第5章 計算量のクラスの間の基本的関係

計算の複雑さについての研究が始まってから約 10 年の 1973 年，それまでにこの分野は急速に発展したが，ボロディン (A.Borodin) は当時までの研究成果についてサーベイした論文の中で次のように述べている．

“… わずかの期間の活発な研究によって，計算の複雑さの理論はきわめて成功裏に発展してきたと思う．しかし，どんな評価もその基本的な不十分性 — 特に，多くの「自然な」認識問題が非多項式的な難しさであることが証明できていないこと — のゆえに軽減されてしまう．…（中略）… これらの中心的問題が近い将来に解決されるであろうと信じる理由はどこにもない．…（中略）… 私は重要問題がすべて解決されたというには程遠いと思う．”

これは非決定性のアルゴリズムが決定性のアルゴリズムに比べて本当に能力が高いのかどうかという問題，その中でも特に $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ 予想が解かれていないことを嘆いているのだと思われる．しかし，実はこういった問題こそが，その解決を目指してその後の計算量理論の発展の原動力になったのである．その成果の一端は本書の後半で述べるが，この章では決定性と非決定性の関係，時間量と領域量の関係について初めて一般的な考察を行なう．

5.1 基本的クラス

まず，**DTIME**, **DSPACE**, **NTIME**, **NSPACE** に関していくつかの重要で興味深いクラスを定義しよう．

$\mathcal{R} = \mathbf{DSPACE}(1)$
$\mathbf{L} = \mathbf{DSPACE}(\log n)$
$\mathbf{P} = \bigcup_{i \geqq 1} \mathbf{DTIME}(n^i)$
$\mathbf{PSPACE} = \bigcup_{i \geqq 1} \mathbf{DSPACE}(n^i)$
$\mathbf{EXP} = \bigcup_{i \geqq 1} \mathbf{DTIME}(2^{n^i})$
$\mathbf{EXPSPACE} = \bigcup_{i \geqq 1} \mathbf{DSPACE}(2^{n^i})$
$\mathbf{REC} = \{L(M) \mid M \text{ は停止性 DTM}\}$
$\mathbf{RE} = \{L(M) \mid M \text{ は任意の DTM}\}$

$\mathbf{NL} = \mathbf{NSPACE}(\log n)$
$\mathbf{NP} = \bigcup_{i \geqq 1} \mathbf{NTIME}(n^i)$
$\mathbf{NPSPACE} = \bigcup_{i \geqq 1} \mathbf{NSPACE}(n^i)$
$\mathbf{NEXP} = \bigcup_{i \geqq 1} \mathbf{NTIME}(2^{n^i})$
$\mathbf{NEXPSPACE} = \bigcup_{i \geqq 1} \mathbf{NSPACE}(2^{n^i})$

停止性 TM (halting TM) とは，任意の入力に対して必ず停止する（すなわち，それ以上動作を続ける命令が存在しない状態に到達する）DTM のことである．停止状態が受理状態なら入力は受理され，受理状態でなければ拒否される．

また，$\mathcal{C}$ を言語のクラスとするとき

$$co\mathcal{C} := \{L^c \mid L \in \mathcal{C}\}$$

と定義する．ここに，L^c は L の補集合（complement; すなわち，$L \subseteqq \Sigma^*$ なら $L^c = \Sigma^* - L$）を表す（Σ はいろいろ考えられるが，特定しない）．

テープヘッドが右方向にしか動かない TM を **1 方向 TM** (1-way TM) という．$O(1)$ 領域限定 1 方向 1 テープのオフライン TM のことを 1 方向有限オートマトン，あるいは単に**有限オートマトン** (**FA**: finite automaton) といい，有限オートマトンによって受理される言語を**正規言語**とか**正規集合**という（正則言語/正則集合ともいう．regular set/language)[†1]．1 方向 FA は，テープヘッドが 1 方向にしか動かないモデルであるが，当然ながら 2 方向のモデルもあり，決定性モデル (**DFA**: deterministic FA) も非決定性モデル (**NFA**: nondeterministic FA) もあり，1 テープモデルだけでなく多テープモデルもある．また，これらそれぞれのモデルに対し，テープヘッドを複数にしたモデル（多ヘッド FA）もあるが，有限オートマトンの場合，1 方向であるか 2 方向であるかや，決定性であるか非決定性であるかや，1 テープであるか多テープであるかによる言語受理能力に差はない．しかし，テープヘッドを複数にすると能力が上がる：

定理 5.1. (1)[†2] TM が 1 方向/2 方向，決定性/非決定性，1 テープ/多テープのいずれであるかに関わりなく，$\mathbf{DSPACE}(1) = \mathbf{NSPACE}(1)$ が成り立つ．

(2) 2 方向多ヘッド DFA（2 方向多ヘッド NFA）によって受理される言語のクラスは $\mathbf{L}$ ($\mathbf{NL}$) に等しい．

(3) $s(n) = o(\log\log n)$ なら $\mathbf{NSPACE}(s(n)) = \mathcal{R}$ である[†3]．

証明の概要 (1) オフラインモデルでは入力テープの書き換えは許されていない．定理 4.1 より，領域量が $O(1)$ の場合にも 1 テープ TM と多テープ TM の能力は等しい．さらに，領域量が $O(1)$ の場合，作業テープ上に書く情報は，作業テープに書かずに状態の中に保持することで代用できるので，作業テープはないとしてもよい．よって，1 方向か 2 方向かは入力テープについてだけ考えればよい．1 方向 FA と 2 方向 FA の言語受理能力が等しいことの証明には，交差列論法（4.3 節）を使う．DFA と NFA の言語受理能力が等しいことを示すには，NFA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ と等価な DFA $M' = (Q', \Sigma, \Gamma, \delta', q_0', F')$ を次のように定義すればよい：

$$Q' := 2^Q, \quad q_0' := \{q_0\}, \quad F' := \{Q'' \subseteqq Q \mid Q'' \cap F \neq \emptyset\},$$
$$a \in \Sigma,\ P \in Q' \text{ に対し } \delta'(P, a) := (\{q \mid p \in P,\ (q, a, 1) \in \delta(p, a)\}, a, 1).$$

(2) k ヘッド DFA / NFA M のヘッド位置 $h \leqq n$（n は入力の長さ）は長さが $O(\log_2 n)$ の 2 進数で表すことができるので，$L(M) \in \mathbf{DSPACE}(\log n) / \mathbf{NSPACE}(\log n)$.

逆に，$\log n$ 領域限定の TM M を模倣する 2 方向多テープ FA M' を次のように構成する．M は 1 テープで，テープ記号は $0, 1$ だけであるとしても一般性を失わない（なぜか？）．M のテープの内容が $w_1 \dot{a} w_2$（$\dot{a}$ はヘッドがテープ記号 a の上にあることを表す）のとき，$|w_1|, |w_2| \leqq k \log n$（$k$ は定数）なので，w_1, w_2 を 2 進数だとみなすと，その値はたかだか n^k である．任意の正整数 $m < n^k$ は $m = \sum_{i=0}^{k-1} a_i n^i$ ($a_i \in \{0, \ldots, n-1\}$) と一意的に表すことができるので，$w_1, w_2$ の内容はそれぞれに k 個のヘッドを使えば，M' におけるそれらのヘッド位置 $a_0 + 1, \ldots, a_{k-1} + 1$

[†1] 有限オートマトンは作業領域を $O(1)$ しか使わないので，作業テープの内容は有限制御部に状態として保持することができ，作業テープは不要である．したがって通常は，作業テープのない，有限個の状態だけからなるモデルを有限オートマトンといい，それが名前の由来である（当初は，有限状態オートマトン (finite-state automaton) と呼んだ）．

[†2] M.O.Rabin and D.Scott, Finite automata and their decision problems, *IBM Res. Develop.* 3, pp.115–125, 1959. 詳細は，例えば，守屋悦朗（著）『形式言語とオートマトン』，サイエンス社，2001 を参照されたい．

[†3] 実は，このことは交代性 TM（ATM: 本書第 2 巻の「並列計算」の項で述べる）でも成り立つことが知られている：K.Iwama, ASPACE($o(\log\log n)$) is regular, *SIAM J. Comput.* 22, pp.136–146, 1993.

によって表すことができる．M' が M の 1 ステップを模倣するためには，それぞれ k 本のヘッドで表された $|w_1|$ と $|w_2|$ を「2 倍する」,「2 倍して 1 足す」,「2 で割る」に相当するように動かせばよい．M 自体のヘッド位置も記憶しておくために，M' はもう 1 つのヘッドを使う．

(3) 簡単のために，DTM の場合，すなわち $\mathbf{DSPACE}(o(\log\log n)) = \mathcal{R}$ を示す．L が正規言語でないなら，すなわち L を受理する DFA が存在しないなら，L を受理するどんな DTM も $\Omega(s(n))$ 領域を使うことを示せばよい．

交差列論法（定理 4.6 参照）で証明する．$L \notin \mathcal{R}$ を受理する $s(n)$ 領域限定の DTM M を考える $(s(n) \neq O(1))$．$s(n)$ 領域を使う，長さが最小の入力 $x \in L(M)$ を考える $(|x| = n)$．入力 x の i 番目の記号と $i+1$ 番目の記号の境目における交差列の個数は，$c_1, c_2 \geqq 2$ を定数として，たかだか $c_1^{c_2^{s(n)}}$ 個しかない（$\because$ 入力が x から到達し得る ID の個数，および ID から ID への遷移の仕方が何通りあるかを考えよ）．もし $s(n) = o(\log\log n)$ だとすると，$c_1^{c_2^{s(n)}} = o(n)$ であるから，鳩の巣原理より，$x = uvw$ かつ u, v の境界における交差列と v, w の境界における交差列が等しいような $u, v, w\,(v \neq \lambda)$ が存在する．よって，$x = uvw \in L(M) \Longleftrightarrow x' := uw \in L(M)$ が成り立つ．$|x'| < |x|$ かつ x' も $s(n)$ 領域を使うとしてよいので（なぜか？），これは x の最小性に反す．

因みに，$\mathbf{DSPACE}(\log n) - \mathcal{R} \neq \emptyset$ であることが知られている． □

問 5.1. (1) 定理 5.1 (1) の証明において，$L(M') = L(M)$ であることを証明せよ．
(2) 定理 5.1 (2), (3) の証明における「なぜか？」に答えよ．また，(2) における「2 倍する」,「2 倍して 1 足す」,「2 で割る」はどういうことを表しているのか？

問 5.2. アルファベット $\{a, b\}$ 上の次の言語を考える．
(1) $L := \{a, b\}^*\{a\}\{a, b\}^n$ を受理する DFA と NFA を示し，状態の個数を比較せよ．
(2) AnBn $:= \{a^n b^n \mid n \geqq 0\}$ を受理する 2 ヘッド DFA および $\log n$ 領域限定 DTM を示せ．

$O(n)$ 領域限定 TM のことを**線形有界オートマトン** (**LBA**, linear bounded automaton) といい，非決定性 LBA によって受理される言語を**文脈依存言語** (context-sensitive language) という．1 章で述べたように，文脈依存言語をはじめとする形式言語と形式文法の理論は，自然言語の数学的研究のために N.Chomsky が始めたものである．歴史的にはまったく研究目的が異なる分野で定義された 2 つの概念 — アルゴリズムの数学的モデルである TM（＝オートマトン）と，自然言語の数学的モデルである形式文法 — には実は密接な関係がある．すなわち，

定理 5.2. (1) L を受理する TM M が存在する $(L = L(M)) \iff L$ を生成する句構造文法 G が存在する $(L = L(G))$．
(2) $L \in \mathbf{NSPACE}(n) \iff L$ を生成する文脈依存文法が存在する．
(3) $L \in \mathbf{NSPACE}(1) \iff L$ を生成する正規文法が存在する． □

が成り立つ．**文脈依存文法** (context-sensitive grammar) とは，どの書き換え規則 (α, β) も $|\alpha| \leqq |\beta|$ を満たしている句構造文法 $G = (V, T, P, S)$ のことであり[†4]，**正規文法** (regular grammar) とは，どの書き換え規則 (α, β) も $\alpha \in V$, $\beta \in TV \cup T \cup \{\lambda\}$ を満たしている句構造文法のことである．

問 5.3. 次のそれぞれを示せ/求めよ．
(a) 3 の倍数である正整数を表す言語 TRI $:= \{x \in \{0, 1, 2, \ldots, 9\}^+ \mid$ 10 進数表現した整数 x は 3 で割り切れる$\} \in \mathcal{R}$ を受理する有限オートマトンと，生成する正規文法．

[†4] 正確には，この形の文法は単調文法 (monotone grammar) といい，文脈依存文法と言語生成能力が等しい．

(b) AmBn := $\{a^m b^n \mid m \geqq n\}$ を受理する NTM と，生成する文脈依存文法．因みに，AmBn $\notin \mathcal{R}$ である．

(c) タリー表現した平方数の集合 SQR := $\{1^{n^2} \mid n \geqq 1\} \in \mathbf{DTIME}(n^{O(1)}) \cap \mathbf{DSPACE}(\log n)$ を満たす DTM と，文脈依存文法．NTM の場合はどうか？ 因みに，SQR $\notin \mathcal{R}$ である．

与えられた語 x が言語 L に属すかどうか ($x \in L$?) を判定する**決定問題** (decision problem) を**所属性判定問題** (membership problem) という．決定問題を解くアルゴリズム（= DTM）が存在するとき，その問題は**決定可能** (decidable; 可解, recursively solvable) であるという．ここで，決定問題とは答が yes または no であるような問題のことであったことを思い出そう．

REC は**帰納的言語** (recursive set/language) のクラスであり，**RE** は**帰納的可算言語** (recursively enumerable set/language) のクラスである．これらの名前の由来は次の定理にある：

定理 5.3. (1) L が帰納的言語 $\iff$ L に対する所属性判定問題は決定可能である．
(2) L が帰納的可算言語 $\iff$ L のすべての元を列挙する手続きが存在する． □

【注】「帰納的 (recursive)」という術語は,「再帰的」という意味以外に「アルゴリズムがある」という意味でも用いられる．したがって，帰納的言語（帰納的集合）とは「与えられた要素がその言語（集合）に属すかどうかを判定するアルゴリズムが存在する集合」という意味である．

一方，可算 (enumerable) とは文字通り，その言語（集合）の要素すべてを「列挙」することができる，すなわち，そのための**手続き**（答が yes のときは必ず停止するが，答が no のときは停止しないかもしれないようなアルゴリズムのこと）が存在するという意味である．手続きについては 1.2 節を参照のこと．

問 5.4. 定理 5.3 の (1) を証明せよ．すなわち，$x \in L$ なら状態 yes で停止し，$x \notin L$ なら状態 no で停止する DTM を示せ．

問 5.5. 定理 5.3 の (2) を証明せよ．すなわち，L のすべての元を**出力テープ**（書き込みだけができる特定の作業テープのこと）の上に書き出す DTM M を示せ（この TM は入力は（使わないので）何でもよく，計算は永久に続くようなものである）.（ヒント：$L \subsetneqq \Sigma^*$ のとき，M は $x \in \Sigma^*$ を系統的に生成して，L を受理する TM に x を入力して $x \in L$ か否かを判定する．この際，1 つの x の計算を最後まで続けると，無限ループに陥るために $x \notin L$ である場合に問題が生じるので，その回避策を考えよ.）

5.2 非決定性 vs 決定性，時間量 vs スペース量

与えられた問題を解くアルゴリズム（与えられた言語を受理する TM）を設計しようとするとき，“非決定性” の機能があると設計が容易である[†5]．では，非決定性によって本当に計算能力が上がるのであろうか？ この問題は計算量理論においてもっとも基本的で重要な問題でありながら，十分良くわかってはいない．実際，$\mathbf{L} \stackrel{?}{=} \mathbf{NL}$ 問題，LBA 問題（$\mathbf{NSPACE}(n) \stackrel{?}{=} \mathbf{DSPACE}(n)$），$\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ 問題など，いくつかの特定のケース（既出）は有名な未解決問題である．

TM（NTM でも DTM でもよい）M_1 が与えられたとき，M_1 を**模倣** (simulate) する TM とは $L(M_1) = L(M_2)$ となるような M_2 のことをいう．時間量に関しては，DTM による NTM の模倣法として次のような自明の方法（NTM の計算木上の可能な道すべてをなぞってみる方法）しか知られていない．

[†5] 例えば，有限オートマトンを例にすると，状態の個数が n の NFA で受理できるのに，それを受理するどんな DFA も状態の個数が 2^n でなければならない言語が存在する．すなわち，DFA の方が NFA よりも複雑になる．例えば，問 5.2 (1) はそのような言語の例である．

定理 5.4. M を $f(n)$ 時間限定 NTM とすると，M を模倣する $c^{f(n)}$ 時間限定 DTM M' が存在する（$c > 0$ は定数）．すなわち，$\mathbf{NTIME}(f(n)) \subseteqq \mathbf{DTIME}(c^{f(n)})$ が成り立つ．

証明 M のテープの本数を k，状態の個数を s，テープ記号の個数を t とする．M が取り得る ID の個数はたかだか

取り得る状態の個数 $\times\, k$ 個のヘッド位置に関する場合の数 $\times\, k$ 本のテープの内容に関する場合の数

である．M は $f(n)$ 時間限定だから，どのテープ上でもたかだか $f(n)+1$ 個のマス目しか使わないから，この値はたかだか $s(f(n)+1)^k t^{kf(n)}$ である．ここで，n は入力の長さ．$d = s(t+1)^{3k}$ とおくと，任意の $n \geqq 1$ に対して（$f(n) \geqq n$ なので）

$$s(f(n)+1)^k t^{kf(n)} \leqq d^{f(n)}$$

である．

M がある ID から次の ID へ遷移するとき選択できる可能性がたかだか r 通りであるとする．M' は $k+r+1$ 本のテープをもつ．入力 w ($|w| = n$) が与えられたとき M' は，w に対する初期 ID から M が到達可能な ID すべてを重複しないように第 $k+1$ テープ上に生成する．この生成は，図 5.2.1 に示した計算木を**幅優先探索** (breadth-first search) するような順序で行なう．

すでに ID $I_1, \ldots, I_i$ が生成済みで，そのうちの $I_1, \ldots, I_{j-1}$ の子のすべてが生成済みとする（図 5.2.1 参照）．ID 間には境界を表すマス目があるが，特に I_{j-1} の右側の境界マス目に印を付けておく．

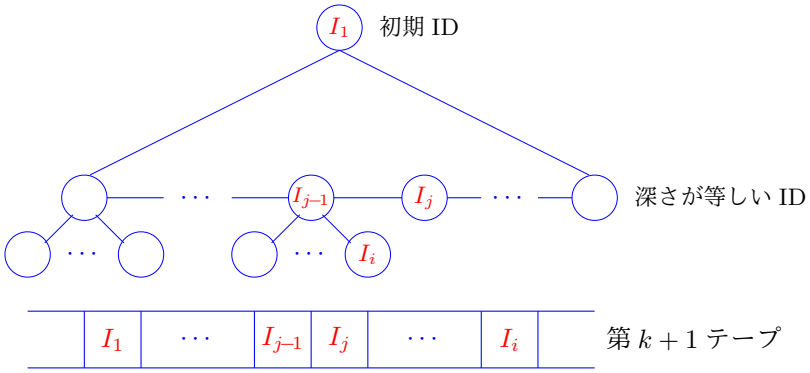

図 5.2.1: M の計算木と ID の生成

I_j のすべての子を次のように生成する．M' は第 $k+1$ テープヘッドを I_j の左端に移し，I_j の子となり得る ID を 1 つずつ第 $k+2 \sim k+r+1$ テープ上に生成する（r 個より少ないこともある）．次に，テープ $k+1 \sim k+r+1$ のヘッドを左端に戻し，生成された子 ID の中に $I_1 \sim I_i$ に等しいものがないかどうかを調べる．$I_1 \sim I_i$ のどれとも等しくなかったものだけを I_i の右にコピーし，I_j の右側の境界マス目に印を付ける．第 $k+1$ テープに ID を 1 つ追加する操作は，それまでに第 $k+1$ テープに書かれた ID の個数に比例する時間で行なうことができる．1 つの ID は長さが $O(1+k(f(n)+1))$ の文字列（すなわち，状態を表す記号と，テープの内容を表す k 個の文字列（各文字列は $\alpha \uparrow \beta$ なる形で，$\uparrow$ はヘッドの位置を表す記号) を連接したもの)）で表すことができる．したがって，最終的にできるリストの長さ ℓ は

$$\ell = O(d^{f(n)}(1+k(f(n)+1)))$$

であり，M' が使う時間は $O\left(\sum_{i=1}^{\ell} i\right) = O(\ell^2)$ である．$\ell^2 \leqq c^{f(n)}$ となる定数 c が存在する．

上記の生成の途中で M の受理 ID が生成されたら M' は入力を受理する．受理 ID が生成されないまま M の全 ID の生成が終了したら M' は入力を拒否する． □

次の条件を満たす $f(n)$ 時間限定 DTM M が存在するとき，関数 $f(n) : \mathbb{N} \to \mathbb{N}$ は**時間構成可能** (time constructible) であるという[†6]：

> 任意の $n \in \mathbb{N}$ と長さ n の任意の入力 w に対し，M の w に対する計算はちょうど $f(n)$ ステップで停止する（出力テープの $f(n)$ 個のマス目に印をつける）

同様に，任意の $n \in \mathbb{N}$ と長さ n の任意の入力 w に対し，w に対する計算が，ある 1 つの決められたテープ上でちょうど $f(n)$ 領域を使うような $f(n)$ 領域限定 DTM M が存在するとき，$f(n)$ は**領域構成可能** (space constructible) であるという．定義より，時間構成可能なら領域構成可能である．n^k, c^n, $n!$ など，時間量や領域量として現れる関数のほとんどは時間構成可能であり，領域構成可能でもある．

例 5.1. (1) n^2 が時間構成可能であることを示そう．次のように動作する DTM M を考える．まず，入力の先頭 1 文字を読み（入力ヘッドは右へ動かす），記号 '1' を第 2 テープに書く（第 2 テープヘッドは動かさない）．以後，入力ヘッドを右に 1 動かすごとに次の操作を行なって停止する（入力が空語のときは何もしないで停止する）：□ に出会うまで第 2 テープヘッドを左に動かし，次いで，'1' が続く限り右に動かし，'1' の右隣りの □ を '1' に書き換える（ヘッドは右端の '1' の上に残す）．この操作の冒頭で，入力ヘッドも右に 1 動かし，入力が尽きたかどうかをチェックしておく（1 ステップ余計に使うわけではなく，第 2 テープヘッドを左に動かすのと同時に行なう）．操作開始前に '1' が i 個あった場合，この操作にちょうど $2i-1$ ステップかかる．したがって，停止するまでにかかる時間は $\sum_{i=1}^{n}(2i-1) = n^2$ である．

M は 1 ステップ進むごとに出力テープの 1 マスに印をつける．

(2) $f_1(n)$, $f_2(n)$ がそれぞれ領域構成可能関数ならば，$f_1(f_2(n))$, $f_1(n)+f_2(n)$, $f_1(n)f_2(n)$, $2^{f_1(n)}$ もそれぞれ領域構成可能関数である． □

問 5.6. (1) 時間構成可能関数であることを示せ ： (a) $f_1(n) = 2n$ (b) $f_2(n) = n^3$ (c) $f_3(n) = 2^n$
(2) 領域構成可能関数であることを示せ ： (a) $f_4(n) = \lceil \log_2 n \rceil$ (b) $f_5(n) = f_1(n) + f_2(n)$

さて，任意の $f(n)$ に対して，

$$\mathbf{DTIME}(f(n)) \subseteqq \mathbf{NTIME}(f(n)), \quad \mathbf{DSPACE}(f(n)) \subseteqq \mathbf{NSPACE}(f(n))$$
$$\mathbf{DTIME}(f(n)) \subseteqq \mathbf{DSPACE}(f(n)), \quad \mathbf{NTIME}(f(n)) \subseteqq \mathbf{NSPACE}(f(n))$$

が成り立つことは定義から自明である．以下に，自明でない基本的結果を示す．

[†6] これは，$f(n)$ は入力の長さ n だけに依存して定まるということである．このようなとき（そのような TM は）入力に依存しない (oblivious) という．

本書で用いた "時間構成可能性" の定義を，文献によっては "fully time constructible" と呼んでいる．その場合，もう少し弱い条件を time constructible の定義にする．領域構成可能についても同様である．問 5.6 を解いてみればわかるように，領域構成可能関数に比べると，時間構成可能関数を実際に DTM で定義するのは易しくない．しかし，幸いにも，$f(n) = \omega(n)$ なら，

本書の意味で $f(n)$ が時間構成可能 $\iff$ $f(n)$ を $O(f(n))$ 時間で出力する DTM が存在する

であることが知られている： K.Kobayashi, On proving time constructibility of functions, *Theor. Comput. Sci.* 35, pp.215–225, 1985.

定理 5.5. (1) 任意の $f(n)$ に対して $\mathbf{NTIME}(f(n)) \subseteqq \bigcup_{c>0} \mathbf{DSPACE}(c^{f(n)})$.
特に，$f(n)$ が時間構成可能な関数なら $\mathbf{NTIME}(f(n)) \subseteqq \mathbf{DSPACE}(f(n))$ が成り立つ．

(2) $f(n)$ が時間構成可能関数なら $\mathbf{DTIME}(f(n)) \subseteqq \mathbf{DSPACE}(f(n)/\log f(n))$.

(3) 任意の $f(n)$ に対して $\mathbf{NSPACE}(f(n)) \subseteqq \bigcup_{c>0} \mathbf{DTIME}(c^{\log n + f(n)})$. したがって，$\mathbf{DSPACE}(f(n)) \subseteqq \bigcup_{c>0} \mathbf{DTIME}(c^{\log n + f(n)})$ も成り立つ．

証明 (1) 定理 5.4 より，$\mathbf{NTIME}(f(n)) \subseteqq \bigcup_{c>0} \mathbf{DTIME}(c^{f(n)})$. したがって，上述の包含関係より $\mathbf{NTIME}(f(n)) \subseteqq \bigcup_{c>0} \mathbf{DSPACE}(c^{f(n)})$.

$f(n)$ が時間構成可能である場合には，$f(n)$ 時間限定 NTM M を模倣する DTM M' を次のように構成する：長さ n の入力に対し，M' は最初に各テープ上の大きさ $f(n)$ の領域に印を付ける（$f(n)$ が時間（領域）構成可能だから，そのようにできる）．次いで，長さが $f(n)$ 以内の M の動作命令の列を系統的に生成し，それが M の受理計算になるかどうかを検証する，ということを繰り返す．1 つでもそのような受理計算が見つかったら直ちに停止して入力を受理するが，生成した動作命令の列が M の受理計算でなかった場合は，すでに使った領域を消去して，そこに次の動作命令の列を生成して模倣を続ける．$O(f(n))$ 領域を使えば，長さ $f(n)$ 以内の M の動作命令の列すべてを系統的に順次生成でき，かつそれぞれの動作命令の列を 1 ステップずつ実行してみることができる．

(2) TM は 1 ステップでテープヘッドはたかだか 1 コマしか左右に移動しないので，模倣をする際にはテープの内容全てを保持しながら行なう必要は必ずしもない．そこで，模倣すべき TM のテープ内容をある長さ（例えば，$\sqrt{f(n)}$）のブロックに分割して記憶することにより領域使用量を減らすことが可能となる．ある時点における模倣において，ヘッドが保持しているブロックをはみ出る場合には保持するブロックを変える必要が出てくるが，そういうときには計算を最初からやり直す．ブロックとブロックの間の遷移関係は，ブロックを頂点とするグラフで表現する．詳細は省略する[†7]．

(3) 定理 4.1 より，M は 1 作業テープオフライン NTM と仮定してもよい．M の状態の個数を s，テープ記号の個数を t とすると，長さ n の入力 x に対して M が取り得る ID の個数はたかだか $(n+2)\cdot s\cdot f(n)\cdot t^{f(n)}$ である．$(n+2)\cdot s\cdot f(n)\cdot t^{f(n)} \leqq d^{\log n + f(n)}$ となる正整数 d が存在する．

入力 x に対する M の ID を頂点とする有向グラフ $G_{M,x}$ を

$$(I_1, I_2) \in E(G_{M,x}) \iff I_1 \vdash_M I_2$$

と定義する．この $G_{M,x}$ を，入力 x に対する M の**様相グラフ** (configuration graph) と呼ぶ．M と x から $G_{M,x}$ のコード $\overline{G_{M,x}}$ を次のように生成する DTM M_1 を構成することができる．$\overline{G_{M,x}}$ は辺すなわち隣接する 2 頂点の対 (I_1, I_2) すべてを列挙した文字列

$$\overline{G_{M,x}} = \cdots \#\#I_1\#I_2\#\#\cdots$$

で表す．I_1, I_2 は M の ID（を適当に符号化した文字列）であり，$\#$ は区切りを表すための特別な記号である．M の ID 1 つの長さは $O(f(n))$ であり，$G_{M,x}$ の頂点（$= M$ の ID）はたかだか $d^{\log n + f(n)}$ 個しかないから

$$|\overline{G_{M,x}}| = O((d^{\log n + f(n)})^2 f(n))$$

[†7] J.E.Hopcroft, W.J.Paul and L.G.Valiant, On time vs. space and related problems, *Proc. 16th IEEE Symp. on the Foundations of Comput. Sci.* pp.57–64, 1975.

とできる．M_1 は定理 5.4 の証明と同様な方法で $G_{M,x}$ の辺すなわち (I_1, I_2) すべてを重複しないようにチェックしながら生成する（ついでに，次のステップで $G_{M,x}$ における到達可能性問題を解くときに使うために，全頂点のリストを別のテープ上に生成しておく）．定理 5.4 の証明と同じ論法で，M_1 の時間量は $O\big(|\overline{G_{M,x}}|^2\big)$ であることがいえる．

有向グラフの到達可能性問題を解く，時間量が O(頂点数 + 辺数) のアルゴリズムが存在するが，このアルゴリズムの時間量は，有向グラフの任意の頂点から隣接する頂点を求めることや任意の頂点がすでにたどられたか否かを判定することがそれぞれ $O(1)$ 時間でできるとした場合のものである．$G_{M,x}$ は M_1 が生成した文字列 $\overline{G_{M,x}}$ と全頂点のリストを表す文字列（その長さは $O\big(|\overline{G_{M,x}}|\big)$ である）とで与えられるが，この $G_{M,x}$ に対する到達可能性問題を解く $O\big(|\overline{G_{M,x}}|^2\big)$ 時間限定 DTM M_2 を構成することができる．

以上の議論により，x が M に受理されるかどうかを時間量

$$O\big(|\overline{G_{M,x}}|^2\big) = O(c^{\log n + f(n)})$$

で判定する DTM が存在することが示された（$c > 0$ は適当な定数）． □

系 5.1. **L** $\subseteqq$ **NL** $\subseteqq$ **P** $\subseteqq$ **NP** $\subseteqq$ **NPSPACE**.

問 5.7. $t(n)$ が時間構成可能関数，$s(n)$ が領域構成可能関数である場合，問 3.5 の 2 つの $t(n)$ 時間限定/$s(n)$ 領域限定の定義は一致することを示せ．

問 5.8. n を（有向）グラフの頂点数＋辺数とする．(有向）グラフの任意の頂点から隣接する頂点を求めることや任意の頂点がすでにたどられたか否かを判定することがそれぞれ $O(1)$ 時間でできるとする．
(1) （有向）グラフの到達可能性問題を解く，$O(n)$ 時間限定のアルゴリズム（= DTM）を示せ．
(2) （有向）グラフの到達可能性問題を解く，$O(\log n)$ 領域限定の非決定性アルゴリズム（= NTM）を示せ．

5.3 サヴィッチの定理

NTM を DTM で模倣する際，時間量に関しては定理 3.1 に述べたような自明な模倣 (**NTIME**$(t(n))$ $\subseteqq \bigcup_{c>0}$ **DTIME**$(c^{t(n)})$) しか知られていないが，領域量に関してはもっと良い模倣の方法が存在する．

定理 5.6 (サヴィッチの定理)．[†8] $s(n) \geqq \log n$ を領域構成可能関数とする．M が $s(n)$ 領域限定 NTM なら，M を模倣する $O(s(n)^2)$ 領域限定 DTM M' が存在する．すなわち，**NSPACE**$(s(n)) \subseteqq$ **DSPACE**$(s(n)^2)$.

証明 M の作業テープの本数を k とし，状態の個数を q，テープ記号の個数を r とすると，長さ n の入力に対して，M の異なる ID の個数はたかだか $q \cdot (n+2) \cdot s(n)^k \cdot r^{ks(n)}$ である．$s(n) \geqq \log n$ であるから，

$$q \cdot (n+2) \cdot s(n)^k \cdot r^{ks(n)} \leqq c^{s(n)}$$

が任意の $n \geqq 1$ に対して成り立つような定数 c が存在する．長さ n の入力 x が与えられたとき，M' は $O(s(n))$ 領域を使い，これらの ID を系統的に順次生成することができる．

I_1, I_2 を M の ID とする．M が I_1 からたかだか k ステップで I_2 に到達できるならば

[†8] W. J. Savitch, Relationship between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* 4, pp.177–192, 1970.

$$I_1 \overset{\leqq k}{\Longrightarrow} I_2$$

と書くことにする．定義より，$i \geqq 1$ のとき

$$I_1 \overset{\leqq 2^i}{\Longrightarrow} I_2 \quad \Longleftrightarrow \quad \exists I_3 \left[I_1 \overset{\leqq 2^{i-1}}{\Longrightarrow} I_3 \overset{\leqq 2^{i-1}}{\Longrightarrow} I_2 \right]$$

である．このことに基づき，M' は 入力 x に対する M の初期 ID $Init_M(x)$ から受理 ID へ到達できるかどうかを再帰的に判定する（図 5.3.1 に，再帰的手続き $M'(x)$ として示した）．

```
procedure M'(x)
begin
    h ← ⌈log_2 c⌉s(n);
    forall 長さ s(n) 以下の M の受理 ID I_f do
        if reachable(Init_M(x), I_f, h) then x を受理して終了する end
    end;
    x を拒否する
end

function reachable(I_1, I_2, i)
begin /* I_1 ⟹^{≦2^i} I_2 かどうかを判定する */
    if i = 0 and (I_1 = I_2 or I_1 ⊢_M I_2) then return true end;
    if i ≧ 1 then
        forall 長さ s(n) 以下の M の ID I_3 do
            if reachable(I_1, I_3, i − 1) and reachable(I_3, I_2, i − 1) then return true end
        end
    end;
    return false
end
```

図 5.3.1: M' による M の模倣

関数 reachable(I_1, I_2, i) の計算木の高さ（＝再帰の最大深さ）は i であり，ステップ数が 2^i 以下の M の計算がそれに対応する．入力 x に対する M の ID の個数 $\leqq c^{s(n)}$ であるから，x が M に受理される必要十分条件は，ステップ数 $\leqq c^{s(n)}$ なる受理計算が存在することである．よって，M' は

$$Init_{M,x} \overset{\leqq 2^h}{\Longrightarrow} I_f$$

が成り立つか否かをすべての受理 ID I_f に対して調べれば十分である．ここで，$h = \lceil \log_2 c^{s(n)} \rceil$ である．以上の考察より，$L(M') = L(M)$ である．

最後に M' の領域量を考えよう．関数 reachable(I_1, I_2, i) の変数（reachable の中だけで局所的に使われている変数）は I_1, I_2, i だけであるから，その領域量は $O(2s(n) + \log h) = O(s(n))$ である（h は，2 進数で表すことにすると，必要な領域量は $O(\log h)$ である）．したがって，M' の領域量は，定理 0.5 より，

$$\begin{aligned} & \lceil reachable(Init_M(x), I_f, h) \text{ の計算木の高さ} \rceil \times \lceil reachable \text{ の局所変数の領域量} \rceil \\ = \ & O(h \cdot s(n)) = O(s(n)^2) \end{aligned}$$

である．よって，$\mathbf{NSPACE}(s(n)) \subseteqq \mathbf{DSPACE}(O(s(n)^2))$ である．最後に，定理 4.2（線形領域圧縮定理）を適用すると $O(\cdot)$ を取り除くことができる． □

系 5.2. $\mathbf{PSPACE} = \bigcup_{i \geqq 1} \mathbf{NSPACE}(n^i)$.

問 5.9. 定理 5.6 において，なぜ $s(n)$ は領域構成可能である必要があるのか？ また，なぜ $s(n) \geqq \log n$ なのか？

問 5.10. 問 5.8 とサヴィッチの定理より，(有向) グラフの到達可能性問題を解く，$O(\log^2 n)$ 領域限定の決定性アルゴリズム（= DTM）が存在する．具体的に示せ．

5.4 イマーマン・セレプトセーニィの定理

2.4 節で，DTM M を変換機と考え，入力 x に対する出力 $M(x)$ を定義した．M が NTM の場合には，$M(x)$ は次のように定義される言語である:

$$y \in M(x) \Longleftrightarrow M \text{ が } x \text{ を受理して停止したときの出力テープの内容が } y.$$

DTM の場合と同様に，出力テープの使用量は領域量にカウントしない．

さて，グラフの到達可能性問題は $O(\log n)$ 領域限定の非決定性アルゴリズム (NTM) で解くことができる（問 5.8）．この問題を拡張して，x から到達可能な頂点の個数を求めることも $O(\log n)$ 領域限定の非決定性アルゴリズムで解くことができることを以下で証明する．このような，個数を求める問題を一般に**数え上げ問題** (counting problem) という．

補題 5.1. 頂点数が n のグラフ G と頂点 x が与えられたとき，G において x から到達可能な頂点の個数 $M(G, x)$ を出力する $O(\log n)$ 領域限定の NTM M が存在する．

証明 x から長さが k 以下の道に沿って到達可能な頂点の集合を $S(k)$ で表す．明らかに $|S(0)| = 1$ ($S(0) = \{x\}$) であり，$|S(n-1)|$ は x から到達可能な頂点の個数である．以下の非決定性アルゴリズム (NTM M) では $|S(0)|, |S(1)|, \ldots$ を順次計算するが，それぞれの k について，$|S(k)|$ の計算には $|S(k-1)|$ しか使わない．

はじめに，頂点 v が与えられたとき，$v \in S(k-1)$ かどうかを $O(\log n)$ 領域で判定する非決定性アルゴリズムを考える:

function member_guess(G, x, v, k)
begin
1. $v_0 \leftarrow x$;
2. **for** $i \leftarrow 1$ **to** $k-1$ **do**
   2.1. G の頂点を 1 つ非決定的に（'$v_{i-1} \in S(i-1)$ に隣接する' と推測して）選び v_i とする;
   2.2. **if** $(v_{i-1}, v_i) \notin E(G)$ **then** 入力を拒否して停止 **end**
3. **end**;
4. **if** $v_{k-1} = v$ **then return true else** 拒否して停止 **end**

end

上記の関数 member_guess では，$v \in S(k-1)$ であっても，行 2.1 において v_i の非決定性の推測に失敗すると入力を拒否してアルゴリズムが終了してしまうが，$v \in S(k-1)$ であるなら推測に成功することも必ずあるので，非決定性の受理の定義より，$v \in S(k-1)$ なら入力は受理される（関数値は **true** になる）ことに注意する．この関数が使う領域量のうち n に依存するものは，i を記憶しておくための $O(\log n)$ だけである．

次に，頂点 u が与えられたとき $u \in S(k)$ かどうかを $|S(k-1)|$ だけを使って判定する非決定性アルゴリズムを示す（以下に示す関数 path_exist や関数 counting_GAP では，$|S(0)|, |S(k-1)|, |S(k)|$ などをあたかも変数のごとく扱っているが，これはあくまでもわかりやすさを重視した便法である）．

```
function path_exist(G, x, u, k, |S(k − 1)|)
begin
  1. p ← 0; /* S(k − 1) の元のうち，考慮済みのものの個数 */
  2. path_found ← false; /* x から u への道があったか？ */
  3. forall v ∈ V(G) do
    3.1. if member_guess(G, x, v, k) then /* v ∈ S(k − 1) のとき */
      3.1.1. p ← p + 1;
      3.1.2. if (v, u) ∈ E(G) then path_found ← true end
    3.2. end
  4. end;
  5. if p < |S(k − 1)| then 拒否して停止 else return path_found end
end
```

$|S(k-1)|$ が正しく求められているとしよう．関数 path_exist は，行 3.1.1 において $v \in S(k-1)$ であるような v の個数をカウントし，行 3.1.2 において v と u が隣接していたら（このとき $u \in S(k)$ である）変数 path_found を **true** にする．行 3 の forall 文が終了したとき $p = |S(k-1)|$ であれば，$S(k-1)$ のすべての頂点が v として考慮されたことを意味し，そのうちの少なくとも 1 つの v と u とが隣接していたら（この場合，$u \in S(k)$ である）行 3.1.2 によって変数 path_found の値は **true** になっており，どの v も u と隣接していなかったら（この場合，$u \notin S(k)$ である）path_found の値は初期値 **false** のまま変化していない．よって，path_found の値をそのまま path_exist の関数値とすればよい（行 5 の **else** 節）．$p < |S(k-1)|$ の場合には，$v \in S(k-1)$ であるにもかかわらず，行 3.1 において呼び出した member_guess が推測に失敗して停止してしまったために考慮しそこなった v が存在することを意味する．この場合，$u \in S(k)$ かどうかを正しく判定することができないのでアルゴリズムをそこで停止させる．以上の考察により，アルゴリズムが途中で停止することなく path_exist が計算した値は必ず正しい，ということがわかる．関数 path_exist は $O(\log n)$ 領域しか使わない．

最後に，M は $|S(0)|, |S(1)|, \ldots$ を次のように順次計算し，$|S(n-1)|$ を計算結果として出力して終了する．

```
procedure counting_GAP(G, x)
begin
  |S(0)| ← 1;
  for k ← 1 to n − 1 do /* n = |V(G)| */
    |S(k)| ← 0;
    forall u ∈ V(G) do
      if path_exist(G, x, u, k, |S(k − 1)|) then |S(k)| ← |S(k)| + 1 end
    end
  end;
  |S(n − 1)| を出力し，入力 (G, x) を受理して停止する
end
```

$|S(k)|$ を計算するにあたって M は $|S(k-1)|$ だけを記憶しておく（k の値ごとに計算用の領域を繰返し再利用する）．したがって，M が使う領域量は $O(\log n)$ で十分である．こうして（M が途中で停止することなく）求められた $|S(n-1)|$ が，x から到達可能な頂点の個数に等しいことは明らかであろう（$|S(k)|$ が，x から長さ k 以下の道に沿って到達可能な頂点の個数であることを k に関する帰納法で証明せよ）．すなわち，入力 (G, x) が M に受理されたときの出力は必ず正しい． また，どんな (G, x) に対しても，member_guess が v_i の推測にすべて成功して $p = |S(k-1)|$ となる

場合が必ず存在するので，そのとき path_exist も counting_GAP も途中で停止せず，M は (G,x) を受理して正しい最終計算結果を出力する．よって，任意の (G,x) に対し，$M(G,x)=|S(n-1)|$ であり，$S(n-1)$ の元は「x から到達可能な頂点」である． □

問 5.11. 任意の $f(n)$ に対して $\mathbf{DSPACE}(f(n))$ や $\mathbf{DTIME}(f(n))$ が補集合をとる演算で閉じていることを示せ．

この問は容易に示すことができるが，非決定性のクラスに対しても同じことが成り立つかどうかは長い間未解決であった．次の定理は $\mathbf{NSPACE}(f(n))$ についての肯定的な答である[†9]．特に，$f(n)=n$ という特別な場合は “文脈依存言語のクラスは補集合で閉じているか?” という問題であり，形式言語理論の研究が始まった 1950 年代以来の有名な未解決問題であったが，これによって解決された．

定理 5.7 (イマーマン・セレプトセーニィの定理)**.** $s(n)\geqq\log n$ が領域構成可能関数ならば

$$\mathbf{NSPACE}(s(n))=co\mathbf{NSPACE}(s(n))$$

が成り立つ．

証明 $L\in\mathbf{NSPACE}(s(n))$ で，L を受理する $s(n)$ 領域限定 NTM を M とする．L の補集合 L^c を受理する $s(n)$ 領域限定 NTM $\bar{M}$ が存在することを示せばよい．

定理 5.5 (3) の証明において述べたように，長さ n の入力 x に対して M が取り得る ID の個数はたかだか $c^{\log n+s(n)}$ である（$c>0$ は定数）．特に $s(n)\geqq\log n$ なら，これは $\leqq(c^2)^{s(n)}$ である．よって，補題 5.1 より，入力 x に対する M の様相グラフ $G_{M,x}$ の到達可能性問題の数え上げ版を解く $O(\log(c^2)^{s(n)})=O(s(n))$ 領域限定 NTM が存在する．

入力 x が与えられたら $\bar{M}$ は $G_{M,x}$ の到達可能性問題を補題 5.1 のように解く．すなわち，counting_GAP$(G_{M,x},Init_M(x))$ を実行する．$Init_M(x)$ は入力 x に対する M の初期 ID である．$\bar{M}$ は $G_{M,x}$ の全辺を記憶しておくわけではなく，$(I_1,I_2)\in E(G_{M,x})$ かどうかを知る必要が生じるたびに M の遷移関数を調べる．したがって，$\bar{M}$ が使う領域量は補題 5.1 の実行に必要な領域量 $O(s(n))$ だけで十分である．

counting_GAP$(G_{M,x},Init_M(x))$ を実行する過程で，ある $S(k)$ に受理 ID がカウントされたら（すなわち M が x を受理したら）$\bar{M}$ は x を拒否して停止する．M が受理 ID に入ることなく $S(n-1)$ の計算が終わったら x を受理する．

$x\in L(M)$ とすると $Init_M(x)\vdash_M^k I_f$ となる受理 ID I_f と整数 k が存在する．ゆえに，I_f は $S(k)$ にカウントされ，$\bar{M}$ における拒否の定義より $x\notin L(\bar{M})$ である．$x\notin L(M)$ とすると，初期 ID $Init_M(x)$ から到達可能な受理 ID は存在しないから $\bar{M}$ は途中で x を拒否して停止することなく $|S(0)|,|S(1)|,\ldots$ を順次計算し，$S(n-1)$ を計算し終わったところで x を受理して停止する．よって，$L(\bar{M})=L(M)^c$ である． □

最後に，帰納的言語と帰納的可算言語との関係について述べておく．まず，補集合をとる演算で閉じているかどうかという観点からいうと，

問 5.12. **REC** は補集合で閉じていることを示せ．

[†9] この定理はイマーマン (N.Immerman) とセレプトセーニィ (R.Szelepcsényi) がほとんど同時に独立に証明した:
N.Immerman, Nondeterministic space is closed under complementation, *SIAM J. Comput.* 17, pp.760–778, 1988.
R.Szelepcsényi, The method of forced enumeration for nondeterministic automata, *Acta Inform.* 26, pp.279–284, 1988.

は容易に示すことができる．一方，帰納的可算言語の補集合は帰納的言語と次のような密接な関係にある：

定理 5.8. $\mathbf{REC} = \mathbf{RE} \cap co\mathbf{RE}$.

証明 $\mathbf{REC} \subseteqq \mathbf{RE}$ は自明．また，問 5.12 より，$\mathbf{REC} \subseteqq co\mathbf{RE}$ である．

逆を示すために $L \in \mathbf{RE} \cap co\mathbf{RE}$ とすると，L を受理する DTM M と L^c を受理する DTM M^c が存在する．次のように動作する DTM M' を考える: 入力 x が与えられたら，M' は M と M^c の動作を交互に 1 ステップずつ模倣し，M が x を受理したら M' は x を受理し，M^c が x を受理したら M' は x を拒否する．どんな x も M または M^c に必ず受理されるから，M' は停止性 TM である．よって，$L \in \mathbf{REC}$. □

帰納的言語でない帰納的可算言語が存在することを後に示す．このことと定理 5.8 より，$\mathbf{RE}$ は補集合で閉じていないことが導かれる．

問 5.13. 補集合を取る演算で閉じている非決定性領域量のクラスの例として次のことを証明せよ：$co\mathbf{NSPACE}(1) = \mathbf{NSPACE}(1)$. これは定理 5.7 からは導かれないことに注意する．実は，有限オートマトンのクラスは共通部分を取る演算でも閉じている．

MEMO