

I216 Computational Complexity  
and  
Discrete Mathematics

by

Prof. Ryuhei Uehara

and

Prof. Atsuko Miyaji

# 1216 計算量の理論と離散数学

上原隆平、宮地充子

# Computational Complexity

- Goal 1:
  - “*Computable Function/Problem/Language/Set*”
- Goal 2:
  - How can you show “*Difficulty of Problem*”
    - There are *intractable* problems even if they are computable!
      - because they require too many resources (time/space)!
    - Technical terms;
      - The class NP, P≠NP conjecture, NP-hardness, reduction

# 計算量の理論

- ゴール1:
  - “計算可能な関数/問題/言語/集合”
- ゴール2:
  - 「問題の困難さ」を示す方法を学ぶ
    - 計算可能な問題であっても、手におえない場合がある！
      - 計算に必要な資源(時間・領域)が多すぎる時
    - 関連する専門用語;
      - クラスNP,  $P \neq NP$ 予想, NP困難性, 還元

# 5. Computational Complexity

## 5.1. Time Complexity Classes

### 5.1.3. Time complexity of a problem

**Definition:** For a function  $t(n)$  over natural numbers, the set of all sets (i.e. recognition problems) with time complexities  $O(t(n))$  is called  **$O(t(n))$ -time complexity class**, and it is denoted by **TIME( $t(n)$ )**. Such a function  $t(n)$  is called a time limit.

## 5.2. Representative time complexity classes

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

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

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

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...

# 5. 計算量の理論

## 5.1. 計算時間の評価

### 5.1.3. 問題の時間計算量

**定義:** 自然数上の関数  $t(n)$  に対して, 時間計算量  $O(t(n))$  の集合(認識問題)全体の集合を  **$O(t(n))$ 時間計算量クラス** とよび,  **$TIME(t(n))$**  とかく. こうした関数  $t(n)$  は制限時間と呼ぶ.

## 5.2. 代表的な計算量クラス

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

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

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

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...

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

P: polynomial  $\times$  polynomial  $\rightarrow$  polynomial

E: linear power of 2  $\times$  polynomial  $\rightarrow$  linear power of 2

EXP: poly. power of 2  $\times$  poly.  $\rightarrow$  poly. power of 2

Ex.5.2: Complexity class of PRIME

Ex.4.7  $\rightarrow$  PRIME  $\in$  TIME( $2^l$ )

Thus, PRIME  $\in$  E

$O(l^6)$  time algorithm puts it into P!!

**Def.5.1:** T: set of time limits

$\bigcup_{t \in T} \text{TIME}(t)$ : T time complexity class

$\rightarrow$  It is denoted by TIME(T).

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

**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  $\times$  多項式  $\rightarrow$  多項式

E: 2の線形乗  $\times$  多項式  $\rightarrow$  2の線形乗

EXP: 2の多項式乗  $\times$  多項式  $\rightarrow$  2の多項式乗

例5.2: PRIMEの計算量クラス

例4.7  $\rightarrow$  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

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

$\rightarrow$ これをTIME(T)と表す.

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



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

**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  $T_1 \subseteq T_2$ ,  $\text{TIME}(T_1) \subseteq \text{TIME}(T_2)$

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

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

Now, for any times  $t_1, t_2$ ,

$t_1 = O(t_2)$  implies  $\text{TIME}(t_1) \subseteq \text{TIME}(t_2)$

Therefore,

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

Therefore,  $\text{TIME}(T_1) = \text{TIME}(T_2)$

Q.E.D.

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

**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,  $\text{TIME}(T_1) \subseteq \text{TIME}(T_2)$

$p$ : 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,  $p(l) = O(l^k)$

ここで、すべての制限時間  $t_1, t_2$  に対し、

$t_1 = O(t_2)$  ならば  $\text{TIME}(t_1) \subseteq \text{TIME}(t_2)$

したがって

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

よって  $\text{TIME}(T_1) = \text{TIME}(T_2)$

証明終

# 5. Computational Complexity

## 5.2. Representative Time Complexity Classes

### 5.2.2. Representative problems and their complexity

#### 5.2.2.1. Problem of evaluating propositional expression (PROP-EVAL)

**Input:**  $\langle F, \langle a_1, a_2, \dots, a_n \rangle \rangle$

- $F$  is an extended propositional expression
- $(a_1, a_2, \dots, a_n)$  is a truth assignment to  $F$

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

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

# 5. 計算量の理論

## 5.2. 代表的な時間計算量クラス

### 5.2.2. 代表的な問題とその計算量

#### 5.2.2.1. 命題論理式の評価 (PROP-EVAL)

入力:  $\langle F, \langle a_1, a_2, \dots, a_n \rangle \rangle$

・  $F$  は拡張命題論理式

・  $(a_1, a_2, \dots, a_n)$  は  $F$  への真偽値割当て

質問:  $F(a_1, a_2, \dots, a_n) = 1$ ?

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

# 5. Computational Complexity

## 5.2. Representative Time Complexity Classes

### 5.2.2. Representative problems and their complexity

#### 5.2.2.1. Problem of evaluating propositional expression (PROP-EVAL)

Input:  $\langle F, \langle a_1, a_2, \dots, a_n \rangle \rangle$

- $F$  is an extended prop. expression
- $(a_1, a_2, \dots, a_n)$  is a truth assignment to  $F$

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

PROP-EVAL  $\in$  P

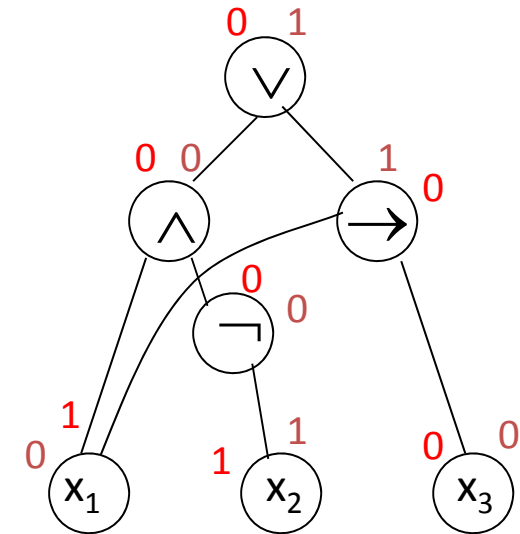
Construct a *computation tree* from a code of  $F$ .

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

Once computation tree is built,  
we can easily obtain the value

$F(a_1, a_2, \dots, a_n)$  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

# 5. 計算量の理論

## 5.2. 代表的な時間計算量クラス

### 5.2.2. 代表的な問題とその計算量

#### 5.2.2.1. 命題論理式の評価 (PROP-EVAL)

入力:  $\langle F, \langle a_1, a_2, \dots, a_n \rangle \rangle$

・  $F$  は拡張命題論理式

・  $(a_1, a_2, \dots, a_n)$  は  $F$  への真偽値割当て

質問:  $F(a_1, a_2, \dots, a_n) = 1$ ?

PROP-EVAL  $\in$  P

$F$  のコードから計算木を作る.

構築に必要な時間は  $O(|F|^3)$ .

ひとたび計算木ができると,

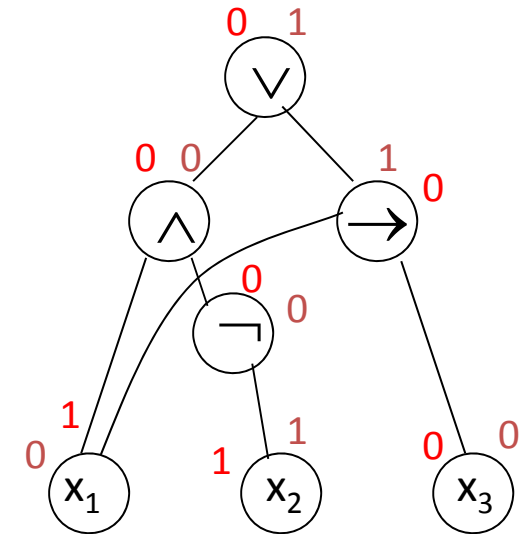
ボトムアップに計算すると

$F(a_1, a_2, \dots, a_n)$  の値は簡単に求まる.

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

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

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



計算木

# 5. Computational Complexity

## 5.2. Representative Time Complexity Classes

### 5.2.2. Representative problems and their complexity

#### 5.2.2.2. Satisfiability (SAT)

**Input:**  $\langle F \rangle$   $F$  is 2-conjunctive normal form

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

Conjunctive Normal Form (CNF)

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

- described by  $\wedge$  of  $\vee$  of literals.

$k$  SAT: Each clause contains  $k$  literals

exactly/at most

We can define 3SAT, 4SAT similarly.

SAT consists of any CNF.

ExSAT consists of any extended propositional expression.

# 5. 計算量の理論

## 5.2. 代表的な時間計算量クラス

### 5.2.2. 代表的な問題とその計算量

#### 5.2.2.2. 充足可能性 (SAT)

入力:  $\langle F \rangle$   $F$  は 2-和積標準形

質問:  $F(a_1, a_2, \dots, a_n) = 1$  とする真偽値割当てが存在するか?

和積標準形 (CNF)

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

- リテラルの  $\vee$  の  $\wedge$  で表現される式

$k$  SAT: 各項が  $k$  個のリテラルを含む

3SAT, 4SAT も同様に定義できる.

ちょうど/たかだか

SAT は任意の CNF を許す

ExSAT は拡張命題論理式 ( $\vee, \wedge, \rightarrow, \leftrightarrow$ ) を許す



# 5. Computational Complexity

## 5.2. Representative Time Complexity Classes

### 5.2.2. Representative problems and their complexity

#### 5.2.2.3. Graph reachability problem (ST-CON)

**Input:**  $\langle G, s, t \rangle$  : an undirected graph  $G$ ,  $1 \leq s, t \leq n (= |G|)$

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

#### 5.2.2.4. Euler cycle problem (DEULER)

**Input:**  $\langle G \rangle$ : a directed graph  $G$

**Question:** Does  $G$  have an Euler cycle?

#### 5.2.2.5 Hamiltonian cycle problem (DHAM)

**Input:**  $\langle G \rangle$ : a directed graph  $G$

**Question:** Does  $G$  have a Hamiltonian cycle?

Actually,  
directed/undirected  
cycle/path  
do not matter

- **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.

# 5. 計算量の理論

## 5.2. 代表的な時間計算量クラス

### 5.2.2. 代表的な問題とその計算量

#### 5.2.2.3. グラフの到達可能性問題 (ST-CON)

入力:  $\langle G, s, t \rangle$ : 無向グラフ  $G$ ,  $1 \leq s, t \leq n (= |G|)$

質問:  $G$  は  $s$  から  $t$  への経路を持つか?

#### 5.2.2.4. オイラー閉路問題 (DEULER)

入力:  $\langle G \rangle$ : 有向グラフ  $G$

質問:  $G$  はオイラー閉路を持つか?

#### 5.2.2.5. ハミルトン閉路問題 (DHAM)

入力:  $\langle G \rangle$ : 有向グラフ  $G$

質問:  $G$  はハミルトン閉路を持つか?

実際には,  
有向/無向  
閉路/パス  
という違いは問題にならない

- 閉路とは両端点を共有する経路.
- オイラー閉路とはすべての辺をちょうど一回通る閉路.
- ハミルトン閉路とはすべての頂点をちょうど一回通る閉路.

# 5. Computational Complexity

## 5.2. Representative Time Complexity Classes

### 5.2.2. Representative problems and their complexity

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?

# 5. 計算量の理論

## 5.2. 代表的な時間計算量クラス

### 5.2.2. 代表的な問題とその計算量

以下は既知:

- 以下の問題は P の要素:
  - ✓ PROP-EVAL, 2SAT, ST-CON, DEULER
- 以下の問題は E の要素だが...
  - ✓ 3SAT, DHAM

P と E の間(?)にあるクラス **NP**

# 5. Computational Complexity

## 5.3. Class NP

### 5.3.\*. Nondeterministic computation

Some problems (like 3SAT, DHAM, etc.) have a common and natural property;

- once you get a solution, you can check it efficiently
  - without solution, it seems to be quite difficult; you may check all possibilities
- 
- Many natural problems have this property in the real problems.
  - This property leads us to the notion of “nondeterministic computation”

# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.\*. 非決定性計算とは

(3SAT, DHAMといった)ある種の問題には、次のような共通で自然な性質がある;

- ひとたび解が得られると、その正当性は簡単にチェックできる
- 解を見つけるのは大変そうに思える。可能な場合をしらみつぶしに調べる必要がありそうに見える。
- 現実の自然な問題の多くはこの性質をもつ。
- この性質を表現するのが「非決定性計算」

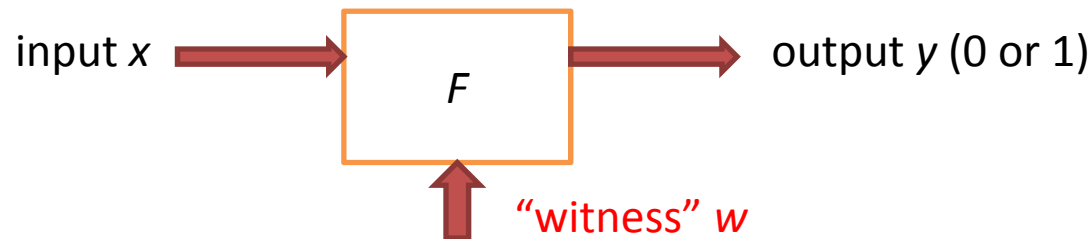
# 5. Computational Complexity

## 5.3. Class NP

### 5.3.\*. Nondeterministic computation

“Nondeterministic computation”

- From the viewpoint of Function:



$L$  is called an NP set if there is a function  $F$  s.t.

1. For each  $x$ , there is a binary string “witness”  $w$  s.t.
2.  $|w|$  is bounded by a polynomial of  $|x|$
3.  $F$  recognizes  $x \in L$  with  $w$  in polynomial time of  $|x|$  and  $|w|$

c.f. : NP=Nondeterministic Polynomial

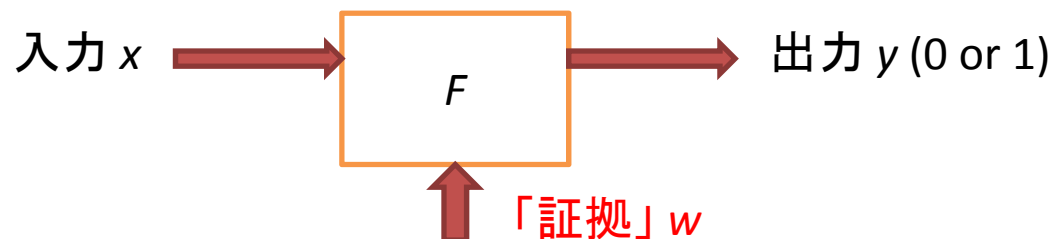
# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.\*. 非決定性計算とは

「非決定性計算」

- 関数の観点からみると:



以下の関数 $F$ が存在するとき $L$ はNP 集合と呼ばれる:

1. 各 $x$ に対して, 2進列の「証拠」 $w$ が存在
2.  $|w|$  は  $|x|$  の多項式で上から抑えられる
3.  $F$  は  $|x|$  と  $|w|$  の多項式時間で  $w$  を使って  $x \in L$  を認識する

c.f. : NP = Nondeterministic Polynomial



# 5. Computational Complexity

## 5.3. Class NP

### 5.3.\*. Nondeterministic computation

“Nondeterministic computation”

- From the viewpoint of Logic:

Suppose that we have a polynomial  $q$  and polynomial time computable predicate  $R$  for a set  $L$  such that

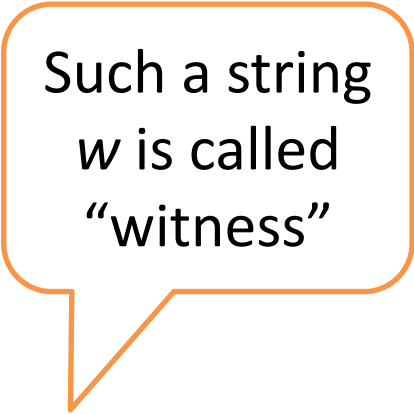
for each  $x \in \Sigma^*$ ,  $x \in L \leftrightarrow \exists w \in \Sigma^* : |w| \leq q(|x|) [R(x, w)]$

i.e., 
$$L = \{x : \exists w \in \Sigma^* [ |w| \leq q(|x|) \wedge R(x, w) ]\}$$

Then,  $L$  is called an NP set, and the problem of recognizing  $L$  is called an **NP problem**.

Also, the whole set of NP sets is called the **class NP**.

c.f. : NP = Nondeterministic Polynomial



Such a string  $w$  is called “witness”

# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.\*. 非決定性計算とは

「非決定性計算」

- 論理の視点からみると:

集合  $L$  に対して多項式  $q$  と多項式で計算できる述語  $R$  があり、以下を満たすとする:

for each  $x \in \Sigma^*$ ,  $x \in L \leftrightarrow \exists w \in \Sigma^* : |w| \leq q(|x|) [R(x, w)]$

つまり,  $L = \{x : \exists w \in \Sigma^* [ |w| \leq q(|x|) \wedge R(x, w) ]\}$

このとき  $L$  は NP 集合とよばれ,

$L$  の認識問題は NP 問題とよばれる.

また NP 集合全体の集合をクラス NP とよぶ.

c.f.: NP = Nondeterministic Polynomial

この文字列  $w$  を「証拠」とよぶ

# 5. Computational Complexity

## 5.3. Class NP

### 5.3.\*. Nondeterministic computation

“Nondeterministic computation”

- From the viewpoint of Turing Machine:

Suppose that Turing machine has “nondeterministic choice” that admits us to two possible choices at the same time;

i.e., it has “one of two cases (0) and (1)” statement.

- A nondeterministic choice allows to assume of two choices and it will be “*true*” if “*at least one of them is true*”.

Then, NP problem  $L$  can be recognized by a nondeterministic Turing machine in polynomial time.

A “nondeterministic choice” is a kind of parallel computing that generates two branches.

c.f. : NP=Nondeterministic Polynomial

# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.\*. 非決定性計算とは 「非決定性計算」

- チューリングマシンの視点から見ると:

チューリングマシンの「非決定性選択」では、二つの選択肢を「同時に」二つとも選ぶことができる；つまり「場合(0)と場合(1)のいずれか」という命令がある。

- 非決定性選択は二つの選択肢のうち、「いずれか一方が真」ならば真になる。

このときNP 問題  $L$  は、非決定性チューリング機械で多項式時間で受理できる問題。

「非決定性選択」はある種の並列計算とみなすこともでき、二つの計算プロセスの生成と考えてもよい。

# 5. Computational Complexity

## 5.3. Class NP

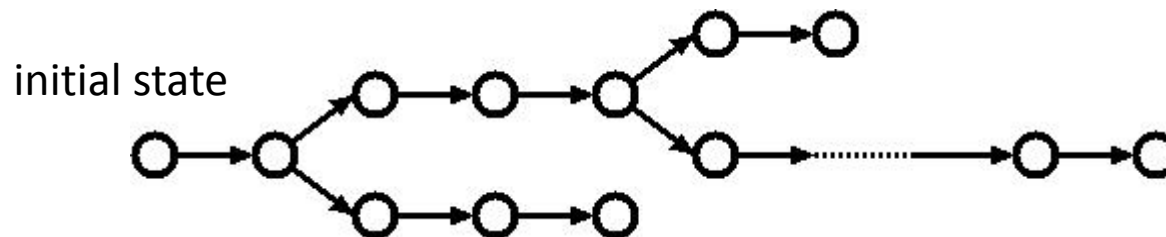
### 5.3.\*. Nondeterministic computation

- From the viewpoint of the computation tree of a Turing Machine:

- Computation tree of a deterministic Turing machine forms a path;



- Computation tree of a *nondeterministic* Turing machine forms a *tree*;



- each computation halts in an accept/reject state or loop.
- it accepts if the tree has at least one "accept" in poly-length.

# 5. 計算量の理論

## 5.3. クラス NP

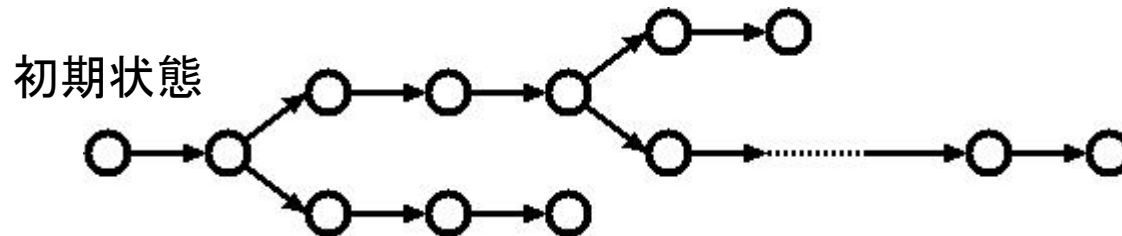
### 5.3.\*. 非決定性計算とは

- チューリングマシンの計算木の観点からみると:

- 決定性のチューリングマシンの計算木はパス(一本道);



- 非決定性のチューリングマシンの計算木は木;



- 各計算プロセスは受理/拒否状態になるか無限ループ
- 木が多項式長の範囲で受理状態を一つでももてば受理.



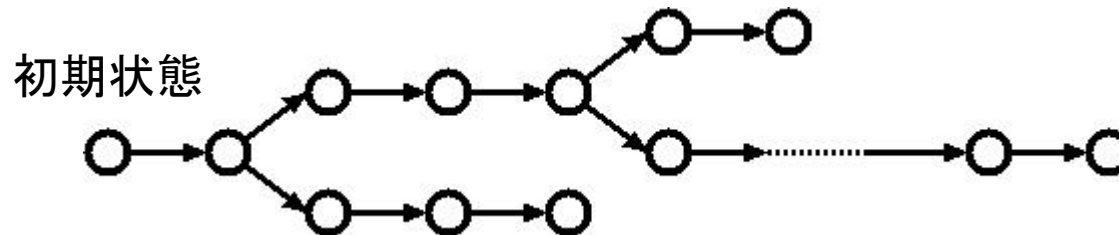
# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.\*. 非決定性計算とは

• チューリングマシンの計算木の観点からみると:

• 非決定性のチューリングマシンの計算木は木;



- 各計算プロセスは受理/拒否状態になるか無限ループ
- 木が多項式長の範囲で受理状態を一つでももてば受理.

証拠  $w$  は正しい選択枝の列を与える

**NP 問題  $L$**  とは非決定性チューリング機械で多項式時間で認識できる言語. つまり, 受理状態に至る  $n$  の多項式長の計算パスが存在すればよい.



# 5. Computational Complexity

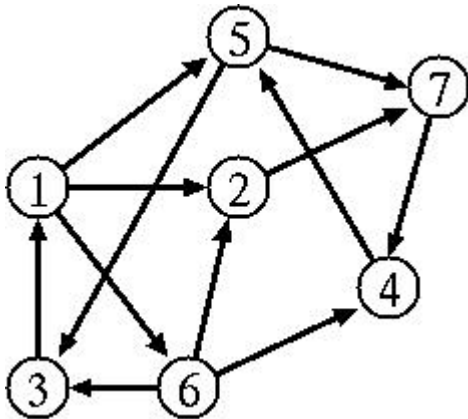
## 5.3. Class NP

### 5.3.1. Representative NP problems

- Hamiltonian cycle problem (DHAM)

**Input:**  $\langle G \rangle$ : a directed graph  $G$

**Question:** Does  $G$  have a Hamiltonian cycle?



- We can certainly check all possible permutations of  $n$ , that counts up to  $n! \sim n^n \dots$  it takes exponential time.
- If  $G$  has a Hamiltonian cycle  $C$ , and we have it as a witness, we can check that it surely a Hamiltonian cycle.

# 5. 計算量の理論

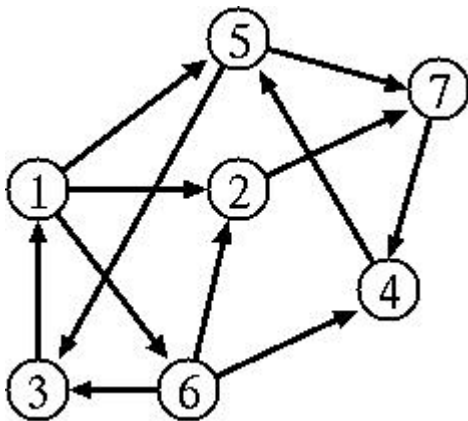
## 5.3. クラスNP

### 5.3.1. 代表的なNP問題

- ハミルトン閉路問題 (DHAM)

入力:  $\langle G \rangle$ : 有向グラフ  $G$

質問:  $G$  はハミルトン閉路をもつか?



- 原理的には  $n$  の順列をすべて試せばよいが, 可能な組合せの数は最大で  $n! \sim n^n \dots$  指数時間かかってしまう.
- もし  $G$  がハミルトン閉路  $C$  をもつならこれを証拠にすれば, 効率よくそれをチェックすることができる.

# 5. Computational Complexity

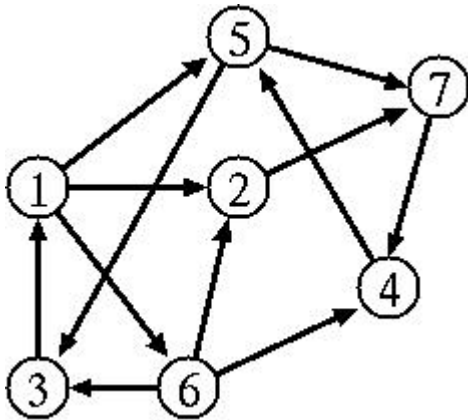
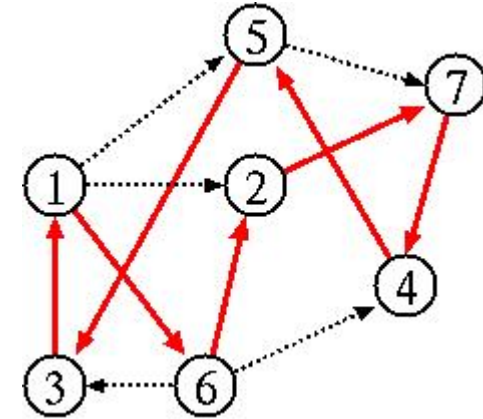
## 5.3. Class NP

### 5.3.1. Representative NP problems

- Hamiltonian cycle problem (DHAM)

**Input:**  $\langle G \rangle$ : a directed graph  $G$

**Question:** Does  $G$  have a Hamiltonian cycle?



- We can certainly check all possible permutations of  $n$ , that counts up to  $n! \sim n^n \dots$  it takes exponential time.
- If  $G$  has a Hamiltonian cycle  $C$ , and we have it as a witness, we can check that it surely a Hamiltonian cycle.

# 5. 計算量の理論

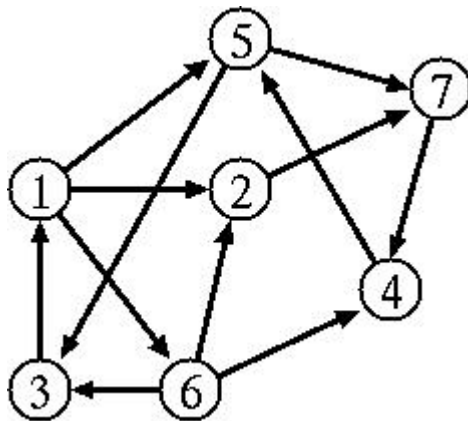
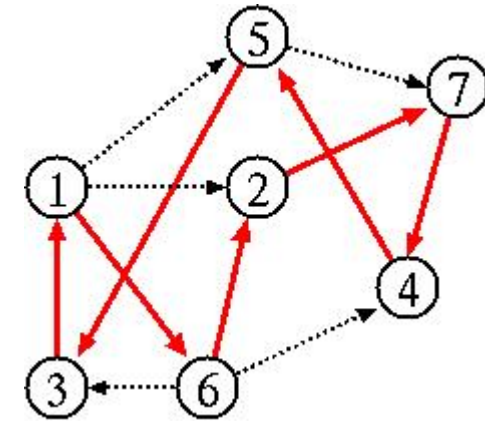
## 5.3. クラス NP

### 5.3.1. 代表的な NP 問題

- ハミルトン閉路問題 (DHAM)

入力:  $\langle G \rangle$ : 有向グラフ  $G$

質問:  $G$  はハミルトン閉路をもつか?



- 原理的には  $n$  の順列をすべて試せばよいが, 可能な組合せの数は最大で  $n! \sim n^n \dots$  指数時間かかってしまう.
- もし  $G$  がハミルトン閉路  $C$  をもつならこれを証拠にすれば, 効率よくそれをチェックすることができる.

# 5. Computational Complexity

## 5.3. Class NP

### 5.3.1. Representative NP problems

- SAT, *kSAT*, *ExSAT* (Satisfiability)

**Input:**  $\langle F \rangle$   $F$  is conjunctive normal form

**Question:** Any assignment  $s$ . t.  $F(a_1, a_2, \dots, a_n) = 1$ ?

- If  $F$  is satisfiable by an assignment  $A$ , and we have it as a witness, we can check it in polynomial time by the same way as the PROP\_EVAL.
- We can certainly check all possible assignments of  $(a_1, a_2, \dots, a_n)$ . The assignments are  $2^n$ , that takes exponential time.

# 5. 計算量の理論

## 5.3. クラスNP

### 5.3.1. 代表的なNP問題

- SAT,  $k$ SAT, ExSAT (充足可能性)

入力:  $\langle F \rangle$   $F$  は和積標準形命題論理式

質問:  $F(a_1, a_2, \dots, a_n) = 1$ となる割当ては存在?

- $F$ を充足する割当て  $A$ があるなら,  
それを証拠として使い, PROP\_EVALのときと同じ方法で  
多項式時間でチェックできる.
- もちろん  $(a_1, a_2, \dots, a_n)$ のすべての可能な割当てを  
チェックすることはできるが, 可能な割当ての個数は  
 $2^n$ なので, 指数時間かかる.

# 5. Computational Complexity

## 5.3. Class NP

### 5.3.2. Another aspect of the NP problems

- What does it mean by being an NP set?
  - Using  $q$  and  $R$  satisfying the predicate characterizing an NP set, we can determine “ $x \in L$ ?” in the following way.

```
for each  $w \in \Sigma^{\leq q(|x|)}$  do
  if  $R(x, w)$  then accept end-if
end-for;
reject;
```

If we enumerate and check all possible strings of length at most  $q(|x|)$ , we can accept or reject them.

Here note that there are  $2^{q(|x|)}$  (exponentially many) such strings.

We may think that those sets recognizable as above are NP sets.

# 5. 計算量の理論

## 5.3. クラス NP

### 5.3.2. NP問題を別の視点から見る

- NP 集合であることの意味は?

- 命題述語論理によるNP 集合の特徴付けで出てきた  $q$  と  $R$  を使うと、「 $x \in L?$ 」という質問に次のアルゴリズムで答えることができる.

```
for each  $w \in \Sigma^{\leq q(|x|)}$  do
  if  $R(x, w)$  then accept end-if
end-for;
reject;
```

長さ高々  $q(|x|)$  のすべての文字列を辞書式に列挙してチェックすれば、受理または拒否を判断できる.

ただし、こうした文字列は  $2^{q(|x|)}$  (指数関数的) 通りある.

こうしたアルゴリズムで認識できる集合をNP集合と考えてもよい.



# 5. Computational Complexity

## 5.3. Class NP

### 5.3.3. More representative NP problems

- Knapsack Problem (KNAP)

**Input:**  $n+1$  tuple of natural numbers  $\langle a_1, a_2, \dots, a_n, b \rangle$

**Question:** Is there a set of indices  $S \subseteq \{1, \dots, n\}$  s.t.  $\sum_{i \in S} a_i = b$  ?

- Bin Packing Problem (BIN)

**Input:**  $n+2$  tuple of natural numbers  $\langle a_1, a_2, \dots, a_n, b, k \rangle$

**Question:** Is there a partition of a set of indices  $U = \{1, \dots, n\}$  into  $U_1, \dots, U_k$  such that  $\sum_{i \in U_j} a_i \leq b$  for each  $j$ ?

- Vertex Cover Problem (VC)

**Input:** pair of undirected graph  $G$  and natural number  $k$   $\langle G, k \rangle$

**Question:** Is there a vertex cover of  $k$  vertices over  $G$ ?

**Vertex Cover**  $S$  contains at least one of  $u$  and  $v$  for each edge  $\{u, v\}$ .

# 5. 計算量の理論

## 5.3. クラスNP

### 5.3.3. 代表的なNP問題再び

- ナップサック問題 (KNAP)

入力: 自然数の $n+1$ 個組 $\langle a_1, a_2, \dots, a_n, b \rangle$

質問: 添え字の集合 $S \subseteq \{1, \dots, n\}$ で $\sum_{i \in S} a_i = b$  を満たすものはあるか?

- ビン詰め問題 (BIN)

入力: 自然数の $n+2$ 個組 $\langle a_1, a_2, \dots, a_n, b, k \rangle$

質問: 添え字の集合 $U = \{1, \dots, n\}$ の分割 $U_1, \dots, U_k$ で $\sum_{i \in U_j} a_i \leq b$  を満たすものはあるか?

- 頂点被覆問題 (VC)

入力: 無向グラフ $G$ と自然数 $k$ の組 $\langle G, k \rangle$

質問:  $G$ 上に大きさ $k$ の頂点被覆は存在するか?

頂点被覆  $S$  とは, 各辺 $\{u, v\}$ に対して $u, v$ の少なくともどちらか一方をふくむ頂点集合

# 5. Computational Complexity

## 5.4. Class coNP

### Definition

A set  $L$  is in coNP if and only if its complement belongs to NP.

### Theorem

For every set  $L$ , the following conditions are equivalent.

- (a)  $L \in \text{coNP}$
- (b) The set  $L$  can be represented as

$$L = \{x : \forall w \in \Sigma^* : |w| \leq q(|x|)[Q(x, w)]\}$$

by using some polynomial  $q$  and polynomial-time computable predicate  $Q$ .

[Note] It is nonsense to define coP since it is equal to P.

# 5. 計算量の理論

## 5.4. クラスcoNP

### 定義

集合  $L$  が coNP に属する必要十分条件は、その補集合がNPに属すること。

### 定理

任意の集合  $L$  に対して、以下の二つは同値である。

(a)  $L \in \text{coNP}$

(b)  $L$  は多項式  $q$  と多項式時間で計算できる述語  $Q$  を使って

次のように書ける:  $L = \{x : \forall w \in \Sigma^* : |w| \leq q(|x|)[Q(x, w)]\}$

[注意] coP はPと同値であることがすぐにわかるので、定義しても無意味。

# 5. Computational Complexity

## 5.5. Relations in the Complexity Classes

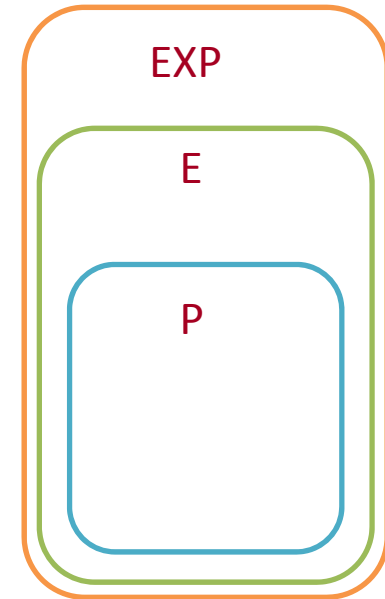
Theorem  $P \subseteq E \subseteq EXP$

Proof: Obvious from the definition.

Theorem  $P \subsetneq E \subsetneq EXP$

Proof: Out of scope in this class...  
(Brief idea: We can use *diagonalization* to show a hierarchy theorem that says  $\text{TIME}(t_1(n)) \subsetneq \text{TIME}(t_2(n))$  for, e.g.,  $t_1(n)^3 = O(t_2(n))$ ).

We have a *proper* hierarchy



# 5. 計算量の理論

## 5.5. 計算量クラスの関係

定理  $P \subseteq E \subseteq EXP$

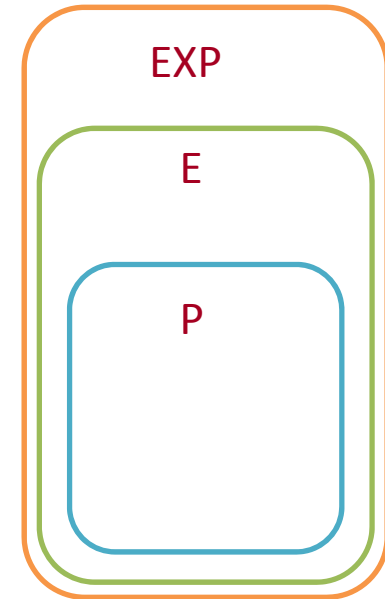
証明: 定義より明らか.

定理  $P \subsetneq E \subsetneq EXP$

証明: 本書の範囲を超えるので省略.  
(アイデアの概略: 対角線論法を巧妙に  
使うと, 例えば  $t_1(n)^3 = O(t_2(n))$  といった関数に  
対して次の階層定理を示すことができる.

$$TIME(t_1(n)) \subsetneq TIME(t_2(n))$$

真に異なる階層構造  
が成立する



# 5. Computational Complexity

## 5.5. Relations in the Complexity Classes

### Theorem

(1)  $P \subseteq NP$ ,  $P \subseteq coNP$  ( $\therefore P \subseteq NP \cap coNP$ )

(2)  $NP \subseteq EXP$ ,  $coNP \subseteq EXP$  ( $\therefore NP \cup coNP \subseteq EXP$ )

Proof (Outline):

(1)  $P \subseteq NP$  ( $P \subseteq coNP$  is similar)

Ignoring the “witness” in the definition of NP, we immediately obtain the definition of P.

(2)  $NP \subseteq EXP$  ( $coNP \subseteq EXP$  is similar)

For the “witness”  $w$  of length  $m$ , we can check all possible strings of length  $m$  in exponential time.

# 5. 計算量の理論

## 5.5. 計算量クラスの関係

### 定理

(1)  $P \subseteq NP$ ,  $P \subseteq \text{coNP}$  ( $\therefore P \subseteq NP \cap \text{coNP}$ )

(2)  $NP \subseteq \text{EXP}$ ,  $\text{coNP} \subseteq \text{EXP}$  ( $\therefore NP \cup \text{coNP} \subseteq \text{EXP}$ )

証明(概略):

(1)  $P \subseteq NP$  ( $P \subseteq \text{coNP}$  も同様)

NPの定義の中の「証拠」を無視すれば、  
Pの定義と同値なものが得られる。

(2)  $NP \subseteq \text{EXP}$  ( $\text{coNP} \subseteq \text{EXP}$  も同様)

長さ $m$ のすべての文字列に対して  
それが長さ $m$ の「証拠」 $w$ になるかどうかを  
指数時間かけてチェックすればよい。



# 5. Computational Complexity

## 5.5. Relations in the Complexity Classes

### Theorem

(1)  $NP \subseteq coNP \rightarrow NP = coNP$

(2)  $coNP \subseteq NP \rightarrow NP = coNP$

(3)  $NP \neq coNP \rightarrow P \neq NP$

**Note:** From (3), proof for  $NP \neq co-NP$  is harder than that for  $P \neq NP$ .

Proof :

(1)  $NP \subseteq coNP \rightarrow NP = coNP$

By assumption, it is sufficient to show that  $coNP \subseteq NP$ .

We will prove  $L \in NP$  for any  $L \in coNP$ .

$$\begin{aligned} L \in coNP &\Leftrightarrow \overline{L} \in NP && \text{(by Definition)} \\ &\rightarrow \overline{L} \in coNP && \text{(} NP \subseteq co-NP \text{)} \\ &\Leftrightarrow L \in NP && \text{(Definition and } L = \overline{\overline{L}} \text{)} \end{aligned}$$

# 5. 計算量の理論

## 5.5. 計算量クラスの関係

### 定理

- (1)  $NP \subseteq coNP \rightarrow NP = coNP$
- (2)  $coNP \subseteq NP \rightarrow NP = coNP$
- (3)  $NP \neq coNP \rightarrow P \neq NP$

注: (3)より $NP \neq co-NP$  の証明は $P \neq NP$ の証明よりも難しい.  
証明:

(1)  $NP \subseteq coNP \rightarrow NP = coNP$

仮定より  $coNP \subseteq NP$  を示せばよい.

そこで任意の  $L \in coNP$  に対して  $L \in NP$  を示す.

$$\begin{aligned} L \in coNP &\Leftrightarrow \overline{L} \in NP && \text{(定義より)} \\ &\rightarrow \overline{L} \in coNP && \text{(} NP \subseteq co-NP \text{)} \\ &\Leftrightarrow L \in NP && \text{(定義と } L = \overline{\overline{L}} \text{より)} \end{aligned}$$

# 5. Computational Complexity

## 5.5. Relations in the Complexity Classes

### Theorem

$$(1) NP \subseteq coNP \rightarrow NP = coNP$$

$$(2) coNP \subseteq NP \rightarrow NP = coNP$$

$$(3) NP \neq coNP \rightarrow P \neq NP$$

**Note:** From (3), proof for  $NP \neq coNP$  is harder than that for  $P \neq NP$ .

**Proof:** (3)  $NP \neq coNP \rightarrow P \neq NP$

Contraposition:  $P = NP \rightarrow NP = coNP$

If we assume  $P=NP$ , for any  $L$  we have

$$L \in NP \Leftrightarrow L \in P \quad (P = NP)$$

$$\Leftrightarrow \bar{L} \in P \quad (P = coP)$$

$$\Leftrightarrow \bar{L} \in NP \quad (P = NP)$$

$$\Leftrightarrow L (= \bar{\bar{L}}) \in coNP \quad (\text{Definitions of NP/coNP})$$

$\therefore NP = coNP$

Q.E.D.

# 5. 計算量の理論

## 5.5. 計算量クラスの関係

### 定理

- (1)  $NP \subseteq coNP \rightarrow NP = coNP$
- (2)  $coNP \subseteq NP \rightarrow NP = coNP$
- (3)  $NP \neq coNP \rightarrow P \neq NP$

注: (3)より $NP \neq coNP$ の証明は $P \neq NP$ の証明よりも難しい.

証明: (3)  $NP \neq coNP \rightarrow P \neq NP$

以下の対偶を示す:  $P = NP \rightarrow NP = coNP$

$P=NP$ と仮定すると, 任意の集合  $L$  に対して以下を得る

$$\begin{aligned} L \in NP &\Leftrightarrow L \in P \quad (P = NP) \\ &\Leftrightarrow \bar{L} \in P \quad (P = coP) \\ &\Leftrightarrow L \in NP \quad (P = NP) \\ &\Leftrightarrow L (= \bar{\bar{L}}) \in coNP \quad (NP/coNPの定義より) \end{aligned}$$

$\therefore NP = coNP$

Q.E.D.

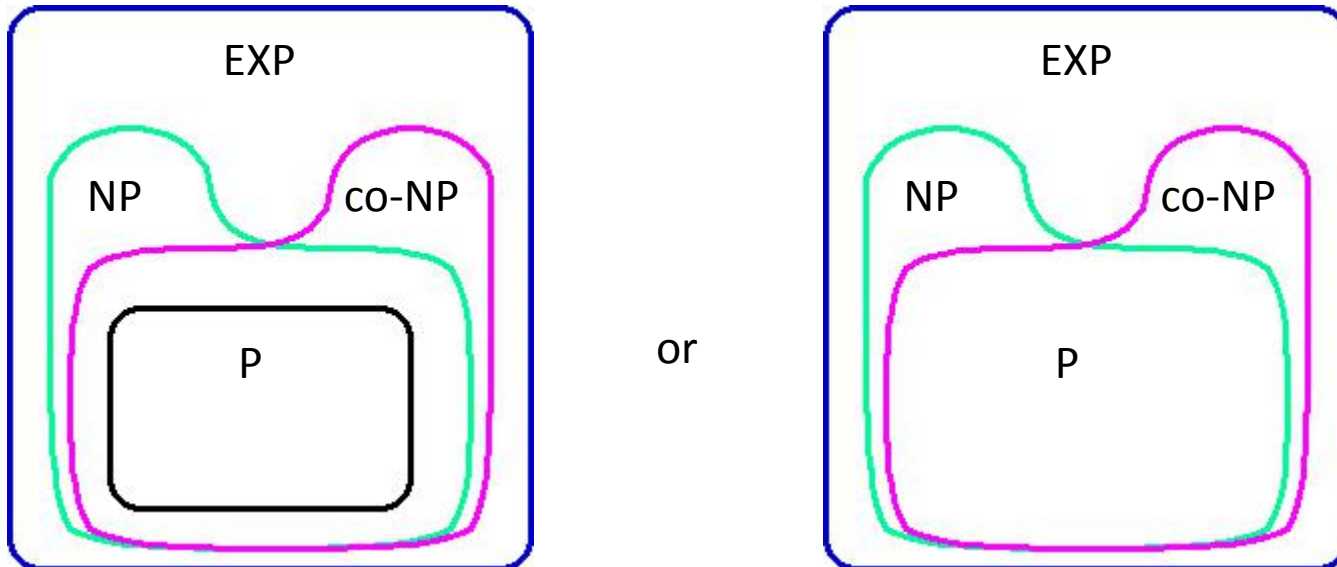
# 5. Computational Complexity

## 5.5. Relations in the Complexity Classes

### Theorem

- (1)  $NP \subseteq coNP \rightarrow NP = coNP$
- (2)  $coNP \subseteq NP \rightarrow NP = coNP$
- (3)  $NP \neq coNP \rightarrow P \neq NP$

We strongly believe that  $P \neq NP$ , and then we have



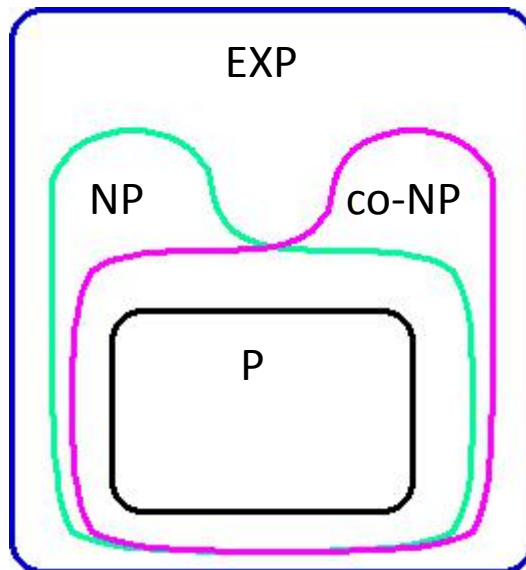
# 5. 計算量の理論

## 5.5. 計算量クラスの関係

### 定理

- (1)  $NP \subseteq coNP \rightarrow NP = coNP$
- (2)  $coNP \subseteq NP \rightarrow NP = coNP$
- (3)  $NP \neq coNP \rightarrow P \neq NP$

$P \neq NP$ が成立すると強く信じられているので、以下の構造になっていると予想される。



または

