計算量理論入門——完全問題と「複雑さ」

河村彰星

第一日　計算可能性

■問題と機械　この講義の前半では主に，入力された文字列のうち何らかの条件に適うもののみを受理したいという形をした問題を扱います．つまり問題とは文字列の集合（計算理論では**言語**とも呼びます）に他なりません．例えば十進法で素数を表す文字列 2, 3, 5, 7, 11, …からなる集合を PRIME としましょう．0, 1, …, 9 からなる与えられた文字列が言語 PRIME に属するか答えよ（素数性の判定）という問が，ここで扱う問題の一例です．もう一つ例を挙げると，a, b からなる文字列のうち連続する三文字として aba を含むものの全体 ABA は一つの言語です．こちらは PRIME よりも簡単に判断できそうな感じがします．

実際，与えられた文字列が ABA に属するか知るには，「aba」がどこまで出たか注視しながら読み進めればよい．その手順は図 1 で表されます．図に描かれた四つの場所 q_0，q_1，q_2，q_3 を**状態**といいますが，どの状態からも a と b の矢印が出ておりますので（q_3 から出る「a, b」は二本の矢印をまとめて記したものです），「始」と書かれた状態 q_0 から始め，入力された文字列を読み進めつつ，各時点の文字に従って矢印を辿ることにします．例えば入力 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 と遷移し，二重丸のついた状態 q_3 に至ります．

この図 1 のような仕掛を**有限状態機械**といい，機械が文字列を読んだとき二重丸の状態

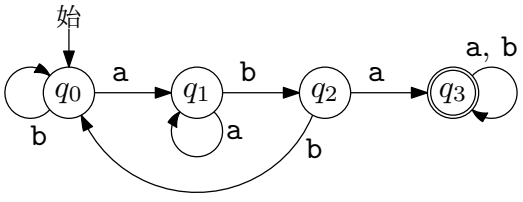

図 1　言語 ABA を認識する有限状態機械．

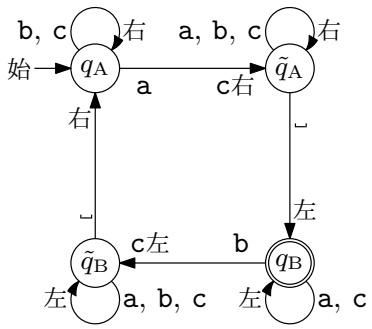

図 2　MORE-A を認識するチューリング機械．a を一つ c に書換えながら右端まで進むことと，b を一つ c に書換えながら左端まで進むことを繰返し，a か b の一方が尽きたと判ると停止する．

で終ることを（その文字列を）**受理**するといいます．図 1 をよく見ると，この機械が aba を含む文字列を受理し，含まない文字列は受理しないことが納得されましょう．これを，この機械が言語 ABA を**認識**すると呼ぶことにします．

これは言語 ABA が十分に「簡単」だからこそ可能なことでした．有限状態機械によって認識される言語の全体を REG で表すことにしましょう．REG に属するのは，言語のうち ABA のようにごく単純なものだけです．例えば a と b からなる文字列のうち a が b よりも多く現れるものからなる言語 MORE-A は，REG に属しないことが比較的容易に証明できます．有限個の状態では，a が b よりもどれだけ多いか正しく覚えられないのです．

■チューリング機械　これではあまりに非力ですから，もっと強力な計算の仕組を考えましょう．計算中に入力を読むだけでなく，自ら文字を書込む機能を加えた図 2 の機械は，MORE-A を認識します．先程は入力を右へ読み進めるだけでしたが，今度は（同じく左端の文字から始めますが）右にも左にも動くことができ，次に動く向きが矢の先に記されています（図 1 をこの書き方に合せるなら，すべての矢に「右」と書くことになります）．さらに例えば「c 右」は，現在の位置にある文字を c に書換えてから右へ進むことを表します（この c のように入力にない文字も使えます）．今度は一読みして終りではないので，入力文字列は左右に無限に長いテープ上に書かれており，何も書かれていない欄には ␣（空白文字）が置かれていると考えることにします．図 2 では各状態から文字 a, b, c, ␣ を読んだときの動作を表す矢印が出ていますが，足りない所もあり，その場合（例えば状態 q_{B} で ␣ を読んだ場合）には計算が停止（終了）します．二重丸の状態で停止すれば受理です．停止せずいつまでも動き続ける場合（図 2 では起りませんが，あり得ることです）は受理していないとすることにしましょう．

この機械の仕組は 1930 年代にイギリスの数学者チューリング(A. Turing)が人の計算

する様子を模して考案したものです．**チューリング機械**は，書込み機能のない機械に比べ，言語を認識する能力が高いと判りました．すなわち，チューリング機械によって認識される言語の全体を CE と書くと，REG $\subsetneq$ CE が成立ちます．このように問題の複雑さを，それがどのような機械で解けるかによって分類してゆくことが本講義のテーマです．

■機械＝算法　チューリング機械は高い能力をもち，あらゆる計算手順（算法（アルゴリズム））を表せると言われています．文字を一つ一つ読み書きするだけの簡素な機構でありながら，情報処理の本質をとらえるに十分なほど強力なのです．計算と呼べそうな仕組は他にもいろいろに考えられ，実際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**や**ラムダ計算**など，数や式に対する操作を抽象化した概念が登場しましたが，どうやらいかなる仕掛をこしらえてもチューリング機械と等価になる（その計算機構で認識できる言語は結局 CE に属する）らしい．およそ機械的な処理手順は皆チューリング機械（やそれと等価な仕組）で書ける，というこの原理は**チャーチとチューリングの定立**（テーゼ）と呼ばれます．勿論「機械的な処理」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代物ではありませんが，広く受入れられております．このためチューリング機械そのものの定義の詳細は今後さほど重要ではありません．図 2 のごとく機械を正確に書いても読みにくいので，これからは手順が解りやすいように言葉で説明して計算法を述べることにします．

例えば素数（を表す文字列）の言語 PRIME を認識する機械はどうでしょうか．与えられた正整数 X が素数か知りたければ，$Y = 2, 3, \cdots, X-1$ 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 X から少し離れた場所）に整数 Y を置くことにし，それを「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ります．後者の割り算の中では引き算を使いますが，それも機械の規則として書く．このように計算手順を部品ごとに組立ててゆくと，PRIME が CE に属することが確かめられます．

素数判定では Y を $X-1$ まで調べれば終りですが，必ずしもこのようにあらかじめ終りの見える計算ばかりとは限りません．例えば次の問題 SR を考えます．入力として a と b からなる奇数個の文字列（を「,」で区切って並べたもの）$u_1,v_1,u_2,v_2,\ldots,u_m,v_m,w$ が与えられます．これは各 $i = 1, 2, \cdots, m$ について「文字列中に現れる u_i を一つ v_i に書換える」という操作ができることを表します．文字列 w をうまく書換えて空文字列（長さ 0 の文字列）に到達できるか問う問題が SR です（どの操作を何度使ってもよい）．例えば aa,bbb,aba,a,bb,,aababab は，次の書換え列が存在するため，SR に属します．

$$\texttt{aababab} \xrightarrow[\texttt{a に}]{\texttt{aba を}} \texttt{aabab} \xrightarrow[\texttt{a に}]{\texttt{aba を}} \texttt{aab} \xrightarrow[\texttt{bbb に}]{\texttt{aa を}} \texttt{bbbb} \xrightarrow[\text{空文字列に}]{\texttt{bb を}} \texttt{bb} \xrightarrow[\text{空文字列に}]{\texttt{bb を}} \text{（空文字列）}$$

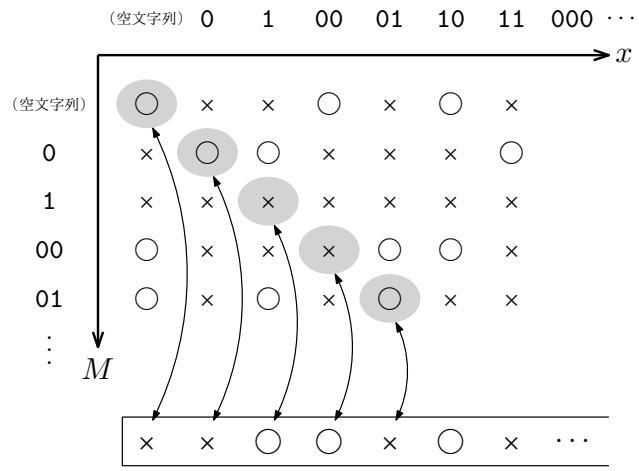

図 3　(M, x) が EVAL に属するか否かを記した表（機械 M が入力 x を受理するとき○，しないとき×）．対角線上の成分を反転して定めた下段の行に一致する行は存在しない．

この問題 SR は，ABA や MORE-A よりは難しそうですが，やはり CE に属します．もし空文字列へ到達可能なら，操作の施し方をすべて（見落しのないように然るべき順番で）実際に試してゆけばいずれ見つかるからです．このやり方では，SR に属しない入力が与えられた場合いつ諦めてよいやら知れず延々と動き続けることになりますが，それで構いません．言語を認識するというのは，その言語に属する文字列が入力されたら必ず受理すべし，しかし属しない文字列なら停止せずとも良い（誤って受理するのは不可）という要求だったからです．なお必ず計算を終えて結論を下すことまで要求する場合は認識ではなく**判定**という言葉を使います（認識のことを半判定ともいいます）．

■万能性と認識不能問題　かようにさまざまな計算を表せるチューリング機械ですが，何でも認識できるわけではありません．CE に属しない言語の例は次のように作れます．まず機械とは図 2 のような有限の設計図ですから文字列で表せます．ここでは 0 と 1 からなる文字列を入力とするチューリング機械を考えることとし，そのような機械をやはり 0 と 1 からなる文字列で表す方法を一つ定めておきます．さて，文字列を漏れなく一列に並べて縦軸と横軸に排した表を考え（図 3），縦軸の M と横軸の x にあたる成分には，文字列 M（が表す機械）が文字列 x を受理するか否かを○×で書込みます．つまり M の行にある○×は機械 M の挙動（どの文字列を受理するか）を表します．ここで表の対角線上の成分の○×を反転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ません．すなわちこの行で○のついた文字列からなる言語は，どの機械にも認識されないと判りました．

この証明（**対角線論法**と呼ばれます）で用いたのは，機械を有限の文字列で書けること

だけでした．そもそも言語を認識するとは，無数にあり得る入力すべてに有限の手立てで正解しようとする企てであり，それにはどうしても限界があるというわけです．

しかし機械が文字列で表せるのは便利なことでもあります．これまで「MORE-A を認識する機械」「PRIME を認識する機械」のように特定の問題を解く機械を考えてきましたが，実際の計算機（コンピュータ）は一台でさまざまの用を足します．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，問題に合せた算譜（プログラム）を，汎用・万能の計算機に読ますことによって実現されます．これは理論的には，言語

$$\mathrm{EVAL} = \{\, (M, x) \mid \text{機械 } M \text{ は文字列 } x \text{ を受理する} \,\}$$

を認識する機械が存在する（EVAL が CE に属する）お蔭といえます．この機械は，二つの文字列 M，x（を適切に一つの文字列 (M, x) に符号化したもの）を受取ると，あたかも機械 M に入力 x を与えたかのごとく振舞うわけですから，まさに**万能機械**の役割を果しています．機械を表す文字列を解釈・実行すること自体が，チューリング機械で書ける機械的な作業なのです．

EVAL は CE に属しますが，先程の対角線論法により，その補集合 $\overline{\mathrm{EVAL}}$ は CE に属しません．また，機械というのは局所的に文字の書換えを行っているだけですから，その動きを問う問題 EVAL は文字列の書換え問題 SR に似ており，この類似に着目すると $\overline{\mathrm{SR}}$ も CE に属しないことが判ります．先程 SR を認識した方法では，SR を判定する（入力された文字列が SR に属しないことも確実に断定する）ことはできていないと申しましたが，それは計算法をいかに工夫しようとも不可能なのです．

第二日　計算量の級

■時間と空間の制限　現実の計算機の能力がチューリング機械でとらえられるというチャーチとチューリングの定立は，計算量（計算にかかる時間や，計算に用いる記憶（メモリ）の量）に関しても成立ちます．つまり，現実に短時間で実行できる計算法は，停止までに行う遷移の回数が少ない機械で表されるし，小さな記憶空間で実行できる計算法は，訪れるテープ上の欄の個数が少ない機械で表される（ことが経験的に知られている）のです．これらの量が機械の仕組の詳細にあまり依存しないという事実は，時間や空間の制限が，実際上の関心事であるばかりでなく，問題に内在する複雑さを測る本質的な尺度であることを示唆します．計算量の重要性は世の中で計算機が広く使われ始めた 1950〜60 年代に明らかになり，以来さかんに研究されるようになりました．

一日目に考えた，書込み機能のない有限状態機械は，チューリング機械に対し，入力を

入力の長さ $n=$	10	30	50	100	1000	10000	
n	1 秒以内	1 秒以内	1 秒以内	1 秒以内	1 秒以内	1 秒以内	多項式時間
n^2	1 秒以内	1 秒以内	1 秒以内	1 秒以内	1 秒以内	1 秒	
n^3	1 秒以内	1 秒以内	1 秒以内	1 秒以内	10 秒	2.8 時間	
n^5	1 秒以内	1 秒以内	1 秒以内	1.7 分	116 日	3 万年	
2^n	1 秒以内	11 秒	130 日	4 百兆年			
$n!$	1 秒以内	8 京年					

図 4　一秒に 10^8 回（一億回）の処理ができる計算機で，長さ n の入力に対して n 回，n^2 回，n^3 回，n^5 回，2^n 回，$n!$ 回の処理を行うのにかかる時間．多項式時間でない計算法は，n が大きくなるにつれて急速に所要時間が増え，実行することがほぼ不可能になる．

ひと通り読むだけの時間しか許さない，あるいは記憶空間の使用を全く許さないという，厳しい制限を課したものといえます．計算量理論では主に，そこまで強くはない一定の制限の下で何が計算できるか，つまり REG と CE の間がどうなっているかを扱います．

■多項式時間と多項式空間　計算量を測るときは，入力文字列 x の大きさ（長さ）$|x|$ に応じてどう増大するかに着目します（図 4）．なかんづく計算効率の最も重要な分れ目と考えられているのは，時間が入力長の多項式に収まる（すなわち多項式 p が存在し，任意の入力 x に対して機械が $p(|x|)$ 回以内の遷移で停止する）か否かです．多項式時間のチューリング機械によって認識される言語全体を P で表します．また遷移の回数の代りにテープ上で訪れた欄の個数を測ることにより，同様に多項式空間で認識できる言語の集合 PSPACE が定義されます．P や PSPACE のように一定の制限下で解ける問題を集めた集合を**計算量級**（クラス）と呼びます．

P に属するのは，入力長と同程度の手間で解ける容易な問題です（$|x|^3$ や $|x|^{100}$ の手間を「同程度」と呼ぶのは乱暴な話ですが，ここでは鷹揚な心でそう考えることにしましょう）．P に属するかどうかが，概ね問題が「現実的に解ける」か否かの境目だと考えられています．対して PSPACE の問題は，記憶領域は入力長と同程度しか使いませんが，時間に制限はないので，入力の内容から生ずる長い変化の行く末を調べたり，組合せで作られる厖大な可能性を考え尽したりという，大変な手間を要するかもしれません．

例えば言語 SR は文字列の書換えに関する問題でしたが，PSPACE には属しません．書換えを施すうちに初めの文字列より長くなることもあり，限られた空間では調べ切れないのです．そうならぬよう，長さの増える書換え規則は認めない（入力に現れる「u_i を v_i

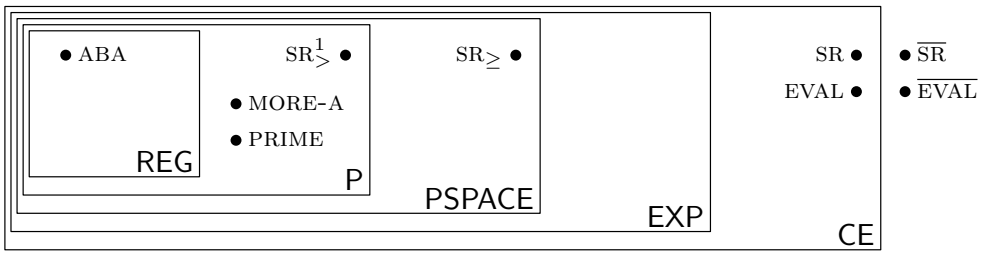

図 5　計算量級の包含関係と，本稿で扱った言語．但し P と PSPACE とが本当に異なるか，また PSPACE と EXP（指数時間で認識可能な言語全体）とが本当に異なるかは判っていない．

に書換える」という規則において必ず v_i が u_i と同じ長さ以下）とした問題 $\mathrm{SR}_\geq$ を考えますと，これは PSPACE に属します（長さが伸びないからといって，各段階で規則の施し方が複数あり得ますから，それをいかにして調べ尽すかは明らかではないのですが，うまく工夫すると多項式空間で済ますやり方があるのです）．ただ，それでも夥しい回数の書換えを要する可能性はまだありますから，P に属するかは判りません．これに対し，u_i よりも v_i が（一文字以上）短いとし，更に規則の施し方が一意である場合に制限した問題 $\mathrm{SR}^1_>$ は P に属します．書換えの度にどんどん短くなるのであれば，もとの文字列の長さと同程度の手間で調べ終えられるからです．このように，問題をどう限定するとどこまで容易になるのか，計算量級の言葉で整理できます．

素数の問題 PRIME は，先述の通り X 未満に X の約数があるか調べ尽せば判定できるので PSPACE に属しますが，この調べ方では時間がかかります．P に属するというには，X を十進法で書いたときの桁数（約 $\log X$）の多項式の時間で済さないといけません．それが可能なのかどうか，長らく未解決でしたが，大勢の人がさまざまな整数論の知識を持込んで挑んだ末，今世紀に入ってインドの研究者らが新たな計算法を発見し（AKS 素数判定法），PRIME が P に属することが示されました．

一方，問題が P に属しないことを示すには，頑張ったけれど速い計算法が見つかりませんでしたというのでは証明になりません．いかなる方法を用いても絶対に不可能と立証するのは難しいことです．じつは今日なお，PSPACE に属する問題のどれ一つとして，P に属しないと証明されたものはありません．図 5 の PSPACE は P よりも広く描いてありますが，本当に PSPACE $\neq$ P であることは示されていないのです．きっと異なるはずだと久しく予想されながら未だ決着を見ない級は他にも数多く，それらを分離する証明法を編み出すことは計算量理論数十年来の宿題となっています．

■非決定性多項式時間　さて，先程 PSPACE は指数的に多くの可能性を考える問題を含んでいると言いましたが，その膨大な可能性について何を答として得たいのかという観点

で多項式時間計算との関係を指定することで定まる部分級として，NP や RP などがあります (以下の定義から判るように，$\mathsf{P} \subseteq \mathsf{RP} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$ という関係にあります)．例えば上述の愚直な素数判定の手順では，指数個ある整数のうちに約数が存在するかのみを調べていました．このように書ける問題の級が NP です．すなわち言語 A が NP に属するとは，言語 $B \in \mathsf{P}$ と多項式 p が存在して任意の文字列 x で次を満すことをいいます．

- もし $x \in A$ ならば，$|r| \leq p(|x|)$ なる文字列 r が存在して $(x, r) \in B$
- もし $x \notin A$ ならば，$|r| \leq p(|x|)$ なる任意の文字列 r について $(x, r) \notin B$

例えば素数判定問題の補言語 $\overline{\text{PRIME}}$ は，この B として正整数 x とその 2 以上 x 未満の約数 r との組全体をとればよいので，NP に属します．NP はこのように

> 厖大な候補の中に，条件を満すものがあるかどうか問う問題
> (但しその厖大さのみが難しさなのであって，候補 r の一つ一つについてならこの「条件」$(x, r) \in B$ の成否の判断は $|x|$ の多項式時間で済むほど容易である)

からなるといえます．もう一つ NP に属する問題の例を挙げますと，先程の文字列書換え問題 $\text{SR}_{\geq}$ と $\text{SR}^1_{>}$ の間にある問題 $\text{SR}_{>}$，すなわち SR 同様だが各書換え規則において書換え先 v_i が書換え前 u_i よりも真に短いとしたものもその一つです．$\text{SR}^1_{>}$ とちがって適用可能な書換え規則が毎回幾つもある所が難しいわけですが，書換え列が与えられてしまえば，それが規則に合った正しい書換えになっているか確かめるのはたやすいからです．この書換え列として，もとの文字列以下の長さのもののみを考えれば良い所が重要です．

このように，多項式時間／空間で問題が「解ける」という P や PSPACE とは違い，NP の定義は○○の手間をかければ問題が「解ける」というものではありません (NP について，命名の由来でもある歴史的経緯からしばしば「非決定性多項式時間で解ける」などとも言い表されますが，実際には「○○で解ける」という意味ではないので，本来は○○を「多項式時間」のままにして，どちらかというと「解ける」という動詞の方を何か別の言葉に変えた方が正確です)．

しかし，NP に似た形の定義でありながら，現実的に「解ける」という意味をもつのが RP です．これは上の NP の定義の条件を

- もし $x \in A$ ならば，$|r| \leq p(|x|)$ なる文字列 r の 9 割以上について $(x, r) \in B$
- もし $x \notin A$ ならば，$|r| \leq p(|x|)$ なる任意の文字列 r について $(x, r) \notin B$

としたものです．すなわち NP の問題のうち，厖大な候補の中に条件を満すものが，**もしあるならば沢山ある**ことが保証されているものを意味します．こうなっていれば，候補 r を一つ一つ試すのではなく，ランダムに選んでやってみるという**乱択**を用いた算法により高確率で正解が出せます (このランダムな選択を何度かやり直すことで正解率は幾らでも上げられますから，「9 割」の代りにどの 1 未満の正定数にしても RP の意味は変りま

せん)．ですから，先程は現実的に解けるのは P の問題までだと言いましたが，より詳しくいうとこの範囲は RP やその周辺まで拡大して考えても良いかもしれません．PSPACE や NP とは違って，RP はどちらかというと P に一致する可能性も大きいと予想されています．この予想のように，ランダムな選択などというものが本当に要るのか (何かの手順で「てきとうに」決めた数列を，乱数として利用してもあまり「実害はない」のではないか) という問も，計算量理論の一大テーマです．

第三日　帰着と完全問題

■神託機械と帰着　或る問題 B を解くと判っている手順を，その中身を気にせずに，別の問題 A を解く手順の一部として利用することは，実際の問題解決でもよく行われます．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ります．B の解決に係る困難はさて措き，他の部分は簡単なのか，相対的に考えるのです．

これは言語 B を**神託**として用いるチューリング機械による計算として定式化できます．この機械は，計算の途中で好きな文字列を指定し，それが B に属するか教えてもらうことができます．この B の判定に要する時間や空間は考えず，即座に答が (神から告げられるかのごとく) もたらされるとするのです．B を神託として A を認識する多項式時間の機械が存在するとき，A は B に**帰着**するといい，$A \leq^{\mathsf{P}} B$ と書くことにします．

これは「もし仮に問題 B が (多項式時間・空間などの制限下で) 解けたら A も解ける」ということですから，不等号の表すように，A の複雑さが B 以下であると解してよいでしょう．二日目で $\overline{\mathrm{SR}}$ が CE に属しないと判ったのも，$\mathrm{EVAL} \leq^{\mathsf{P}} \mathrm{SR}$ という帰着によるものと見なせます．また P に属するか不明な問題どうしの間でも帰着関係が詳しく調べられています．$A \leq^{\mathsf{P}} B$ ならば，A や B が P に属するか属しないか四通りのうち，A が属せず B が属するという可能性が除かれるわけです．

■完全問題　このような帰着に基づく複雑さの整理に特に役立つのが**困難性**の概念です．PSPACE に属するすべての言語 A について $A \leq^{\mathsf{P}} B$ が成立つとき，B は PSPACE **困難**であるといいます．この場合もし B が P に属せば PSPACE = P となってしまいますので，逆にいえば B は多項式時間で解けないことがかなり確実といえます．例えば先に PSPACE の問題の例として挙げた言語 $\mathrm{SR}_{\geq}$ は，じつは PSPACE 困難でもあることが証明できます．こうなると $\mathrm{SR}_{\geq}$ は PSPACE の中でも最大の複雑さをもつ代表的な問題の一つであり，これを以て $\mathrm{SR}_{\geq}$ の複雑さは或る意味でぴたりと同定されたといえましょう．同

様に $\mathrm{SR}_>$ は NP **困難**です．他にも多くの級について，このような「最難の問題」（その級の**完全問題**と呼ばれます）が知られています．

なおこの講義に出てくる殆どの具体的問題の帰着では，じつは神託に多くの質問をする必要はなく，最後の一回質問してその答の通りにそのまま受理・不受理を決めるという形で十分です．このように制限された帰着 $\leq^{\mathsf{P}}_{\mathrm{m}}$ を**多対一帰着**といいます．困難性は本当は PSPACE-$\leq^{\mathsf{P}}$ 困難などと帰着を含めて言うべき概念ですが，しばしば省かれます．特に NP は $\leq^{\mathsf{P}}_{\mathrm{m}}$ については閉じている（$A \leq^{\mathsf{P}}_{\mathrm{m}} B \in \mathsf{NP}$ ならば $A \in \mathsf{NP}$）が $\leq^{\mathsf{P}}$ について閉じていないと考えられるため，NP 完全問題というときは普通 NP-$\leq^{\mathsf{P}}_{\mathrm{m}}$ 困難なものを指します．

■二型計算　神託機械のもう一つの応用として，計算理論による複雑さの分析を，整数，文字列，グラフといった離散的な対象を扱う問題だけでなく，実数をはじめとする連続的な対象をやりとりする計算に適用する，**計算可能解析学**の考え方を紹介します．

実数を受取り実数を答える函数を計算したいなら，まず実数を算法に「入力として与える」必要がありますが，もちろん実数は有限の文字列では書けず，近似によってのみ捉えられる対象です．数学的には実数とはどんどん近づいてゆく有理数の列（コーシー列）であるというものでした．そこで実数を算法に入力するとは，そのような無限の有理数列を与えること，すなわち算法が幾らでも必要な精度でその実数の近似値を得られるようにしてやることだと考えることにしましょう．また算法が実数を出力するというのも，利用者からの求めに応じて幾らでも必要な桁数でその実数の近似値を書き出すことと考えます．ここで実数 t の n 桁近似とは，t から距離 10^{-n} 以内にある小数点以下 n 桁の数（大体 2 つあります）のこととしておきます．つまり算法 M が実函数 $f\colon [0,1] \to \mathbb{R}$ を計算するとは，任意の $t \in [0,1]$ について，もし

- M が質問として 0^n（0 を n 個並べた文字列）を投げかけると，利用者は t の n 桁近似の一つを答えてやる

ならば，

- 利用者が質問 0^m を投げかけると，M は $f(t)$ の m 桁近似の一つを答えてくれる

ことをいいます（図 6）．この算法 M を形式的には神託機械と考えることで，実函数の多項式時間計算可能性を定義します（この多項式とは m の多項式ということになります）．

計算可能な函数は連続なものに限られます．実際，算法は $f(t)$ の m 桁近似を答えるまでに t の或る n 桁近似しか知りませんから，このような計算が可能であるには，そもそも t を幅 $2 \cdot 10^{-n}$ の区間に局限することで $f(t)$ が幅 $2 \cdot 10^{-m}$ の区間内に局限される必要があり，これは所謂イプシロン・デルタ論法による連続性の定義そのものです．

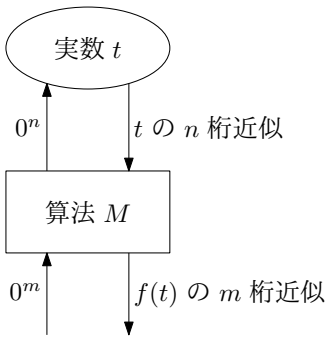

図 6　実函数 f を計算する算法 M.

この実函数の計算のような形で外界とやりとりしながら行う情報処理を，**二型**の計算と呼びます．零型とは整数などのように（有限の）文字列で表されるデータであり，一型とは零型の対象を入力とする写像や述語，例えば文字列から文字列への函数とか，整数の集合とかいったデータを指します．実数も有理数の無限列（すなわち $\mathbb{N}$ からの写像）と考えると一型のデータであり，実函数はそれを入出力とするから二型というわけです．

実数の（二変数函数の計算も同様に二つの神託へ自由に質問を許すものと定義して）四則演算や sin，exp，log といった普通に現れる連続な函数の多くは簡単に多項式時間計算可能と判ってしまいます．興味のある問は，そのような特定の函数というより，様々な数学的構成から生じ得る複雑さがどの程度か，というものです．例えば $f\colon [0,1]^2 \to \mathbb{R}$ に対して $\mathrm{MAX}(f)\colon [0,1] \to \mathbb{R}$ を

$$\mathrm{MAX}(f)(x) = \max\{\, f(x,y) : y \in [0,1] \,\}$$

で定義すると，「任意の多項式時間計算可能な f に対して $\mathrm{MAX}(f)$ もまた多項式時間計算可能である」という命題は $\mathsf{NP} = \mathsf{P}$ と同値であることが示せます．また，「任意のリプシッツ連続な多項式時間計算可能函数 $f\colon [0,1] \times [-1,1] \to \mathbb{R}$ に対して，微分方程式

$$g(0) = 0 \qquad\qquad g'(t) = f(t, g(t))$$

の解 $g = \mathrm{SOLVE}(f)\colon [0,1] \to \mathbb{R}$ も（もし値が $[-1,1]$ 内に収まるなら）多項式時間計算可能である」という命題は $\mathsf{PSPACE} = \mathsf{P}$ と同値です．これは「最大値をとる」演算子 MAX や「微分方程式を解く」演算子 SOLVE がそれぞれ NP や PSPACE と同等の複雑さをもつと解釈でき，実際この講義では扱いませんが適切に拡張した定式化の下でこれらは「NP 完全」「PSPACE 完全」であると示せます．「閉区間上の連続函数は最大値をもつ」「リプシッツ連続な微分方程式は解をもつ」という存在定理が，どこまで低い計算量で成立つ単純な主張であるかという限界を述べた結果ともいえるでしょう．

このように，機械や文字列に関する問題だけでなく，多様な分野の数学的対象に関する情報処理が，計算量級に基づく複雑さの類型の中に位置づけられます．三日目の講義では幾つかの完全問題を取り上げ，各問題のいかなる側面がそれぞれの複雑さを孕んでいるのか考えます．

参考文献

計算量理論の定評ある入門書を挙げておきます．

- M. Sipser. *Introduction to the Theory of Computation*, 3rd edition. Cengage Learning, 2012.
  ［邦訳］マイケル・シプサ著，田中圭介・藤岡淳監訳，阿部正幸・植田広樹・太田和夫・渡辺治訳『計算理論の基礎　原著第 3 版』共立出版（令和 5 年）
- 岩間一雄『アルゴリズム理論入門』朝倉書店（平成 26 年）
- O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

三日目に触れた計算可能解析についての解説は

- V. Brattka, P. Hertling and K. Weihrauch. A tutorial on computable analysis. In *New Computational Paradigms: Changing Conceptions of What is Computable*, S. B. Cooper, B. Löwe, and A. Sorbi, Eds. Springer, pp. 425–491, 2008.

本講義の資料の一部はこちらにあります．

`https://www.kurims.kyoto-u.ac.jp/~kawamura/t/ssmp/`