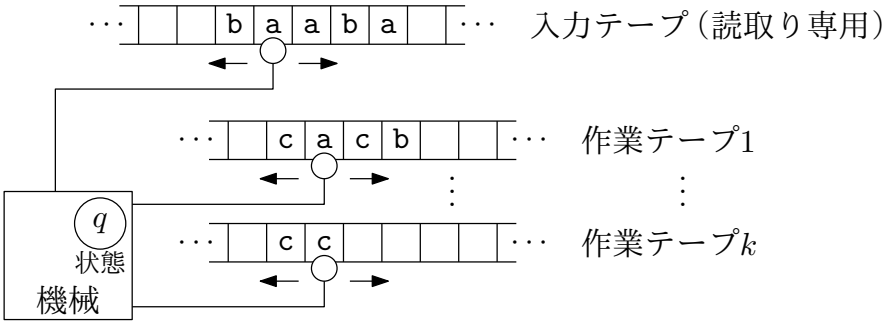


図 2.6　k 本の作業テープをもつチューリング機械.

力・作業テープ上で）動く向きを定める．形式的には，

定義 3　チューリング機械(Turing machine)は次のものにより指定される．

- 有限個の状態の集合 Q．ただし始状態と呼ばれる状態 $q_{\text{始}} \in Q$ と，受理状態と呼ばれる状態の集合 $Q_{\text{受理}} \subseteq Q$ が定まっている．
- 有限個の文字の集合 Σ.
- Σ の文字すべてと空白文字 ␣ を含む，有限個の文字の集合 Γ.
- 作業テープの本数 $k \in \mathbb{N}$.
- 遷移規則と呼ばれる函数 $\delta\colon Q \times \Gamma^{1+k} \to (Q \times \Gamma^k \times \{ ㊧, ㊨ \}^{1+k}) \cup \{ \textbf{止} \}$.

Σ 上の文字列 x をこの機械に入力したときの動作を以下で定める．初め機械は始状態 $q_{\text{始}}$ にあり，各テープ上の一区劃を見ている．入力テープ上のその位置から右に向って入力文字列 x が，連続する区劃に一字ずつ書かれている．他の各区劃や作業テープ上の各区劃には ␣（空白）が書かれていると考える．機械は次の遷移を繰り返す．

> 現在の状態が q であり，読んだ文字が入力テープ上で σ，各作業テープ上で $\sigma_1, \cdots, \sigma_k$ であるとき，もし $\delta(q, \sigma, \sigma_1, \ldots, \sigma_k)$ が
>
> - **止** ならば，機械は停止する．
> - $(q', \sigma'_1, \ldots, \sigma'_k, d, d_1, \ldots, d_k) \in Q \times \Gamma^k \times \{ ㊧, ㊨ \}^{1+k}$ という形ならば，機械は状態を q' に変え，各作業テープ上の現在の位置にある文字を $\sigma'_1, \cdots, \sigma'_k$ に書換えてから，入力テープと各作業テープ上でそれぞれ d, d_1, …, d_k の向きに一区劃だけ進む．

機械が停止し，そのときの状態が $Q_{\text{受理}}$ に属すれば，機械は文字列 x を**受理**したという．

定義 2 は定義 3 で $k = 0$ とした場合に当る．$k > 0$ のチューリング機械には作業テープ上に文字を書く機能があり，これにより有限状態機械では認識できなかった言語を認識できるようになる（「認識」の意味はやはり 2.1.1 節の枠内の通りである）．例えば 2.1.2 節の言語 $\text{ANBN} = \{ \mathtt{a}^n\mathtt{b}^n : n \in \mathbb{N} \}$ は有限状態機械には認識されなかったが，図 2.7 のチューリング機械によって認識される．

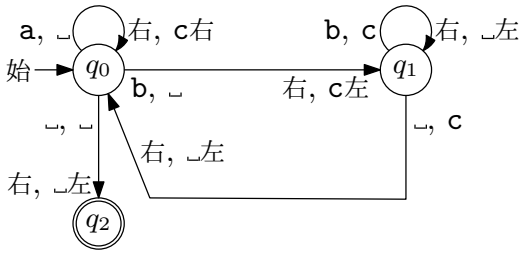


図 2.7　ANBN を認識するチューリング機械．状態 q から q' への矢の根元に文字 σ, σ_1 があり，矢の先に d, $\sigma'_1 d_1$ があるとき「状態 q で両テープ上にそれぞれ文字 σ, σ_1 を読むと，作業テープに文字 σ'_1 を書いて両テープをそれぞれ向き d, d_1 に動かし，次の状態は q' とする」を表す $(\delta(q,\sigma,\sigma_1) = (q',\sigma'_1,d,d_1))$．そのような矢がない (q,σ,σ_1) については $\delta(q,\sigma,\sigma_1) =$ **止** である．

この機械は作業テープを一本もつが，入力テープ上では常に右へ右へと読み進める．読んだ文字が a ならば作業テープ上に c を書いて右に行く．初めて b を読むと，作業テープ上では左へ戻る．以後は b を見る毎に作業テープの c を一つ消して左へ進む．入力を読み終った次の時刻にちょうど作業テープ上の c を消し終った場合のみ受理する．

他にも多くの言語をチューリング機械で認識することができる．例えば素数（を表す文字列）の全体 PRIMES を認識する機械は次の方針で作ることができる．与えられた正整数 X が素数か知りたければ，$Y = 2$，3，4，…について順に，X が Y の倍数か調べればよい．そのためには，テープ上の特定の場所に整数 Y を置くことにしてその数を「2 から順に増やしてゆく」手順や，その各 Y について「X が Y で割り切れるか筆算のような方法で確かめる」手順を機械で実現することになる．後者の筆算の中では減算を使うから，それも機械の遷移規則として書く．

2.2.2　チャーチ・チューリングのテーゼ

チューリング機械を定義したのは，算法というものを正確に定式化し，その能力を調べるためであった．しかも有限状態機械のような限定に縛られず，いかなる算法をも実行し得る仕組を作りたかったのである．このようにあらゆる算法を漏れなく考え尽すというのは中々の難事業であるように思われる．しかし驚くべきことに，以下に述べる理由から，チューリング機械はこの目的に適っていると考えられている．

まず，2.1.3 節で有限状態機械について述べたと同様に，言語がチューリング機械で認識できるという性質もまた，定義の細部によらない頑健な概念である．現実にある計算機の仕組や，人が紙を用いて計算する様子を真似ると，チューリング機械に例えば次の機能を加えることも考えられるが，そのような変更を加えても言語を認識する能力には影響がないのである．

- 各時刻にテープ上を左か右に一歩だけ動くとする代りに，指定した歩数だけ飛ぶ機

能を加える

- テープの代わりに平面的に広がった計算用紙をもつ
- 一定の桁数の整数どうしを一度の遷移で加算・減算できるようにする

このことを証明するには，これらの変種を明確に定義した上で，そのような変種の機械を与えられたときに元の機能しか使わずに同じ問題を解く機械に作り直す方法を述べればよい．それは面倒なのでここではやらないが（簡単なものは問 2.3 で扱った），多くの変種について示されているのである．さらに，チューリング機械による計算可能性は，チューリングと同時代に**クリーニ**(S. Kleene)や**チャーチ**(A. Church)が考案した**再帰函数**(recursive function)や**ラムダ計算**(lambda calculus)など，数や式に対する機械的な処理を捉えた他の概念とも等価であることが知られている．

そればかりか，チューリング機械の能力を超える現実的な計算の仕組は，今日に至るまで見つかっていない．考え得るすべての算法が悉く定義 3 のチューリング機械で表されるとは俄かに信じ難いかもしれないが，よくよく考えるとこの定義は単純でありながら情報処理の本質を捉えているのである．チューリングの論文でも，規則に従って計算をする者が取り得るあらゆる行動がこの機械の動きとして表せそうだという理由が説明されている．およそ明確に記述された処理手順は皆チューリング機械で書ける，というこの主張は**チャーチ・チューリングのテーゼ**(Church–Turing Thesis)と呼ばれている．勿論「明確に記述された手順」が厳密な概念でない以上，この主張は数学的に証明できる類のものではないが，広く受入れられている．

このことから，チューリング機械の定義そのものの詳細は，今後の議論にあまり重要でない．今後は「機械」と「算法」を全く同じ意味で使う．図 2.7 のごとく定義に忠実に従った機械を書いても読み難いだけなので，これからは計算手順として解り易いように算法を述べることにする．また，「言語 A を認識するチューリング機械が存在する」を単に「言語 A は認識可能である」という．今後定義する，問題を機械が○○するという形の各概念についても同様に，それを行うチューリング機械が存在することを○○可能という．

■チューリング機械と現実の計算　しかし本章ではここまで，言語を認識する（定義は 2.1.1 節の枠内）という形の問題のみを扱ってきた．現実に解くべき問題はもっと多様ではないか，と読者は考えるかもしれない．例えば 1.3 節では「与えられたグラフの最大独立集合を一つ求める」という「問題」が出て来たが，これは言語を認識するという形の要求ではない．

これを本章の意味での問題として扱うには，まず入力を文字列とせねばならない．それには何らかの取り決め，例えば附録 A.3 節の末尾の方法に従って，グラフを文字列に符号化することにすればよい．素数の問題 PRIMES も，正整数を十進法という符号化法により文字列で表したのであった．このように考えると，問題とはグラフや整数の表記法を固定

して初めて決まるのであり，入力を十進法で表したときの素数判定と，二進法で表したときの素数判定は，厳密には異なる問題である．したがって表記法を明示せずに素数判定の問題などと呼ぶのは本来まずいのだが，通常それはあまり気にするに及ばない．素数か否かの判断という本質部分の難しさに比べれば，二進法と十進法の間の変換は容易だからである．無論このことは何を「容易」と見做すかによるが，二進と十進の変換くらいは今後考える如何なる尺度においても容易であるから気にしないのである．そこで以降では，細部まで表記法を指定する煩を避け，本質に支障のない限り多少の曖昧さは気にせず日本語で問題を記述することにする[*6]．例えば「整数 n を十進法で表した文字列」を単に n と呼んだり，「適当な符号化法を用い，容易に復号できるよう複数の文字列 $p_1, \cdots, p_k$ を書き並べたもの」を単に $(p_1, \ldots, p_k)$ と書いたりする．

最大独立集合問題が言語の認識とは異なるもう一つの点は，真偽を問うのではなく，何かを探し出して答えよという形をしている所である．これまでの枠組で扱えるのは例えば言語

$$\textsc{is} = \{\, (G, l) : \text{グラフ } G \text{ に大きさ } l \text{ 以上の独立集合が存在する} \,\}$$

の認識であり，元の目標とは少し異なる．しかし何かを発見することと存在を判定することは密接に関わっており，IS のような言語の認識の難しさは，様々な形の問題の困難さをそれなりに代表している（このことは問 3.7，3.8 や 6 章でも触れる）．とはいえ文字列を出力する問題を扱いたい場合はあるので，そのやり方についても 2.3.1 節で述べる．

2.2.3　認識不能な言語

このようにチューリング機械は強力であるが，認識不能な言語も存在する．ここではその例を一つ作ろう．

まず，チューリング機械は定義 3 にある要素から定まるものであり，文字列で表せることを注意しておく．ここでは入力字母 Σ が $\Sigma = \{0, 1\}$ であるような機械のみを考えることとし，そのような機械を文字列で表す方法を決めておく[*7]．更にこの記述中の各文字を適当に符号化して $\{0, 1\}$ 上の文字列で表す．機械 M を表す $\{0, 1\}$ 上の文字列をやはり

[*6] ただ二進法ではなく羅列表示（整数 n を文字列 0^n で表す）にしてしまうと著しく冗長になるため，後に計算量を論ずる際には影響が出る．そこで特に断らないとき整数は十進法や二進法で表すなど，計算対象は自然で簡潔な表し方で符号化するという注意は必要である．

[*7] 定義 3 では文字や状態の名に好きな文字を用いてよいことにしていたが，機械が何らかの機能を実現する上では単にどの状態でどの文字を読むとどの状態になるかという関係のみが重要であり，文字や状態の名は何でもよいので，ここでは一定の文字集合のみを使って書いた機械を考える．すなわち 0 と 1 以外の文字は $\gamma 0$，$\gamma 1$，$\gamma 10$，$\gamma 11$，$\gamma 100$，…，また状態は $q0$，$q1$，$q10$，$q11$，$q100$，…と二進法の番号で名づけるなどと決めておく．

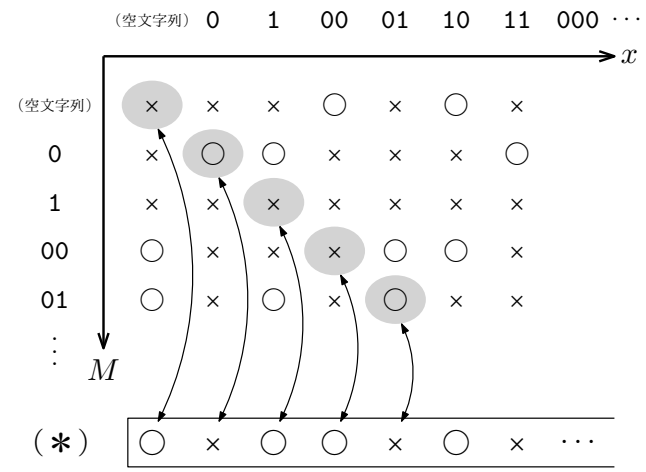


図 2.8　$(D, x) \in$ EVAL か否かを記した表（機械 D が入力 x を受理するとき○，しないとき×）．対角線上の成分を反転して定めた下段の（＊）に一致する行は存在しない．

M と呼ぶことにしよう．ここで言語

$$\text{EVAL} = \{\, (M, x) \in \{0,1\}^* \times \{0,1\}^* : \text{機械 } M \text{ は文字列 } x \text{ を受理する} \,\}$$

を考える[*8]（文字列 M がそもそも機械の記述になっていないときは便宜上 $(M, x) \notin$ EVAL としておく）．後に 3.3.1 節で見るように EVAL は認識可能である．

しかしその補言語（附録 A.2 節）$\overline{\text{EVAL}}$ は認識可能でない．このことを証明しよう．

$\overline{\text{EVAL}}$ を認識する機械 N が存在したとする．認識の定義（2.1.1 節の枠内）から，N は入力 (M, x) に対して次の挙動を示す．

$(M, x) \in$ EVAL ならば受理せず，$(M, x) \notin$ EVAL ならば受理する．

この N に変更を加えて次のような機械 D を作る．D の入力文字集合は $\{0, 1\}$ である．D は文字列 $x \in \{0,1\}^*$ を受け取ると，それを複写して文字列 (x, x) を作り，これを入力とする N の動きをする．この変更は N に作業テープを一本加え，状態や規則を少し書換えることで実現できる．こうして作った機械 D は，もとの機械 N との違いを考えると，入力 x に対して次の挙動を示す．

$(x, x) \in$ EVAL ならば受理せず，$(x, x) \notin$ EVAL ならば受理する．　　（＊）

これが任意の x で成り立つのだから，特に $x = D$ のときにも成り立つはずである．しかし，もし $(D, D) \in$ EVAL ならば前半が成り立たないし，さもなくば後半が成り立たない．これは矛盾である．

これで背理法により $\overline{\text{EVAL}}$ の認識不能性が示されたが，聊か技巧的で解り難かったかもしれない．この議論は図 2.8 のように理解できる．図の上段は縦軸にすべての文字列 D

[*8] ここで (M, x) とは，2.2.2 節で述べたように，適当な区切り文字，例えば # を用いて，M と x を並べたものである．つまり正確にいえば EVAL は $\{0, 1, \#\}$ 上の言語である．

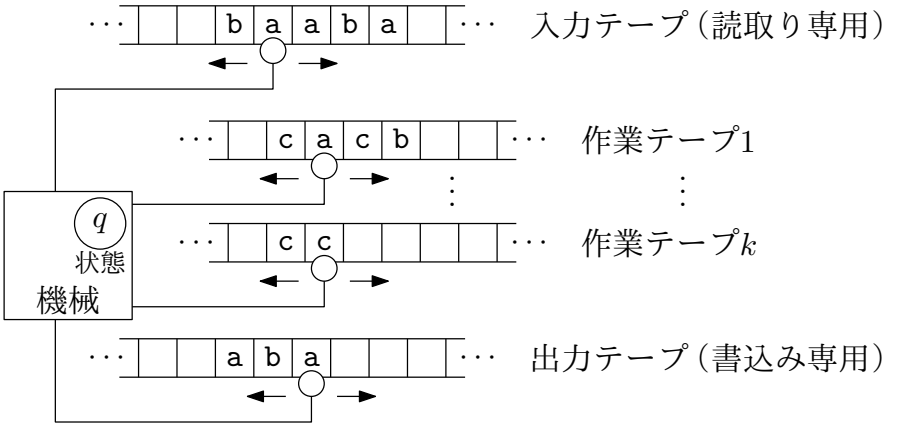


図 2.9　出力テープをもつチューリング機械.

を，横軸にもすべての文字列 x を並べ，各成分に $(D, x) \in \text{EVAL}$ の成否を記した表である．表の対角線上の成分の○×を逆にして下段のごとく書き並べると，これは表中のどの行とも完全には一致しないことになる．ところが証明中の条件 $(*)$ が述べているのは，まさに各入力 x に対してこの下段の挙動を示すということであった．故にそのような機械は存在しない．この証明法は，表の対角線上の成分に着目したことから，**対角線論法**と呼ばれる．

2.3　入出力と問題の形

ここまでは「言語」を「認識」するという形の問題のみを扱ったが，ここでは少し異なる形の問題をどのように扱うかを述べる．

2.3.1　（多価）函数の計算

各入力文字列に対して適切な文字列を出力せよという形の問題は，形式的には**多価函数** (multi-valued function) によって表される．(Σ^* 上の) 多価函数 A とは，各 $x \in \Sigma^*$ に対し，空でない集合 $A[x] \subseteq \Sigma^*$ を定めるものである．入力が $x \in \Sigma^*$ であるときに許される出力の全体を $A[x]$ と書くのである．例えば最大独立集合問題 (→ 1.3 節) では，文字列 `6,1,2,3,2,3,5,4,3,4,6,5,6,6,3` (図 A.2.1 のグラフを表す) が与えられたら，文字列 `1,4,5` または文字列 `2,4,5` を答えるのが正解である．各個例 x について $A[x]$ がただ一つの元より成る (正解が一つである) ときは，要するに A は Σ^* から Σ^* への単なる (一価の) **函数** (function) である．このときはその $A[x]$ の唯一の元を $A(x)$ で表す．

このように文字列を答えることを要求する問題を扱うには，定義 3 の機械に出力テープを加えた機械を考える (図 2.9)．遷移規則は出力テープに書く文字と動きも指定するた

め，函数 $\delta\colon Q \times \Gamma^{1+k} \to (Q \times \Gamma^{k+1} \times \{ ㊧, ㊨ \}^{1+k+1}) \cup \{ \textbf{止} \}$ により与える．機械が停止し，そのとき出力テープ上で機械が見ている区劃を左端とする連続する区劃に文字列 $y\llcorner$（但し $y \in \Sigma^*$）があるとき，機械は文字列 y を**出力**(output)したという．

> 機械 M が多価函数 A を**計算**(compute)するとは，任意の個例 x に対して
> - M に x を入力すると，$A[x]$ に属する文字列を出力する
>
> ことをいう．

2.3.2　言語の判定

機械が言語 A を認識することの定義（2.1.1 節の枠内）は，A に属する文字列は受理し，属しない文字列は受理しないというものであったが，この「受理しない」では停止しないことも許されていた．A に属する文字列と属しない文字列の扱いが対称でないといえる．これに対し，入力 x が A に属するときも属しないときも停止することを要求したのが次の定義である．

> 機械 M が言語 A を**判定**(decide)するとは，任意の個例 x に対して，
> - もし $x \in A$ ならば，M は x を受理（して停止）し，かつ
> - もし $x \notin A$ ならば，M は x を受理せずに停止する
>
> ことをいう（「判定する」を「**決定**する」ともいう）．

問 2.5　言語 $A \subseteq \{0,1\}^*$ が判定可能であるには，次の函数 $\chi_A\colon \{0,1\}^* \to \{0,1\}^*$ が計算可能であることが必要十分であることを示せ．

$$\chi_A(x) = \begin{cases} 1 & x \in A \text{ のとき} \\ 0 & x \notin A \text{ のとき} \end{cases}$$

注意　2.2.2 節で述べた通り，「…が○○可能」は「…を○○する機械が存在する」を意味する．この定義に即して機械から機械を適切に作る方法を与えることによって証明を述べよ．

問 2.6　言語 $A \subseteq \{0,1\}^*$ について，

(1) もし A が判定可能ならば，A も $\overline{A}$ も認識可能であることを示せ．

(2) もし A と $\overline{A}$ が認識可能ならば，A が判定可能であることを示せ．

注意　問 2.5 の注意と同じ．

2.2.3 節で見たように $\overline{\text{EVAL}}$ は認識不能なので，問 2.6 (1) より EVAL は判定不能である．

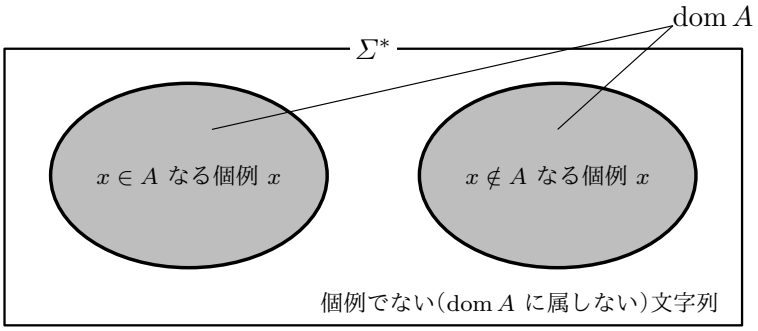


図 2.10　約束つき言語 A．文字列のうち約束 $\mathrm{dom}\, A$ に属するもののみが A の個例であり，個例でない文字列 $x \in \Sigma^* \setminus \mathrm{dom}\, A$ については $x \in A$ であるか $x \notin A$ であるかが定まっていない．

2.3.3　約束つき問題

この節は初めて読む際には飛ばしても差支えない．

これまでの問題では個例（入力）として任意の文字列 $x \in \Sigma^*$ が与えられるとしたが，実際には一部の文字列のみを個例として想定している（つまり入力が Σ^* の或る部分集合に属することが予め「約束」されている）場合がある．そのような問題を**約束つき問題**(promise problem)という．正確にいうと約束つき言語 A とは，

- 入力として与えられ得る文字列の集合 $\mathrm{dom}\, A \subseteq \Sigma^*$ が定まっており，
- 各 $x \in \mathrm{dom}\, A$ に対し，$x \in A$ であるか否かが定まっている

ものである[*9]（図 2.10）．集合 $\mathrm{dom}\, A$ を**約束**(promise)といい，その元のみをこの問題の**個例**と呼ぶ．個例 $x \in \mathrm{dom}\, A$ に対してのみ，$x \in A$ か $x \notin A$ かを正しく判断することが要求される．同様に約束つき多価函数 A は，

- 約束（入力として与えられ得る文字列の集合）$\mathrm{dom}\, A \subseteq \Sigma^*$ が定まっており，
- 各 $x \in \mathrm{dom}\, A$ に対し，入力が x のときに許される出力の候補として，空でない集合 $A[x] \subseteq \Sigma^*$ が定まっている

ものである．これまでの言語や多価函数は，約束つき問題のうち約束が Σ^* であるものに当る．

例えば先に挙げた素数判定問題 PRIMES も，日本語での説明に「与えられた正の整数に対し」とあるからには，$\{0,1\}^*$ 上のすべての文字列を受付けるよりも，十進法で正の整数を表す文字列全体を約束 $\mathrm{dom}\,\mathrm{PRIMES}$ とする約束つき言語と考えるのが自然かもしれない．入力が $\mathrm{dom}\,\mathrm{PRIMES}$ に属しないとき，すなわち正整数の十進表示でないとき，先程は受理してはならないとしていたが，今度はそのようなときには受理してもしなくても良いというのである．この意味では約束つき言語の方が要求が緩く，見かけの上では認識し易くなっている．しかし実際には与えられた文字列が正整数の十進表示であるか否か調べるのはどうせ容易だから，困難さに殆ど違いはない．このように約束にわざわざ注意を払うに及ばない場合も多いので，本稿では簡単のため問題といえば約束なし問題を指すとしておく．

ただ，約束つき問題についても話が一応つながるように諸定義は述べる．具体的には

[*9] この箇条書き第二項の「$x \in A$」は通常の集合への帰属と同じ記号であり，$x \in A$ が成立たないことを「$x \notin A$」で表すが，A が約束つき問題であるときは，これらの記法「$x \in A$」「$x \notin A$」を $\mathrm{dom}\, A$ の元 x に対してしか使わないことにする．

- これまで及び 4.2.1，4.2.2，4.2.3 節などで定義される「問題（言語や多価函数）A を○○する」という形の言葉は，A の各個例 x について正しく処理をすることを以て定義されるので，約束つき問題 A に対してはその「各個例 x について」を「各 $x \in \mathrm{dom}\, A$ については」と解釈すればよい．
- 5.1 節では他の問題が「問題 B に帰着する」ことを定義するが，これは B を解く算法があればそれを利用できるという意味なので，B が約束つきであるときには，脚註*1 のように，その利用を B の約束に属する個例に限ることにすればよい．

このようにすれば今後の議論の殆どは約束つき問題についても成立つ．そうでない場合には特に「（約束なし）問題では…」と括弧に入れて断ることにする．約束つき問題を使った方が説明し易い事柄については上述の語法で述べる（問 4.12，5.7，5.16，7.3，7.6，7.10）が，それまでは約束を取り立てて意識せずに読み進めても差支えない．

第 3 章

時間と空間

本章では機械に時間や空間の制約を課したときにどのような計算ができるかを考える．分けても重要な制約は，計算時間を入力長の多項式以内に制限するというものである．他にも時間や空間の制約を幾つか扱い，その制約の下で解ける問題や，制約どうしの間の関係を見る．

3.1 時間・空間の制限

機械 M は字母 Σ 上の文字列を入力として受付けるとする．M に $x \in \Sigma^*$ を入力するとやがて停止し，それまでの遷移の回数が $t \in \mathbb{N}$ 以下であるとき，M は x において t **時間限定**であるという．また、M に $x \in \Sigma^*$ を入力すると停止し，それまでにどの作業テープ上でも機械が $t \in \mathbb{N}$ 個以下の欄しか訪れていないとき，M は x において t **空間限定**であるという．停止しないときは t 時間限定ないし t 空間限定でないとすることを注意しておく．作業テープ上で一区劃進むにも遷移一回を要するので，t 時間限定ならば t 空間限定である．また出力テープをもつ t 時間限定の機械 M において，入力 x に対する出力の長さは同様の理由から t を超えない．

第 1 章で触れたように，入力 x の大きさ（文字列としての長さ）$|x|$ に応じて計算時間・空間がどのように増えるかに注目したい[*1]．函数 $\tau\colon \mathbb{N} \to \mathbb{N}$ に対し，M が τ **時間限定**であるとは，任意の文字列 $x \in \Sigma^*$ において M が $\tau(|x|)$ 時間限定であることをいう．M が τ **空間限定**であることも，同様に任意の文字列 x において $\tau(|x|)$ 空間限定であることと定義する．なお函数 $\tau\colon \mathbb{N} \to \mathbb{N}$ の代りに式 $\tau(n)$ を使って $\tau(n)$ 時間限定などと書くことも多い．例えば「この機械は τ 時間限定である．ここで τ は $\tau(n) = 3n^2$ で定義され

[*1] 厳密に言うと 1 章で触れた尺度（入力された数列の長さ，入力されたグラフの頂点数など）は，その入力（数列，グラフなど）を文字列として書いたときの長さとは少し異なる．しかし本稿では後に見るように計算量を大雑把に測りたいだけなので，この程度の細かい違いはあまり気にしないことにする．

る」などと言うのは煩わしいので，単に「この機械は $3n^2$ 時間限定である」と（何の説明もなく n という記号が出て来るのはおかしいのだが，入力長をそう表すことを暗黙の了解として）記してしまうのである．更に O 記法を用いて「この機械は $\mathrm{O}(n^2)$ 時間限定である」のようにも書く．すなわち函数 $t\colon \mathbb{N} \to \mathbb{N}$ に対し，M が $\mathrm{O}(t(n))$ 時間限定であるとは，$\tau(n) = \mathrm{O}(t(n))$ なる $\tau\colon \mathbb{N} \to \mathbb{N}$ が存在して M が $\tau(n)$ 時間限定であることをいう．空間限定についても同様である．3.2 節と 3.4 節ではそれぞれ，このように時間と空間を限定した計算について考える．

2.2.2 節で述べたように，チューリング機械は現実の計算可能性によく対応している（チャーチ・チューリングのテーゼ）．このことは時間・空間を考えに入れても，かなりの程度まで成立つ．つまりチューリング機械の消費時間・空間は，それが表す算法を現実の計算機で実行したときに使われる時間や記憶領域の量に或る程度近い．

例えば 1.1 節（整列，行列乗算）では計算量を整数に対する比較や乗算の回数で測った．この尺度は厳密にはチューリング機械での消費時間とは異なる．一回の整数演算を実現するにもチューリング機械では二進表示した整数に一ビットづつ操作を加えることになるから時間がかかる上，数の演算そのもの以外にも例えば配列中の適当な位置まで注目点を移動させるなどの手間が要るからである．とはいえ整数が一定値以内（例えば 32 ビット）であれば一演算にかかる時間も何らかの定数で抑えられるし，単なる移動にかかる時間の占める割合は計算全体から見れば大きくない場合が多い．

整数演算の回数で数えた計算量もそれなりに計算の手間を捉えた尺度ではあるのだが，様々な問題について計算の手間を統一的に調べるためには，「整数の比較」「加算と乗算」といった特定の演算を前提としない定義をしたい．そのために様々な算法を実装できる汎用の機械として，便宜上チューリング機械を用いるのである．以下では特に断らない限り，チューリング機械と通常の計算機を同一視し，消費時間・空間も概ねその計算機上のものと考えて差支えない．

3.2　時間の制限

本節では時間限定を課せられた機械で解かれる問題について考える．

「問題（言語・多価函数）A を○○する△△時間限定のチューリング機械が存在する」を「A は△△時間○○可能である」という．$\mathrm{O}(t(n))$ 時間認識可能な言語の全体を $\mathsf{Time}(t(n))$ で表し，$\mathrm{O}(t(n))$ 時間計算可能な多価函数（2.3.1 節）の全体を $\mathsf{FTime}(t(n))$ で表す（この n は 3.1 節の時間限定の定義の直後で注意したように入力長を表す記号である）[*2]．

[*2] この FTime や直後の FP が F で始まっているのは function（函数）の頭文字である．文献によっては，これを多価函数ではなく（一価）函数の集合とするものや，言語の級 P のみを主に扱うもの，その記法 P を函数や多価函数にも流用するものなどがある．

$\mathsf{Time}(t(n))$ や $\mathsf{FTime}(t(n))$ は，機械を通して定義されてはいるが，あくまで問題（言語や多価函数）からなる集合であることに注意しよう．まず機械 M に文字列 x を入力したときの消費時間は x ごとに異なるが，M が τ 時間限定であるにはすべての文字列 x について計算が消費時間 $\tau(|x|)$ 以内で停止する必要があるのだった．言語 $A \subseteq \Sigma^*$ を認識する機械 M にも色々あるが，A が $\mathsf{Time}(t(n))$ に属するにはそのいずれかが $\mathrm{O}(t(n))$ 時間限定であればよいのであった．つまり $A \in \mathsf{Time}(t(n))$ とは

或る機械 M と $\tau(n) = \mathrm{O}(t(n))$ なる函数 $\tau\colon \mathbb{N} \to \mathbb{N}$ とが存在し，

如何なる入力 $x \in \Sigma^*$ を M に与えたときも，

- 受理するか否かは $x \in A$ か否かに一致し，かつ
- 消費時間は $\tau(|x|)$ である

という入り組んだ意味をもつ主張である．

3.2.1 多項式時間

1.3 節で述べたように入力長の多項式以内という時間制約は特に重要である．次の問 3.1 の㋑㋺のいずれか（故に両方）が成立つとき機械 M は多項式時間限定であるという．

問 3.1 機械 M について，次の㋑と㋺が同値であることを示せ．

㋑ 非負整数を係数とする或る多項式 $p\colon \mathbb{N} \to \mathbb{N}$ が存在し，M は p 時間限定である．

㋺ 或る $k \in \mathbb{N}$ が存在して，M は $\mathrm{O}(n^k)$ 時間限定である．

多項式時間認識可能な言語の全体を P で，多項式時間計算可能な多価函数の全体を FP で，それぞれ表す．問 3.1 ㋺により，

$$\mathsf{P} = \bigcup_{k\in\mathbb{N}} \mathsf{Time}(n^k) \qquad\qquad \mathsf{FP} = \bigcup_{k\in\mathbb{N}} \mathsf{FTime}(n^k)$$

である．この $\mathsf{Time}(t)$，$\mathsf{FTime}(t)$ や P，FP のように計算資源（ここでは時間）の制限によって定まる問題（や約束つき問題[*3]）の集合を**計算量級**(complexity class)という．なお個々の $\mathsf{Time}(n^3)$ などは機械の細かな定義に多少依存する（例えば脚註*5 で述べたテープが一本か否かで変る）が，それを纏めた級 P や FP は自然で安定した概念である．

問題が多項式時間で解ける（P や FP に属する）ことは「現実的な時間で解ける」ことに経験上かなり近いと考えられている．勿論実際上の効率の良さは計算機の速さや算法の用途によるし，多項式といえども $\mathrm{O}(n^{1000})$ 時間限定の算法は速いとはいわれまいが，それでも多項式であるか否かは重要である．指数時間を要しそうだった問題に多項式時間限定の算法が見つかったとなれば，それは何らかの工夫が功を奏して劇的に効率が改善したわ

[*3] ここでは P などの級を約束なし問題の集合としたが，同じ記号を約束つき問題（2.3.3 節）を含めた級に流用することもある．

けであり，更に改良を加えれば高速な算法が見つかることが多い．$\mathrm{O}(n^{1000})$ 時間限定の算法だけが知られているなどという状況は，実際には滅多にないのである．

FP は二つの (一価) 函数の合成について閉じている．すなわち FP に属する任意の函数 A, $B\colon \Sigma^* \to \Sigma^*$ に対し，合成 $A \circ B$ も FP に属する．このことを確かめよう．仮定より A, B を計算する多項式時間限定の機械 M, N が存在する．多項式 p, $q\colon \mathbb{N} \to \mathbb{N}$ が存在して，M は p 時間限定，N は q 時間限定である．次のように動作する新たな機械 K を考える．K は M や N よりも多くの作業テープを有している．K は入力 x を受取ると先ず N (が x を入力されたとき) と同じ動きをする．但し N が出力テープに書込むときには代りに，N に無い余分な作業テープに出力を書込む．この作業テープを中間結果テープと呼ぼう．N が停止すると，中間結果テープには $B(x)$ が書込まれている．その後 K は M と同じ動きをする．但し M が入力テープを読むときには，代りに中間結果テープから読取る．このようにすれば出力テープに $A(B(x))$ が書込まれる．このときの消費時間は，N に入力 x を与えたときの消費時間と，M に入力 $A(x)$ を与えたときの消費時間の和，すなわち $q(|x|) + p(|B(x)|)$ 程度である．ここで N の出力長は消費時間を超えない (3.1 節冒頭) ことから $|B(x)| \leq q(|x|)$ であるので，K の消費時間は $q(|x|) + p(q(|x|))$ という $|x|$ の多項式で抑えられる．

4**問 3.2**　$\mathsf{FTime}(n^3)$ は二つの函数の合成について閉じているか．証明とともに述べよ．

6**問 3.3**　函数 $A\colon \Sigma^* \to \Sigma^*$ に対し函数 A^{iter} を

$$A^{\mathrm{iter}}(u, 0^n) = \underbrace{A(A(\cdots(A}_{n\ \text{回}}(u))\cdots)) \qquad u \in \Sigma^*,\ n \in \mathbb{N}$$

で定める[*4]．FP は iter について閉じているか．すなわち，A が FP に属すれば必ず A^{iter} もまた FP に属すると言えるか．

3.2.2　多項式時間で解かれる問題の例

以下では幾つか多項式時間で解かれる問題の例を考えよう．

二つの正整数 x, y (を十進法で表す文字列を，適当な区切り文字を挟んで並べたもの) が与えられたとき，その和 $x + y$ (を十進法で表す文字列) を求める函数 ADD-TWO を考える．この函数は，まず x を一本目の作業テープに，y を二本目の作業テープに書き写した上で，筆算をするかのごとく両者を同時に下位桁から順に読みながら，繰上りの有無

[*4] 2.3.3 節で述べたように，A^{iter} は入力として $(u, 0^n)$ という形をしたもののみを想定する約束つき函数と考えてもよいし，そうでない入力に対しては決った値，例えば 0 を返す約束なし函数と考えてもよい．入力がこの形であるか判定するのは容易だからである．以後この類のことは一々断らない．

を状態に保持しつつ $x+y$ を下位桁から求めて出力テープに書き出すという機械で計算できる．入力長 n は概ね x の長さと y の長さの和に等しく，この機械は $\mathrm{O}(n)$ 時間限定である．故に ADD-TWO は $\mathsf{FTime}(n)$ に属し，従って FP にも属する．

では与えられた幾つかの正整数 $x_1, \cdots, x_k$ の和 $x_1 + \cdots + x_k$ を求める函数 ADD-MANY は如何だろうか．先程は入力に正整数が丁度二個（区切り文字が一個）だけ書かれていると決っていたが，今度は何個であっても和を正しく計算せよというのである．この函数 ADD-MANY は，先程 ADD-TWO のために行った手順を繰返すことで計算できる．つまりまず x_1 と x_2 をそれぞれ一本目，二本目の作業テープに写してから同様に和 $x_1 + x_2$ を計算し（但しその結果を出力テープでなく三本目の作業テープに書き），それと x_3 とをそれぞれ一本目，二本目の作業テープに写してやはり同じ手順により和 $(x_1 + x_2) + x_3$ を求める，というように最後まで繰返せばよい．この繰返しは $k-1$ 回行われるが，この回数 $k-1$ は多くとも入力長 n 以内であるから，結局この機械は $\mathrm{O}(n^2)$ 時間限定であると言える（より短い時間で解く機械も作れるが）．故に ADD-MANY も FP に属する．

二つの正整数 x, y の積 xy を求める函数 MULT-TWO も多項式時間計算可能である．これも，加算よりは複雑になるが筆算を真似た方法を機械で行えばよい．筆算では紙を縦横の二次元に使うところを，一方向のみに広がる作業テープで代用するための工夫を要するが，うまくやれば $\mathrm{O}(n^2)$ 時間限定のチューリング機械として実現できる（より短い時間で解く方法があることも知られている）．故に MULT-TWO も FP に属する．

与えられた幾つかの正整数 $x_1, \cdots, x_k$ の積を求める函数 MULT-MANY も，MULT-TWO のための手順を繰返すことにより多項式時間計算可能である．

[6]**問 3.4** 以上では整数を十進法で表したが, 代りに羅列表示（正整数 n を $\mathsf{0}^n$ で表す）を使ってみよう．ADD-TWO′, ADD-MANY′, MULT-TWO′, MULT-MANY′ をそれぞれ, ADD-TWO, ADD-MANY，MULT-TWO，MULT-MANY と同様の内容だが，入力・出力ともに整数を羅列表示で表すとした問題とする．例えば $2+4+1=7$ なので ADD-MANY′$(\mathsf{00\#0000\#0}) = \mathsf{0000000}$ である．この四つの問題はそれぞれ FP に属するか．

他にも多くの「効率よく解ける」問題は，チューリング機械により厳密に定義した意味でも，FP に属する（多項式時間計算可能である）．例えば与えられた正整数 x, y に対し，x を y で除した商と余を求める函数 DIV も，筆算を真似た方法により多項式時間計算可能であると示すことができる．与えられた正整数 x, y の最大公約数を求める函数は，この DIV と 1.2 節の互除法により，やはり多項式時間計算可能である．

出力数 1 の回路 C（附録 A.4 節）と，その入力数 k に合った割当 $(b_1, \ldots, b_k) \in \{\mathsf{0}, \mathsf{1}\}^k$ とが与えられたとき，$(b_1, \ldots, b_k)$ が C を充足するか問う問題 CEVAL を考える．入力される回路 C が附録 A.4 節末にあるように代入命令の列で書かれていることを考えれば，それに従って各変数の値を順に求めればよく，それは多項式時間で行うことができるか

ら，CEVAL $\in$ P が成立つ．

一方，逆に回路だけを与えられて，それを充足する割当を一つ答えよ（無いならば無いと答えよ）という問題 FIND-CSAT を考えると（この問題は多価函数として表される），そのように容易な方法は無さそうである．実は，この FIND-CSAT が FP に属するか否かは未解決であり，5.2 節で見るように，恐らくは属しないであろうと予想されている．

13+2**問 3.5**　しかし FIND-CSAT の回路の形を非常に単純なものに限った問題は FP に属することをこの問では示そう（これも 1.3 節の最大独立集合問題と同様に，与えられる個例が一定の性質を満すという約束が付くと問題が解き易くなる状況の一例といえる）．附録 A.4 で述べたように論理式は回路の一種であるが，論理式の中でも更に制限された形のものをここでは考える．

命題変数またはそれに $\neg$ を付けたものをリテラルという．リテラル二つを $\vee$ で結んだものを二選言節という．幾つかの二選言節を $\wedge$ で結んだものを 2CNF 式という．例えば

$$(\neg \mathtt{x}_3 \vee \mathtt{x}_2) \wedge (\mathtt{x}_1 \vee \mathtt{x}_3) \wedge (\neg \mathtt{x}_2 \vee \neg \mathtt{x}_4)$$

は 2CNF 式である．

(1) 入力数 k の 2CNF 式 φ に対し，次の有向グラフ（A.3 節）G を考える．

- 各リテラル $\mathtt{x}_1, \neg\mathtt{x}_1, \cdots, \mathtt{x}_k, \neg\mathtt{x}_k$ に対応して頂点は $2k$ 個ある．
- 頂点 L から L' への辺があるのは，$(\overline{L} \vee L')$ または $(L' \vee \overline{L})$ が φ の二選言節として現れるときとする．ここで $\overline{L}$ は L の $\neg$ の有無を入れ替えたものを表す．

このとき φ が充足割当をもたないためには，或る命題変数とその否定の両方を通る閉路が G にあることが必要十分であることを示せ．

(2) 与えられた 2CNF 式の充足割当を見つける（充足割当が無いときは無いと答える）問題が FP に属することを示せ．

3.2.3　判定問題について

言語 $A \subseteq \Sigma^*$ が認識可能であるとは，或る機械 M が存在し，任意の個例 $x \in \Sigma^*$ について次が成立つことであった．

- もし $x \in A$ ならば，M は x を受理する
- もし $x \notin A$ ならば，M は x を受理しない

この「受理しない」は，「受理状態以外で停止する」場合と，「停止しない」場合を含んでいた．後者を許すのは重要である．許さなければ判定可能（2.3.2 節）の定義になってしまうのであった．

一方，A が多項式時間認識可能であることの定義では，M を多項式時間限定とするから（そして「○○時間限定」の定義においては M がすべての入力に対して停止することを要求したから），結局 $x \in A$ でないときも「停止しない」の方は起らない．つまり「多項式時間認識可能」と「多項式時間判定可能」は同じ意味になる．この点では時間制限が付いた方が却って単純な状況になっているわけである．

4**問 3.6** 太郎君はこの説明がどうも腑に落ちない．寧ろ $x \in A$ でないときは M の停止を要求しない方が，認識可能という概念の多項式時間版として適切なのではないか．つまり，認識可能の定義（の「受理する」を，問 2.3 (3)に従って「停止する」に置き換えたもの）に現れる「停止する」のみに時間の制限を課して，次のようにしてはどうかと考えた．

> 言語 $A \subseteq \Sigma^*$ が「多項式時間認識可能」であるとは，或る多項式 p と或る機械 M が存在し，任意の個例 $x \in \Sigma^*$ について次が成立つことである．
>
> - もし $x \in A$ ならば，M に x を入力すると停止し消費時間が $p(|x|)$ 以下である
> - もし $x \notin A$ ならば，そうでない

しかし，言語 A が太郎君の意味で多項式時間認識可能であることは，よく考えると結局 $A \in \mathsf{P}$ と同じことである．これは何故か．

このことから以下では言語に属するか否か問う問題を（認識問題と呼んでも構わないのだが）判定問題と呼んだり，その内容を「……であるか判定する問題」のごとく説明したりする．例えば附録 A.2 節末の言語 PRIMES を説明するとき，「与えられた正整数が素数かどうか判定する問題 PRIMES」のような言い方で済ますことがある．

3.2.2 節で例示した問題は，CEVAL が言語（すなわち判定問題）であった他は函数（や多価函数）の形のものが多かったが，次節以降では簡単のため判定問題に限って話を進めることがある．これは以下の問 3.7，3.8 に見るように，問題が FP に属するか否か考えたい場合でも，判定問題の級 P を考察するだけでその目的が或る程度は果されるからである．

6**問 3.7** 函数 $A\colon \{0,1\}^* \to \{0,1\}^*$ に対し言語 BIT_A を次で定める．

$$\mathrm{BIT}_A = \big\{\, (x,i,b) \in \{0,1\}^* \times \mathbb{N} \times \{0,1\} : \\ |A(x)| > i \text{ であり } A(x) \text{ の第 } i \text{ ビットは } b \text{ である} \,\big\}$$

但し初めのビットを第 0 としている．

A が FP に属するには，BIT_A が P に属し，かつ或る多項式 p が存在して任意の $x \in \{0,1\}^*$ で $|A(x)| \leq p(|x|)$ が成立つことが，必要十分であることを示せ．

与えられた回路が充足割当をもつか問う問題を CSAT とする．先程の FIND-CSAT と違って，充足割当が存在する場合ただ存在すると言えばよく，実際にその割当の一つを答

えることは要求されていないという問題である．しかし実は，もし CSAT が多項式時間認識可能（P に属する）ならば FIND-CSAT も多項式時間計算可能である（FP に属する）．

問 3.8　このことを示せ．

ヒント　入力数 n の回路 C の $\mathtt{x}_n$ を $\mathtt{0}$, $\mathtt{1}$ に固定することで，入力数 $n-1$ の回路 C_0, C_1 がそれぞれ得られる．もし $C \in$ CSAT ならば C_0 か C_1 の少くとも一方は CSAT に属するし，この C_0 または C_1 の充足割当があればそれを使って C の充足割当が作れる．

3.3　計算の摸倣

本節ではチューリング機械（やそれと等価な計算機構）の重要な性質である**万能性**について見る．これは機械の従う手順を文字列として書き下しておき，それをやはりチューリング機械により解釈・実行できる——しかもそれが効率よくできる——という性質である．

ここでは簡単のため言語を認識する問題のみを考えることとし，出力テープを有せず，また字母 Γ が $\{\mathtt{0},\mathtt{1},\textvisiblespace\}$ であるような機械を考える．$\{\mathtt{0},\mathtt{1}\}^*$ 上の認識可能な言語は，このような機械によっても認識される．

3.3.1　計算の符号化と万能機械

これまで「与えられた文字列に aba が現れるか」「与えられた文字列（の表す整数）が素数か」などの特定の問題を解く機械について考えてきたが，読者はこれに違和感を覚えたかもしれない．実際のコンピュータは一台でさまざまな計算を行う．つまり，個々の問題を解くことは普通，その問題に専用の計算機を製造することによってではなく，問題に合わせた算譜（プログラム）を書く（それを汎用の計算機に読ませる）ことによって，実現されている．これを**算譜内蔵方式**という．

何故このようなことが可能なのであろうか．これは理論的には，任意のチューリング機械を**摸倣**(simulate)するチューリング機械が存在すること，すなわち言語の認識問題について言えば 2.2.3 節の言語

$$\text{EVAL} = \{\, (M,x) \in \{\mathtt{0},\mathtt{1}\}^* \times \{\mathtt{0},\mathtt{1}\}^* : \text{機械 } M \text{ は文字列 } x \text{ を受理する} \,\}$$

が認識可能であることによる．

このことは万能性と呼ばれるチューリング機械の著しい性質である．その意味を考えてみよう．機械 U が存在し，任意のチューリング機械 M と任意の文字列 x に対して

- M が x を受理するとき，U は (M,x) を受理する
- M が x を受理しないとき，U は (M,x) を受理しない

が成立つというのである．すなわち U は，M と x の両方が入力として与えられると，あたかも機械 M に文字列 x を入力したかのような結果になる機械である．これにより，さまざまな機械 M による処理が，M そのものをハードウェアとして用意せずとも，M を記述した文字列（算譜，プログラム）を U という一台の**万能**(universal)な機械に与えることによって実現できることになる．

EVAL が認識可能であることは以下のように考えると判る．機械 M に何らかの長さ $n \in \mathbb{N}$ の入力 $x \in \{\mathtt{0},\mathtt{1}\}^n$ を与えて動作させると，各時点の

- 機械の内部状態（Q の元）
- 入力テープ上の読取り位置
- k 本の各作業テープについて，注目点よりも左に書かれた文字列と，注目点から右に書かれた文字列

が定まり，爾後の動作はこれ（と x）のみから決する．これらの組を**時点表示**と呼ぶことにする．例えば図 2.6 に描かれている状況の時点表示は $(q, 2, \mathtt{c}, \mathtt{acb}, \ldots, \mathtt{c}, \mathtt{c})$ である．M の時点表示の全体を

$$ID_{M,n} = Q \times \{0, 1, \ldots, n, n+1\} \times (\Gamma^*)^{2k}$$

で表す．M への入力 $x \in \Sigma^n$ と時点表示 $d \in ID_{M,n}$ とが与えられると，次の時刻の時点表示 $next_M(x, d) \in ID_{M,n} \cup \{\, \textbf{受}, \textbf{拒} \,\}$ が定まる．但し **受** と **拒** はそれぞれ，既に受理状態で停止したこと，受理状態以外で停止したことを表す．便宜上 $next_M(x, \textbf{受}) = \textbf{受}$, $next_M(x, \textbf{拒}) = \textbf{拒}$ とすると，計算を始めて t 遷移後の時点表示 d_t は

$$\begin{aligned} d_0 &= (q_{\text{始}}, 1, \varepsilon, \varepsilon, \ldots, \varepsilon, \varepsilon) \\ d_{t+1} &= next_M(x, d_t) \end{aligned}$$

で定まり，消費時間 t 未満で受理するという条件は $d_t = \textbf{受}$ と書ける．順に d_0, d_1, d_2, $\cdots$ を計算することによりそのような t が存在するか（つまり M が入力 x に対して停止するか）調べることができる．このように M の規則を解釈しながら規則通り実行するという処理が機械的な手順として書ける（それを行うチューリング機械が作れる）のは，チャーチ・チューリングのテーゼを信ずるならば納得のゆくことであろう．これにより EVAL は認識可能である．

但し 2.2.3 節で見たように $\overline{\text{EVAL}}$ は認識不能であり，EVAL を一定の限られた時間で認識することはできない．M の計算を摸倣していて中々停止しない場合，このまま根気よく続ければやがて停止するのか，それともずっと停止しないのかを，一般に有限時間内に知る術は無いということである．

勿論，M の遷移を t 回行えと予め指定されれば，その回数分の摸倣をする（d_t を求める）ことはできる．従って次の言語 TIME-EVAL は判定可能である．

$$\text{TIME-EVAL} = \{\, (M, x, \mathtt{0}^t) \in \{\mathtt{0},\mathtt{1}\}^* \times \{\mathtt{0},\mathtt{1}\}^* \times \{\mathtt{0}\}^* : \text{機械 } M \text{ は文字列 } x \text{ を消費時間 } t \text{ 以内で受理する} \,\}$$

判定可能であるばかりでなく，上述の摸倣を効率的に行うよう注意すると，TIME-EVAL は P に（更には $\mathsf{Time}(n \log n)$ にも）属すると判る．つまり，U に (M, x) を入力したときの計算が，M に x を入力したときに比べて，あまり大きくない手間でできる．

3.3.2　機械と回路

以下では後の幾つかの議論の便宜のため，上でも述べた「機械の動きは単純な規則で書ける」ことをもう少し詳しく考察し，特に機械の動きを回路（→附録 A.4）で実現できることを見ておこう．具体的には，次の多価函数 MAKE-CIRC が FP に属する．

入力　機械 M と羅列表記された整数 n, t との組 $(M, 0^n, 0^t)$.

出力　n 入力 1 出力の回路 C であって，$x \in \{\mathtt{0},\mathtt{1}\}^n$ を入力すると，次が成立つとき 1，成立たないとき 0 を出力するもの．

M に x を入力すると消費時間 t 以内で受理する．

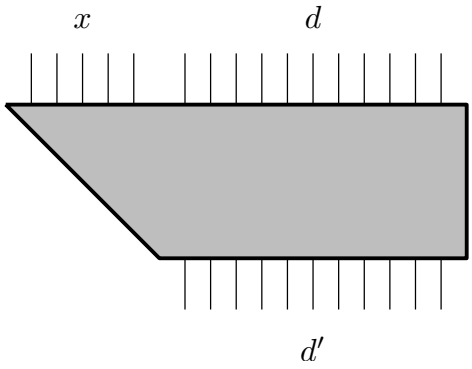
この C は入力 x に対して「機械 M と同じことを答える回路」である．しかし無限にあり得る入力を一台で引受けるチューリング機械と違い，回路は決った長さの入力を受付けることしかできないので，大きさ n（や t）毎に別々の回路を用意する形にしてある．

この MAKE-CIRC $\in \mathsf{FP}$ と，与えられた回路を多項式時間で実行できること（CEVAL $\in \mathsf{P}$, 3.2.2 節）とから，機械の動きを「回路を作ってそれを実行する」ことで実現できる．実際，$(M, x, \mathtt{0}^t) \in$ TIME-EVAL であるか判定するには，回路 $C \in$ MAKE-CIRC$[(M, \mathtt{0}^{|x|}, \mathtt{0}^t)]$ を作ってから $(C, x) \in$ CEVAL であるか判定すればよい．このことからも 3.3.1 節末の TIME-EVAL $\in \mathsf{P}$ が判る．

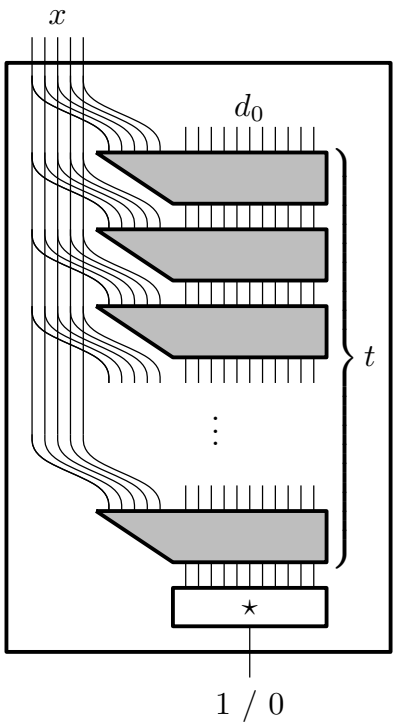
さて，MAKE-CIRC $\in \mathsf{FP}$ の証明の概略を述べよう．機械 M に入力 $x \in \{\mathtt{0},\mathtt{1}\}^n$ を与えたときの消費空間が $s \in \mathbb{N}$ 以内ならば，時点表示は上述の $ID_{M,n}$ の有限な部分集合

$$ID_{M,n,s} = Q \times \{0, 1, \ldots, n, n+1\} \times (\Gamma^s)^{2k}$$

に留まる．これは一定の長さ——具体的には $\mathrm{O}(\log|Q| + \log n + ks)$ 程度——のビット列で表される情報である．そこで，任意の $x \in \{\mathtt{0},\mathtt{1}\}^n$ と任意の時点表示 $d \in ID_{M,n,s}$ とを入力すると次の時点表示 $next_M(x, d)$ を出力するような回路を作ることができる（図 3.1.1）．この一回の遷移を表す回路を図 3.1.2 のごとく組合せると，指定した回数の遷移の繰返しを表す回路が得られる．但し図 3.1.2 中の $\star$ は **受** を入力されたときのみ 1 を出力する回路であり，n, s から容易に作ることができる．



3.1.1 文字列 x と時点表示 d とを与えると次の時点表示 d' を答える回路．与えられた組 $(M, 0^n, 0^s)$ に対し，機械 M の動きを表すこのような回路を作ることは容易である．数 n, s はこの回路の寸法，すなわち扱う入力 x と時点表示 d, d' の大きさを指定している．



3.1.2 図 3.1.1 の回路を t 個使って作った回路．与えられた組 $(M, \mathtt{0}^n, \mathtt{0}^t)$ からこの回路を作ること（MAKE-CIRC）は容易である．

図 3.1 機械の動きを摸倣する回路．

[3]**問 3.9** チューリング機械の遷移一回分を表す図 3.1.1 の回路の構成すべての確認は省くが，試しに少しだけ作ってみよう．この回路は例えば「入力テープ上の指定された位置に書かれた文字を読取る」必要があるが，それを単純化した次のような機能を実現することを考える．4 ビットからなる列 $\mathtt{d}_0\mathtt{d}_1\mathtt{d}_2\mathtt{d}_3 \in \{\mathtt{0},\mathtt{1}\}^4$ の第 i ビット $\mathtt{d}_i$ を取出したい．但し i は二進法で 2 ビット $\mathtt{p}_1\mathtt{p}_0 \in \{\mathtt{0},\mathtt{1}\}^2$ により指定されている．6 つのビット $\mathtt{d}_0$, $\mathtt{d}_1$, $\mathtt{d}_2$, $\mathtt{d}_3$, $\mathtt{p}_1$, $\mathtt{p}_0$ を入力するとビット $\mathtt{d}_i$ を出力する回路を描け．

[8]**問 3.10** 各 $n \in \mathbb{N}$ について入力数 n，出力数 1 の回路 C_n を定めたもの $(C_n)_{n\in\mathbb{N}}$ を回路族という．この回路族が $\{\mathtt{0},\mathtt{1}\}^*$ 上の言語 A を認識するとは，各 $n \in \mathbb{N}$ について，長さ n の任意の A の個例 x に対し，C_n に x を入力すると $x \in A$ か否かが出力されることをいう．回路族 $(C_n)_{n\in\mathbb{N}}$ が多項式時間構成可能であるとは，数 $n \in \mathbb{N}$ を羅列表記した文字列 0^n が与えられたときに C_n を求める問題が，FP に属することをいう．

言語が P に属するには，或る多項式時間構成可能な回路族によって認識されることが必要十分であることを示せ（MAKE-CIRC $\in$ FP や CEVAL $\in$ P を用いよ）．

3.3.3　対角線論法と階層定理

ここでは，級 P に属しないが，指数時間認識可能すなわち

$$\mathsf{EXP} = \bigcup_{k\in\mathbb{N}} \mathsf{Time}\big(2^{n^k}\big)$$

に属する言語を構成しよう．これまでにも FP に属しない函数は幾つか挙げたが，その理由はいずれも「答が長くて制限時間内に書けないから」という馬鹿馬鹿しいものであった．そこで出力がなくても多項式時間で解けない問題を作ろうというのである．

まずは時間制限を考えない 2.2.3 節での認識不能な言語の構成を想い出そう．縦軸にすべての機械 D を，横軸にすべての文字列を並べ，D に対応する行と x に対応する列の交る所に $(D,x) \in \textsc{eval}$ すなわち「D に文字列 x を入力すると受理するか」を書込んだ表を作る（図 2.8）．作りたい言語 H は，これらの機械 D のいずれにも認識されないもの，つまり表の任意の行 D に対して或る個例 x があり，$x \in H$ であるか否かが第 D 行 x 列の内容と異なるようなものである．このためには例えば H を次で定めればよい．

$$H = \{\, x : (x,x) \notin \textsc{eval} \,\}$$

するとどの機械 D も H を正しく認識しない（入力が D であるときに過つ）のであった．

さて，この議論に時間制限を加えて，$\mathsf{EXP} \setminus \mathsf{P}$ に属する問題 $H_{\exp}$ を構成しよう．同様な表を考える．但し今度は縦軸に並べる機械 D として多項式時間限定なもののみを考えればよい．$H_{\exp}$ をうまく定義していずれの行とも喰い違うようにしたいわけだが，今度は $H_{\exp} \in \mathsf{EXP}$ を成立たせたいという所が聊か難しい．というのも $H_{\exp}$ を先の H と同じ作り方にしてしまうと，限られた時間で $H_{\exp}$ を認識できない．各機械 D は多項式時間限定だが，その多項式は D ごとにまちまちなので，$H_{\exp}$ を計算する時間を一括りに抑える役には立たないのである．そこで D を「入力 D において」過たすという方針は諦め，代りに「入力 $(D,0)$, $(D,00)$, $(D,000)$, …のいずれかにおいて」過たせよう（或る入力で過てばよかったのだから，これで十分である）．$H_{\exp}$ を次で定める．

$$H_{\exp} = \{\, (x,0^k) : (x,(x,0^k),0^{2^k}) \notin \textsc{time-eval} \,\}$$

6+5**問 3.11**　（1）これで確かに $H_{\exp} \notin \mathsf{P}$ が成立つことを示せ．

（2）3.3.2 節の $\textsc{time-eval} \in \mathsf{P}$ を使って $H_{\exp} \in \mathsf{EXP}$ を示せ．

これで $\mathsf{P} \subsetneq \mathsf{EXP}$ が示されたが，実は更に議論を適切に精密化すると，$1 \le a < b$ なる任意の実数 a, b について $\mathsf{Time}(n^a) \subsetneq \mathsf{Time}(n^b)$ がわかる．この $\mathsf{P} \subsetneq \mathsf{EXP}$ や $\mathsf{Time}(n^a) \subsetneq \mathsf{Time}(n^b)$ のように資源の量を増やすと解ける問題が真に増えるという形の事実を総称して**階層定理**(hierarchy theorem)という．

3.4 空間の制限

機械の消費空間というときは作業テープ上の欄の個数のみを数え，入力テープ（や出力テープ）は度外視するのであった（3.1 節）．

時間限定に関する定義（3.2 節）と同様に，「問題 A を○○する△△空間限定のチューリング機械が存在する」を「A は△△空間○○可能である」という．$\mathrm{O}(s(n))$ 空間認識可能な言語の全体を $\mathsf{Space}(s)$ で表し，$\mathrm{O}(s(n))$ 空間計算可能な多価函数の全体を $\mathsf{FSpace}(s)$ で表す．多項式時間のときと同様に，判定問題（言語）について考えるだけで或る程度は空間限定計算の性質が表れて来るので，以下では問 3.17 など一部を除き主に判定問題を扱うことにする．なお空間についても 3.3.3 節に似た議論により階層定理が成立つ（例えば $\mathsf{Space}(n^5)$ は $\mathsf{Space}(n^3)$ よりも真に大きい）．

問 2.4 で示したように，有限状態機械（2.1 節）は作業テープをもたないチューリング機械と同じ働きができ，従って $\mathsf{Space}(1)$ は正則な言語の全体に一致する．以下ではより大きな作業領域を使って認識される言語について考える．

問 3.12 しかし上の定義に正確に従うなら $\mathsf{Space}(1)$ は，作業テープを一本ももたないチューリング機械ではなく，作業テープをもってはいるがその欄を定数個しか訪れないチューリング機械によって認識される言語からなる．何故これを同じと言って良いのか．

3.4.1 対数空間と多項式空間

対数空間認識可能，多項式空間認識可能な言語の全体をそれぞれ L，PSPACE で表す．すなわち

$$\mathsf{L} = \mathsf{Space}(\log n) \qquad \mathsf{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{Space}(n^k)$$

対数空間限定の機械は消費空間が入力長よりも短い．機械の立場からすると，入力があまりに巨大で記憶し切れないが，必要に応じて一部を読みに行くことは許されているという状況である．機械の定義において入力テープを消費空間に含めなかったのは，かような状況を考えるためである．なお時間に関しては，入力を最後まで読み切る暇も与えないのはさすがに奇妙であるから，「対数時間限定」というものは普通考えない．L は（上述の正則言語全体 $\mathsf{Space}(1)$ を別とすれば）本講義で扱う最も小さい計算量級である．

問 3.13 （1）2.2.1 節の図 2.7 の機械は，対数空間限定か．

（2）この機械が認識する言語 ANBN は L に属するか．

3.1 節の初めに述べたように，機械が（作業テープ一本につき）一つの欄を新たに訪れるには少くとも一回の遷移を要するから，多項式時間限定の機械は多項式空間限定である．従って $\mathsf{P} \subseteq \mathsf{PSPACE}$ が成立つ．

一方，次のように時点表示の個数に着目すると，対数空間限定の機械は多項式時間限定でもあることがわかる．M を対数空間限定の機械とすると，$\tau(n) = \mathrm{O}(\log n)$ なる函数 $\tau\colon \mathbb{N} \to \mathbb{N}$ が存在し，M は τ 空間限定である．このとき入力 x に対する M の計算は，$ID_{M,|x|,\tau(|x|)}$ に属する時点表示の列で表される（3.3.2 節）が，この列に同じ時点表示が再び現れることはない．もし或る時点表示 d から一度以上の遷移を経て再び d に戻るならば，更に同じ回数の遷移の後にもまた d，以下ずっと繰返しとなり M の停止性に反するからである．ところが集合 $ID_{M,|x|,\tau(|x|)}$ の大きさは $|Q| \cdot (|x|+2) \cdot |\Gamma|^{2k\tau(|x|)}$ であり，これは M と τ とから決る或る多項式 p を使って $p(|x|)$ で抑えられる．故に M は多項式時間限定である．これにより $\mathsf{L} \subseteq \mathsf{P}$.

[4]**問 3.14**　同様に時点表示の個数に着目することで，PSPACE が EXP（→ 3.3.3 節）に含まれることを示せ．

以上より次が成立つ．

$$\mathsf{L} \subseteq \mathsf{P} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$$

実際にはこの隣同士がいずれも真に異なる（つまり $\mathsf{L} \subsetneq \mathsf{P} \subsetneq \mathsf{PSPACE} \subsetneq \mathsf{EXP}$）であろうと予想されているが，その証明は得られていない．なお二つ隣同士が異なること，すなわち $\mathsf{P} \neq \mathsf{EXP}$ や $\mathsf{L} \neq \mathsf{PSPACE}$ は，階層定理から判る．

[5]**問 3.15**　空間限定の定義では機械が停止することを要求した（3.1 節冒頭）．すなわち L の定義には，任意の入力に対して必ず停止する機械のみが使われている．

太郎君（問 3.6 と同じ人物である）は，停止しない機械であっても，作業テープ上で訪れる欄の個数が限られているならば，空間限定を満していると考えた方が良いのではないかと考えた．つまり，言語 A が対数空間認識可能であることを，或る多項式 p と機械 M とが存在し，任意の個例 $x \in \Sigma^*$ に対して次を満すことを以て定義することを考えた．

- $x \in A$ ならば，M に x を入力したときの計算において，どの作業テープ上でも，$\log p(|x|)$ 個以内の欄しか訪れない
- $x \in A$ ならば，そうではない

しかし，言語 A がこの意味で対数空間認識可能であることは，よく考えると結局 $A \in \mathsf{L}$ と同じことである．これは何故か．

3.4.2 多項式空間で解かれる問題の例

与えられた回路 C が充足割当をもつか問う問題 CSAT は，3.2.3 節で述べたように恐らく P には属しないが，PSPACE には属する．C を充足する割当 $b \in \{0,1\}^k$ があるか，$(0,\ldots,0)$ から $(1,\ldots,1)$ まで 2^k 通りの b を順に作業テープに記憶しながらすべて調べ上げればよい（個々の b が C を充足するか問う問題 CEVAL は，3.2.2 節で触れた通り多項式時間認識可能である）．これには指数時間かかるが，空間は多項式で抑えられる．実は更に一般化した次の問題を多項式空間で解くことができる．

出力数 1 の回路 C と，その入力数 k と同じ個数の量化子の列 $Q = Q_1 \ldots Q_k \in \{\forall, \exists\}^k$ との組 (C, Q) が言語 CQSAT に属するとは，条件

$$Q_k b_k . \, Q_{k-1} b_{k-1} \ldots Q_2 b_2 . \, Q_1 b_1 . \, (C, b_1, b_2, \ldots, b_{k-1}, b_k) \in \text{CEVAL}$$

を満すときとする．ここで $\forall b$ は「b が 1 と 0 のいずれであっても」を，また $\exists b$ は「b が 1 と 0 のいずれかにうまく決めると」を意味する．例えば C が図 A.5 の回路であるとき，$(C, \exists\forall\forall) \in$ CQSAT である．「$\mathtt{x}_3$ と $\mathtt{x}_2$ に如何なる真理値を割当てても，それに応じて $\mathtt{x}_1$ にうまく真理値を割当てれば，C の出力は 1 となる」という条件は，図 A.5 右の表を調べると確かに正しいからである．

先程の CSAT は，与えられた回路 C が $(C, \exists\exists\cdots\exists) \in$ CQSAT を満すか問う問題であり，実質的に CQSAT を特殊な入力に限った問題だったと言える．CQSAT も各割当 $b \in \{0,1\}^k$ について $(C, b) \in$ CEVAL か（すなわち b が C を充足するか）すべて調べ上げて適切に集計することにより解けるので，CQSAT $\in$ PSPACE である．

CEVAL の代りに任意の P に属する言語を考えても同じことが言える．すなわち，或る $R \in$ P を以て

$$A = \left\{ (Q,u) : Q_k b_k . \, Q_{k-1} b_{k-1} \ldots Q_2 b_2 . \, Q_1 b_1 . \, (u, b_1 b_2 \cdots b_{k-1} b_k) \in R \right\}$$

と書ける言語 A は，PSPACE に属する．

CQSAT は二者の間の勝負を解析していずれに必勝法があるか答える問題と言える．$\forall$ の付いた変数への割当をうまく選んで回路の出力を 0 にしようとする A 氏と，$\exists$ の付いた変数への割当をうまく選んで回路の出力を 1 にしようとする E 氏がおり，量化子の列は両者に与えられる手番を表している．上の例では，まず A 氏が $\mathtt{x}_3$ と $\mathtt{x}_2$ の真偽を選ぶ権利を与えられるが，そこで 4 通りの割当のうちいずれを選んでも，それに応じて E 氏が $\mathtt{x}_1$ への割当を適切に選べば，回路の出力を真にすることができた．

問 3.16 同じように二者間の勝敗を判定する問題の例を見よう．有向グラフ $D = (V, E)$

上のしりとり[*5]とは，先手が好きな辺 $e_1 \in E$ を選び，後手がこの e_1 に繋がる（つまり e_1 の終点と e_2 の始点が等しい）辺 $e_2 \in E$ を選び，先手が e_2 に繋がる辺 $e_3 \in E$ を選び，…と繰返し，先に辺を選べなくなった者が敗れるという勝負である．与えられた有向グラフ上のしりとりで両者が最善を尽したとき先手が勝つか判定する問題 SIRITORI が，PSPACE に属することを示せ．

3.4.3　対数空間で解かれる問題の例

L は言語からなるが，対数空間計算可能な多価函数の全体を $\mathsf{FL} = \mathsf{FSpace}(\log n)$ で表す．出力テープは空間限定の対象外だが，上と同じ議論により $\mathsf{FL} \subseteq \mathsf{FP}$ であり，特に長さ n の入力に対する出力の長さは，n の或る多項式では抑えられる．問 3.7 と同様に，函数 $A\colon \{0,1\}^* \to \{0,1\}^*$ が FL に属するには，BIT_A が L に属し，かつ或る多項式 p が存在して任意の $x \in \{0,1\}^*$ で $|A(x)| \leq p(|x|)$ が成立つことが必要十分である．

函数 A, B が FL に属するとき，合成 $A \circ B$ もまた FL に属するが，この理由としては，同じことを 3.2.1 節で FP について示したときとは異なる説明を要する．A, B をそれぞれ計算する対数空間限定の機械 M, N を用いて，$A \circ B$ を計算したい，つまり入力 x に対して $A(B(x))$ を求めたいわけだが，このとき N の手順により文字列 $y = B(x)$ をまず計算して結果を作業テープに書いてから，それを入力とする M の手順により $A(y)$ を計算したのでは，この途中結果 y を書くだけで $|y|$ 程度の空間を使ってしまうことになる．これを避けるには，y 全体を記録してから N を動かすのではなく，N が y の或る桁を読む度に，M を用いてその桁を計算することにすれば良い．つまり，$A \circ B$ を計算する機械は，M の手順に従って計算を進めるのだが，その際に現在 M が y の何桁目を見ているか作業テープの一本に（十進法で，すなわち $\mathrm{O}(\log |y|)$ 程度の空間を使って）保持しておき，M の一遷移を行うために毎回 N を動かして y のその桁を求めるのである．時間の面では効率が悪いかもしれないが，消費空間は M と N の消費空間と $\log |y|$ のいずれか程度で抑えられる．

3.2.2 節で見た加算や乗算について再び考えよう．与えられた（十進法で表された）二つの非負整数 x, y の和 $x + y$ を求める問題 ADD-TWO は FL に属する．と言っても，3.2.2 節で ADD-TWO $\in$ FL を示すために述べたやり方は対数空間限定ではないので，もっと作業テープを節約する工夫が要る．数 x や y を作業テープに一旦書き写すという贅沢はできないので，入力テープに置いたままで，x と y の対応する桁を見に行ってその和を出力テープに書込む．これは今何桁目を扱っているかを作業テープに保持しながら行えばでき

[*5] 字の集合を V とし，字 $u \in V$ で始まり $v \in V$ で終る語を u から v への辺とすれば，これは所謂しりとりと解釈できる．但しこの場合，両端点が等しい辺や，端点を同じくする複数の辺も存在することになる．

る．これで和 $x+y$ を下位の桁から順に出力することはできるが，それができれば上位から順に出力することもできる．何故なら与えられた入力文字列の前後をあべこべにして出力する函数は明らかに FL に属し，上で見たように FL は二つの函数の合成について閉じているからである．

1+4+4 **問 3.17** (1) 与えられた幾つかの非負整数の和を求める問題 ADD-MANY が FP に属することは 3.2.2 節で見たが，同じ算法により ADD-MANY が FL に属するとは言えない．何故か．

(2) しかし ADD-MANY は FL に属することを示せ．

(3) 与えられた二つの非負整数の積を求める問題 MULT-TWO が FL に属することを示せ．

2.2.2 節で，二進法と十進法の間の変換は「容易」であると述べた．実際それを行う函数は FL に属する．しかし実をいうとこれは全く明らかでなく，長いこと未解決であった末に示された[*6]．このことを認めると，FL に属するか否かを考える上では，問題を定義する際に整数を十進法で表すか二進法で表すかをあまり気にしなくてよいことが判る．先述のように FL は函数の合成について閉じているので，或る問題が FL に属するなら，それとこの表記法変換とを合成したものも FL に属するからである．

12 **問 3.18** 命題論理式 φ (A.4 節) と，その入力数 k に合った割当 $b \in \{0,1\}^k$ とが与えられたとき，b の下での φ の真理値を答える問題 FEVAL を考える．これは CEVAL (3.2.2 節) を論理式に制限したものである．FEVAL $\in$ L を示せ．

[*6] 次の論文で問題 MULT-MANY (→ 3.2.2 節) が FL に属することが示されたことにより決着した．

A. Chiu, G. Davida and B. Litow. Division in logspace-uniform NC^1. *RAIRO – Theoretical Informatics and Applications*, 35:259–275, 2001.

第 4 章

非決定性と乱択

前章では様々な資源制約 (時間・空間の限定) を課した機械で解ける問題の集まりとして P，L，PSPACE などを扱った．本章でも新たに RP，BPP，NP など幾つかの級を扱うが，これらは資源制約を変えることによってではなく，同じ資源制約を課した機械を用いつつ，その機械と問題との関わりを変えることによって定義される級である．例えば或る言語が BPP に属することは，P と同じく多項式時間限定の機械を使って定義されるが，P の定義とは違ってこの機械は一つの入力に対して確率的・非決定的に複数通りの動作をし，その中で高い確率で正解を答えればよいとするのである．本章ではこのような級を幾つか定義し，それぞれに属する問題や，級の間の関係について調べる．

2 章で取決めたように機械 M が言語 $A \subseteq \Sigma^*$ を認識するとは，

- もし $x \in A$ ならば，M は入力 x を受理する
- もし $x \notin A$ ならば，M は入力 x を受理しない

が成立つことであった．本章ではこの「認識」の代りに幾つかの「○○認識」を考える．

その際に注意すべきことの一つが，一般に言語 A を○○認識することとその補言語 $\overline{A}$ (附録 A.2 節) を○○認識することは異なるという点である．「認識」については，A と $\overline{A}$ が異なるとはいっても通常の時間制限の下では結局「判定」と同じであり (3.2.3 節)，A を判定する機械の受理と不受理を入れ替えて $\overline{A}$ を判定する機械が得られるが，以下ではこのようなことができるとは限らない．すなわち，言語の集合 (計算量級) C に属する言語の補言語の全体を coC で表すと，前章までの言語の級はみな明らかに補について閉じていた (P = coP，L = coL など) が，以下ではそうとは限らない．

問 4.1 C を言語の集合とする．もし C ⊆ coC ならば C = coC となることを示せ．

	多項式時間限定の機械により	対数空間限定の機械により
認識される	P(3.2.1 節)	L(3.4.1 節)
R 認識される	RP(4.2.1 節)	RL(4.3 節)
N 認識される	NP(4.2.2 節)	NL(4.3 節)
BP 認識される	BPP(4.2.3 節)	BPL

図 4.1　前章では時間と空間の制約のしかたの違いにより幾つかの級を定義したが，今度はその機械が問題に対して何をなすべきかという違いによって新たな級を定義する．例えば RP の定義では，多項式時間限定の機械を考えるのは P と同じだが，その機械によって問題が「認識」ではなく「R 認識」されることを要求する．

4.1　乱択機械

前章では**決定性**(deterministic)チューリング機械を考えた．すなわち各時点において，各テープの現在の注目点にある記号と機械の内部状態とのみに依存して，次に各テープに何を書込み注目点をいずれの向きに移し内部状態を何にするかが一通りに決るのであった．故に入力文字列 x から受理または不受理の結果が一通りに決った．機械が言語 A を認識するとは，A の任意の個例 x に対し，この受理・不受理が正答すなわち x が A に属するかしないかに一致することをいうのであった．

これを拡張して，各時点で次の動きが常に二通りあり，これを独立に確率 1/2 で選ぶ**乱択**(randomized)チューリング機械[*1]を考える．形式的には 2.2.1 節の定義 3 にある遷移規則を $\delta\colon Q \times \Gamma^{1+k} \to \big((Q \times \Gamma^k \times \{\,\textcircled{左}, \textcircled{右}\,\}^{1+k}) \cup \{\,\textbf{止}\,\}\big)^2$ という形にするということである．一つの入力文字列 x に対しても，結果は途中の各時点でこの二通りのうちいずれを選んだかに依る．出力のみならず停止するまでの時間もこの分岐に依存するが，乱択チューリング機械が多項式時間限定であると言った場合には，或る多項式 p が存在し，どの入力 x に対しても分岐によらず $p(|x|)$ 回以内の遷移で停止に至ることを意味するものとする．次の 4.2 節では多項式時間限定の乱択機械を使い，その受理・不受理と正答 $A(x)$ との関係を指定することにより，RP, NP, BPP などの計算量級を定義する (図 4.1)．

各時点で遷移が二通りあるという代りに，次のごとく乱数を初めに選んでおくと考えても同じことである．すなわち機械に入力，作業テープとは別に一本の「乱数テープ」があり，その注目点は各時点で必ず右へ一歩動くものとする．計算の始まりと同時に乱数テープに無限のビット列 r が与えられる．r の各ビットは独立に確率 1/2 で 0

[*1] **非決定性**(nondeterministic)チューリング機械ともいう．

または 1 であり，各時点での遷移は，各テープ（乱数テープも含む）の注目点に書かれた記号と機械の内部状態とから一通りに（決定的に）決る．形式的にいえば遷移規則を $\delta\colon Q \times \Gamma^{1+k} \times \{0,1\} \to (Q \times \Gamma^k \times \{ ㊧, ㊨ \}^{1+k}) \cup \{ \textbf{止} \}$ の形になる．この仕組でも前段落のように各時刻でその都度 1 ビットを選んでゆく機械と全く同じ働きができるわけである．

またここで乱数テープ上の r を無限文字列としたけれども，入力 x に対する計算時間が例えば多項式 $p(|x|)$ で抑えられていれば，実際には r の初めの $p(|x|)$ 文字しか使われない．されば「入力を x とし，遷移先を各時点で確率 $1/2$ で選ぶ」計算は，「長さ $p(|x|)$ のビット列 $r \in \{0,1\}^{p(|x|)}$ を一様な確率で選んだ上で，入力 x と乱数 r を読み取りながら決定的に進む」計算と同じことである（片方の意味での機械があれば，それと結果の分布が全く同じであるような，他方の意味での機械が作れる）．そこで以下では議論の都合上このように一定長の乱数を初めに選んでおくという考え方で話を進めることもある．

なおここで乱択機械は各時点で二つの遷移から等確率で選ぶ（或いは上述の乱数先取の考え方でいえば，r の各桁が 1 か 0 かの二通りである）ものとしたが，これをうまく利用すると 2 の冪でない n に対しても，n 通りの遷移に概ね等確率で分岐できるので，以下ではこの点は気にせず n 個の遷移をちょうど確率 $1/n$ で選ぶこともできるかのごとく考えてよい．

4.2 多項式時間

ここでは多項式時間限定の乱択機械による計算を扱う．

4.2.1 片側誤り

与えられた文字列 x が言語 A に属するか否かを，機械は高い確率で正しく答えればよいとした定義の一つが次のものである[*2]．

> 乱択機械 M が言語 A を **R 認識**するとは，任意の個例 x に対して次が成立つことをいう．
>
> - もし $x \in A$ ならば，M が入力 x を受理する確率は $1/2$ 以上．
> - もし $x \notin A$ ならば，M が入力 x を受理することはない．

[*2] この「R 認識」や以下の「N 認識」「BP 認識」は本演習でのみ使う言葉だが，級の名 RP, NP, BPP は一般的である．それぞれ randomized，nondeterministic，bounded probabilistic に由来するが，これは歴史的経緯からそう名づけられただけであり，必ずしも意味を適切に表しているわけではないので余り気にしない方が良い．

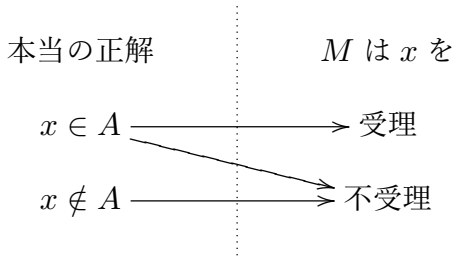


図 4.2　言語 A を R 認識する機械 M．片側にのみ誤る可能性がある．

この「確率は 1/2 以上」「……ことはない」は，乱択機械 M が用いる乱数列 r から生ずる確率的挙動に関して言ったものである．つまり先述 (4.1 節) のように r が集合 $\{0,1\}^{p(|x|)}$ から選ばれるとき，この $2^{p(|x|)}$ 個のうち M をして x を受理せしめるものが，それぞれ $2^{p(|x|)-1}$ 個以上である，0 個である，という意味である．

多項式時間 R 認識可能な言語の全体を RP で表す．明らかに $\mathsf{P} \subseteq \mathsf{RP}$.

問 4.2　この定義を読んだ人々の会話である．空欄に「する」か「しない」を入れよ．

太郎君　機械 M には，与えられた文字列 x が言語 A に属するか否かを，受理するかしないかによって答えてもらいたいわけですね．

先生　そうです．「受理する」は基本的には「属すると言っている」と解釈できます．しかし M の動きは入力 x だけでなく乱数 r にも依存しており，r の値によっては誤ることもあります．

花子さん　どのような誤りが許されるかが，R 認識の定義に書かれていますね．

先生　その条件が非対称であることに注意しましょう（図 4.2）．x が A に属＿**イ**＿とき，M は乱数 r の値によっては x を受理＿**ロ**＿という誤りを犯すことが (確率 1/2 以内で) あるが，逆の誤り方はしないと述べています．

太郎君　つまり機械 M が x が A に属＿**ハ**＿と言ったら完全に信用できるが，属＿**ニ**＿と言ったときは誤っているかもしれないということですね．

先生　その通り．誤る向きを一方に限った**片側誤り**(one-sided error)の形をしています．なお「両側誤り」については 4.2.3 節で扱います．ところで「R 認識」はこの資料を書いた K 助教 (当時) が勝手に拵えた言い方です．

花子さん　何故 K さんはそういう変てこりんな言葉を使うのでしょうか．

先生　きっと RP を始めとする色々な級が，以下この「○○認識」という部分を入替えることで定義されることを強調したいのでしょう．しかし他所では通じませんから注意して下さい．

確率 1/2 以内で誤るというと頼りないが，実はこの 1/2 という数そのものは RP の定義

において本質的ではない．というのも，上の R 認識の定義の中の「……確率は 1/2 以上」の代りに「……確率は $1-\alpha$ 以上」としたものを「R_α 認識」と呼ぶことにすると，例えば言語 A を R 認識（つまり $\mathrm{R}_{1/2}$ 認識）する機械 M があるとき，A は次の機械 N によって $\mathrm{R}_{1/32}$ 認識される．

> N は入力 x を受け取ると，M に x を入力したときの計算を 5 回，別々の乱数列 5 つを用いて行う．この 5 回の実行のうち一つでも受理すれば，N は受理する．

同様に $0<\alpha<1$ なるどの定数 α を使っても多項式時間 R_α 認識可能性の概念は変らない．

このように RP に属する問題は，乱数さえあれば多項式時間で例えば確率 99.9% で答えられるというのであるから，或る意味で P の問題と同様に実際上効率よく解けるといえるかもしれない．但し $\mathsf{RP}=\mathsf{P}$ が成立つか否かはわかっていない．

なお R 認識（や後述の N 認識）は「片側誤り」であり定義が非対称であるから，4 章冒頭で述べたように問題とその補問題をはっきり区別せねばならない．$\mathsf{RP}=\mathsf{coRP}$ であるか否かはわかっていない．

6+6**問 4.3** それでも誤りを恐れる人は，次のように「乱数を使うが絶対に誤らずに解ける」ような問題の級を考えたくなるかもしれない．

言語 A が ZPP に属するとは，或る多項式 q と乱択機械 N とが存在し，A の各個例 x に対して次が成立つことをいう．

- N の x に対する消費時間の（機械の行う乱択による）期待値は $q(|x|)$ 以下である[*3]．
- もし $x\in A$ ならば，N が x を受理する確率は 1 である．
- もし $x\notin A$ ならば，N が x を受理する確率は 0 である．

言語 A が ZPP' に属するとは，或る多項式 p と乱択機械 M とが存在し，A の各個例 x に対して次が成立つことをいう．但し M の状態は受理状態，拒否状態，留保状態の三つに分かれており，それぞれの状態で停止することを M が受理，拒否，留保すると呼ぶ．

- M の x に対する消費時間は（機械の行う乱択の結果に依らず）$p(|x|)$ 以下である．
- もし $x\in A$ ならば，M に x を入力すると，受理または留保する．
- もし $x\notin A$ ならば，M に x を入力すると，拒否または留保する．
- 入力 x に対し M が留保する確率は 1/2 に満たない．

(1) $\mathsf{ZPP}=\mathsf{ZPP}'$ を示せ．

(2) $\mathsf{ZPP}'=\mathsf{RP}\cap\mathsf{coRP}$ を示せ．

[*3] 本講義では 3 章以降，機械は如何なる入力と如何なる乱数に対しても一定時間で停止するものとしており，この ZPP の定義が唯一の例外である．

1+4+6**問 4.4**　入力数 k の算術式とは，変数 $\mathtt{x}_1$，$\mathtt{x}_2, \cdots$，$\mathtt{x}_k$ 及び定数 0，1，-1 から二項演算子 $+$，$\times$ を使って構成される式をいう．論理式（附録 A.4 節）と似たようなものだが，算術式は整数を係数とする多項式を表している．与えられた算術式が多項式として零でないか問う問題 $\overline{\mathrm{PIT}}$ を考える[*4]．

（1）全部展開して確かめれば良いからこの問題は P に属する，と言えないのは何故か．

（2）$\overline{\mathrm{PIT}} \in \mathsf{RP}$（すなわち $\mathrm{PIT} \in \mathsf{coRP}$）を次の補題から示せ．

補題　整数を係数とする零でない k 変数多項式 P があり，その全次数[*5]は d であるとする．また B を正の整数とする．このとき次が成立つ．但し Pr は r_1，$r_2, \cdots$，r_k を独立に B 以下の正整数から一様に選ぶときの確率を表す．

$$\Pr[\,P(r_1, r_2, \ldots, r_k) = 0\,] \leq \frac{d}{B}$$

（3）この補題を k に関する帰納法で示せ．

■乱択算法と約束つき問題　2.3.3 節に述べたように，これまではすべての文字列に対して答の定まった「約束なし問題」を主に扱ってきた．しかし RP のごとき級を考える際には，次の事情により，約束つき問題も含める方がよい場合がある（説明の都合上ここでは P，RP に対応する約束つき問題の級を $\widehat{\mathsf{P}}$，$\widehat{\mathsf{RP}}$ で表す）．

約束つき言語 A が $\widehat{\mathsf{P}}$ に属するとは，多項式時間限定の機械 M が存在して，A の約束 $\mathrm{dom}\,A$ に属する文字列 x に対しては，M に x を入力したときの受理・不受理が，$x \in A$ であるか否かに一致することをいうのであった．しかしこの機械 M は，$\mathrm{dom}\,A$ に属しない文字列を入力したときも受理・不受理のいずれかではあるのだから，何らかの約束なし言語 B を認識している．結局 $\widehat{\mathsf{P}}$ に属する任意の約束つき言語 A は，P に属する或る約束なし言語 B を約束 $\mathrm{dom}\,A$ 上に制限しただけのものである．つまり $\widehat{\mathsf{P}}$ は単に「P に属する言語を何らかの集合に制限して得られる約束つき言語の全体」であるので，P とは別に $\widehat{\mathsf{P}}$ を取り立てて考察する必要は薄かった（なお同様のことは次に扱う級 NP についても成立つ）．

しかし同じ議論は RP では成立たない（成立つか判らない）．つまり $\widehat{\mathsf{RP}}$ に属する約束つき言語 A を R 認識する機械 M を修正して，$\mathrm{dom}\,A$ 上で A に一致する約束なし言語を R 認識する機械を得るということができない．M は $\mathrm{dom}\,A$ に属しない入力に対しては受理確率が $1/2$ 以上でも 0 でもないかもしれず，これは容易には修正し難いからである．

このため R 認識（や後に扱う BP 認識）に関しては，約束つき言語をきちんと考えに入れて議論した方がよい場合がある．今後はそのような場合，断った上で約束つき言語の級を扱うことがある．

[*4] 多項式同一性判定(polynomial identity testing)の頭文字である．与えられた算術式が零であるか問う問題がそう呼ばれている（二つの多項式 P と Q が同一か判定せよというのは結局 $P - Q = 0$ であるか問うているわけで，実質上「同一性判定」と「零判定」は同じことである）．

[*5] すべての変数を合せて数えた次数．例えば $x^3yz^2 + 4y^5$ の全次数は 6 である．

4.2.2 非決定性と検証

先程の R 認識の定義における「1/2 以上の確率で」を「正の確率で」に変えたものが次の定義である.

> 乱択機械 M が言語 A を **N 認識**するとは，任意の個例 x に対して次が成立つことをいう.
>
> - もし $x \in A$ ならば，M が入力 x を受理することがある.
> - もし $x \notin A$ ならば，M が入力 x を受理することはない.

この「……ことがある／ない」はやはり，乱択機械 M が用いる乱数列 r から生ずる確率的挙動に関して言ったものである.

多項式時間 N 認識可能な言語の全体を NP で表す．明らかに $\mathsf{RP} \subseteq \mathsf{NP}$ である.

この定義を少し言換えてみよう．4.1 節で述べたように，多項式時間限定の乱択機械 M による計算は，多項式長の乱数列 r を初めに選んでから計算を始める多項式時間限定の決定的な機械 M' で代えることができる．この r は乱数テープでなく入力テープに書いてあるとしてもよいから，結局問題 A が NP に属するとは，次を満す多項式 p と多項式時間限定の決定的な機械 M' とが存在することに同値である.

- もし $x \in A$ ならば，或る $r \in \Sigma^{p(|x|)}$ が存在し，M' は入力 (x, r) を受理する.
- もし $x \notin A$ ならば，そのような r は存在しない.

こう考えると文字列 r は，個例 x が A に属することの「証拠」のごときものといえる．組 (x, r) が与えられたら，この r が証拠として正しいかは M' により多項式時間でわかる——そしてそのような正しい証拠 r は $x \in A$ であるときにのみ存在する——というのである.

例えば与えられた回路が充足可能か問う問題 CSAT (問 3.8) は NP に属する．この場合，個例 C が充足可能であることの証拠として C の充足割当を用いればよい．回路 C と文字列 r が与えられたとき，r が C を充足する割当であるかは多項式時間で判る (3.2.2 節の $\textsc{ceval} \in \mathsf{P}$)．その充足割当 r は当然ながら $\varphi \in \textsc{csat}$ であるときのみ存在する.

また独立集合問題 IS (2.2.2 節末) も NP に属する．この問題は与えられた (無向) グラフと整数の組 (G, l) について，$G = (V, E)$ に大きさ l 以上の独立集合 $S \subseteq V$ が存在するか問う問題であった．この場合，集合 S (を，要素を書き並べることにより表した文字列) を証拠として用いればよい．与えられた (G, l, S) に対し，S が G における大きさ l 以上の独立集合であるか判定することは容易だからである.

N 認識の定義は形の上では R 認識に似るが，$x \in A$ のときの正解確率の下界を定数で与えず，ただ 0 でないと言うのみである点が大いに異なる．上述の「証拠」の考え方でいえば，各入力 x に対し，数ある乱数 r のうち**少くとも一つ**が証拠となる（機械 M' を受理させる）と述べているに過ぎない．その証拠 r を見出すのは極めて難しいかもしれないから，**N 認識ができても実際に問題を解く役に立つわけではない．**この点で，繰返しにより誤り確率を減らすことができた R 認識とは異なる状況である．RP が或る意味で（完全な乱数が使えれば高確率で正しく）効率よく解ける問題の集まりであるに対し，NP は何らかの実際上の解き易さを捉えようとしたものではない．では何故この級を考えるかといえば，世の中には何かを「見出し」たい状況がよくあり，それを計算問題として定式化すると NP に属する（つまり，見出したものが条件に適するか判断することは多項式時間でできる）ことが多いためである．

NP は P や RP よりも真に大きい（$\mathsf{RP} \subsetneq \mathsf{NP}$）と予想されている．また $\mathsf{NP} \neq \mathsf{coNP}$ とも予想されている．しかしいずれも証明は得られていない．特に $\mathsf{P} \neq \mathsf{NP}$ という予想の決着には米国のクレイ数学研究所により百万ドルの懸賞金が懸けられている．$\mathsf{NP} \cap \mathsf{coNP} = \mathsf{P}$ か否かも判っていない．

4**問 4.5**　与えられた無向グラフ（A.3 節）G, H に対し，G の部分グラフであって H に同型なものが存在するか問う問題が NP に属することを示せ．

注意　上述のように，或る問題が NP に属するという主張は，証拠の適否を判定する機械の存在を意味する．個例 (G, H) に対する証拠 r として何を使うのか，また与えられた (G, H, r) に対して機械は何を行うのか，明確に述べること．

2+13**問 4.6**　PRIMES は与えられた正の整数が素数であるか問う問題である．

（1）$\textsc{primes} \in \mathsf{coNP}$ を示せ（つまり $\overline{\textsc{primes}} \in \mathsf{NP}$ を示せ）．

（2）$\textsc{primes} \in \mathsf{NP}$ を示せ．次の事実を証明せずに使ってよい．
正の整数 n が素数であるには，整数 a が存在し，$a^{n-1} \equiv 1 \pmod{n}$ かつ $n-1$ の各素因数 p に対して $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ となることが，必要十分である．

注意 1　問 4.5 の注意と同じ．

注意 2　入力の整数は二進法で書かれている．

なお実は $\textsc{primes} \in \mathsf{P}$ であることがわかっている[*6]．

[*6] M. Agrawal, N. Kayal, N. Saxena. PRIMES is in P. *Annals of Mathematics* 160:781–793, 2004.

4.2.3 両側誤り

R 認識や N 認識の条件は片側誤りであったが，**両側誤り**(two-sided error)を考えることもできる．

> 乱択機械 M が言語 A を **BP 認識**するとは，任意の個例 x に対して次が成立つことをいう．
> - もし $x \in A$ ならば，M が入力 x を受理する確率は $3/4$ 以上．
> - もし $x \notin A$ ならば，M が入力 x を受理する確率は $1/4$ 以下．

多項式時間 BP 認識可能な言語の全体を BPP で表す．

RP は $\mathrm{R}_{1/4}$ 認識によって定義してもよかった (4.2.1 節) ことに注意して，上の BPP の定義と比べると，RP $\subseteq$ BPP が判る．また BP 認識の誤り確率についても次のことが成立つ．

10**問 4.7** $0 < p < 1/2$ を定数とする．BP 認識の定義における $3/4$ と $1/4$ をそれぞれ $1-p$ と p に置換えても BPP は変らないことを示せ．必要なら問 A.3 で扱った事実を用いよ．

BPP = P が成立つか否かも未解決であるが，どちらかというと成立つと予想されている．これは「乱数は真に必要なのか (決定的算法で真の乱数を真似ることができるのか)」という問といえる．

2+15**問 4.8** 文字列からなる列 $(z_k)_{k\in\mathbb{N}} = (z_0, z_1, \ldots)$ を助言という．機械 M が助言 $(z_k)_{k\in\mathbb{N}}$ により言語 A を認識するとは，A の任意の個例 x に対し，M に $(x, z_{|x|})$ を入力すると，$x \in A$ ならば受理し，$x \notin A$ ならば受理しないことをいう．助言 $(z_k)_{k\in\mathbb{N}}$ が多項式長であるとは，或る多項式 p が存在して任意の $k \in \mathbb{N}$ で $|z_k| \leq p(k)$ となることをいう．多項式時間限定な機械が多項式長の助言によって認識する言語の全体を P/*poly* で表す．

(1) P/*poly* に属するが P に属しない問題を一つ挙げよ．

(2) BPP $\subseteq$ P/*poly* を示せ．

(2)のヒント 問 4.7 の確率 p を更に減らして，任意の多項式 q に対し，入力長 n のときの誤り確率を $1/2^{q(n)}$ 以下にすることもできることをまず示す．

9**問 4.9** RP の定義において $x \in A$ と $x \notin A$ のときの確率の隔たりを薄くすることで NP の定義を得た．同様に BPP の定義において確率の隔たりを薄くしてみる．言語 A が PP に属するとは，或る多項式時間限定の乱択機械 M が存在し，任意の個例 x に対して次が成立つことをいう．

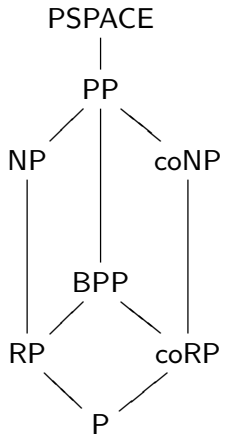


図 4.3　多項式時間限定の乱択機械によって定義される級の包含関係．線で結ばれた下の級は上の級に含まれる．

- もし $x \in A$ ならば，M が入力 x を受理する確率は $1/2$ よりも大きい．
- もし $x \notin A$ ならば，M が入力 x を受理する確率は $1/2$ 以下．

$\mathsf{NP} \subseteq \mathsf{PP} \subseteq \mathsf{PSPACE}$ を示せ (図 4.3)．

4.3　対数空間

以上の多項式時間を対数空間に置換えて級 RL, NL などを考えることができる．つまりこれらは対数空間限定の機械によってそれぞれ R 認識ないし N 認識される言語の全体である．明らかに $\mathsf{L} \subseteq \mathsf{RL} \subseteq \mathsf{NL}$．また $\mathsf{NL} \subseteq \mathsf{P}$ が成立つことを後に問 5.18 (1)で見る．

問 4.10[5]　次の事実が知られる．

> 頂点数 n，辺数 m の連結な無向グラフ (A.3 節) G があり，u をその頂点の一つとする．一人の気侭な旅人が今日 u から出発したのち，現在いる頂点に隣接する頂点から等確率で一つ選んでそこへ移ることを毎日繰返す．するとこの旅人が G の頂点をすべて訪れるまでの日数の期待値は $4mn$ 以下である．

これを用いて $\textsc{upath} \in \mathsf{RL}$ を示せ．但し UPATH は，無向グラフと二頂点 s, t とが与えられたとき，s から t への路が存在するか問う判定問題である．

なお実は $\textsc{upath} \in \mathsf{L}$ であることがわかっている[*7]．

[*7] O. Reingold. Undirected connectivity in log-space. *Journal of the ACM* 55, Article 17, 2008.

2+12 **問 4.11** 有向グラフ（A.3 節）とその二頂点 s, t とが与えられたとき，s から t への路が存在するか問う判定問題 DPATH（3.2.2 節）を考える．

（1）DPATH $\in$ NL を示せ．

（2）DPATH $\in$ Space$((\log n)^2)$ を示せ．

(2)のヒント 「頂点 a から頂点 b へ長さ 2^k 以内の路がある」を $R_k(a,b)$ で表す．R_k を R_{k-1} で表して再帰する．

DPATH は(1)により多項式時間で解かれ，また(2)により対数多項式空間で（すなわち或る $k \in \mathbb{N}$ について $\mathrm{O}((\log n)^k)$ 空間で）解かれることがわかった．しかし DPATH が「多項式時間かつ対数多項式空間」で解かれるか否かは未解決である．

次に示すように DPATH は coNL にも属する．

5+9+6 **問 4.12** 本問では議論の都合上(1)(2)で約束つき言語（2.3.3 節）を扱い，NL も約束つき言語の級と考える．

有向グラフ G において頂点 s から i 歩以内で到達できる頂点の全体を $\Gamma^i_G(s)$ で表す．G の頂点数を n とすると $\Gamma^n_G(s)$ は s から到達できる頂点全体である．

（1）次の約束つき言語が NL に属することを示せ．

約束 入力は有向グラフ G とその二頂点 s, v と非負整数 i, m との組であり，$m = |\Gamma^i_G(s)|$ を満す．

答 $v \notin \Gamma^i_G(s)$ か．

（2）次の約束つき言語が NL に属することを示せ．

約束 入力は有向グラフ G とその頂点 s と非負整数 i, m, m' との組であり，$m = |\Gamma^i_G(s)|$ を満す．

答 $m' = |\Gamma^{i+1}_G(s)|$ か．

（3）(1)(2)で得た算法を使って $\overline{\text{DPATH}} \in$ NL を示せ．

注意 4.1 節で述べた「乱数テープ」の考え方を使う場合は，乱数テープは注目点を右へしか動かせない一方通行であることに注意せよ．

第 5 章

帰着と完全問題

前章までに様々な計算資源の制限下で処理「できる」問題を集めた級を定義した．逆に問題が処理し難く複雑であると主張するにはどうしたらよいか．その方法の一つは，問題 A が問題 B に帰着する（B が解ければこれを使って A も解ける）という概念を使って問題どうしの難しさを比べることである．多くの難しそうな問題 A が B に帰着するならば，B は恐らく難しいと言ってよかろう．この考えに立って各計算量級の中でも最も難しいといえる問題を完全問題という．NP などの主な級については，本章で見るように多くの完全問題が知られている．

5.1 節では帰着の一つである多対一帰着を定義し，これを使って 5.2, 5.3 節では各級の完全問題を見る．前章に引続き本章でも主に言語（判定問題）のみを扱う．

5.1 多対一帰着

1.5 節で述べたように，問題の絶対的な困難さ（計算の手間の下界）を示すことは一般に難しい．しかし二つの問題を比べて相対的な困難さを調べることができる場合がある．

言語 A から言語 B への**多対一帰着**(many-one reduction)とは，函数 T であって任意の x について

$$x \in A \iff T(x) \in B$$

が成立つ[*1]ものをいう（図 5.1）．そのような T であって FP に属するものが存在するとき，A は B に**多項式時間多対一帰着（カープ帰着）**するといい，$A \leq_{\mathrm{m}}^{\mathsf{P}} B$ と書く．

これは，A を解くにはその入力 x に函数 T を施して得られる $T(x)$ を入力として B を解けばよいと述べており，つまり（T を計算する手間を無視すると）「B が解ければ

[*1] T が約束つき言語 A から約束つき言語 B への帰着であることの定義は，任意の $x \in \mathrm{dom}\, A$ に対して $T(x) \in \mathrm{dom}\, B$ であり，かつ任意の $x \in \mathrm{dom}\, A$ についてこの関係 $x \in A \iff T(x) \in B$ が成立つこととする．

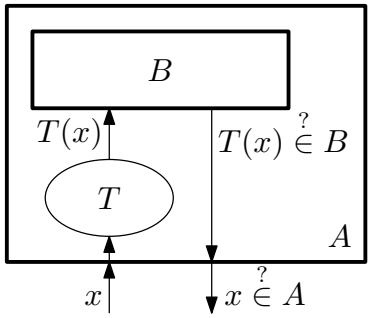


図 5.1　A から B への多対一帰着 T は，A の各個例 x を，答が一致する B の個例 $T(x)$ に，変換する函数である．$x \in A$ か調べるには，$T(x) \in B$ か調べれば良い，という関係が成立つ．

A が解ける」「A の難しさは B 以下」と言える[*2]．実際もし B が P に属するならば，A も (3.2.1 で FP が函数の合成について閉じていることを示したので) P に属する．ただ帰着の利はむしろ次の問のごとく B が P に属するか不明なときも相対的に二つの問題の難しさを比べられる所にある．

関係 $\leq^{\mathsf{P}}_{\mathrm{m}}$ は推移的，すなわち $A \leq^{\mathsf{P}}_{\mathrm{m}} B \leq^{\mathsf{P}}_{\mathrm{m}} C$ ならば $A \leq^{\mathsf{P}}_{\mathrm{m}} C$ が成立つ．

なお同様に FL に属する多対一帰着の存在を以て $\leq^{\mathsf{L}}_{\mathrm{m}}$ を定義することも勿論できる．これは $\leq^{\mathsf{P}}_{\mathrm{m}}$ よりも細かい (すなわち $A \leq^{\mathsf{L}}_{\mathrm{m}} B$ ならば $A \leq^{\mathsf{P}}_{\mathrm{m}} B$ が成立つ) 帰着ということになる (尤も $\mathsf{L} \neq \mathsf{P}$ がまだ証明されない以上，本当に「より細かい」かはわからないが)．既に P に属することがわかっている問題の難しさを比べるには，$\leq^{\mathsf{P}}_{\mathrm{m}}$ 帰着では役に立たないから，より詳細な $\leq^{\mathsf{L}}_{\mathrm{m}}$ などを使うべきである．以下ではそのような細かい比較を要する場合のみ $\leq^{\mathsf{L}}_{\mathrm{m}}$ を使うことにするが，実際には $A \leq^{\mathsf{P}}_{\mathrm{m}} B$ が示されたときには大抵 (例えば次の二問においても実は) $A \leq^{\mathsf{L}}_{\mathrm{m}} B$ も成立つ．

2+2**問 5.1**　与えられた無向グラフ G と正整数 k とに対し，G に大きさ k 以下の点被覆があるか問う判定問題を VC とする．但し無向グラフ $G = (V, E)$ の頂点の集合 $C \subseteq V$ が**点被覆**(vertex cover)であるとは，任意の辺 $uv \in E$ に対して u, v のいずれかが C に属することをいう．

与えられた無向グラフ G と正整数 l とに対し，G に大きさ l 以上の独立集合があるか問う判定問題が IS であった (2.2.2 節末)．与えられた無向グラフ H と正整数 l とに対し，H が大きさ l 以上の閥 (A.3 節) をもつか問う判定問題を CLQ とする．

(1) VC $\leq^{\mathsf{P}}_{\mathrm{m}}$ IS 及び IS $\leq^{\mathsf{P}}_{\mathrm{m}}$ VC を示せ．

(2) IS $\leq^{\mathsf{P}}_{\mathrm{m}}$ CLQ 及び CLQ $\leq^{\mathsf{P}}_{\mathrm{m}}$ IS を示せ．

[*2] ただ，そのような意味をもつ比較の仕方としては他にも考えられる．実際 7.1 節では別の帰着を扱う．ここで多対一帰着を使うのは，後に 5.2 節で NP 完全性を論ずるときなどに都合が良いからである．

注意 帰着の定義は 5.1 節冒頭で述べた通りである．帰着としてここではどの函数を用いるのか，はっきり述べること．

これにより VC と IS と CLQ とは，皆が P に属するか，或いは一つも属しないかであることがわかる．なおそのいずれであるかは未解決である．

1+5**問 5.2** 有限個の品物の集合 E であって各品物 $e \in E$ の重さ $w_e \in \mathbb{N}$ が定まったものが与えられたとき，E を二つの集合に重なりなく分割して重さの和を等しくできるか問う判定問題を PARTITION とする．

同様の品物集合 E と正整数 c との組 (E, c) が与えられたとき，E の部分集合であって重さの和が丁度 c であるものが存在するか問う判定問題を SUBSET-SUM とする．

(1) PARTITION $\leq^{\mathsf{P}}_{\mathrm{m}}$ SUBSET-SUM を示せ．

(2) SUBSET-SUM $\leq^{\mathsf{P}}_{\mathrm{m}}$ PARTITION を示せ．

2+1+8**問 5.3** 問 3.5 の 2CNF 式や問題 2CNFSAT と同様に，三選言節の連言を 3CNF 式といい，与えられた 3CNF 式が充足可能か問う問題を 3CNFSAT という．また与えられた (3CNF 式とは限らない) 命題論理式が充足可能か問う問題を FSAT という．更に，与えられた (論理式とも限らない) 回路が充足可能か問う問題を CSAT (問 3.8) というのであった．

(1) 3CNFSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ FSAT 及び FSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ CSAT を示せ．

(2) $\{0,1\}^k$ から $\{0,1\}$ への任意の函数は，k 変数の或る論理式によって実現される (問 A.6)．A 君はこの事実から CSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ FSAT が成立つと考えた．与えられた回路 C に対し，それと同じ函数を実現する論理式 (の一つ) を $T(C)$ とすれば，函数 T は CSAT から FSAT への帰着となるからである．A 君の考えはどこが誤っているか．

(3) CSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ 3CNFSAT を示せ．

問 5.1 ではグラフに関する問題どうし，問 5.2 では数の和を合せる問題どうし，問 5.3 では論理式と論理回路に関する問題どうしを比べ，それぞれ難しさの度合が或る意味で等しいのを確かめたわけであるが，次の問 5.4，5.5 より，問 5.3 の各問題が問 5.1，5.2 の各問題に多項式時間帰着することも判る．このように一見かなり異なる手合の問題を比べられるのが帰着という考え方の面白い所である．実は逆に問 5.1，5.2 の各問題が問 5.3 の各問題に多項式時間帰着することも 5.2.2 節で判る．

7**問 5.4** 3CNF 式 $\varphi = (l_{11} \vee l_{12} \vee l_{13}) \wedge \cdots \wedge (l_{s1} \vee l_{s2} \vee l_{s3})$ に対し $T(\varphi) = (G_\varphi, s)$ とする．但し G_φ とは，各 $(j,p) \in \{1,\ldots,s\} \times \{1,2,3\}$ に対応する頂点 v_{jp} をもち，辺を次の二つの場合に設けて得られるグラフである (図 5.2)．

- 各 $j \in \{1,\ldots,s\}$ に対し，$v_{j1}v_{j2}$，$v_{j2}v_{j3}$，$v_{j3}v_{j1}$ は辺である．

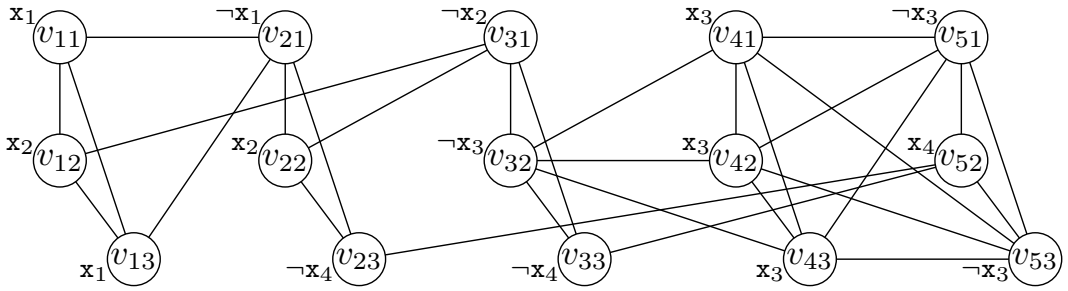


図 5.2　$s = 5$ 個の節からなる 3CNF 式 $\varphi = (\mathtt{x}_1 \vee \mathtt{x}_2 \vee \mathtt{x}_1) \wedge (\neg\mathtt{x}_1 \vee \mathtt{x}_2 \vee \neg\mathtt{x}_4) \wedge (\neg\mathtt{x}_2 \vee \neg\mathtt{x}_3 \vee \neg\mathtt{x}_4) \wedge (\mathtt{x}_3 \vee \mathtt{x}_3 \vee \mathtt{x}_3) \wedge (\neg\mathtt{x}_3 \vee \mathtt{x}_4 \vee \neg\mathtt{x}_3)$ に対して問 5.4 で構成するグラフ G_φ．例えば $l_{12} = \mathtt{x}_2$, $l_{31} = \neg\mathtt{x}_2$ であるから v_{12} と v_{31} の間には辺がある．

- 各変数 x と，$l_{jp} = x$ なる各 j, p と，$l_{j'p'} = \neg x$ なる各 j', p' とに対し，$v_{jp}v_{j'p'}$ は辺である．

この函数 T が 3CNFSAT から IS への多対一帰着であることを証明せよ．

なおこの T は多項式時間計算可能であるので，3CNFSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ IS が判った．

[7]**問 5.5**　3CNFSAT の個例として変数 $\mathtt{x}_1, \cdots, \mathtt{x}_k$ を使った 3CNF 式 $\varphi = (l_{j1} \vee l_{j2} \vee l_{j3}) \wedge \cdots \wedge (l_{s1} \vee l_{s2} \vee l_{s3})$ が与えられたとき，$T(\varphi) = (E_\varphi, c_\varphi)$ と定める．ここで

$$c_\varphi = \sum_{i=1}^{k} 10^{s+i-1} + \sum_{j=1}^{s} 3 \cdot 10^{j-1}$$

であり，E_φ は次の ($2k + 2s$ 個の) 品物からなる集合である (図 5.3)．

- 各 $i = 1, \cdots, k$ に対し，重さ

$$10^{s+i-1} + \sum_{j=1}^{s} \sum_{p=1}^{3} \begin{cases} 10^{j-1} & l_{jp} = \mathtt{x}_i \text{ のとき} \\ 0 & \text{然らざるとき} \end{cases}$$

の品物 $e_{\mathtt{x}_i}$ と，重さ

$$10^{s+i-1} + \sum_{j=1}^{s} \sum_{p=1}^{3} \begin{cases} 10^{j-1} & l_{jp} = \neg\mathtt{x}_i \text{ のとき} \\ 0 & \text{然らざるとき} \end{cases}$$

の品物 $e_{\neg\mathtt{x}_i}$.

- 各 $j = 1, \cdots, s$ に対し，重さ 10^{j-1} の品物 a_j, b_j.

この函数 T は 3CNFSAT から SUBSET-SUM への多対一帰着であることを示せ．

なおこの T は多項式時間計算可能であるので，3CNFSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ SUBSET-SUM が判った．

[1+13]**問 5.6**　論理回路の $\wedge$, $\vee$ の代りに $+$, $\times$ を使ったものを**算術回路**と呼ぼう．但し $\neg$ に当るものはなく，定数 0, 1, -1 が使える．回路を A.4 節の方法で表したと同様に，入力数 k，素子数 n の算術回路は次のように命令を並べたものである．

$e_{\mathtt{x}_1}$	1 0 0 0 0 2
$e_{\neg\mathtt{x}_1}$	1 0 0 0 1 0
$e_{\mathtt{x}_2}$	1 0 0 0 0 1 1
$e_{\neg\mathtt{x}_2}$	1 0 0 0 1 0 0
$e_{\mathtt{x}_3}$	1 0 0 0 3 0 0 0
$e_{\neg\mathtt{x}_3}$	1 0 0 2 0 1 0 0
$e_{\mathtt{x}_4}$	1 0 0 0 1 0 0 0 0
$e_{\neg\mathtt{x}_4}$	1 0 0 0 0 0 1 1 0
a_1	1
b_1	1
a_2	1 0
b_2	1 0
a_3	1 0 0
b_3	1 0 0
a_4	1 0 0 0
b_4	1 0 0 0
a_5	1 0 0 0 0
b_5	1 0 0 0 0
c_φ	$=$ 1 1 1 1 3 3 3 3 3

図 5.3 $k = 4$ 個の変数 $\mathtt{x}_1$, $\mathtt{x}_2$, $\mathtt{x}_3$, $\mathtt{x}_4$ から作られる $s = 5$ 個の節からなる 3CNF 式 $\varphi = (\mathtt{x}_1 \vee \mathtt{x}_2 \vee \mathtt{x}_1) \wedge (\neg\mathtt{x}_1 \vee \mathtt{x}_2 \vee \neg\mathtt{x}_4) \wedge (\neg\mathtt{x}_2 \vee \neg\mathtt{x}_3 \vee \neg\mathtt{x}_4) \wedge (\mathtt{x}_3 \vee \mathtt{x}_3 \vee \mathtt{x}_3) \wedge (\neg\mathtt{x}_3 \vee \mathtt{x}_4 \vee \neg\mathtt{x}_3)$ に対し，問 5.5 で構成する品物 18 個の重さと，目標とする重み和 c_φ．例えば $e_{\neg\mathtt{x}_3}$ の下から 5 桁目が 2 であるのは，$\neg\mathtt{x}_3$ が φ の第 5 節に 2 度現れることを表す．

- $i = 1, \cdots, k$ については，i 番目の命令は $\mathtt{x}_i := \mathtt{input}$ である．
- $i = k+1,\ k+2,\ \cdots,\ n$ については，i 番目の命令は次のいずれかの形をしている．

  $\mathtt{x}_i := c$（但し c は $\mathtt{0}$ または $\mathtt{1}$ または $-\mathtt{1}$）

  $\mathtt{x}_i := \mathtt{x}_j + \mathtt{x}_k$（但し $j,\ k$ は i 未満の正整数）

  $\mathtt{x}_i := \mathtt{x}_j \times \mathtt{x}_k$（但し $j,\ k$ は i 未満の正整数）

これを通常の数式の列として解釈すると，最後の変数 $\mathtt{x}_n$ には $\mathtt{x}_1, \cdots, \mathtt{x}_k$ の多項式で表されるものが入ることになるが，これをこの算術回路の表す多項式と呼ぶことにする．与えられた算術回路に対し，それが表す多項式が零であるか問う判定問題を A とする．

特に入力数 0 の算術回路が表す多項式は単なる整数である．与えられた 0 入力の算術回路に対し，それが表す数が 0 であるか問う判定問題を A_0 とする．

(1) 或るひと曰く「A_0 を解くには論理回路の評価 (3.2.2 節の CEVAL) と同じように単に順に実行すればよい．故に $A_0 \in \mathsf{P}$」と．何が誤りか．

(2) $A_0 \leq^{\mathsf{P}}_{\mathrm{m}} A$ は明らかである．$A \leq^{\mathsf{P}}_{\mathrm{m}} A_0$ を示せ．

この問 5.6 の問題 A, A_0 も P に属するか否かは不明だが，一方が属するなら他方も属することがわかったわけである．

[16]**問 5.7**　$\{0,1\}$ 上の約束つき言語 A と函数 $g\colon \mathbb{N} \to \mathbb{N}$ に対し，約束つき言語 B を

$$\operatorname{dom} B = \{\, x10^{g(|x|)} : x \in \operatorname{dom} A \,\}$$

及び

$$\text{各 } x \in \operatorname{dom} A \text{ について} \qquad x10^{g(|x|)} \in B \iff x \in A$$

により定める．つまり B は A の「入力長を g の分だけ水増しした」問題である．このとき（g によらず）$B \leq_{\mathrm{m}}^{\mathsf{P}} A$ が成立つのは明らかだが，逆が成立つか考えよう．

$A \notin \mathsf{P}$ とし，g が超多項式的である（すなわち任意の $k \in \mathbb{N}$ について $g(n) = \Omega(n^k)$）とする．このとき $A \leq_{\mathrm{m}}^{\mathsf{P}} B$ とはならないことを示せ．

注意　或る特定の帰着 T が FP に属せずというのみでは不十分である．

ヒント　「A から B への帰着を再帰的に使って A を解く」

[3+9]**問 5.8**　（1）PSPACE に属する任意の言語は $\mathsf{Space}(n)$ に属する或る言語に多項式時間多対一帰着することを示せ．

（2）$\mathsf{P} \neq \mathsf{Space}(n)$ を示せ．

なお $\mathsf{P} \subseteq \mathsf{Space}(n)$ や $\mathsf{P} \supseteq \mathsf{Space}(n)$ が成立つか否かはわかっていない．

(1)のヒント　「水増し」（問 5.7）

5.2　NP 完全問題

或る問題を解くのが困難であると主張するには，本来ならばその問題が何らかの級に属しないことを証明できればよいのだが，既に述べたように，P や NP のごとく互に異なると予想される計算量級も，本当に異なるとは示されていない．つまり P に属していなそうな NP 問題であっても，そのことを証明できていないのである．

そこで帰着の概念が役立つ．多項式時間帰着の関係 $\leq_{\mathrm{m}}^{\mathsf{P}}$（5.1 節）を用いて NP に属する問題どうしを比べたとき，最も困難であるというのが次に定義する NP 完全性である．

5.2.1　NP 完全性

問題 $B \in \mathsf{NP}$ が NP **完全**(NP-complete)であるとは，任意の $A \in \mathsf{NP}$ に対して $A \leq_{\mathrm{m}}^{\mathsf{P}} B$ が成立つことをいう．

[2+1+1]**問 5.9**　（1）$A \leq_{\mathrm{m}}^{\mathsf{P}} B$ かつ $B \in \mathsf{NP}$ ならば $A \in \mathsf{NP}$ が成立つことを示せ．

(2) $A \leq^{\mathsf{P}}_{\mathrm{m}} B \in \mathsf{NP}$ かつ A が NP 完全ならば B もまた NP 完全であることを示せ.
(3) もし或る NP 完全問題が P に属すれば, $\mathsf{NP} = \mathsf{P}$ であることを示せ.

つまり NP 完全な問題は或る意味で NP のすべての問題の難しさを代表するといえる.

NP 完全問題は存在する. 例えば次の問題 NTIME-EVAL がそれである.

入力 乱択機械 M と文字列 x と羅列表記された数 $t \in \mathbb{N}$ との組 $(M, x, 0^t)$.

答 M に x を入力して無限乱数列 r に従って動作すると消費時間 t 以内で受理するような r はあるか.

まず NTIME-EVAL は NP に属する. これは 3.3.1 節末で見た TIME-EVAL $\in$ P という事実, すなわち与えられた機械を効率よく摸倣できるという万能性が, N 認識についても成立つということであり, 同様の議論から判る.

一方 NTIME-EVAL が NP 完全であることを示すため, 勝手な $A \in \mathsf{NP}$ を考える. 或る多項式 p と乱択機械 M とが存在し, M は p 時間限定であって A を N 認識する. すると $x \in A$ であるか否かは, $(M, x, 0^{p(|x|)}) \in$ NTIME-EVAL の真偽に一致する. つまり, $T(x) = (M, x, 0^{p(|x|)})$ で定まる函数 T が, A から NTIME-EVAL への多対一帰着になっている. この T は明らかに多項式時間計算可能だから $A \leq^{\mathsf{P}}_{\mathrm{m}}$ NTIME-EVAL. この A は任意であったから NTIME-EVAL は NP 完全である.

ところで 4.1 節で述べたように乱択機械は, 与えられた $\{0, 1\}$ 上の乱数列を読みながら決定的に動く機械と考えてもよいのであった. つまり上述の NTIME-EVAL の入力 M を決定的な機械とし,「M に x を入力して無限乱数列 r に従って動作すると消費時間 t 以内で受理するような r は」を「長さ t のビット列 r であって, M に (x, r) を入力して動作すると消費時間 t 以内で受理するようなものは」に代えても実質的に同じ問題であり, やはり NP 完全である. 更に x を無くした次の問題 TIME-SAT も NP 完全である.

入力 (決定的) 機械 N と羅列表記された数 $s \in \mathbb{N}$ との組 $(N, 0^t)$.

答 N に消費時間 t 以内で受理される文字列はあるか.

実際, NTIME-EVAL $\leq^{\mathsf{P}}_{\mathrm{m}}$ TIME-SAT が成立つ. これは, NTIME-EVAL の個例 $(M, x, 0^t)$ が与えられると, M の入力を x に固定して後は長さ t の乱数列を受付けるのみとした機械 N に変換できるからである. つまり N は $\{0, 1\}$ を入力字母とする機械であり, M と同じ動きをするが, M が乱数テープを読む代りに N は入力テープを読み (但し空白文字を読んだときは受理せずに停止することとする), M が入力テープを読む代りに N は入力テープに x が書かれているかのように動作する. これは M の入力テープ上での位置を N の状態に保持することによって可能である. $T(M, x, 0^t) = (N, 0^t)$ とすれば, T は NTIME-EVAL から TIME-SAT への多項式時間計算可能な多対一帰着となる.

5.2.2　様々な NP 完全問題

様々な分野の自然な問題にも NP 完全なものが数多くある．或る問題 A が NP 完全ならば，問 5.9 (2)により，$A \leq^{\mathsf{P}}_{\mathrm{m}} B$ なる別の問題 $B \in \mathsf{NP}$ も NP 完全とわかる．以下この方法で多くの問題の NP 完全性を示そう．

まず与えられた回路が充足可能か問う判定問題 CSAT は NP 完全である（**クックの定理**）．これには TIME-SAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ CSAT を示せばよい．機械の動きを表す回路を作る問題 MAKE-CIRC が FP に属したから (3.3.2 節)，TIME-SAT の各個例 $(M, 0^t)$ に対して $T(M, 0^t) \in$ MAKE-CIRC$[(M, 0^t, 0^t)]$ となるような函数 $T \in \mathsf{FP}$ が存在する．この回路 $T(M, 0^t)$ が充足可能であるのは，M によって消費時間 t 以内で受理される文字列が存在するときであるから，T は TIME-SAT から CSAT への多対一帰着である．

CSAT が NP 完全なので，問 5.3 より FSAT や 3CNFSAT も NP 完全である．独立集合問題 IS については問 5.4 で 3CNFSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ IS を示した．これにより IS も NP 完全である．更に問 5.1 より VC や CLQ も NP 完全である．問 5.5 より 3CNFSAT $\leq^{\mathsf{P}}_{\mathrm{m}}$ SUBSET-SUM であるから，SUBSET-SUM も NP 完全である．

問 5.10　次の問題 KNAPSACK が NP 完全であること示せ．

KNAPSACK の個例は品物の有限集合 U と正整数 B, $K \in \mathbb{N}$ との組であって各品物 $u \in U$ について重量 $s(u) \in \mathbb{N}$ と価値 $v(u) \in \mathbb{N}$ が定まったものである．これに対して

$$\sum_{u \in S} s(u) \leq B$$

（重量の和が容量 B を超えない）かつ

$$\sum_{u \in S} v(u) \geq K$$

（価値の和が目標値 K に達する）なる $S \subseteq U$ が存在するかを判定したい．

ヒント　既に扱った NP 完全問題のいずれかからの帰着を作る．

問 5.11　問 4.5 の問題 SUBGRAPH-ISOM が NP 完全であることを示せ．

ヒント　問 5.10 のヒントと同じ．

問 5.12　次の問題 SC が NP 完全であることを示せ．

SC の個例は，有限集合 X とその部分集合の集まり $\mathfrak{S}$ と正整数 k との組 $(X, \mathfrak{S}, k)$ である．これに対し，大きさ k 以下の $\mathfrak{C} \subseteq \mathfrak{S}$ であって，X の任意の元が $\mathfrak{C}$ の或る元に属するようなものが，存在するか判定したい．

ヒント　問 5.10 のヒントと同じ．

12**問 5.13** 正整数 a で割ると r 余る整数全体を $a\cdot\mathbb{Z}+r=\{\,a\cdot m+r : m\in\mathbb{Z}\,\}$ で表すことにする．次の問題が NP 完全であることを示せ．

正整数と整数の組を幾つか並べた列 $((a_1,r_1),\ldots,(a_k,r_k))$ が与えられたとき，

$$\bigcup_{i=1}^{k}(a_i\cdot\mathbb{Z}+r_i)$$

に属しない整数が存在するか判定したい．

ヒント 問 5.10 のヒントと同じ．

5.2.3 NP と自己帰着

級 NP や NP 完全問題の構造についてもう少し見てゆこう．

15**問 5.14** 言語 $A\subseteq\{0,1\}^*$ が羅列的であるとは，$A\subseteq\{0\}^*$ であることをいう．

P $\neq$ NP とする．羅列的で NP 完全な言語が存在しないことを示せ．

ヒント 問 3.8 のヒントにあるように $\varphi\in$ CSAT となるのは $\varphi_0\in$ CSAT 又は $\varphi_1\in$ CSAT が成立つときである．これを再帰的に使うと CSAT を解くことはできるが，毎回二手に分かれるから指数的な手間がかかる（変数 $x_1,\cdots,x_i$ を固定する仕方は 2^i 通りある）．CSAT が羅列的な言語に帰着することを用いてこの爆発を抑える．

15**問 5.15** もし NP $\subseteq$ BPP ならば NP $=$ RP となることを示せ．

ヒント 問 4.8 のヒントのように BPP の誤り確率を著しく減らすことができる．もし NP $\subseteq$ BPP ならばこのことより CSAT が低い誤り確率（但し両側誤り）で判定できる．これを片側誤りにしたい．つまり或る論理式を充足可能と答えるのは本当に充足割当があるのを確かめてからにしたい．そのために問 5.14 のヒントにある φ の充足可能性と φ_0, φ_1 の充足可能性との関係を繰返し用いる．

問 5.16 NP $\neq$ P ならば，NP に属するが P に属せず NP 完全でもない問題が存在すること（ラードナーの定理[*3]）を示そう（図 5.4）．本問では議論の都合上まず約束つき判定問題（2.3.3 節）を扱い，P や NP も約束つき判定問題からなる級と考えるが，最後の(4)で結論は約束なし問題に限っても成立つことを示す．

以下では NP $\neq$ P と仮定し，A を $\{0,1\}^*$ 上の NP 完全な（約束なし）判定問題とする．

非負整数の有限列 $(e_0,\ldots,e_{n-1})$ に対し，約束つき判定問題 $A_{(e_0,\ldots,e_{n-1})}$ を

$$\operatorname{dom}A_{(e_0,\ldots,e_{n-1})}=\{\,x10^{|x|^{e_{|x|}}} : x\in\{0,1\}^{<n}\,\}$$
$$A_{(e_0,\ldots,e_{n-1})}(x10^{|x|^{e_{|x|}}})=A(x)$$

[*3] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the ACM* 22(1):155–171, 1975.
但し問 5.16 で用いた証明はこの論文とは異なる．

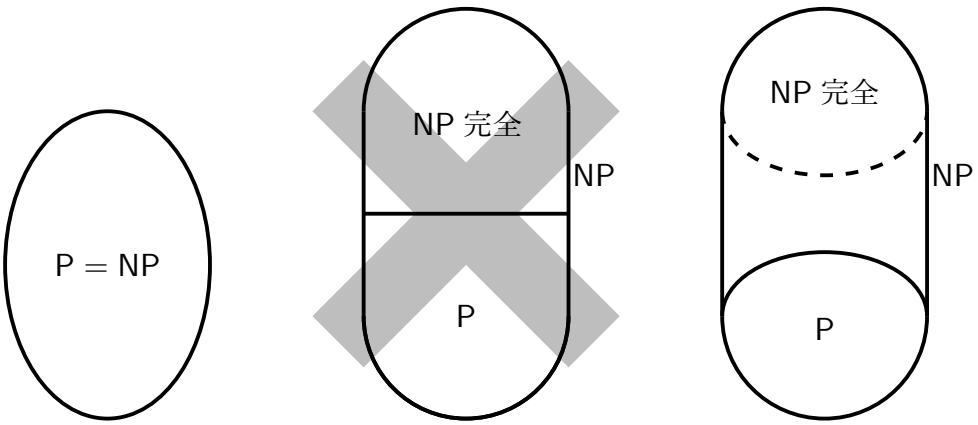


図 5.4　NP ≠ P は示されていないが，中央の図の関係にないことはわかっている．

で定義する．非負整数の無限列 $e = (e_0, e_1, \ldots)$ に対し，すべての $n \in \mathbb{N}$ についてこれらの問題 $A_{(e_0,\ldots,e_{n-1})}$ を合併して得られる約束つき判定問題を A_e と書く．

つまり A_e は A に似た問題だが，A が入力として文字列 x を受取る代りに文字列 $x10^{|x|^{e_{|x|}}}$ を受取るというものである．このように入力が水増し (問 5.7) されているので，A_e は一般に A よりも——数列 e が速く増大すればするほど——易しくなる．極端な話，e が充分に速く増大すれば，$A_e \in \mathsf{P}$ となる．逆に e が有界ならば A_e は A と同じく NP 完全なままである．実はこの逆も成立つ．

(1) 今述べたこと，すなわち任意の e について

$$e \text{ は有界} \iff A_e \text{ は NP 完全}$$

が成立つことを示せ．問 5.7 を用いよ．

本問の目標はラードナーの定理であるから，e として「有界ではないが，速く増大し過ぎることもない」数列を選ぶことで，A_e が NP 完全でも P に属するでもないようにしたい．

(2) 多項式時間限定の機械をすべて並べた列 $M_0, M_1, \cdots$を一つ取る．各 $n \in \mathbb{N}$ に対して e_n を，M_{e_n} が $A_{(e_0,\ldots,e_{n-1})}$ を認識するような最小の数と定める．これにより定まる列 $e = (e_0, e_1, \ldots)$ について次が成立つことを示せ．

$$e \text{ は有界} \iff A_e \in \mathsf{P}$$

(3) e を(2)で選んだ通りとする．A_e が P に属せず NP 完全でもないことを示せ．

以上により A_e は確かに $\mathsf{NP} \setminus \mathsf{P}$ に属し NP 完全でないことがわかったが，本来はそのような約束なし問題の存在を示したいのであった．A_e に似た約束なし問題を得るには，約束 $\mathrm{dom}\, A_e$ に属しない入力に対する値を例えばすべて偽に決めてしまうという方法が考えられるが，そうすると ($\mathrm{dom}\, A_e$ は容易に計算し難いため) NP に属するとは限らなくなってしまう．

(4) (2)において e_n を「最小の」数と定める必要は特に無かったことに着目し，e_n の定め方を修正することで，A_e から所望の約束なし問題を直ちに得られるようにせよ．

5.3 その他の級の完全問題

同様に PSPACE 完全性も定義できる．すなわち問題 $B \in \mathsf{PSPACE}$ が PSPACE **完全** (PSPACE-complete)であるとは，任意の $A \in \mathsf{PSPACE}$ に対して $A \leq_{\mathrm{m}}^{\mathsf{P}} B$ が成立つことをいう．NTIME-EVAL が NP 完全であったのと同様に，与えられた組 $(M, x, 0^s)$ が次を満すか判定する問題 SPACE-EVAL が PSPACE 完全である．

- 機械 M に文字列 x を入力して動かすと受理し，消費空間は s 以下である．

またこの問題を帰着することにより例えば 3.4.2 節の判定問題 CQSAT が PSPACE 完全であることが知られている．

一方，P やそれよりも小さい級における完全性を定義するには，5.1 節で述べたように，FP を使って定義した帰着 $\leq_{\mathrm{m}}^{\mathsf{P}}$ は役に立たないので，対数空間の多対一帰着によって定義した $\leq_{\mathrm{m}}^{\mathsf{L}}$ を使うことにする．問題 $B \in \mathsf{P}$ が P **完全**(P-complete)であるとは，任意の $A \in \mathsf{P}$ に対して $A \leq_{\mathrm{m}}^{\mathsf{L}} B$ が成立つことをいう[*4]．

[6]**問 5.17** 機械の動きを表す回路を作る問題 MAKE-CIRC (3.3.2 節) は，より詳しく見ると FL に属することがわかる．このことを用いて CEVAL が P 完全であることを示せ．

問題 $B \in \mathsf{NL}$ が NL **完全**(NL-complete)であるとは，任意の $A \in \mathsf{NL}$ に対して $A \leq_{\mathrm{m}}^{\mathsf{L}} B$ が成立つことをいう．

有向グラフ G とその二頂点 s, t が与えられたとき，G に s から t への路があるか問う問題 DPATH (問 4.11) は，NL 完全である．つまり勝手な問題 $A \in \mathsf{NL}$ に対し，A から DPATH への対数空間限定な多対一帰着 T_A が存在する．実際，A の個例 x に対し，DPATH の個例 $T_A(x) = (G, s, t)$ を次のように作ればよい．$A \in \mathsf{NL}$ より，或る $s(n) = \mathrm{O}(\log n)$ なる函数 $s\colon \mathbb{N} \to \mathbb{N}$ と s 空間限定な機械 M とが存在して M は A を N 認識する．M に文字列 x を入力したときの時点表示の全体 $ID_{M,x,s(|x|)}$ は，大きさが $p(|x|)$ 以内であった (3.3.2 節)．これに受理・不受理の停止を表す点 **受**, **拒** を加えた集合 $ID_{M,x,s(|x|)} \cup \{\, \textbf{受}, \textbf{拒} \,\}$ を頂点集合とし (3.3.1 節)，それらの間の遷移関係を辺とするグラフを G とする．M が x を受理するのは，G において初期時点表示 $s = d_0$ から受理を表す点 $t =$ **受** への路が存在するときである．以上の構成は x から単純な手順でき，T_A は対数空間計算可能である．

[*4] 本来はどの帰着を使うか明示して NP-$\leq_{\mathrm{m}}^{\mathsf{P}}$ 完全，P-$\leq_{\mathrm{m}}^{\mathsf{L}}$ 完全などと言うべきだが省くことが多い．本演習で現れた NP-$\leq_{\mathrm{m}}^{\mathsf{P}}$ 完全問題はみな，実はほぼ同じ帰着により NP-$\leq_{\mathrm{m}}^{\mathsf{L}}$ 完全でもある．

3+3+3**問 5.18**　DPATH が NL 完全であることを用いて，

（1）$\mathsf{NL} \subseteq \mathsf{P}$ を示せ．

（2）問 4.11 (2)より，$\mathsf{NL} \subseteq \mathsf{Space}((\log n)^2)$ を示せ．

（3）問 4.12 より，$\mathsf{NL} = \mathsf{coNL}$ を示せ（**イマーマン・セレプチェーニの定理**）．

6**問 5.19**　問 5.18 (2)を用いて PSPACE = NPSPACE を示せ．但し NPSPACE は多項式空間限定の機械により N 認識される言語の全体を表す．

ヒント　「水増し」(問 5.7)

11**問 5.20**　2CNFSAT が NL 完全であることを示せ．問 3.5 (1)や問 5.18 (3)の結果を用いるときは明示せよ．

第 6 章

最適化と近似

多価函数を計算する問題 (2.3.1 節) A においては，一つの入力 (個例) x に対して解 $y \in A[x]$ が一般に複数ある．これらの解の間に何らかの意味で優劣があり，中でもなるべく良い解を得たい場合がある．つまり各個例 x に対して適格(feasible)な解の集合 $A[x]$ が決っているが，加えて各解 $y \in A[x]$ の「良さ」の度合を表す非負実数 $c(x, y)$ が定まっており，これをなるべく大きくしたい．このような問題 (A, c) を総称して**最大化**問題といい，c を**目的函数**(objective function)と呼ぶ．

これまでに見て来た判定問題の中には，関連する最大化問題を考えることができるものも多い．例えば独立集合問題 IS (2.2.2 節末) は，与えられた無向グラフ G と正整数 l とに対し，G に大きさ l 以上の独立集合があるか問う判定問題であった．しかし，与えられるのはグラフ G のみであり，その G の独立集合のうち最大のものを求めたいという問題 MAX-IS を考えるのも自然であろう．元々 1.3 節で考えたのもそのような問題であった．判定問題 IS は NP 完全であった (問 5.4) から，P = NP でない限り，IS は多項式時間認識可能でなく，従って最大化問題 MAX-IS において常に目的函数を最大にする解を出力する多項式時間算法は存在しない．

そこで厳密な最大化は諦め，なるべく最大に近い値を達成する近似を目指すことにしよう．最大化問題 (A, c) について，算法 M が多価函数 A を正しく計算する，すなわち各個例 x に対して M は適格な解 $M(x) \in A[x]$ を出力するとする．非負実数 r について，M がこの最大化問題 (A, c) を r **近似**するとは，任意の x と任意の $y \in A[x]$ とについて，この解 $M(x)$ の良さ $c(x, M(x))$ が

$$c\big(x, M(x)\big) \geq r \cdot c(x, y)$$

を満すことをいう．そのような最大の r を M の**近似率** (approximation ratio) という．これは 1 以下の値であって 1 に近いほど良い (これの逆数を近似率と呼ぶ文献もある)．

問 6.1 最大化問題 MAX-IS を 1 近似する多項式時間算法が存在することと，判定問題 IS

が P に属することとが，同値であることを示せ．

2+2**問 6.2**　問 5.10 で扱った背嚢問題 KNAPSACK を最大化の形にした問題 MAX-KNAPSACK を考えよう．この問題 MAX-KNAPSACK では，各品物 $i = 1, \cdots, n$ の重量 $s_i \in \mathbb{N}$ と価値 $v_i \in \mathbb{N}$ と正整数 $B \in \mathbb{N}$ とが与えられる．これに対し重量の和が B を超えないような品物の取り方のうち価値の和のなるべく大きいものを求めたい．

(1) 次の算法は MAX-KNAPSACK を正定数近似（或る $r > 0$ に対し r 近似）するか．

$T = \emptyset$ から始める．品物 i を一つづつ，価値 v_i の降順に見て，これを T に加えても重量の和が B を超えないならば加える．終ったら T を出力する．

(2) (1)の「価値 v_i の降順に」の代りに「重量あたりの価値 $\dfrac{v_i}{s_i}$ の降順に」とした算法は，MAX-KNAPSACK を正定数近似するか．

目的函数 c が「良さ」「利益」でなく「悪さ」「費用」を表しており，これをなるべく小さくしたいという**最小化**問題についても，同様に（「$\geq$」「最大」の代りに「$\leq$」「最小」として）定義する．近似率は存在すれば 1 以上の値となり，1 に近いほど良い．最大化，最小化を合せて**最適化**(optimization)問題と呼ぶ．

1+6**問 6.3**　与えられた無向グラフ $G = (V, E)$ の点被覆（問 5.1）のうち頂点数のなるべく小さいものを求める問題に対し，次の近似算法を考える．

G における極大な取組[*1]を一つ求め，その各辺の両端点すべてを出力する．

(1) この算法が正しいこと，すなわち得られる解が確かに G の点被覆であることを示せ．

(2) この算法の近似率を求めよ．

(2)についての注意　近似率がその値以上であることと以下であることの両方を示すこと．

7**問 6.4**　人数 m と各仕事 $j \in \{1, \ldots, n\}$ の重さ $t_j \geq 0$ が与えられる．すべての仕事をいずれかの人に割当てたい．各人の負荷はその人に割当てた仕事の重さの合計とし，その最大値(最も大変な人の負荷)をなるべく小さくしたい．この問題を次の算法が 4/3 近似することを示せ．

各人に何も割当てられていない状況から始め，仕事を重い順に一つずつ割当てる．その際，既に割当てられた仕事の重さの合計が最も小さい人に割当てる．

NP 完全な判定問題に対応する最適化問題の中でも，近似のし易さには違いがある．例えば判定問題としての最小頂点被覆問題 VC と最大独立集合問題 IS は殆ど同じものであったけれども（問 5.1），与えられたグラフにおけるなるべく大きな独立集合を求める最大化

[*1] 無向グラフ $G = (V, E)$ における取組(matching)とは辺集合 $M \subseteq E$ であって端点に重複なきものをいう．それが極大であるとは，M を含む取組が M 以外に存在しないことをいう．

問題は，NP = P でない限り，（問 6.3 の頂点被覆最小化問題とは違って）多項式時間正定数近似不能であることが知られている．

問 6.5 HC は与えられた無向グラフ（A.3 節）G にハミルトン閉路（すべての頂点を一度づつ通る閉路）が存在するか答える判定問題であり，NP 完全であることが知られている．

NP $\neq$ P ならば次の最小化問題 MIN-TSP を定数近似する多項式時間算法が存在しないことを示せ．

MIN-TSP の個例は，相異なる町 i, $j \in \{1, \ldots, m\}$ 間の旅費 $d(i,j) = d(j,i) \in \mathbb{N} \setminus \{0\}$ で与えられる[*2]．町を一度づつ訪れて帰る旅程すなわち $(1, \ldots, m)$ の並べ替え $(i_1, \ldots, i_m)$ のうち，総費用 $d(i_1, i_2) + d(i_2, i_3) + \cdots + d(i_{m-1}, i_m) + d(i_m, i_1)$ のなるべく小さいものを求めたい．

問 6.6 NP $\neq$ P ならば，問 6.4 の問題を 1 近似する多項式時間算法が存在しないことを示せ．

[*2] 距離の公理（三角不等式）を満すとは限らない．満す場合は近似率 3/2 の多項式時間算法が知られる．

第 7 章

神託と相対化

或る問題 B を解くとわかっている算法を，その中身を気にせずに，別の計算手順の一部として利用することがある．計算量理論においても「仮に問題 B が解けるとして」話を進めると議論の助けになることがある．B の解決に係る困難はさて措き，それに加えて如何なる困難があるか，相対的に調べるのである．このために，動作中に B の答を必要に応じて尋ねる仕組をもつ神託機械を用いる．

7.1 神託機械とチューリング帰着

神託チューリング機械（以下単に**神託機械**と呼ぶ）とは，これまでのチューリング機械の機能に加え，計算中に神託と呼ばれる言語 B に質問をすることができるものである．このために「質問テープ」「回答テープ」があり，特別な状態として $q_{\text{質問前}}$ と $q_{\text{質問後}}$ とをもつ．状態が $q_{\text{質問前}}$ になると，その時に質問テープに書かれている文字列 x に対し，次の時刻には，状態が $q_{\text{質問後}}$ になるとともに，回答テープに $B(x)$ が書込まれる[*1]．つまり B を解くに要する時間や空間は考えず，質問すれば直ちに答が外から与えられるとするのである．神託機械 M に神託 B を与えたものを M^B で表す．

この M^B が言語 A を認識するとき，M を A から B への**チューリング帰着**（Turing reduction）と呼ぶ．特に M が多項式時間限定に取れるとき，A は B へ**多項式時間チューリング帰着**（**クック帰着**）するという．

「B を使うと A が解ける」という概念としては 5.1 節の多対一帰着もあったが，多対一帰着ではその「使い方」が，A の個例 x を B の個例 $T(x)$ に変換するという形に限られており，答 $B(T(x))$ には手を加えずそのまま A の答とした（図 5.1）．チューリング帰着はその制限を除いて計算中いつでも B の助けを求められるようにしたものである（図

[*1] B が約束つき問題（2.3.3 節）である場合，約束に属しない質問 x に対しては毎回出鱈目の答が返されるとし，神託機械はその結果によらず正しく振舞わねばならないとする．

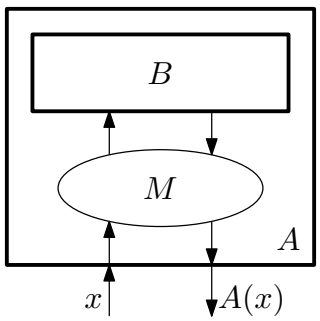


図 7.1　図 5.1 の多対一帰着とは異なり，チューリング帰着においては，神託機械 M が B に多くの質問をし，その結果を使って問題 A に答えることができる．

7.1)．従って当然，もし A が B へ多項式時間多対一帰着するならば，A は B へ多項式時間チューリング帰着する．

問 7.1　coNP の任意の言語は CSAT へ多項式時間チューリング帰着することを示せ．

7.2　相対化

以上では神託機械 M に神託 B を与えた M^B が言語 A を「認識する」ことを考えたが，認識の代りに 4 章の「R 認識」「N 認識」などを考えることもできる．級 P, NP, RP, BPP などの定義で使われた機械 M の代りに，神託機械に問題 B を神託として与えたもの M^B を使ったときに得られる級を，それぞれ P^B，NP^B，RP^B，BPP^B で表す．言語 A が B に多項式時間チューリング帰着するとは，この記法で書けば $A \in \mathsf{FP}^B$ ということになる．問 7.1 で述べたことは $\mathsf{coNP} \subseteq \mathsf{P}^{\mathrm{CSAT}}$ と書ける．

これまでの計算量級に関する議論の多くは，機械に任意の神託 B を附けてもそのまま成立つ．このように「神託 B つきで同じ議論を行う」ことを B による**相対化**(relativization)と呼ぶ．例えば図 4.3 の包含関係は各級に神託 B を与えても全く同じ議論により成立つし，CSAT が NP 完全ということも次のごとく相対化する．

回路やその一種である論理式においても，機械に神託 B を与えたのと同様に，言語 B を表す素子を使った計算を考えることができる．すなわち各 $l \in \mathbb{N}$ に対して l 入力 1 出力の素子 B_l があり，これを回路の中で自由に使ってよい．3.2.2 節の形式的な書き方でいえば，$\mathtt{x}_i = B(\mathtt{x}_{j_1}, \ldots, \mathtt{x}_{j_l})$ の形の命令 ($l \in \mathbb{N}$ は任意，$j_1, \cdots, j_l$ は i 未満の正整数) も許すということである．このように B を使った回路が与えられたとき，その充足可能性を判定する問題を CSAT^B と書くことにする．

この問題 CSAT^B は，B を神託として与えられれば容易に N 認識でき，したがって NP^B に属する．また CSAT が NP 完全であるという 5.2.2 節までの議論も相対化する．すなわ

ち CSAT^B は多項式時間多対一帰着 $\leq^{\mathsf{P}}_{\mathrm{m}}$ の下で NP^B 完全でもある（これは B の現れる回路が多対一帰着により作られるのであり，多対一帰着そのものの計算に神託 B を用いるのではない）．

13+1 **問 7.2** （約束なし）言語 $B \in \mathsf{NP} \cap \mathsf{coNP}$ について

（1）$\mathrm{CSAT}^B \in \mathsf{NP}$ を示し，これにより $\mathsf{NP}^B = \mathsf{NP}$ を示せ．

（2）$\mathsf{P}^B \subseteq \mathsf{NP} \cap \mathsf{coNP}$ を示せ．

5+3 **問 7.3** （1）次の約束つき言語（2.3.3 節と脚註*1）XSAT について $\mathrm{CSAT} \in \mathsf{P}^{\mathrm{XSAT}}$ を示せ．

XSAT の個例は二つの回路の組 (φ, ψ) であって丁度片方のみが充足可能であることが約束されている．これに対し，充足可能なのが φ であるか判定する．

（2）$\mathsf{NP} \neq \mathsf{coNP}$ とする．問 7.2 (2)の主張は，B を約束つき言語にしてしまうと成立たないことを示せ．

2+4+4 **問 7.4** 以上では A から B へのチューリング帰着と言うとき A と B は言語としていた．

A が言語でなく多価函数である場合にも全く同様に考えることができる．すなわち M^B が A を計算するとき，M を A から B への**チューリング帰着**と呼ぶ．例えば問 3.8 では，もし言語 CSAT が P に属せば函数 FIND-SAT が FP に属することを見たが，そのために FIND-SAT が CSAT へ多項式時間チューリング帰着することを示したのであった．

更に神託 B の方も言語に限らず，より一般に多価函数のうち多項式均衡なもの（つまり或る多項式 p が存在し任意の個例 x に対して $B[x]$ の元はみな長さが $p(|x|)$ 以下であるもの）に拡げて考えることができる．すなわち B への質問 x に対しては $B[x]$ の元が一つ返され，機械はいずれの答が選ばれるかによらず正しく振舞わねばならぬものとする[*2]．このように多価函数へのチューリング帰着を定義すると，例えば NP に属する任意の言語は FIND-SAT に多項式時間チューリング帰着する．このことを以て FIND-SAT を（それ自身が判定問題ではないので「NP 完全」とはいわれないが）「NP 困難」と言うことがある．他にもこれまでに見た NP 完全な言語に関連する多くの多価函数，例えば与えられたグラフの最大独立集合を求める多価函数や，与えられた正整数の集合を総和が等しい二集合に分割する方法を求める多価函数も，NP 困難であることがすぐ判る．

一方，与えられた正整数を素因数分解する問題 FACTOR は，FP に属するか不明だが，次の理由から，NP 困難というわけでもなさそうだと考えられている．

[*2] B を多項式均衡な問題に限ったのは，そうでない場合に同じ定義をそのまま採用してしまうと，神託が返す答を読む時間すらない虞があり，良い定義といえなくなるからである．本問の FIND-SAT や FACTOR は多項式均衡なので，神託機械の時間制限を十分大きな多項式にしておけばその心配はない．

(1) 与えられた正整数の組 (n, b) に対し，n に b 未満の素因数があるか判定する問題を FACTOR-BELOW とする．FACTOR が FACTOR-BELOW に多項式時間チューリング帰着することを示せ．

(2) FACTOR-BELOW $\in \mathsf{NP} \cap \mathsf{coNP}$ を示せ．問 4.6 の結果 PRIMES $\in \mathsf{NP}$ を用いよ．

(3) $\mathsf{NP} \neq \mathsf{coNP}$ とする．FACTOR が NP 困難でない（すなわち NP に属するが FACTOR に多項式時間チューリング帰着しない言語が存在する）ことを示せ．問 7.2 を用いよ．

次の問 7.5, 7.6 で見るように，$\mathsf{P}^B = \mathsf{NP}^B$ という主張は B によって成否が変る．故に $\mathsf{P} \neq \mathsf{NP}$ 予想は，「任意の神託で相対化できるような議論」によっては解決し得ない．

1+2**問 7.5**　B を PSPACE 完全な言語（例えば CQSAT）とする．$\mathsf{P}^B = \mathsf{NP}^B$ を示そう．

(1) $\mathsf{PSPACE} \subseteq \mathsf{P}^B$ を示せ．

(2) $\mathsf{NP}^B \subseteq \mathsf{PSPACE}$ を示せ．

1+6+4+4**問 7.6**　$\mathsf{P}^B \neq \mathsf{NP}^B$ なる言語 B の存在を示そう．

(1) 任意の約束つき言語 B に対し，次の判定問題 U_B が NP^B に属することを示せ．

U_B の個例は，文字列 $x \in \{0,1\}^*$ であり，それと同じ長さの $\{0,1\}$ 上の文字列はすべて $\mathrm{dom}\, B$ に属することが約束されている．この個例 x に対し，$y \in B$ なる $y \in \{0,1\}^{|x|}$ が存在するか判定する．

(2) （決定性）多項式時間神託機械 M を一つ取る．約束つき言語 B であって M^B が U_B を認識しない（すなわち U_B の或る個例 x に対する出力が $U_B(x)$ と異なる）ようなものを一つ作れ．

(3) (2)よりも少し強いことを示そう．約束つき言語 C（→ 2.3.3 節）が有限であるとは約束 $\mathrm{dom}\, C$ が有限であることをいう．また約束つき言語 B が C を含むとは，任意の $x \in \mathrm{dom}\, C$ について $x \in \mathrm{dom}\, B$ かつ $x \in B \iff x \in C$ なることをいう．任意の有限な約束つき言語 C と任意の多項式時間神託機械 M とに対して，C を含む有限な約束つき言語 B が存在し，M^B が U_B を解かないことを示せ．

(4) 多項式時間神託機械をすべて枚挙して $M_1, M_2, \cdots$ とし，これらについて順に(3)を繰返し用いることで，$U_B \notin \mathsf{P}^B$ なる（約束なし）言語 B を作れ．

7.3　多項式時間階層

以下では言語の集合 C について $\mathsf{P}^{\mathsf{C}} = \bigcup_{B \in \mathsf{C}} \mathsf{P}^B$ のように書く（NP^{C}，RP^{C} なども同様）．

$\Sigma_0^{\mathsf{p}} = \mathsf{P}$ とし各 $k \in \mathbb{N}$ で $\Sigma_{k+1}^{\mathsf{p}} = \mathsf{NP}^{\Sigma_k^{\mathsf{p}}}$ とする．例えば $\Sigma_3^{\mathsf{p}} = \mathsf{NP}^{\mathsf{NP}^{\mathsf{NP}}}$．列 $\Sigma_0^{\mathsf{p}} \subseteq \Sigma_1^{\mathsf{p}} \subseteq \Sigma_2^{\mathsf{p}} \subseteq \ldots$ を**多項式時間階層**(polynomial-time hierarchy)と呼ぶ．$\mathsf{PH} = \bigcup_{k\in\mathbb{N}} \Sigma_k^{\mathsf{p}}$ とする．

10+3**問 7.7** 任意の $k \in \mathbb{N}$ について，

(1) 言語 A が $\Sigma_{k+1}^{\mathsf{p}}$ に属するには，多項式 p と（約束なし）言語 $R \in \Sigma_k^{\mathsf{p}}$ とが存在し，A の任意の入力 u について次が成立つことが必要十分であることを示せ．

$$A(u) = 1 \iff \exists v \in \Sigma^{p(|u|)}.\, R(u, v) = 0$$

(2) 言語 A が Σ_k^{p} に属するには，多項式 p と（約束なし）言語 $R \in \mathsf{P}$ とが存在し，A の任意の入力 u について次が成立つことが必要十分であることを示せ．

$$A(u) = 1 \iff \exists v_1 \in \Sigma^{p(|u|)}.\, \forall v_2 \in \Sigma^{p(|u|)}.\, \exists v_3 \in \Sigma^{p(|u|)} \ldots Q v_k \in \Sigma^{p(|u|)}.\, R(u, v_1, \ldots, v_k) = 1$$

但し Q は k が奇数なら $\exists$，偶数なら $\forall$.

(1)のヒント $A \in \Sigma_{k+1}^{\mathsf{p}}$ ならば，或る $B \in \Sigma_k^{\mathsf{p}}$ が存在して $A \leq_{\mathrm{m}}^{\mathsf{P}} \mathrm{CSAT}^B$.

この(2)の条件は多項式空間限定の機械で確かめられる（3.4.2 の $\mathrm{CQSAT} \in \mathsf{PSPACE}$ と同様）から，（直接にも示せるが）$\mathsf{PH} \subseteq \mathsf{PSPACE}$ が成立つ．

多項式時間階層は $\mathsf{P} = \Sigma_0^{\mathsf{p}} \subsetneq \Sigma_1^{\mathsf{p}} \subsetneq \Sigma_2^{\mathsf{p}} \subsetneq \ldots$ のごとく各層みな相異なるというのが大方の予想である．もしこれに反して或る層 $k \in \mathbb{N}$ で $\Sigma_k^{\mathsf{p}} = \Sigma_{k+1}^{\mathsf{p}}$ ならば，定義より明らかに以降の層もみな等しく $\Sigma_k^{\mathsf{p}} = \mathsf{PH}$ となる．これを多項式時間階層が「潰れる」という．もし $\mathsf{P} = \mathsf{NP}$ なら勿論すべて潰れるが，そうでなく或る層より上のみ潰れる可能性もある．

7**問 7.8** もし $\mathsf{PH} = \mathsf{PSPACE}$ ならば多項式時間階層が（或る層で）潰れることを示せ．

ヒント PSPACE 完全問題をとる．$\mathsf{PH} = \mathsf{PSPACE}$ ならばそれは或る Σ_k^{p} に属する．

11**問 7.9** $\mathsf{BPP}^{\mathsf{BPP}} = \mathsf{BPP}$ を示せ[*3].

10+6+2**問 7.10** (1) 文字列 $x,\ y \in \{0,1\}^k$ のビット毎の排他論理和を $x \oplus y \in \{0,1\}^k$ で表し，集合 $X,\ Y \subseteq \{0,1\}^k$ に対しては $X \oplus Y = \{\, x \oplus y : x \in X,\ y \in Y \,\}$ とする．

多項式 $p,\ q$ を適当に定めると，任意の $k \in \mathbb{N}$ と集合 $Y \subseteq \{0,1\}^k$ とについて次が成立つことを示せ．組 $s = (s_1, \ldots, s_{p(k)}) \in (\{0,1\}^k)^{p(k)}$ に対して $f(Y, s) = |Y \oplus \{s_1, \ldots, s_{p(k)}\}|/2^k$ と書くことにすると，

- $|Y|/2^k \geq 1 - 1/q(k)$ ならば，過半数の $s \in (\{0,1\}^k)^{p(k)}$ について $f(Y, s) = 1$
- $|Y|/2^k \leq 1/q(k)$ ならば，すべての $s \in (\{0,1\}^k)^{p(k)}$ について $f(Y, s) < 1/2$

[*3] 問 4.8 のヒントのように BPP の誤り確率は著しく減らすことができる．問 7.9, 7.10 では必要ならばこのことを用いてよい．

(2) (1)を用いて $\mathsf{RP}^{\widehat{\mathsf{RP}}} = \mathsf{BPP}$ を示せ．但し $\widehat{\mathsf{RP}}$ は多項式時間限定の機械で R 認識される約束つき言語の級である (2.3.3 節，4.2.1 節末)．

(3) (2)を用いて $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{p}}$ を示せ．

A

記法と用語

A.1 漸近記法

函数の増大する速さを大まかに表すために次の記法を使う．自然数を非負実数に移す函数 f, $g\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ について,

(1) $f(n) = \mathrm{O}(g(n))$ とは，或る $c \in \mathbb{R}_{\geq 0}$ が存在し，十分大きなすべての $n \in \mathbb{N}$ について $f(n) \leq c \cdot g(n)$ が成立つことをいう．

(2) $f(n) = \Omega(g(n))$ とは，$g(n) = \mathrm{O}(f(n))$ であることをいう．

(3) $f(n) = \Theta(g(n))$ とは，$f(n) = \mathrm{O}(g(n))$ かつ $f(n) = \Omega(g(n))$ が成立つことをいう．

つまり $f(n) = \mathrm{O}(g(n))$ は「f の値は定数倍を無視すれば概ね g 以下である」を表す．逆に Ω は「…以上である」を表す．便利な書き方だが,「存在」「すべて」のやや混み入った意味を式の中に隠す上に，二つの物が等しいわけでもないのに等号を使うという聊か怪しい記法であるから，誤解を与えぬよう注意して用いるべきである．この演習では式の右辺でのみ (或いは「…の値は $\mathrm{O}(\log n)$ である」のごとく文の述語として) 使うことにする．

この記法を $f(n) = n^3 - \mathrm{O}(n^2)$ のごとく式の中に入れて使うこともある．これは或る $c \in \mathbb{R}$ が存在し，十分大きなすべての $n \in \mathbb{N}$ で $f(n) \geq n^3 - c \cdot n^2$ という意味になる．つまり O が右辺になるように移項した $n^3 - f(n) = \mathrm{O}(n^2)$ として解釈すればよいのである．

また $f(m,n) = \mathrm{O}(g(m,n))$ のごとく複数の変数をもつ式にも使う．これは或る $c \in \mathbb{R}$ が存在し，十分大きなすべての m, $n \in \mathbb{N}$ で $f(m,n) \leq c \cdot g(m,n)$ が成立つことを意味する．この「十分大きなすべての m, n で」とは「或る m_0，n_0 が存在し，$m \geq m_0$ かつ $n \geq n_0$ なるすべての (m,n) について」ということである[*1].

5**問 A.1** それぞれ，成立つなら○を，成立たないなら×を記せ．理由は不要.

[*1] なお一変数 n のみに使うのであれば，$f(n) = \mathrm{O}(g(n))$ の定義中の「十分大きなすべての $n \in \mathbb{N}$ について $f(n) \leq c \cdot g(n)$」の代りに「すべての $n \in \mathbb{N}$ について $f(n) \leq c \cdot g(n) + c$」としても意味は変らない.

（イ）$n^2 = \mathrm{O}(n)$
（ロ）$n^2 = \Omega(n)$
（ハ）$5n + 3 = \mathrm{O}(n)$
（ニ）$2^{5n+3} = \mathrm{O}(2^n)$
（ホ）$2n = \Theta(3n)$
（ヘ）$2^n = \Theta(3^n)$

以下では便宜上 $\log_2 0 = \log_7 0 = 0$ とする．

（ト）$\log_2 n = \Theta(\log_7 n)$

以下では更に log は $\log_2$ を意味するものとする．

（チ）$\log(n^2) = \mathrm{O}(\log n)$
（リ）$(\log n)^2 = \mathrm{O}(\log n)$
（ヌ）$n(\log n)^2 = \mathrm{O}(n\sqrt{n})$
（ル）$2^{n \log\log n} = \mathrm{O}(n^n)$
（ヲ）$n! = \mathrm{O}(10^n)$

2+2**問 A.2** （1）函数 f, g, $h\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ について，もし $f(n) = \mathrm{O}(g(n))$ かつ $g(n) = \mathrm{O}(h(n))$ ならば必ず $f(n) = \mathrm{O}(h(n))$ が成立つといえるか．

（2）常に正の値をとる任意の函数 f, $g\colon \mathbb{N} \to \mathbb{R}_{>0}$ について，$f(n) = \mathrm{O}(g(n))$ または $f(n) = \Omega(g(n))$ の少くとも一方は必ず成立つといえるか．

1+4+2+4**問 A.3** 正の定数 $p < 1$ と $\delta > 0$ を固定する．確率 p で当る籤（くじ）を独立に l 回引くとき，当りの出る回数 X の期待値は勿論 pl であるが，これよりも割合 δ だけ多めに当る確率を考えよう．つまり $X \geq pl(1+\delta)$ となる確率を $s(l)$ とする．

（1）このとき $s(l) = 2^{-\Omega(l)}$ が成立つ．この主張を漸近記法を使わずに書け．

以下(1)の主張を示そう．

（2）$(1+\delta)^X$ の期待値を求めよ．

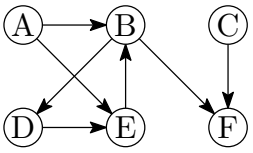
（3）$s(l) \cdot (1+\delta)^{pl(1+\delta)}$ は(2)の値以下である．何故か．

（4）$s(l) = 2^{-\Omega(l)}$ を示せ．

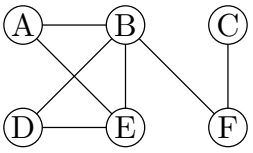
A.2 文字列と言語

この講義では情報処理を文字列に対する操作として扱うので，そのための用語や記法を用意しておく．

字母(alphabet)とは空でない有限集合であり，その元を**文字**(letter)或いは**記号**(symbol)と呼ぶ．字母 Σ に属する文字を重複を許して有限個並べた列 $x = x_1 x_2 \ldots x_n$ を Σ



A.1.1 有向グラフ.



A.1.2 無向グラフ.

図 A.1 グラフの例.

上の**文字列**(string)或いは**語**(word)と呼ぶ．例えば字母 $\{0,1,2,3,4,5,6,7,8,9\}$ 上の文字列には 2 や 39 や 1517 などがある．並べた文字の個数 n を x の**長さ**(length)といい $|x|$ で表す．長さ 0 の文字列を ε で表す．Σ 上の長さ n の文字列全体を Σ^n で表し，文字列全体を Σ^* で表す．例えば字母 $\Sigma = \{0,1\}$ 上の文字列全体は

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$$

である．また文字列 $001 \in \Sigma^3 \subseteq \Sigma^*$ の長さは $|001| = 3$ である．

文字列 $x = x_1 \cdots x_m$ の後に続けて文字列 $y = y_1 \cdots y_n$ を並べて得られる文字列 $x_1 \cdots x_m y_1 \cdots y_n$ を xy で表す．また文字列 x を n 回書き並べた文字列

$$\underbrace{xx \cdots x}_{n}$$

を x^n で表す.

文字列からなる集合，つまり Σ^* の部分集合を，(Σ 上の) **言語**(language)と呼ぶ．例えば $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ として，

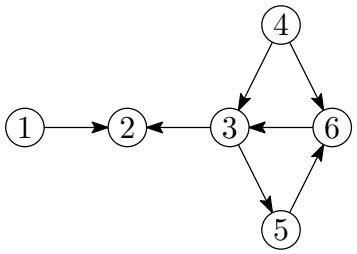
$$\begin{aligned}\text{PRIMES} &= \{\, x \in \Sigma^* : x \text{ は或る素数を十進法で書いたものである} \,\} \\ &= \{2, 3, 5, 7, 11, 13, \ldots\}\end{aligned}$$

は Σ 上の言語である．A に属しない文字列の全体 $\Sigma^* \setminus A$ を，言語 A の**補言語**(complement)と呼び，$\overline{A}$ で表す．例えば $\overline{\text{PRIMES}}$ は，文字列 ε と，文字列 1 と，文字 0 で始まるすべての文字列と，合成数を十進法で書いたすべての文字列とからなる言語である．

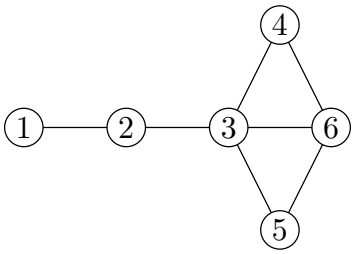
字母 Σ は通常，一続きの議論の中で必要な文字を含む集合に固定して考えればよく，その詳細については支障のない限り一々指定せずに話を進めることがある (2.2.2 節).

A.3 グラフ

グラフ(graph，辺点図)は，図 A.1 のように有限個の物の間の繋がりを表したものである．結ばれている物を**頂点**(vertex)または**点**と呼び，繋がりを**辺**(edge)と呼ぶ．辺は二つの頂点を結ぶが，辺に向きがあるとする場合とないとする場合があり，それぞれ有向



A.2.1 図 A.1.1 に同型な有向グラフ.



A.2.2 図 A.1.2 に同型な無向グラフ.

図 A.2 これらのグラフは，図 A.1 の頂点 A, B, C, D, E, F をそれぞれ 4, 3, 1, 5, 6, 2 に呼び換えた（そして異なる配置の図で表した）に過ぎないから，もとのグラフに同型である.

(directed)グラフ，無向(undirected)グラフという．つまり頂点 u と v とを結ぶもの uv が辺だが，有向グラフでは uv（u から v への辺）と vu（v から u への辺）とを区別し，無向グラフでは区別しない．無向グラフでは，この辺 uv があるとき u と v は**隣接**(adjacent)するという．なお本演習では u から v への辺（無向グラフでは，u と v を結ぶ辺）が複数あることはなく，従って uv といえば辺は高々一本に決るものとする（有向グラフが辺 uv と辺 vu の両方をもつことは許す）．また両端点を同じくする「uu」のごとき辺もないものとする．頂点の集合 V と辺の集合 E とを決めることでグラフ (V, E) は指定される．無向グラフにおいて頂点 u と他の頂点を結ぶ辺の本数を u の**次数**(degree)と呼ぶ．有向グラフでは，頂点 u から他の頂点への辺の本数を u の**出次数**(いで)，他の頂点から u への辺の本数を u の**入次数**(いり)と呼ぶ.

1+1+1**問 A.4** 次のような無向グラフは，それぞれ存在するか.

(1) 頂点が 4 つで，それらの次数が 3, 3, 2, 2 であるグラフ.

(2) 頂点が 5 つで，それらの次数が 3, 3, 3, 2, 2 であるグラフ.

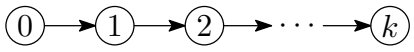
(3) 頂点が 6 つで，それらの次数が 4, 4, 1, 1, 1, 1 であるグラフ.

図 A.1.1 と図 A.2.1 のグラフは頂点の名づけ方が異なるだけであり，適切に呼び換えれば繋がりの形は同じである．このようなとき両者は同型であるという．図 A.1.2 と図 A.2.2 も同様である．つまり二つの有向グラフないし二つの無向グラフ (V_1, E_1), (V_2, E_2) が**同型**(isomorphic)であるとは，或る全単射 $\varphi\colon V_1 \to V_2$ が存在し，任意の $u, v \in V$ について

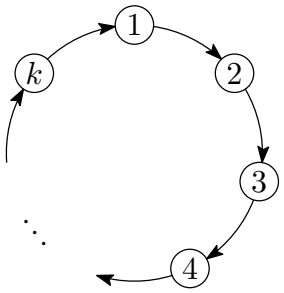
$$uv \in E_1 \iff \varphi(u)\varphi(v) \in E_2$$

が成立つことをいう.

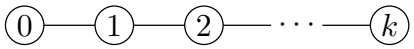
2+2**問 A.5** (1) 頂点が 4 つで，その次数が 3, 3, 2, 2 であるような無向グラフが二つある．これらは必ず同型であるといえるか.



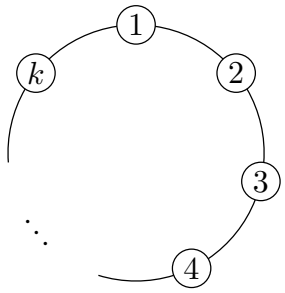
A.3.1 長さ k の路 (有向).



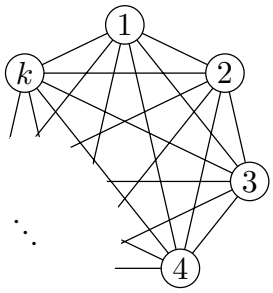
A.3.2 長さ k (≥ 2) の閉路 (有向).



A.3.3 長さ k の路 (無向).



A.3.4 長さ k (> 2) の閉路 (無向).



A.3.5 大きさ k の閥. どの相異なる二頂点間にも辺がある.

図 A.3 路, 閉路, 閥.

(2) 頂点の次数を大きい順に並べた列が全く同じであるような無向グラフが二つある. これらは必ず同型であるといえるか.

図 A.3.1, A.3.2 のものに同型な有向グラフをそれぞれ**路**(path), **閉路**(cycle)と呼ぶ. 図 A.3.3, A.3.4, A.3.5 のものに同型な無向グラフをそれぞれ**路**, **閉路**, **閥**(clique)と呼ぶ. 大きさ k の閥は, k 個の頂点のうち任意の二つが辺で結ばれ, 従って $k(k-1)/2$ 本の辺をもつ無向グラフである.

グラフ (V', E') が (V, E) の**部分グラフ**(subgraph)であるとは, $V' \subseteq V$ かつ $E' \subseteq E$ であることをいう.「グラフ G の部分グラフであって○○であるもの」を単に「G の○○」と呼ぶことがある. 例えば有向グラフ G の閉路と言えば, G において二本以上の辺を順に辿って戻って来る経路であって途中で同じ頂点を二度通らないもののことである. 図 A.1.1 や図 A.2.1 のグラフには長さ 3 の閉路がそれぞれ一つだけある. G の路において図 A.3.1 や図 A.3.3 の頂点 0, k に対応する頂点がそれぞれ u, v であるとき, この路を u から v への路という.

閉路を含まない有向グラフを (英語で directed acyclic graph であることから) **ダグ**(dag)という.

無向グラフが**連結**(connected)であるとは, どの二頂点の間にも路があることをいう.

閉路を含まない無向グラフを**森**(forest)という．連結な森を**木**(tree)という．森（や木）において次数が 1 である頂点を**葉**という．

グラフの計算の対象として扱うには，グラフを文字列で表す（そして機械に入力する）必要があるが (2.2.2 節)，そのときには頂点を番号で呼び，辺をその端点の番号で表すこととし，頂点数を指定した上で集合 E の元をすべて書き並べることにしておけばよい．これによりグラフを $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ,\}$ 上の文字列で表すことができる．例えば図 A.2.1 のグラフは，

```
6,1,2,3,2,3,5,4,3,4,6,5,6,6,3
```

と表される（頂点が 6 つであり，辺 12, 32, 35, 43, 46, 56, 63 があるため）．

A.4 論理式と回路

入力数 k の**命題論理式**(propositional formula)或いは単に**論理式**(formula)とは，**命題変数**と呼ばれる記号 $\mathtt{x}_1, \mathtt{x}_2, \cdots, \mathtt{x}_k$ を**論理結合子**（**素子**）$\wedge$, $\vee$, $\neg$ で適切に繋いだものをいう．例えば

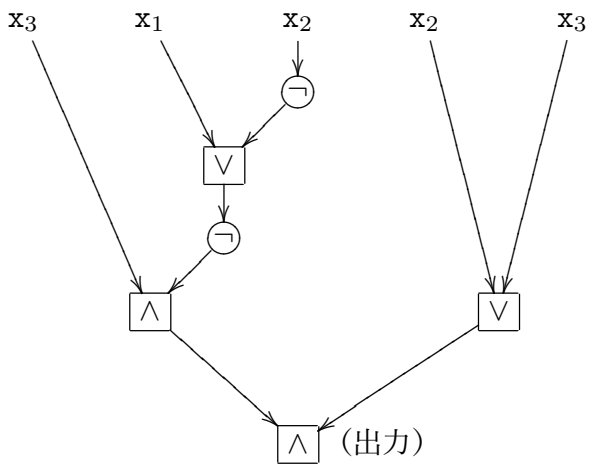
$$\mathtt{x}_3 \wedge \neg(\mathtt{x}_1 \vee \neg\mathtt{x}_2) \wedge (\mathtt{x}_2 \vee \mathtt{x}_3)$$

は論理式である．形式的にいうと論理式の全体は次により定まる．

- 各 $i \in \mathbb{N}$ について，$\mathtt{x}_i$ は論理式である．
- F と G が論理式であるとき $(F \wedge G)$ もまた論理式である．
- F と G が論理式であるとき $(F \vee G)$ もまた論理式である．
- F が論理式であるとき $\neg F$ もまた論理式である．

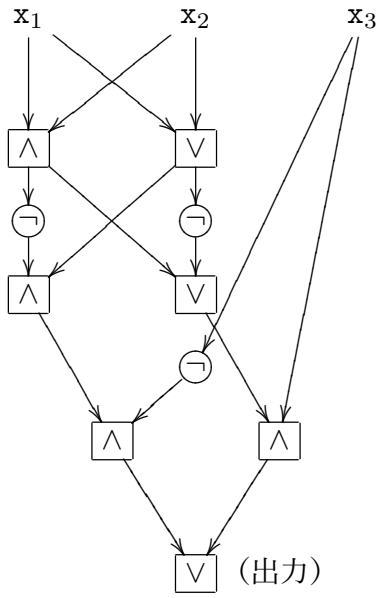
但し誤解の生じない限り括弧を略することがある．例えば規則に従って作った論理式 $((\mathtt{x}_3 \wedge \neg(\mathtt{x}_1 \vee \neg\mathtt{x}_2)) \wedge (\mathtt{x}_2 \vee \mathtt{x}_3))$ を略して上に書いたように表す．論理式はこの構成法に沿って作られた有向グラフ，例えば今挙げた論理式については図 A.4 左にあるものを表すと考えてもよい．

変数 $\mathtt{x}_1, \mathtt{x}_2, \cdots, \mathtt{x}_k$ にそれぞれ真（1 で表す）か偽（0 で表す）を割当て，$\wedge$ を且ツ，$\vee$ を又ハ，$\neg$ を〜ニ非ズの意味で読むと，全体の真理値が定まる．例えば $(\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3)$ に対する $(0, 1, 1)$ という**割当**(assignment)の下では，上に挙げた論理式の値は 1 となる．このように割当が論理式の値を真にすることを，その論理式を**充足**(satisfy)するともいう．各割当についてこのように真理値が決るから，入力数 k の論理式は $\{0, 1\}^k$ から $\{0, 1\}$ への函数を実現していることになる．例えば上に挙げた入力数 3 の論理式は図 A.4 右に示す函数を実現する．この論理式を充足する割当は $(0, 1, 1)$ のみであったことが判る．充足する割当が一つ以上存在するとき，その論理式は**充足可能**(satisfiable)であるという．



$\mathtt{x}_1$	$\mathtt{x}_2$	$\mathtt{x}_3$	$\varphi(\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

図 A.4　入力数 3 の論理式 φ（左）と，それが実現する函数（右）．



$\mathtt{x}_1$	$\mathtt{x}_2$	$\mathtt{x}_3$	$C(\mathtt{x}_1, \mathtt{x}_2, \mathtt{x}_3)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

図 A.5　入力数 3，出力数 1 の回路 C（左）と，それが実現する函数（右）．

回路は論理式を一般化し，或る頂点までで得た途中結果を，一つではなく複数の素子に渡せるようにしたものである．すなわち入力数 k，出力数 l の**回路**(circuit)とは，各頂点に次を満すように命題変数や論理結合子の書かれたダグ（→ A.3 節）をいう（図 A.5 左）．

- 入次数 0 の頂点には変数 $\mathtt{x}_1, \cdots, \mathtt{x}_k$ のいずれかが書かれている．
- 入次数 1 の頂点には ¬ の素子が置かれている．
- その他の頂点は入次数が 2 であり，∧ か ∨ の素子が置かれている．
- 頂点のうち l 個が出力場所として（順序つきで）指定されている．

この回路は入力変数 $\mathtt{x}_1, \cdots, \mathtt{x}_k$ への割当を与えると出力として l 個の真理値が決るから，$\{0,1\}^k$ から $\{0,1\}^l$ への或る函数を実現している（図 A.5 右）．出力数 $l = 1$ の回路 C に

ついては論理式と同様に，割当 $b \in \{0,1\}^k$ が C の出力の値を 1 にすることを b は C を**充足**するといい，そのような充足割当 b が存在するとき C は**充足可能**であるという．

問 A.6 図 A.5 左の回路と同じ函数を（つまり図 A.5 右の表に示された函数を）実現する論理式を作れ．

このように論理式でも $\{0,1\}^k$ から $\{0,1\}$ への任意の函数を実現できるが，そのためには回路よりも多くの素子が必要になる場合がある．

回路そのものを計算の対象として機械に入出力することもある．そのためには回路を文字列で表さねばならないが，その際は例えば次のような命令の列として書けばよい．

- $i = 1, \cdots, k$ については，i 番目の命令は $\mathtt{x}_i := \mathtt{input}$ である．
- $i = k+1, k+2, \cdots, n$ については，i 番目の命令は $\mathtt{x}_i := a(b\mathtt{x}_j \wedge c\mathtt{x}_k)$ の形である．但し j, k は i 未満の正整数であり，a, b, c の位置には $\neg$ を置くか否か選べる（$\vee$ 素子は $\wedge$ と $\neg$ を組合せて表すことができる）．最後から l 個の変数 $\mathtt{x}_{n-l+1}, \cdots, \mathtt{x}_n$ が出力となる．

例えば図 A.5 の回路は $\wedge$ と $\vee$ の素子を左上から順に $\mathtt{x}_4, \mathtt{x}_5, \cdots$ と名づけて $\mathtt{x}_1 := \mathtt{input}$, $\mathtt{x}_2 := \mathtt{input}$, $\mathtt{x}_3 := \mathtt{input}$, $\mathtt{x}_4 := \mathtt{x}_1 \wedge \mathtt{x}_2$, $\mathtt{x}_5 := \neg(\neg\mathtt{x}_1 \wedge \neg\mathtt{x}_2)$, $\mathtt{x}_6 := \neg\mathtt{x}_4 \wedge \mathtt{x}_5$, $\mathtt{x}_7 := \neg(\neg\mathtt{x}_4 \wedge \mathtt{x}_5)$, $\mathtt{x}_8 := \mathtt{x}_6 \wedge \neg\mathtt{x}_3$, $\mathtt{x}_9 := \mathtt{x}_7 \wedge \mathtt{x}_3$, $\mathtt{x}_{10} := \neg(\neg\mathtt{x}_8 \wedge \neg\mathtt{x}_9)$ と表される．