4.3. 階層定理(続き)

定理4.4: 任意の制限時間 t_1, t_2 に対し,

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

DIAG = {<*a*,*w*>: 次の3条件を満たす.

(a) $IsProgram(a)$

(b) $l < t$

(c) eval-in-time(a, $\langle a,w \rangle$, $\overline{t}$) $\neq$ accept }

ただし, $x=\langle a,w \rangle$, $l=|x|$, $t = [\sqrt{t_2(|x|)}\ /\ |a|\ ]$

([ ]は切り捨て)

プログラムA= $\lfloor a \rfloor$ に $x=\langle a,w \rangle$ を入力すると、

$|x| < \sqrt{t_2(|x|)}\ /\ |a|$ $\quad t = \sqrt{t_2(|x|)}\ /\ |a|$ 以内にacceptしない

4.3. Hierarchy Theorem (Cont'd)

Theorem 4.4 For any time limits t_1 and t_2, we have

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

DIAG $= \{\langle a,w\rangle$: the following three conditions are satisfied:

(a) $IsProgram(a)$

(b) $l < t$

(c) eval-in-time$(a, \langle a,w\rangle, \overline{t}) \neq$ accept }

where, $x=\langle a,w\rangle$, $l=|x|$, $t = [\sqrt{t_2(l)} / |a|]$ [ ] denotes round-off

If we input $x=\langle a,w\rangle$ to a program a as an input, $|x| < \sqrt{t_2(|x|)} / |a|$ and it does not accept before time $t=\sqrt{t_2(|x|)} / |a|$.

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.
  $\rightarrow$ time_ A$_0(l) \leqq c_0\, t_1(l)$ .... (1) (c_0;定数)
- 定理の仮定 $\forall c > 0, \overset{\infty}{\forall} l[ct_1(l)^2 \leq t_2(l)]$
  より、lを十分大きくとると、 $c_0 t_1(l) \leq [\sqrt{t_2(l)} / |a_0|]$ ....(2)
- t_1は自然な制限時間であるから、l_0=|<a_0,w_0>|を十分長くすると

$$l_0 \leq [\sqrt{t_2(l_0)} / |a_0|] (\equiv t_0) \quad \text{...........(3)}$$

- (3) と$DIAG$の定義より

<a_0,w_0>$\in DIAG \leftrightarrow$ eval-in-time(a_0,<a_0,w_0>,$\overline{t}_0$) $\neq$ accept ....(4)

Proof of $\boxed{\text{Lemma 4.8: } DIAG \notin \mathrm{TIME}(t_1)}$

To derive contradictions, we assume $DIAG \in \mathrm{TIME}(t_1)$.

- Let A_0 be the program that recognizes *DIAG* in $O(t_1)$ time, with code a_0. $\rightarrow$ time_ $A_0(l) \leqq c_0\, t_1(l)$ .... (1) (c_0;constant)
- By the assumption of theorem $\forall c > 0, \overset{\infty}{\forall} l[ct_1(l)^2 \le t_2(l)]$
  for sufficiently large l, $c_0 t_1(l) \le [\sqrt{t_2(l)} / |a_0|]$ ....(2)
- Since t_1 is a natural limit, for sufficiently long $l_0 = |<a_0, w_0>|$,

$$l_0 \le [\sqrt{t_2(l_0)} / |a_0|] (\equiv t_0) \qquad \text{...........(3)}$$

- By (3) and definition of *DIAG*,

$<a_0, w_0> \in DIAG \leftrightarrow \text{eval-in-time}(a_0, <a_0, w_0>, \bar{t}_0) \neq \text{accept}$ ....(4)

一般の l について

(1) $\mathrm{time\_A}_0(l) \le c_0 t_1(l)$

十分長い l_0=<a_0,w_0>

(2) $\mathrm{c}_0 t_1(l_0) \le [\sqrt{t_2(l_0)} / |a_0|] = t_0$

(4) $< a_0, w_0 > \in DIAG \leftrightarrow \text{eval-in-time}(a_0, < a_0, w_0 >, \overline{t_0}) \ne \text{accept}$

(1)(2)より，$\mathrm{time\_A}_0(< a_0, w_0 >) \le c_0 t_1(l_0) \le [\sqrt{t_2(l_0)} / |a_0|] = t_0$ .....(5)

(5) より、十分長い l_0=<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 時間以内に終わる。つまり

eval-in-timeでの制限時間 t_0 は本質的ではない.

→ eval-in-time(a_0, <a_0,w_0>, $\overline{t_0}$) = eval(a_0, <a_0,w_0>)

(4) に eval-in-time(a_0, <a_0,w_0>, $\overline{t_0}$) = eval(a_0, <a_0,w_0>) を代入：

<a_0,w_0> $\in$ $DIAG$ $\leftrightarrow$ eval(a_0, <a_0,w_0>) $\ne$ accept
$\leftrightarrow$ A_0が<a_0,w_0>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）

For general l

For sufficiently long l_0=<a_0,w_0>

(1) $\text{time\_A}_0(l) \le c_0 t_1(l)$

(2) $c_0 t_1(l_0) \le [\sqrt{t_2(l_0)} / |a_0|] = t_0$

(4) $< a_0, w_0 > \in DIAG \leftrightarrow \text{eval-in-time}(a_0, < a_0, w_0 >, \overline{t_0}) \neq \text{accept}$

By (1)(2), $\text{time\_A}_0(< a_0, w_0 >) \le c_0 t_1(l_0) \le [\sqrt{t_2(l_0)} / |a_0|] = t_0$ .....(5)

From (5), program A_0 halts in t_0 time for sufficiently long l_0=<a_0,w_0>, which means

the time limit t_0 in eval-in-time is not essential.

→ eval-in-time(a_0, <a_0,w_0>, t_0) = eval(a_0, <a_0,w_0>)

Substitute eval-in-time(a_0, <a_0,$\overline{w_0}$>, t_0) = eval(a_0, <a_0,w_0>) to (4):

$$<a_0,w_0> \in DIAG \leftrightarrow \text{eval}(a_0, <a_0,w_0>) \neq \text{accept}$$
$$\leftrightarrow A_0 \text{ does not accept } <a_0,w_0>.$$

Which contradicts the assumption that "A_0 recognizes *DIAG*." Q.E.D.

対角線論法に基づく解釈

$F_1 = \mathrm{O}(t_1)$の時間計算量をもつ認識プログラムすべての集合
$= \{\mathrm{A}_1, \mathrm{A}_2, \ldots\}$
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,

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

$x=<a,w>,\ l = |x|$

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

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

とできる. 各プログラムA_iの入力x_iに対する出力の表を作ると,

	x_1	x_2	x_3	....	x_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

$\mathrm{A}_i(x_i)$の値

x_1	x_2	x_3	....	x_k
R				
	A			
		R		
			....	
				R

$x_i \in DIAG$?の答

対角線で
RとAが逆

$DIAG$を認識するプログラムはこの表に登場できない…矛盾

Interpretation based on Diagonalization

F_1 = a set of all recognizing programs of time complexity $O(t_1)$

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

Let their program codes be $a_1, a_2, \ldots$.

$x=\langle a,w\rangle$, $l = |x|$

Considering an appropriate constant c_i for each a_i, we have

time_$A_i(l) \le c_i t_1(l)$

Moreover, we can take sufficiently long w_i for each a_i and c_i s.t.

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

Putting the outputs of A_i for input x_i in the table:

	x_1	x_2	x_3	....	x_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

values of $A_i(x_i)$

x_1	x_2	x_3	....	x_k
R				
	A			
		R		
			....	
				R

answer to $x_i \in DIAG$?

Compare Diagonals

This table can't include a program recognizing *DIAG*...contradiction.

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)

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

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

If we have

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

then the hierarchy theorem tells us that TIME(t_1) $\subsetneq$ TIME(t_2)

第5章 代表的な計算量クラス

5.1. 代表的な時間計算量クラス

$$\mathrm{P} \equiv \bigcup_{p:\text{多項式}} \mathrm{TIME}(p(l))$$

$$\mathrm{E} \equiv \bigcup_{c>1} \mathrm{TIME}(2^{cl})$$

$$\mathrm{EXP} \equiv \bigcup_{p:\text{多項式}} \mathrm{TIME}(2^{p(l)})$$

C集合: 計算量クラスCに入る集合.
C問題: C集合の認識問題

ある問題がPに入っていないなら、
現実的には手に負えない…

Chapter 5
Representative Complexity Classes

5.1. Representative time complexity classes

$$\mathrm{P} \equiv \bigcup_{p:\mathrm{polynomial}} \mathrm{TIME}(\mathrm{p}(l))$$

$$\mathrm{E} \equiv \bigcup_{c>1} \mathrm{TIME}(2^{cl})$$

$$\mathrm{EXP} \equiv \bigcup_{p:\mathrm{polynomial}} \mathrm{TIME}(2^{p(l)})$$

C set : set in the complexity class C.
C problem : problem of recognizing a C set.

Problems not in P are intractable
from the practical viewpoint…

例5.1: クラスP, E, EXPでは，多項式時間程度の違いは問題ではない.

P: 多項式 × 多項式→多項式

E: 2の線形乗 × 多項式 → 2の線形乗

EXP: 2の多項式乗 × 多項式 → 2の多項式乗

例5.2: PRIMEの計算量クラス

例4.7 → PRIME $\in$ TIME(2^l)

故に，PRIME $\in$ E

余談: 2002年に $O(l^6)$ のアルゴリズムが考案されたので、今ではP

定義5.1. T: 制限時間の集合

$\bigcup_{t \in \mathrm{T}} \mathrm{TIME}(t)$: T時間計算量クラス

→これをTIME(T)と表す.

定理5.1: (1) P = $\cup_{c>0}\mathrm{TIME}(l^c)$, (2) EXP = $\cup_{c>0}\mathrm{TIME}(2^{l^c})$

Ex.5.1: Polynomial makes no serious difference in the classes P, E, EXP.

P: polynomial × polynomial → polynomial
E: linear power of 2 × polynomial → linear power of 2
EXP: poly. power of 2 × poly. → poly. power of 2

Ex.5.2: Complexity class of PRIME
Ex.4.7 → PRIME ∈ TIME(2^l)
Thus, PRIME ∈ E

$O(l^6)$ time algorithm put it in P!!

Def.5.1: T: set of time limits

$\bigcup_{t \in T}$ TIME(t): T time complexity class
→It is denoted by TIME(T).

Theorem5.1 (1) P = $\bigcup_{c>0}$ TIME(l^c), (2) EXP = $\bigcup_{c>0}$ TIME(2^{l^c})

定理5.1: (1) $\mathrm{P} = \cup_{c>0}\mathrm{TIME}(l^c)$,　　(2) $\mathrm{EXP} = \cup_{c>0}\mathrm{TIME}(2^{l^c})$

証明: (2)の証明は省略.

T_1: l^cという形の多項式の集合.

T_2: 多項式の全体

→ $T_1 \subseteq T_2$ なので, $\mathrm{TIME}(T_1) \subseteq \mathrm{TIME}(T_2)$

p: 任意の多項式（pはT_2の任意の要素）

多項式pの最大次数をkとすると, $p(l) = O(l^k)$

定理4.3より,

$\mathrm{TIME}(p(l)) \subseteq \mathrm{TIME}(l^k) \subseteq \mathrm{TIME}(T_1)$

したがって, $\mathrm{TIME}(T_1) = \mathrm{TIME}(T_2)$

証明終

Theorem 5.1: (1) P $= \cup_{c>0}$ TIME(l^c), (2) EXP $= \cup_{c>0}$ TIME(2^{l^c})

Proof: The proof of (2) is omitted.

T_1: set of polynomials of the form of l^c.

T_2: set of all polynomials

→ since $T_1 \subseteq T_2$, TIME(T_1) $\subseteq$ TIME(T_2)

p: arbitrary polynomial (p is any element of T_2)

if the maximum degree of a polynomial p is k, $p(l) = O(l^k)$

From Theorem 4.3,

TIME($p(l)$) $\subseteq$ TIME(l^k) $\subseteq$ TIME(T_1)

Therefore, TIME(T_1) = TIME(T_2)

Q.E.D.

例5.3. 命題論理式評価問題(PROP-EVAL)

入力：<F, < $a_1, a_2, \ldots, a_n$ >>

Fは拡張命題論理式 $\wedge \vee \neg \rightarrow \leftrightarrow$

($a_1, a_2, \ldots, a_n$)は F に対する真理値割り当て

質問： $F(a_1, a_2, \ldots, a_n) = 1$?

(x,y)	$x \rightarrow y$ $(\neg x \vee y)$	$x \leftrightarrow y$ $((x \rightarrow y) \wedge (y \rightarrow x))$
(0,0)	1	1
(0,1)	1	0
(1,0)	0	0
(1,1)	1	1

Ex.5.3. Problem of evaluating propositional expression(PROP-EVAL)

Input : $<F, < a_1, a_2, \ldots , a_n >>$

F is an extended prop. expression

$(a_1, a_2, \ldots , a_n)$ is a truth assignment to F

Question : $F(a_1, a_2, \ldots , a_n) = 1$?

(x,y)	$x \rightarrow y$ $(\neg x \vee y)$	$x \leftrightarrow y$ $((x \rightarrow y) \wedge (y \rightarrow x))$
(0,0)	1	1
(0,1)	1	0
(1,0)	0	0
(1,1)	1	1

例5.3. 命題論理式評価問題(PROP-EVAL)

入力: $<F, < a_1, a_2, \ldots , a_n >>$
Fは拡張命題論理式 $\wedge \vee \neg \rightarrow \leftrightarrow$
$(a_1, a_2, \ldots , a_n)$はFに対する真理値割り当て

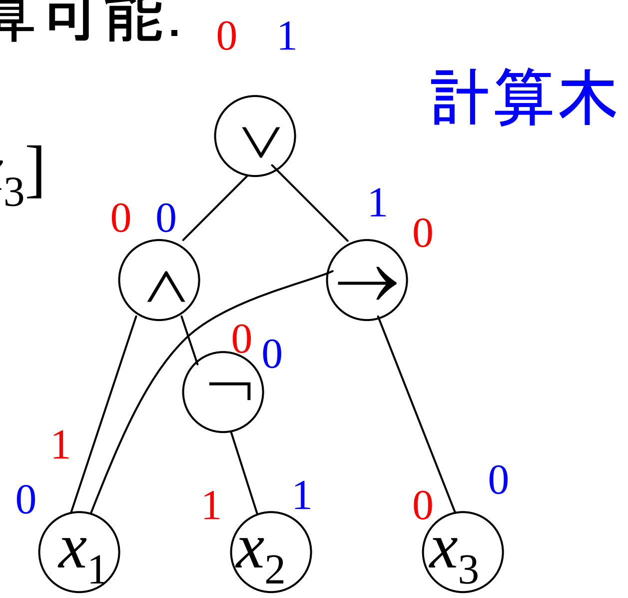
質問: $F(a_1, a_2, \ldots , a_n) = 1$?

拡張命題論理式 F がコード化されたもの $\lceil F \rceil$ から計算木を作る.
計算木は$O(|\lceil F \rceil|^3)$時間で構成できる.
計算木が得られていれば, ボトムアップ式で
$F(a_1, a_2, \ldots , a_n)$ の値は容易に計算可能.

例: $F(x_1, x_2, x_3) = [x_1 \wedge \neg x_2] \vee [x_1 \rightarrow x_3]$

$F(0,1,0)=1$

$F(1,1,0)=0$



よって PROP-EVAL ∈ P

Ex.5.3. Problem of evaluating propositional expression(PROP-EVAL)

Input : $<F, < a_1, a_2, \ldots , a_n >>$

F is an extended prop. expression

$(a_1, a_2, \ldots , a_n)$ is a truth assignment to F

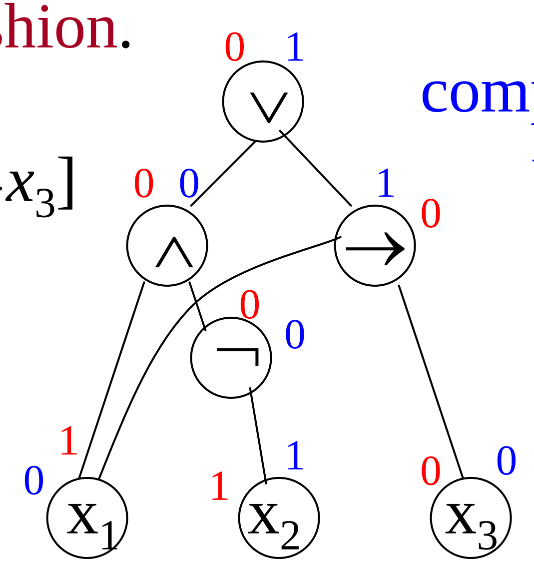
Question : $F(a_1, a_2, \ldots , a_n) = 1$?

Construct a computation tree from a code $\lceil F \rceil$ of ext. prop. expression

It is built in time $O(|\lceil F \rceil|^3)$.

If computation tree is available, we can easily obtain the value $F(a_1, a_2, \ldots , a_n)$ in a bottom-up fashion.

Ex. : $F(x_1, x_2 , x_3) = [x_1 \wedge \neg x_2] \vee [x_1 \rightarrow x_3]$



computation tree

$F(0,1,0)=1$

$F(1,1,0)=0$

Hence PROP-EVAL ∈ P

例5.3. 命題論理式充足性問題:2和積形(2SAT)

入力:$<F>$ Fは2和積形命題論理式

質問: $F(a_1, a_2, \ldots, a_n) = 1$を満たす割り当てがあるか?

和積形:

$F = (\bullet \vee \bullet \vee \ldots \vee \bullet) \wedge (\bullet \vee \ldots \vee \bullet) \wedge \ldots \wedge (\ldots)$

- リテラルの論理和の論理積で表現されたもの

k和積形(k SAT)

- 和積形の各論理和が k 個のリテラルを含む

ちょうど/たかだか

- 3SAT, 4SAT も同様に定義できる。
- SAT: 各論理和のリテラルの個数に制限がないもの
- ExSAT: 入力が拡張命題論理式(→や↔も許す)

Ex. 5.3. 2-Satisfiability (2SAT)

Input : $<F>$ F is 2-conjunctive normal form

Question : Is there any assignment such that $F(a_1, a_2, \ldots , a_n) = 1$?

Conjunctive Normal Form (CNF)

$F = (\bullet \lor \bullet \lor \ldots \lor \bullet) \land (\bullet \lor \ldots \lor \bullet) \land \ldots \land (\ldots)$

- described by $\land$ of $\lor$ of literals.

k SAT

- Each closure contains *k* literals

<u>exactly</u>/at most

- We can define 3SAT, 4SAT similarly.
- SAT consists of any CNF.
- ExSAT consists of any extended propositional expression.

例5.4: 到達可能性問題(ST-CON)

入力: <G,s,t> : 無向グラフG, $1 \leqq s,t \leqq n(=|G|)$

質問: G上で s から t への道があるか?

- 閉路とは、始点と終点が同じである路
- オイラー閉路とは、すべての辺を一度づつ通る閉路
- ハミルトン閉路とは、すべての頂点を一度づつ通る閉路

例5.4: 一筆書き閉路問題(DEULER)

入力: <G>: 有向グラフG

質問: Gはオイラー閉路をもつか?

例5.5: ハミルトン閉路問題(DHAM)

入力: <G>: 有向グラフG

質問: Gはハミルトン閉路をもつか?

Ex. 5.4: Graph reachability problem (ST-CON)

Input: <G,s,t> : an undirectd graph G, $1 \leqq s,t \leqq n(=|G|)$

Question: Does G have a path from s to t?

- Cycle is a path that shares two endpoints.
- Euler cycle is a cycle that visits all edges once.
- Hamiltonian cycle is a cycle that visits all vertices once.

Ex. 5.4: Euler cycle problem (DEULER)

Input: <G>: a directed graph G

Question: Does G have an Euler cycle?

Ex. 5.5 Hamiltonian cycle problem (DHAM)

Input: <G>: a directed graph G

Question: Does G have a Hamiltonian cycle?

知られていること:

- 以下の問題はP:
  - ✓ PROP-EVAL, 2SAT, ST-CON, DEULER
- 以下の問題はEであることはわかっているが…
  - ✓ 3SAT, DHAM

PとEの間にある(?)クラスNP

It is known that:

- The following problems are in P:
  - ✓ PROP-EVAL, 2SAT, ST-CON, DEULER

- The following problems are in E, but…
  - ✓ 3SAT, DHAM

The class NP between P and E?