4.3. 階層定理

定理4.4: 任意の制限時間 t_1, t_2 に対し,

$$\forall c>0,\ \overset{\infty}{\forall} n\,[c t_1(n)^2 \le t_2(n)] \rightarrow \mathrm{TIME}(t_1) \subsetneq \mathrm{TIME}(t_2).$$

制限時間 t が大きな関数であればあるほど, t 時間以内で計算可能な問題は増える.

4.3.1. IsProgram, evalの時間計算量

プログラムのコード化法を正確に定義することが重要.

（約束事） Σ^*型の変数名: $v1, v2, \ldots, pc$. v変数と呼ぶ.

Σ型の変数名: $u1, u2, \ldots$ u変数と呼ぶ.

入力変数としては常に $v1$ を用いる.

出力変数としても $v1$ を用いる.

標準形プログラムAのコード化

$\langle \overline{K}, \overline{L}, \overline{M}$, <文1のコード，文2のコード，...,文$k$のコード>>

K: case文の分岐の数

L, M: u変数，v変数の数.

文k: case文中のk番目の分岐に書かれている文.

各文kを5つ組$\langle op, a1, a2, a3, a4 \rangle$でコード化.

（例）　タイプ　文の形

(3) ui:=head(vj); pc:=k; ➔ $\langle \overline{3}, \overline{i}, \overline{j}, \overline{k}, \overline{\varepsilon} \rangle$

(12) if vi=s then pc:=k1 else pc:=k2 end-if; ➔ $\langle \overline{12}, \overline{i}, s, \overline{k}1, \overline{k}2 \rangle$

4.2.1.で定義した12個のタイプ

例4.10.

```
prog A(input v1: Σ*): Σ*;
var pc, v2: Σ*; u1: Σ;
begin
  pc:=1;
  while文
  halt(v1)
end.
```

while文の中身

```
while pc≠0 do
  case pc of
  1: u1:=head(v1); pc:=2;        タイプ3
  2: if u1=0 then pc:=3 else pc:=1 end-if    タイプ11
  3: v1:=0; pc:=0;     タイプ5
  end-case
end-while;
```

このプログラムのコードは

$$<\overline{3}, \overline{1}, \overline{2}, <<\overline{3}, \overline{1}, \overline{1}, \overline{2}, \varepsilon>, <\overline{11}, \overline{1}, \overline{0}, \overline{3}, \overline{1}>, <\overline{5}, \overline{1}, \overline{0}, \overline{0}, \varepsilon>>>$$

補題4.5: 適当な定数c_{isp}, d_{isp}に対して次の時間で*IsProgram*を計算するプログラムIsProgramが構成できる.

$$\forall a \in \Sigma^* \ [\mathrm{time\_IsProgram}(a) \le c_{\mathrm{isp}}|a| + d_{\mathrm{isp}}\ ]$$

略証:

与えられたコード a に対してIsProgramが調べること.

(a) a が$<\overline{K},\overline{L},\overline{M},<\overline{s_1}, \overline{s_2}, \ldots , \overline{s_N}>>$という形で, しかも $K = N$ か？

(b) 各s_kは正しい文のコードになっているか？

(c) 使われている変数番号, case分岐番号が範囲内に収まっているか？

上記の仕事が入力の長さの線形時間でできることは明らか.

（詳細はテキストp.117 – 118を参照. ）

補題4.6: 適当な定数c_{ev}, d_{ev}に対して次の時間で*eval* を計算するプログラムevalが構成できる.

$$\forall a,x \in\Sigma^* \; [\mathrm{time\_eval}(a, x)\leq c_{ev}|a|\mathrm{time\_a}(x) \times \max\{|a|, |x|, \mathrm{time\_a}(x)\}+ d_{ev}\,]$$

（補注）

a が表すプログラムに x を入力したときの実行時間 time_A(x)は，ふつうは$|a|$および$|x|$より大きい.

このとき，$\max\{|a|, |x|, \mathrm{time\_a}(x)\} = \mathrm{time\_a}(x)$

よって，$\mathrm{time\_eval}(a, x) \leqq c_{ev}|a|\mathrm{time\_a}(x)^2$

a に対応するプログラムの実行時間 t に対して，そのシミュレーションには$|a|^2t^2$ぐらいの時間がかかる。

(あるいはそれくらい時間をかければ十分).

補題4.6の略称

$(\forall\ a,x \in \Sigma^* \ [\mathrm{time\_eval}(a, x) \le c_{ev}|a|\mathrm{time\_a}(x) \times \max\{|a|, |x|, \mathrm{time\_a}(x)\} + d_{ev}\ ]\ )$

変数v1はAの
入力&出力

$\lfloor a \rfloor$ =Aは
{ Σ変数u1,…,uL
Σ*変数v1,…,vM
K 個のcase分岐

・evalの定義

$$\mathrm{eval}(a, x) = \begin{cases} f\_a(x) & \text{IsProgram}(a)\text{のとき,} \\ ? & \text{その他のとき.} \end{cases}$$

・eval(a, x)の計算に必要なこと

(1)IsProgramのチェック

(2) $\lfloor a \rfloor$ の入力 x に対する実行をシミュレートし, $f_a(x)$を求める.

シミュレーションのために変数*u*, *v*を用意.

変数u: プログラムA中の変数u1, ... , uLの値の管理: <u1, ... , uL>

変数v: プログラムA中の変数v1, ... , vMの値の管理: <v1, ... , vM>

uの初期値 = <0, ... , 0>

vの初期値 = <x, ε, ... , ε >　　最初のxはv1=xの値.

シミュレーション終了時の出力変数v1の値はvの第一引数の値.

シミュレーションプログラム

prog eval(input a, x);
% K, L, M, pcの値は1進表記の自然数
begin
  if ¬IsProgram(a) then halt(?);
  K:=get(a, $\overline{1}$);
入力文字列からK,L,Mの値を取得. ssはcase文の集合
  L:=get(a, $\overline{2}$); M=get(a, $\overline{3}$); ss:=get(a, $\overline{4}$);
  u:=<0, ... , 0>; v:=<x, ε, ... , ε >;
L個 M-1個
  pc:=$\overline{1}$;
  while pc $\neq$ 0 do
    s:=get(ss, pc); % case文の系列からpc番目の文を取得
    sに対応する文を実行（pcの更新も含む）
  end-while;
  halt(get(v, 1))
end.
変数v1の値を出力して終了

プログラムevalの時間計算量…シミュレーションの時間計算量

l_a: 入力aの長さ
l_x: 入力 x の長さ
K, L, M: aが表すプログラムA中の
case文の数, u変数の数, v変数の数
l_{maxv}: シミュレーション中のevalの内部変数vの長さの最大値

項目	計算時間	理由
4つのget	$4(c_{\mathrm{get}} + l_{\mathrm{a}} + d_{\mathrm{get}})$	例4.3
IsProgram	$c_{\mathrm{isp}} l_{\mathrm{a}} + d_{\mathrm{isp}}$	補題4.5
u,vの初期化	$c_{\mathrm{cr}}(L+M) + 2d_{\mathrm{cr}} + c_{\mathrm{put}}(\lvert<, \ldots ,>\rvert + l_{\mathrm{x}}) + d_{\mathrm{put}}$	
pcの初期化	1	
while文全体	$\mathrm{time\_A}(x)\{c_1(l_a + l_{\mathrm{maxv}}) + d_1\}$	
halt文のget	$c_{\mathrm{get}} l_{\mathrm{maxv}} + d_{\mathrm{get}}$	例4.3

プログラムevalの時間計算量...シミュレーションの時間計算量

l_a: 入力aの長さ
l_x: 入力 x の長さ
K, L, M: aが表すプログラムA中の
case文の数, u変数の数, v変数の数
l_{maxv}: シミュレーション中のevalの内部変数vの長さの最大値

項目	計算時間	理由
while文全体	time_A(x)$\{c_1(l_a + l_{\mathrm{maxv}})+d_1\}$	

⇨ whileループ1回分の計算: $c_1(l_a + l_{\mathrm{maxv}})+d_1$ 時間以内
ループを回る回数: time_A(x)
∴ while文全体では, time_A(x)$\{c_1(l_a + l_{\mathrm{maxv}})+d_1\}$

よって, 全体では, ある定数c_2, d_2に対し
time_eval(a, x) $\leqq c_2(l_a + l_{\mathrm{maxv}})$ time_A(x) $+d_2$

l_{maxv}の評価（内部変数vの長さの最大値）

t ステップ目でA中のv変数に保持している文字列の内，
最も長い長さを l_t とすると，

12個の基本命令には
[文字列＋文字列]はない

$l_0 = l_x$,

$l_t \leq \max\{|s|, l_{t\text{-}1}+1\} \leq \max\{l_a, l_{t\text{-}1}+1\}$

（ただし，s は a 中の最も長い文字列定数）

したがって，

$l_{\max} = \max\{l_t : 0 \leq t \leq \mathrm{time\_A}(x)\}$

とおくと，

$l_{\max} \leq \max\{l_a, l_x\} + \mathrm{time\_A}(x)$

v の値は<v1, ... , vM>なので

$$l_{\mathrm{maxv}} \leq M\, l_{\max} \leq M(\max\{l_a, l_x\} + \mathrm{time\_A}(x))$$
$$\leq l_a\ \ 2\max\{l_a, l_x, \mathrm{time\_A}(x)\}$$

プログラムevalの時間計算量: ある定数 c_2, d_2 に対し

$$\text{time\_eval}(a, x) \leqq c_2(l_a + l_{\text{maxv}})\ \text{time\_A}(x) + d_2$$

l_{maxv}の評価(内部変数vの長さの最大値):

$$l_{\text{maxv}} \leqq l_a\ 2\max\{l_a, l_x, \text{time\_A}(x)\}$$

$$\begin{aligned}
&\text{time_eval}(a, x) \\
&\quad \leqq c_2(l_a + 2\, l_a \max\{l_a, l_x, \text{time_A}(x)\}) \times \text{time_A}(x) + d_2 \\
&\quad = c_2 l_a\, \text{time_A}(x) \times (1+2\max\{l_a, l_x, \text{time_A}(x)\}) + d_2 \\
&\quad \leqq 3\, \mathrm{c}_2 l_a\, \text{time_A}(x) \times 2\max\{l_a, l_x, \text{time_A}(x)\} + d_2 \\
&\quad = c_{\text{ev}}|a|\, \text{time_A}(x) \times 2\max\{|a|, |x|, \text{time_A}(x)\} + d_{\text{ev}}
\end{aligned}$$

証明終

例4.11: 文“ui:=tail(vj); pc:=k;”をシミュレートするのに要する時間

行うこと	ステップ数の概算
s:=get(ss, pc);	$c_{\mathrm{get}}\|ss\|+d_{\mathrm{get}}=O(l_a)$
文のタイプの判定	O(1)
i:=get(s, 2)	$c_{\mathrm{get}}\|s\|+d_{\mathrm{get}}=O(l_a)$
j:=get(s, 3)	$c_{\mathrm{get}}\|s\|+d_{\mathrm{get}}=O(l_a)$
k:=get(s, 4)	$c_{\mathrm{get}}\|s\|+d_{\mathrm{get}}=O(l_{\mathrm{a}})$
tmp:=get(v, j)	$c_{\mathrm{get}}\, l_v+d_{\mathrm{get}}=O(l_v)$
tmp:=tail(tmp)	O(1)
put(u, i, tmp)	$c_{\mathrm{put}}(l_a+\|\mathrm{tmp}\|)+d_{\mathrm{put}}=O(l_{\mathrm{a}})$
pc:=k	O(1)

よって，time$\leq c(l_a+l_v)+d$,　c,dはl_a , l_v と独立

この c, d は文のタイプごとに違う

そこで，最大のものをc_1 , d_1 とすると，文のタイプに関係なく

　１ステップあたりの時間　$\leq c_1\,(l_a+l_v)+d_1$

$$\leq c_1\;(l_a+l_{\mathrm{maxv}})+d_1$$

今までの議論の応用として，次の関数の計算時間を評価

各$a, x \in \Sigma^*, t \in$ N に対して

$$eval\_in\_time(a, x, \bar{t}) = \begin{cases} f\_a(x), & IsProgram(a) \text{ かつ} \lfloor a \rfloor (x) \text{が} \\ & t \text{ 時間以内に停止するとき} \\ ? & \text{その他のとき} \end{cases}$$

補題4.7: 適当な定数 $c_{\mathrm{evt}}, d_{\mathrm{evt}}$に対して，次の計算量で *eval_in_time*を計算するプログラムeval-in-timeが構成できる.

$\forall a, x \in \Sigma^*, t \in$ N [time_eval-in-time$(a, x, \bar{t}) \leq c_{\mathrm{evt}}|a|^2 t \max\{|x|, t\} + d_{\mathrm{evt}}$]

（略証）

プログラム$\lfloor a \rfloor (x)$の計算時間が t を超えると強制終了するためのカウンタを用いてシミュレーションを実行.

それ以外はevalの評価と同様.

証明終

4.3.2. 階層定理の証明

定理4.4: 任意の制限時間 t_1, t_2 に対し,
$\forall c>0, \overset{\infty}{\forall} n\ [c t_1(n)^2 \leq t_2(n)] \rightarrow \mathrm{TIME}(t_1) \subsetneq \mathrm{TIME}(t_2)$.

仮定より，明らかに$t_1=O(t_2)$であるから，$\mathrm{TIME}(t_1) \subseteq \mathrm{TIME}(t_2)$.
よって，$\mathrm{TIME}(t_2) - \mathrm{TIME}(t_1) \neq \phi$であることを示せばよい.
すなわち，$O(t_2)$時間では認識できるが，$O(t_1)$時間では認識できない集合の存在を示せばよい.

プログラムA=$\lfloor a \rfloor$ に
x=<a,w>を入力すると、
$|x| < \sqrt{t_2(|x|)/|a|}$
$t = \sqrt{t_2(|x|)/|a|}$ 以内にacceptしない

DIAG =
{<a,w>: 次の3条件を満たす.
(a) *IsProgram*(a)
(b) $l < t$
(c) eval-in-time(a, <a,w>, $\overline{t}$) $\neq$ accept }
ただし，x=<a,w>, $l=|x|$, $t = [\sqrt{t_2(|x|)/|a|}]$
([ ]は切り捨て)

補題4.8: $DIAG \notin \mathrm{TIME}(t_1)$

補題4.9: $DIAG \in \mathrm{TIME}(t_2)$

補題4.9の証明:

$x \in DIAG$ を調べるプログラムDIAGの計算時間.

$l = |x|,\ \ t = [\sqrt{t_2(l)}\ /|a|]$

(1) xが$<a,w>$の形をしているか？

(2) $IsProgram(a)$?

(3) $l < t$?

(4) $eval\text{-}in\text{-}time(a, <a,w>, \overline{t}) = eval\text{-}in\text{-}time(a, x, \overline{t}) \neq \mathrm{accept}$?

(1)(2): c_1,d_1を定数として $c_1 l + d_1$時間で判定可能

(3) $l < t$? $t = [\sqrt{t_2(l)}\ /|a|]$はO($t_2(l)$)時間で計算可能.

→演習問題4.12

(a) 2進表記の自然数nから1進表記$\overline{n}$への変換 : O(n)時間

逆の変換 inv-transもO(n)でできる.

(b) n から$\sqrt{n}$ はO($|n|^2$)時間で計算可能→プログラムsqrt

(c) プログラムinv-trans, div, sqrt, transを用いて

$\sqrt{t_2(l)}\ /|a|$の1進表記を求める.

(4) eval-in-time(a,x,$\bar{t}$) ≠ accept?の判定

補題4.7より，$c_{\mathrm{evt}}|a|^2 t \max\{l,t\}+d_{\mathrm{evt}}$時間で判定可能.

$$c_{\mathrm{evt}}|a|^2[\sqrt{t_2(l)}/|a|] \max\{l, [\sqrt{t_2(l)}/|a|]\}+d_{\mathrm{evt}}$$

$$\leq c_{\mathrm{evt}}|a| \max\{l\sqrt{t_2(l)}, t_2(l)/|a|\}+d_{\mathrm{evt}}$$

$$= c_{\mathrm{evt}} \max\{l|a|\sqrt{t_2(l)}, t_2(l)\}+d_{\mathrm{evt}} = \mathrm{O}(t_2(l)).$$

結局，(1)-(4)は$\mathrm{O}(t_2(l))$時間で判定可能.

よって，$DIAG \in \mathrm{TIME}(t_2)$.

補題4.9の証明終

補題4.8 ($DIAG \notin \mathrm{TIME}(t_1)$)の証明:

$DIAG \in \mathrm{TIME}(t_1)$として矛盾を導く.

・$DIAG$ を$O(t_1)$時間で認識するプログラムをA_0, コードを a_0とする.

・$\mathrm{time\_}A_0(l) \leqq c\, t_1(l) + d$ を満たす定数c, d が存在する.

$c_0=c+d$とおく. ただし, $c_0>1$.

→ $\mathrm{time\_}A_0(l) \leqq c_0\, t_1(l)$ .... (1) (if $1 \leqq t_1(l)$)

c_0は定数

定理の仮定: $\forall c>0, \overset{\infty}{\forall} n\, [c t_1(n)^2 \leqq t_2(n)]$ より、

n を十分大きく取ると,

$$(|a_0|^2 c_0^2)\, t_1(n)^2 \leqq t_2(n) \rightarrow c_0\, t_1(n) \leqq \sqrt{t_2(n)}\, /|a_0|$$

また, 自然な制限時間の条件より, $n \leq t_1(n)$

そこで, 十分長い文字列 w_0 を考えると (ただし, $l_0=|<a_0,w_0>|$)

$$c_0\, t_1(l_0) \leq [\sqrt{t_2(l_0)}\, /|a_0|\,]$$

$$t_1(l_0) \leq [\sqrt{t_2(l_0)}\, /|a_0|\,] / c_0 \leq [\sqrt{t_2(l_0)}\, /|a_0|\,]$$

よって,

$$l_0 \leq [\sqrt{t_2(l_0)}\, /|a_0|\,] \text{ ....(2)}$$

(1) $\mathrm{time\_A}_0(l) \le c_0 t_1(l)$ —— 一般の l について

(2) $l_0 \le [\sqrt{t_2(l_0)}/|a_0|]$ —— 十分長い l_0=<a_0,w_0>

$$<a_0,w> \in DIAG \leftrightarrow l \leqq t \wedge \text{eval-in-time}(a_0,<a_0,w>,\bar{t}) \neq \text{accept}$$

$$(w \in \Sigma^*,\ l=|<a_0,w>|,\ t=[\sqrt{t_2(l)}/|a_0|\ ])$$

w_0に対しては第1の条件 $l_0 \le t_0=[\sqrt{t_2(l_0)}/|a_0|\ ]$が満たされているので,

$<a_0,w_0> \in DIAG \leftrightarrow \text{eval-in-time}(a_0,<a_0,w_0>,\bar{t}_0) \neq \text{accept}$ .... (3)

(1), (2)より,

$$\mathrm{time\_A}_0(<a_0,w_0>) \le c_0 t_1(l_0) \le t_0 = [\sqrt{t_2(l_0)}/|a_0|\ ]$$

すなわち, $\mathrm{time\_A}_0(<a_0,w_0>) \le t_0$

これは$<a_0,w_0>$をプログラムA_0に入力したとき,

計算は必ず t_0時間以内に終ることを意味しているから,

eval-in-timeでの制限時間 t_0 は本質的ではない.

つまり, $\text{eval-in-time}(a_0, <a_0,w_0>, t_0) = \text{eval}(a_0, <a_0,w_0>)$

$\langle a_0, w_0 \rangle \in DIAG \leftrightarrow$ eval-in-time$(a_0, \langle a_0, w_0 \rangle, \overline{t_0}) \neq$ accept .... (3)

に

eval-in-time$(a_0, \langle a_0, w_0 \rangle, t_0) =$ eval$(a_0, \langle a_0, w_0 \rangle)$

をあてはめると、

$$\langle a_0, w_0 \rangle \in DIAG \leftrightarrow \text{eval}(a_0, \langle a_0, w_0 \rangle) \neq \text{accept}$$

$\leftrightarrow$ A_0が$\langle a_0, w_0 \rangle$をacceptしない.

これは,「 A_0が$DIAG$を認識する」という仮定に矛盾.

（証明終）

対角線論法に基づく解釈

$F_1 = O(t_1)$の時間計算量をもつ認識プログラムすべての集合

$= \{A_1, A_2, \ldots\}$

それらのプログラムのコードを $a_1, a_2, \ldots$ とする.

各a_iごとに適当な定数c_iを考えると,

$$\mathrm{time\_A}_i(l) \leq c_i\, t_1(l)$$

が成立. さらに, 各a_i, c_i に対し, 十分長いw_i を取ると,

$$c_i t_1(l_i) \leq [\sqrt{t_2(l_i)} / |a_i|\,], \quad l_i \leq [\sqrt{t_2(l_i)} / |a_i|\,]$$

とできる. 各プログラムの入力w_iに対する出力の表を作ると,

	w_1	w_2	w_3	....	w_k
A_1	A	R	A	....	A
A_2	R	R	R	....	A
A_3	A	A	A	....	R
......	......	......	......	......	......
A_k	R	R	A	....	A

$A_i(w_i)$の値

対角線で
RとAが
逆

w_1	w_2	w_3	....	w_k
R				
	A			
		R		
			....	
				R

$w_i \in DIAG$?の答

$DIAG$を認識するプログラムはこの表に登場できない…矛盾

例4.12: TIME(n^2) $\subsetneq$ TIME(n^5)

$\Leftarrow$ $\forall c>0, \forall n \geq c$ [$c(n^2)^2 \leq n^5$]

要するに,

$$\liminf_{n\to\infty} \frac{t_2(n)}{t_1(n)^2} = \infty$$

となれば，階層定理よりTIME(t_1) $\subsetneq$ TIME(t_2)