# 理論計算機科学特論 (2026 年前学期)

## 計算複雑性の基礎

# 第 5 回

# 時間と領域の関係：P ⊆ PSPACE ⊆ EXPTIME

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2026 年 5 月 12 日

最終更新：2026 年 5 月 12 日　11:31

# スケジュール (前半)



1. 計算理論の復習 (4/7)
2. 時間計算量：P, NP, coNP (4/14)
3. 帰着と完全性：NP 完全 (4/21)
4. 領域計算量：L, NL, PSPACE (4/28)

* 休み (祝日) (5/5)

5. **時間と領域の関係：** $P \subseteq PSPACE \subseteq EXPTIME$ (5/12)
6. 階層定理：$P \neq EXPTIME$ (5/19)
7. Ladner の定理：$NP - P = NPC \Rightarrow P = NP$ (5/26)

# スケジュール (後半)

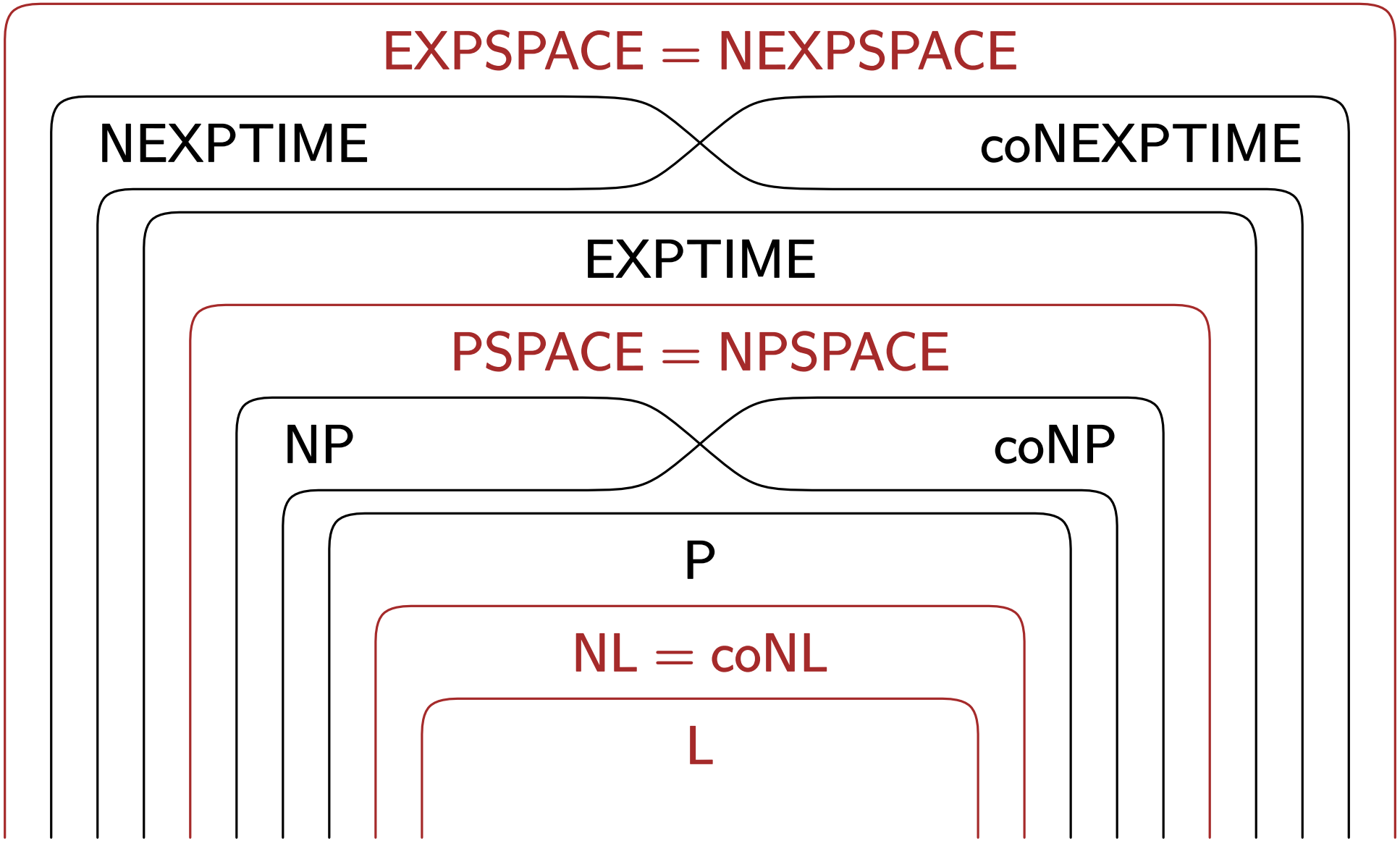


8. Savitch の定理：PSPACE = NPSPACE (6/2)
9. Immerman-Szlepcsényi の定理：NL = coNL (6/9)
10. 多項式階層：$P = NP \Rightarrow P = PH$ (6/16)
11. 交代性計算：AP = PSPACE (6/23)
12. 確率的計算：$P \subseteq BPP \subseteq PP$, $NP \subseteq PP$ (6/30)
13. 対話証明系 (1)：$NP \subseteq MA \subseteq AM$ (7/7)
14. 対話証明系 (2)：$IP \subseteq PSPACE$ (7/14)
15. 対話証明系 (3)：$PSPACE \subseteq IP$ (7/21)

* 休み (授業のない日) (7/28)

# 復習：時間計算量と領域計算量

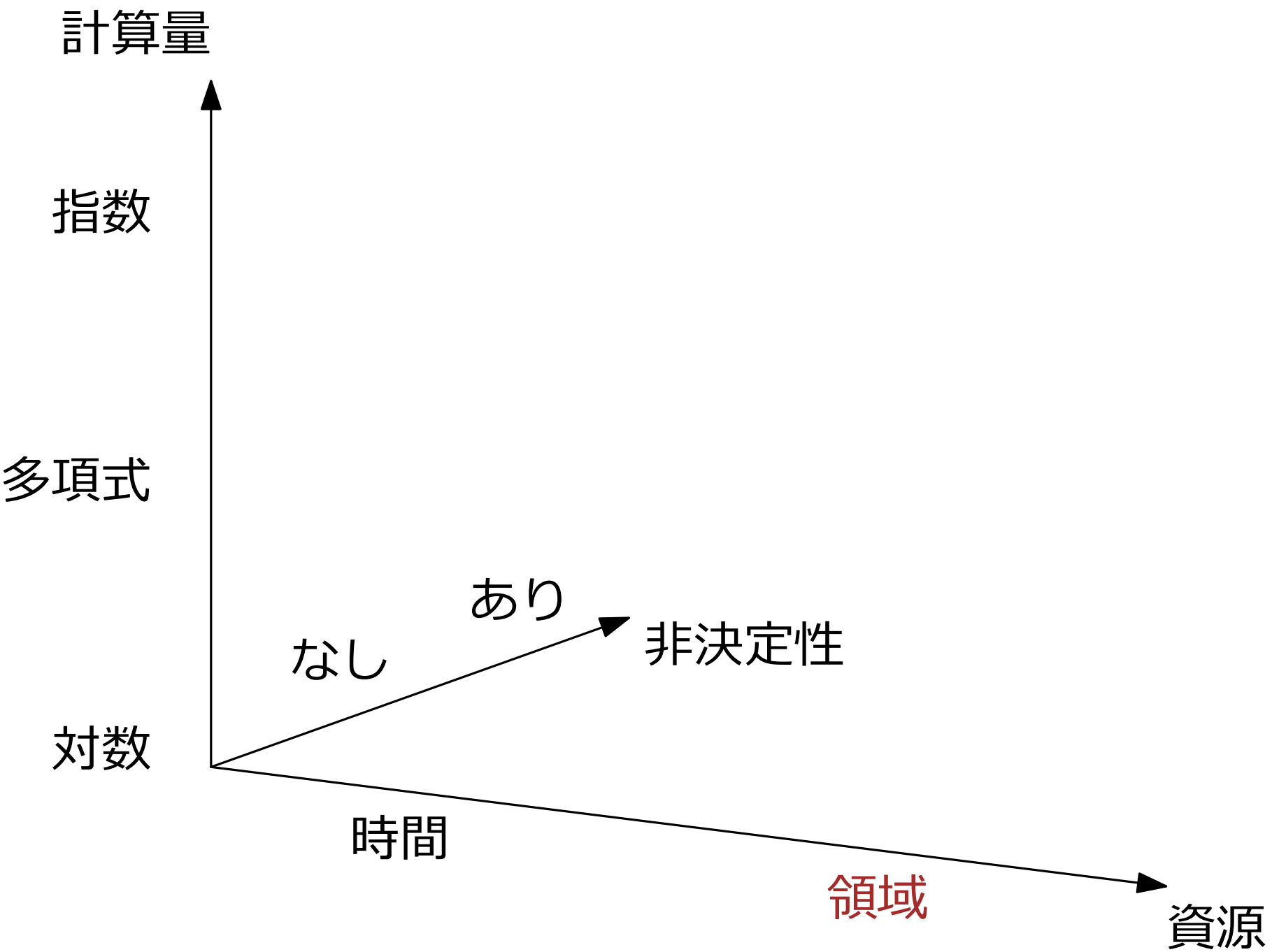





黒は時間複雑性クラス，茶は領域複雑性クラス

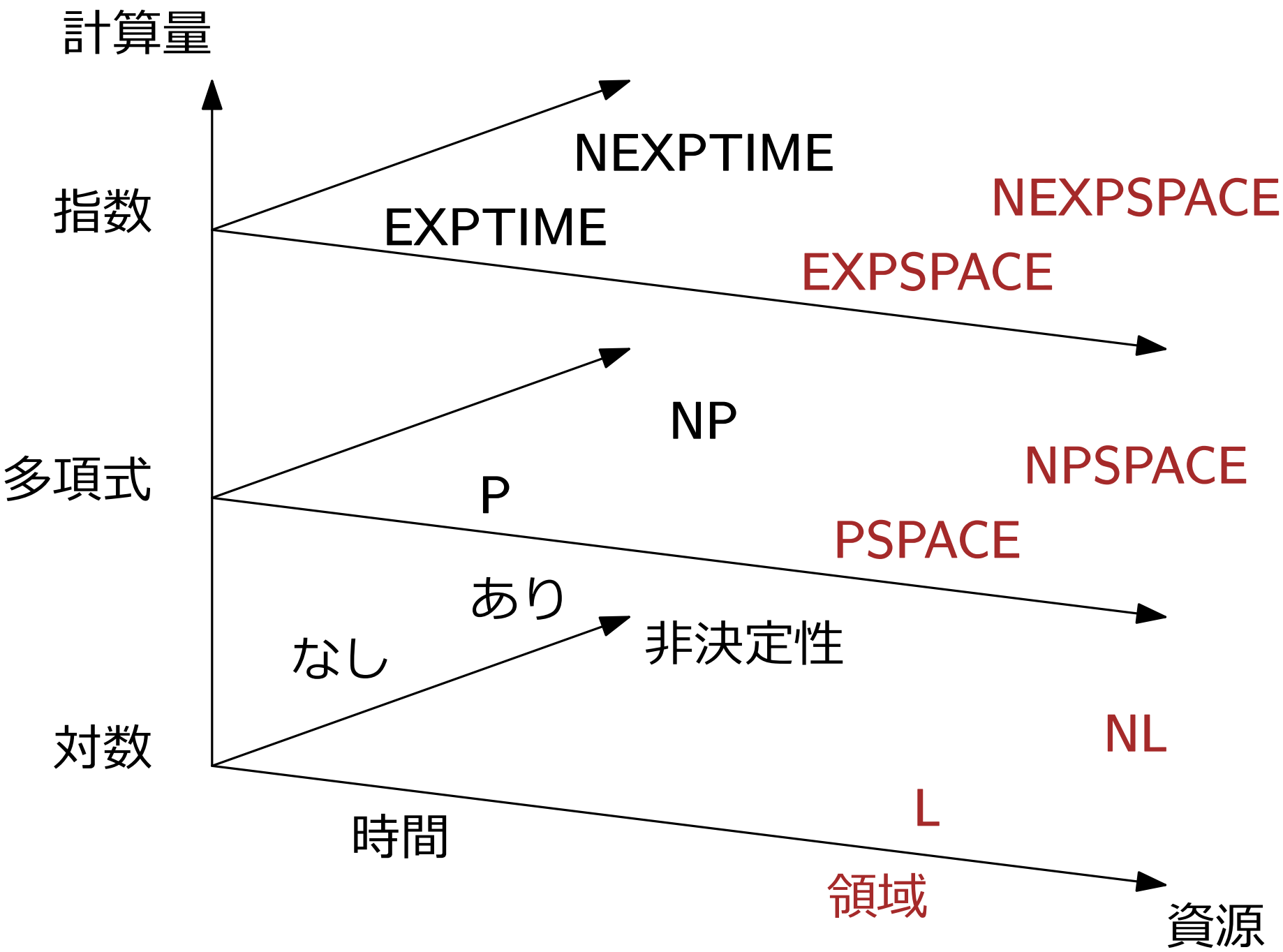
# 復習：三次元的？ に書くと





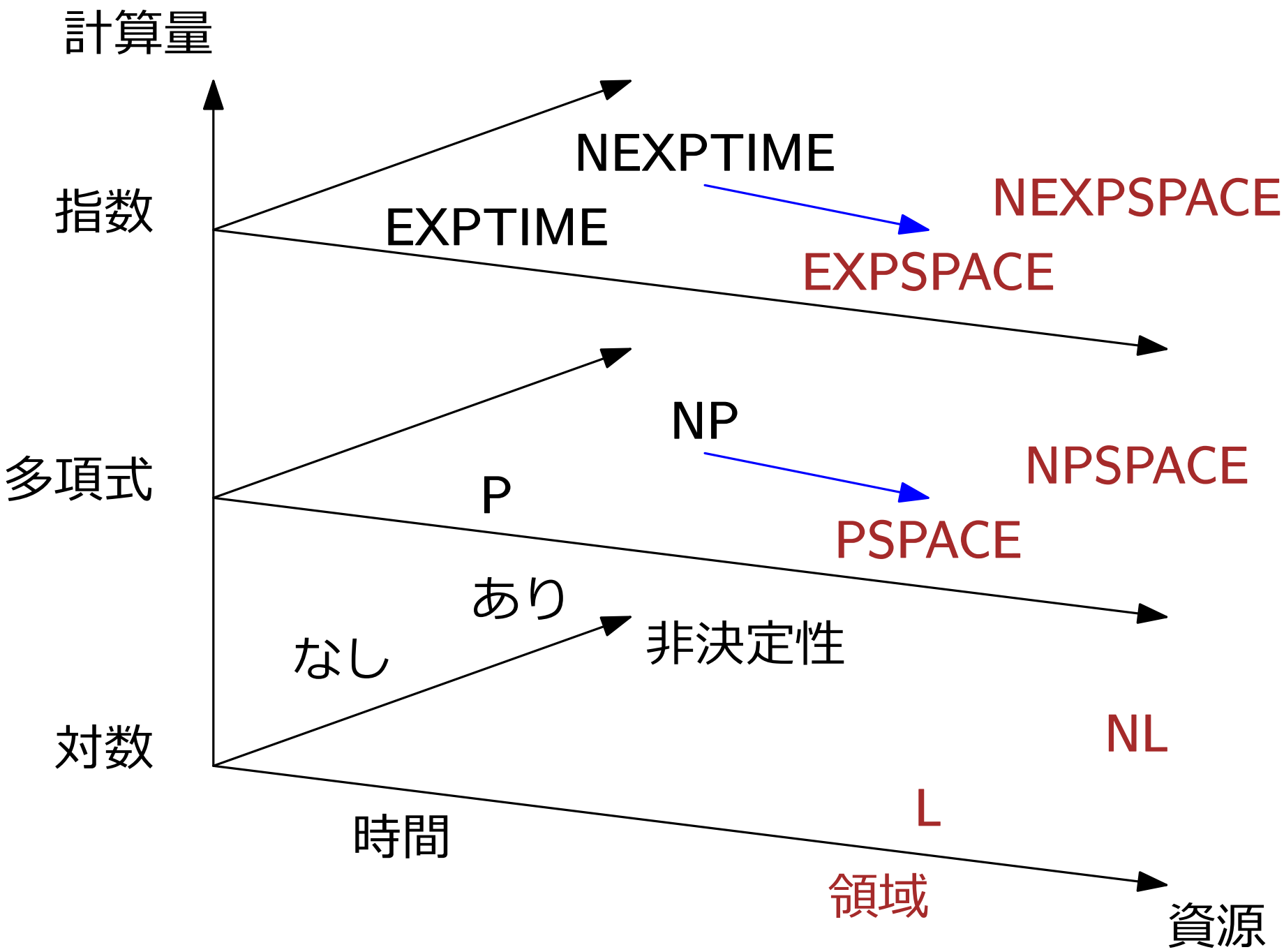
# 復習：三次元的？ に書くと




計算量
指数
多項式
対数
時間
資源
NEXPTIME
EXPTIME
NEXPSPACE
EXPSPACE
NP
P
NPSPACE
PSPACE
あり
なし
非決定性
NL
L
領域

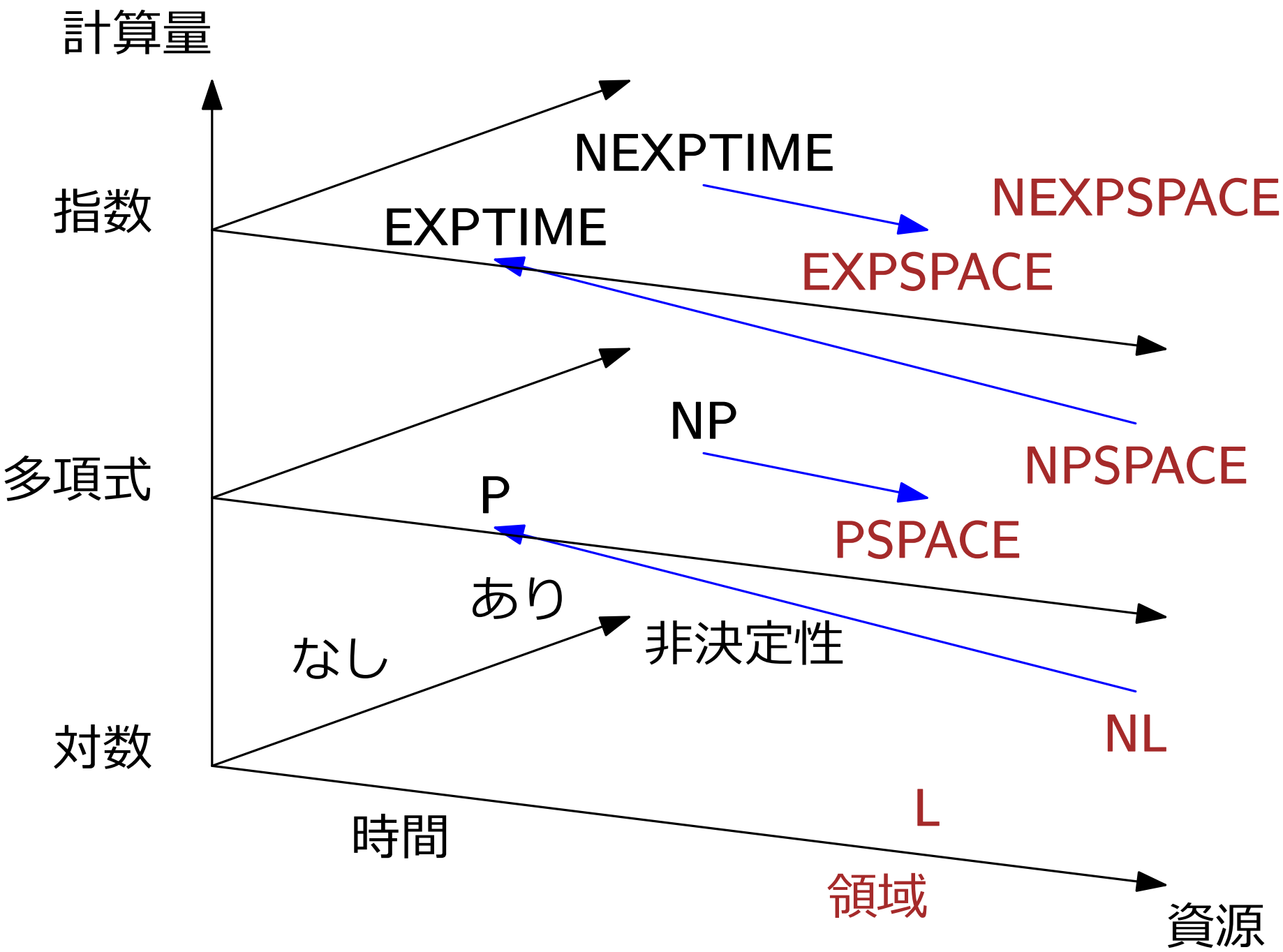
# 復習：三次元的？ に書くと




計算量
指数
NEXPTIME
EXPTIME
NEXPSPACE
EXPSPACE
多項式
NP
P
NPSPACE
PSPACE
あり
なし
非決定性
対数
NL
L
時間
領域
資源

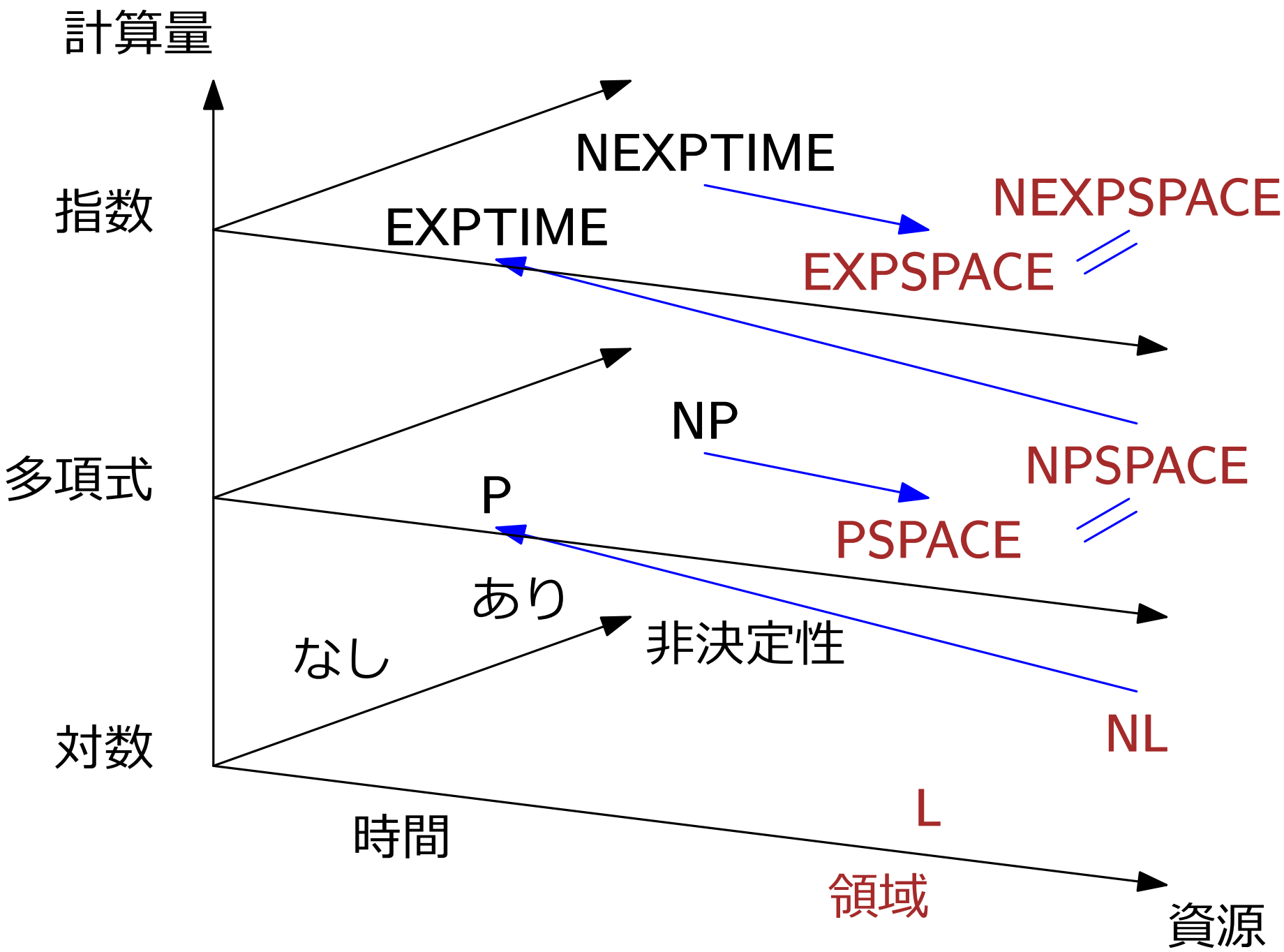
# 復習：三次元的？ に書くと




計算量
指数
多項式
対数
時間
資源
NEXPTIME
EXPTIME
NEXPSPACE
EXPSPACE
NP
P
NPSPACE
PSPACE
あり
なし
非決定性
NL
L
領域

# 復習：三次元的？ に書くと




計算量
指数
多項式
対数
時間
資源
NEXPTIME
EXPTIME
NEXPSPACE
EXPSPACE
NP
P
NPSPACE
PSPACE
あり
なし
非決定性
NL
L
領域

# 本日の内容



## 目標

次の証明を通して，計算複雑性に関する基本的な論法が使えるようになる

- $P \subseteq PSPACE$
  - $EXPTIME \subseteq EXPSPACE$，$NP \subseteq NPSPACE$，...
- $NP \subseteq PSPACE$
  - $NEXPTIME \subseteq EXPSPACE$
- $NL \subseteq P$
  - $NPSPACE \subseteq EXPTIME$
- $P = NP \Rightarrow EXPTIME = NEXPTIME$

# 本日の内容



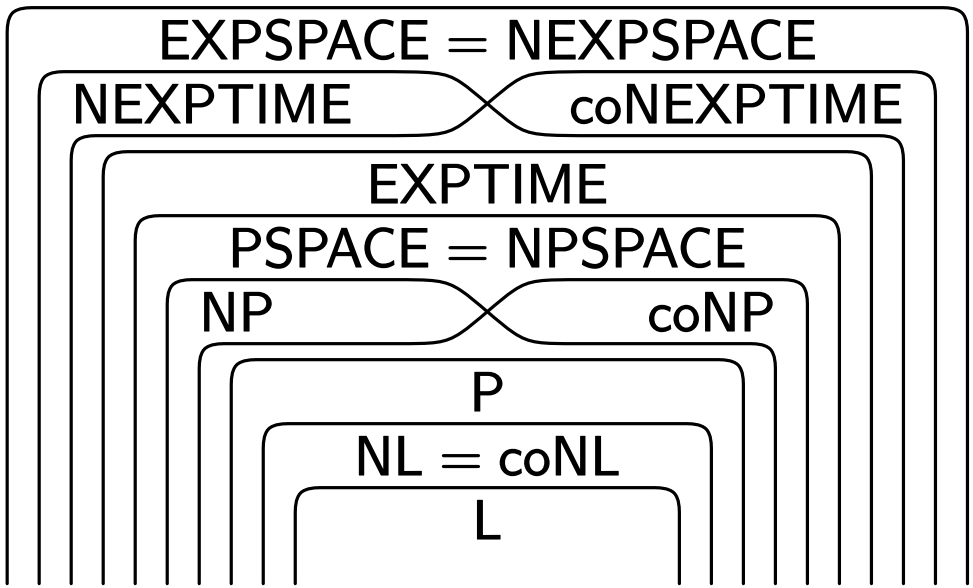
1. $\mathrm{P} \subseteq \mathrm{PSPACE}$
2. $\mathrm{NP} \subseteq \mathrm{PSPACE}$
3. $\mathrm{NL} \subseteq \mathrm{P}$
4. $\mathrm{P} = \mathrm{NP} \Rightarrow \mathrm{EXPTIME} = \mathrm{EXPSPACE}$

# P ⊆ PSPACE



## 性質

P ⊆ PSPACE




## 方針：$\mathcal{C} \subseteq \mathcal{D}$ を証明するには？

次のどちらかを行う

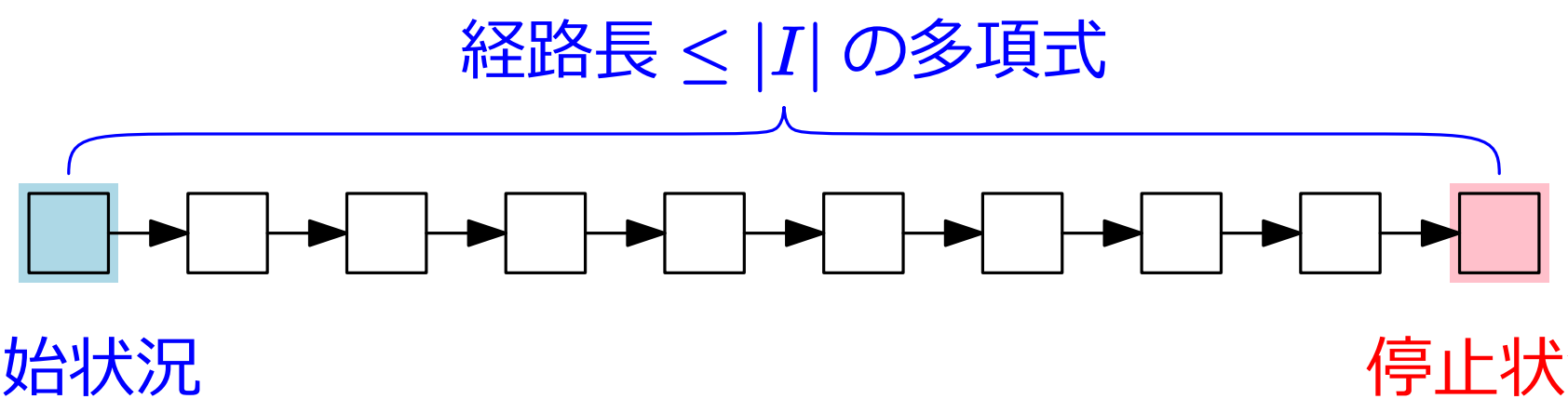
1. $\mathcal{C}$ に属する任意の問題が $\mathcal{D}$ に入ることを証明する
2. $\mathcal{C}$ 困難問題 1 つが $\mathcal{D}$ に入ることを証明する

# $P \subseteq$ PSPACE：方針 1



$P \in$ P として，$A$ は $P$ を解く多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A$ は多項式時間アルゴリズムなので
  $A(I)$ が通る時点状況の数 $\leq |I|$ の多項式
- 各時点状況において，
  $A(I)$ が読み書きを行う作業領域の量 $= O(1)$
- $A(I)$ が使う作業領域の総量 $\leq |I|$ の多項式
- $\therefore P \in$ PSPACE □



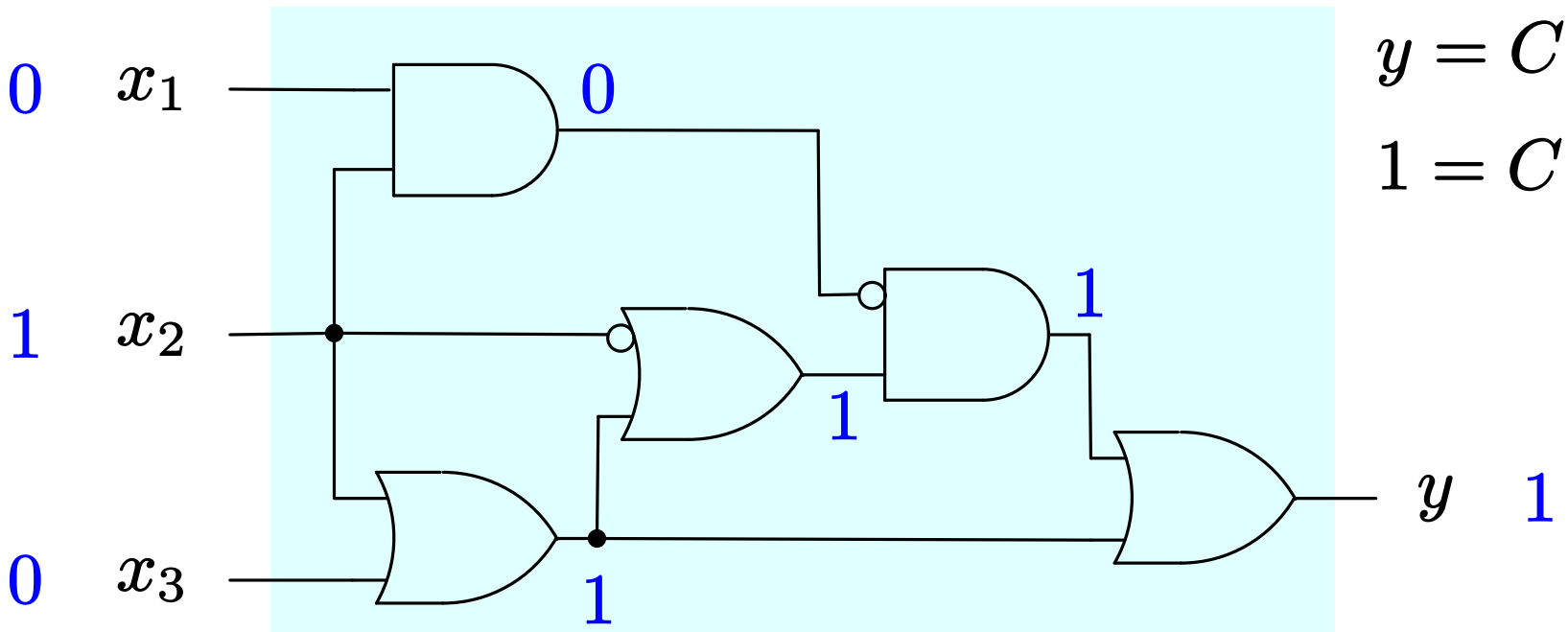
# P ⊆ PSPACE：方針 2



次の P 完全問題を考える

**問題：論理回路評価問題 (circuit value problem)**

**入力：** 論理回路 $C$，割当 $\alpha$
**出力：** $C(\alpha) = 1 \Rightarrow$ Yes
$C(\alpha) = 0 \Rightarrow$ No



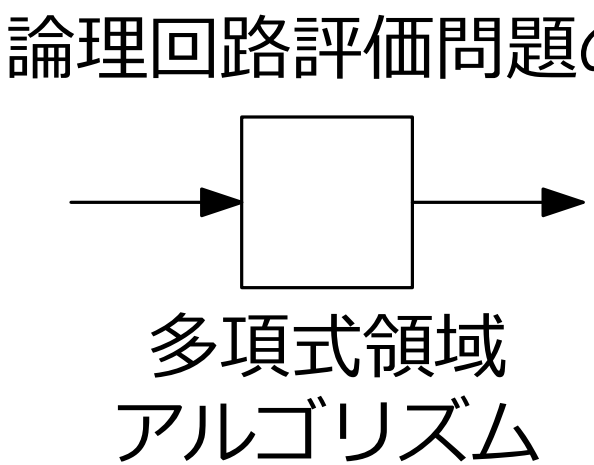

$y = C(x_1, x_2, x_3)$

$1 = C(0, 1, 0)$

# P ⊆ PSPACE：方針 2 の原理



論理回路評価問題 ∈ PSPACE と証明できたら · · ·



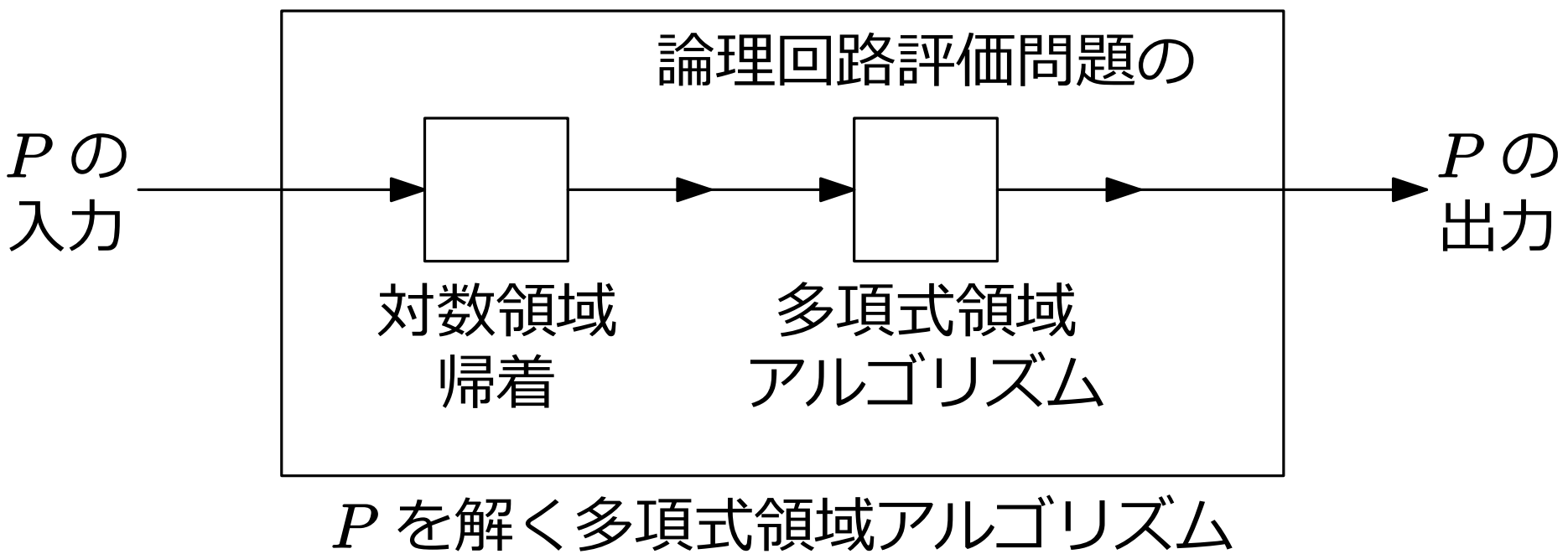
# P ⊆ PSPACE：方針 2 の原理



論理回路評価問題 ∈ PSPACE と証明できたら・・・

論理回路評価問題が P 困難であるので,

任意の問題 $P \in$ P が多項式領域アルゴリズムで解ける



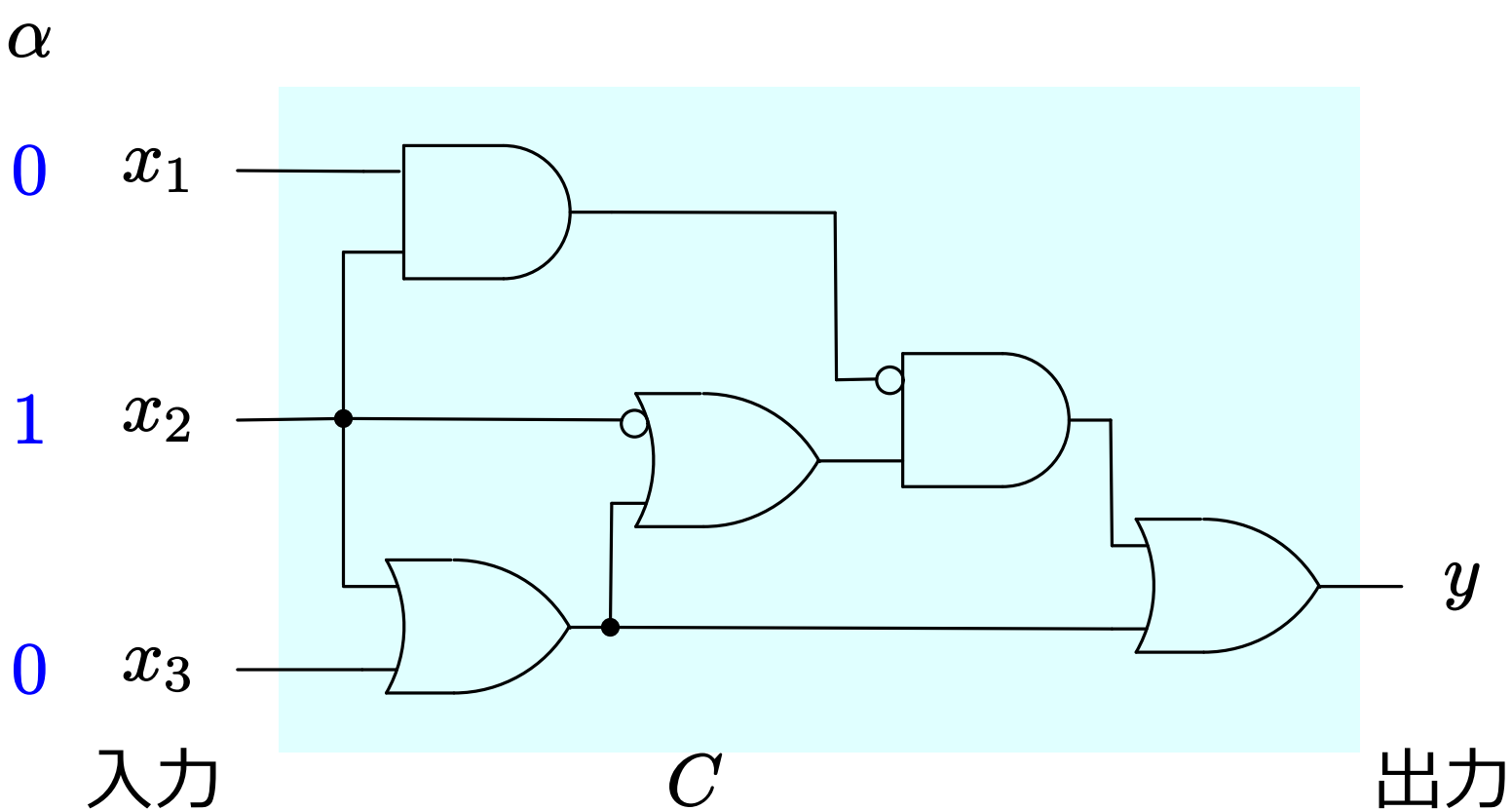
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



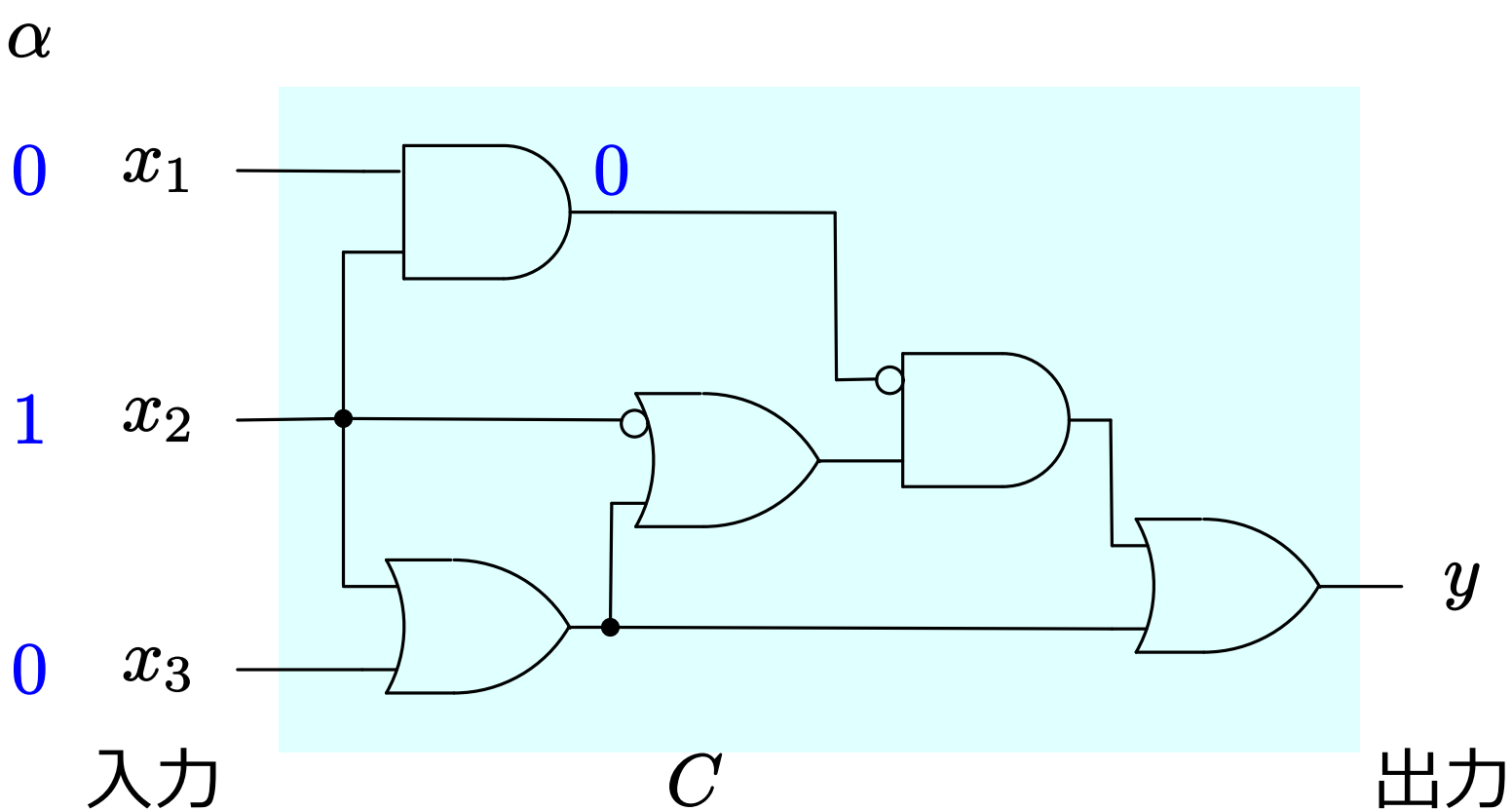
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



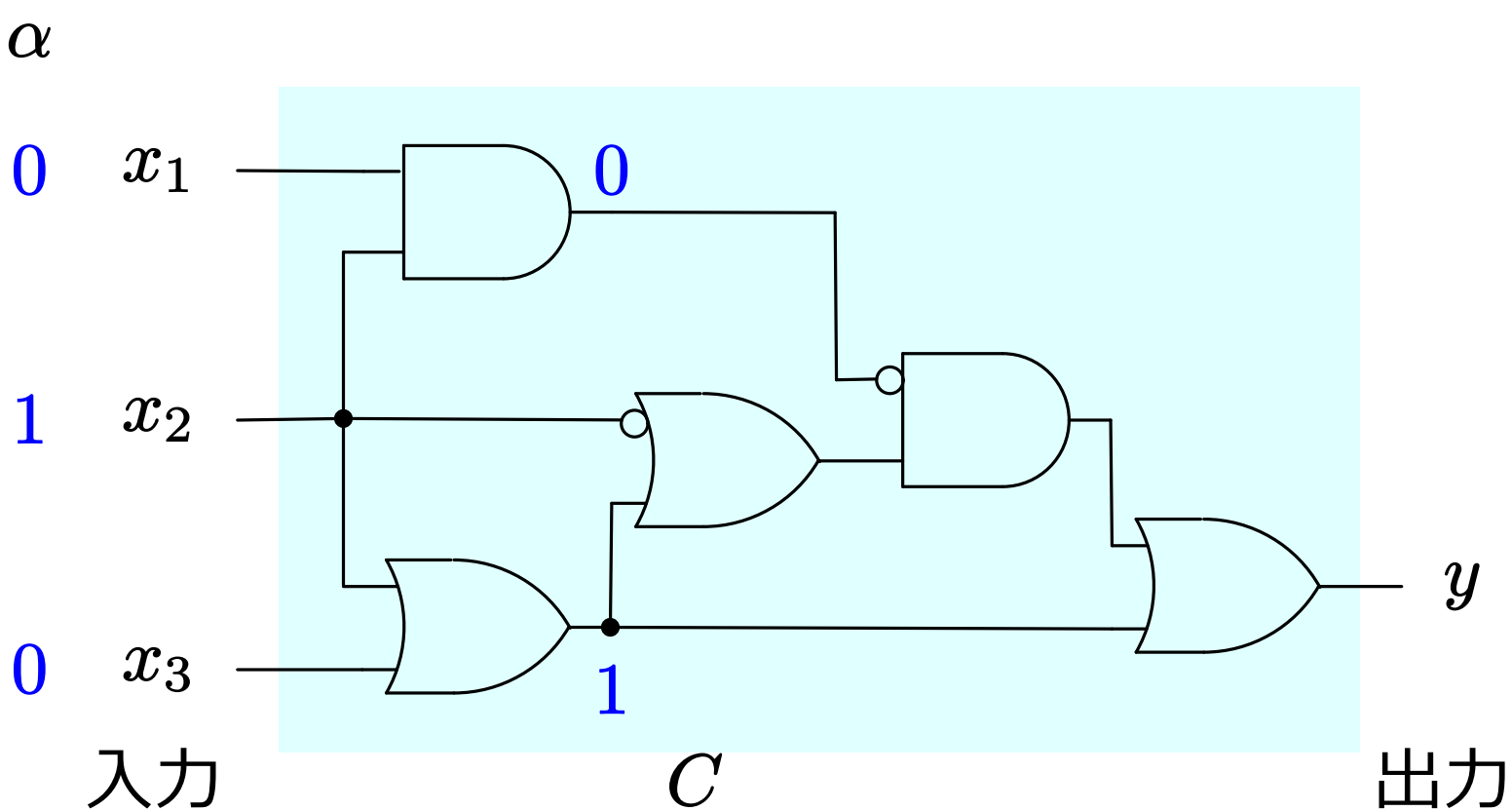
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



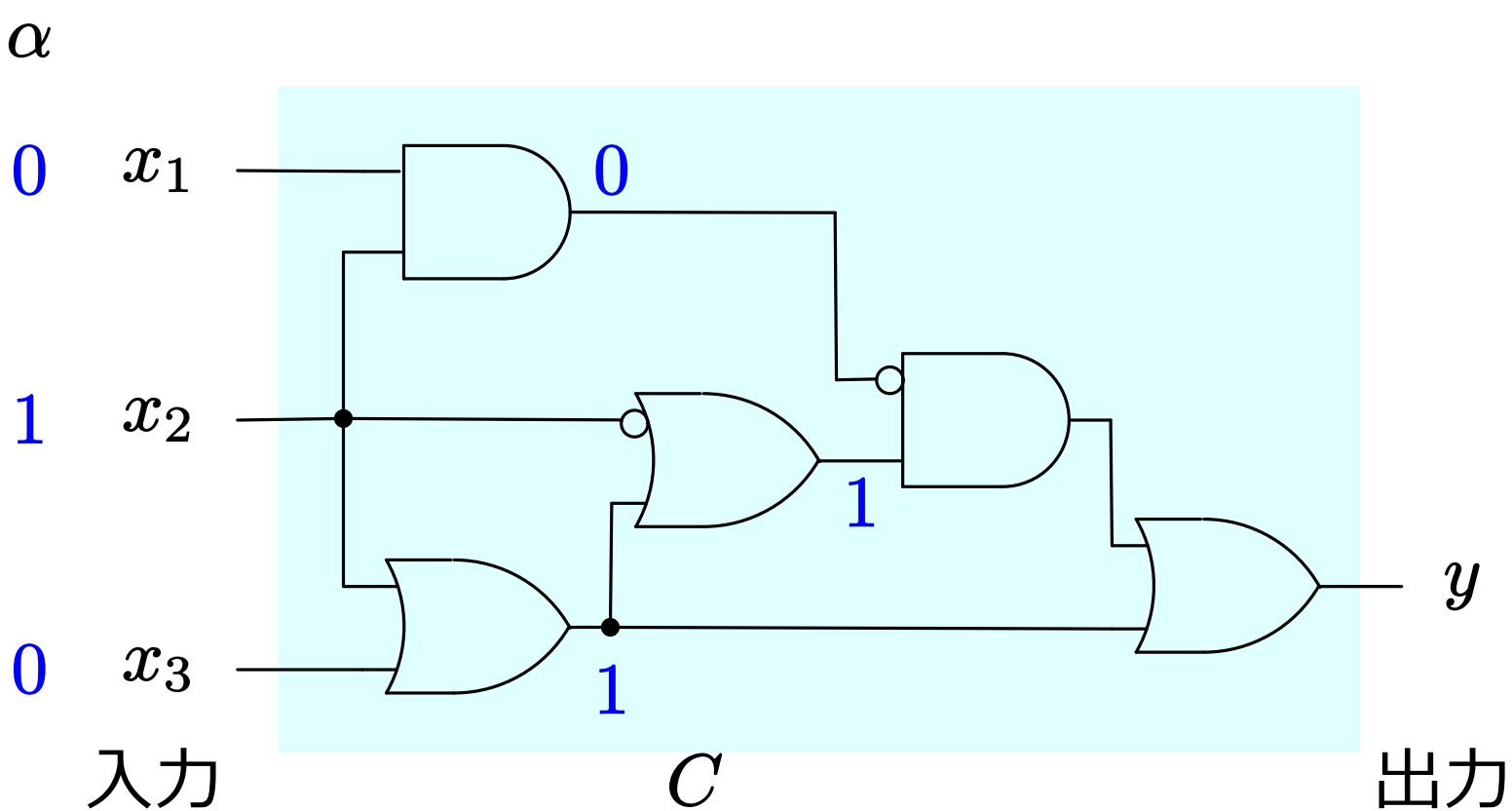
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



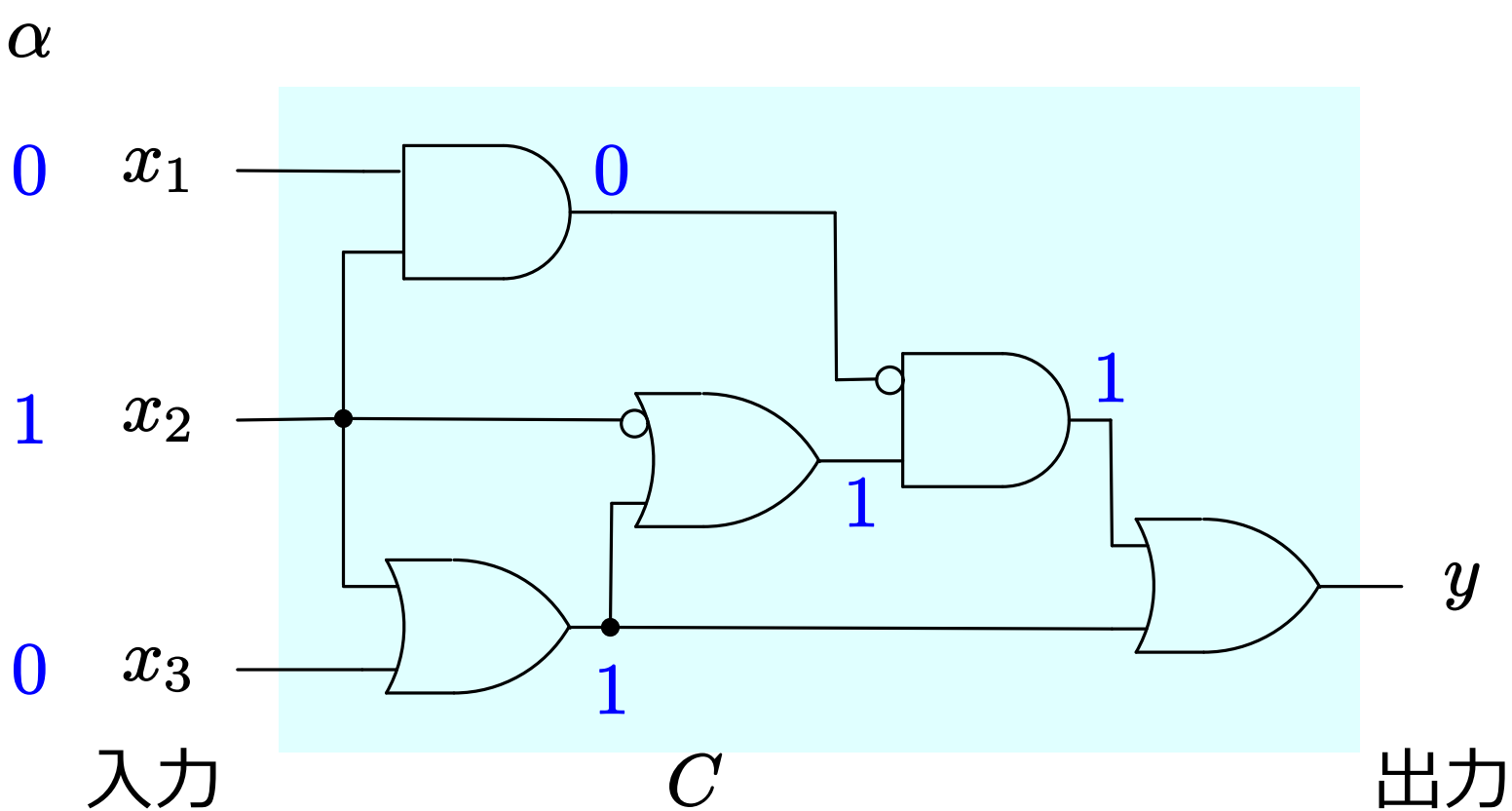
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



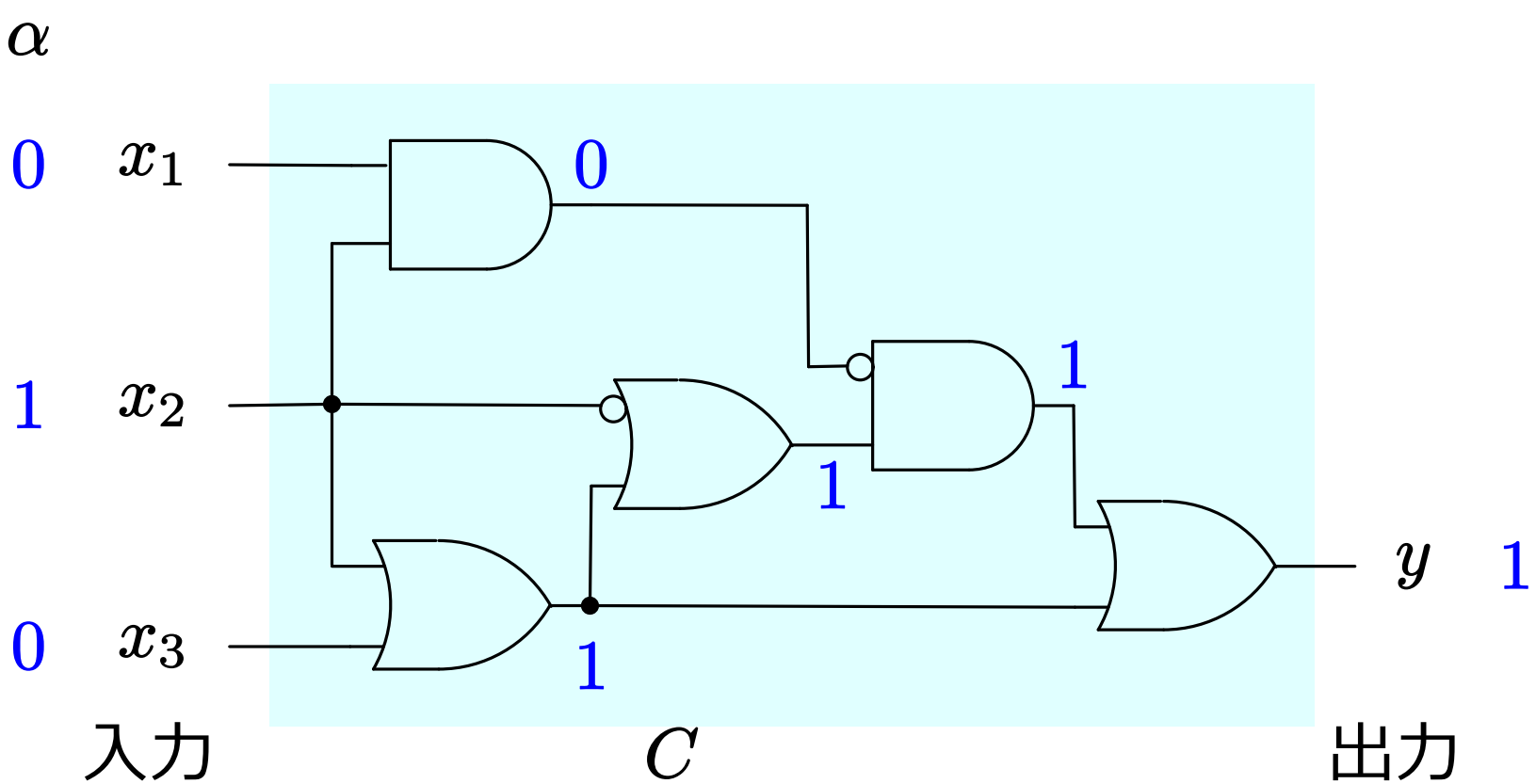
# 論理回路評価問題を解く



入力された論理回路 $C$ と割当 $\alpha$ に対して，

「左から順」に，素子の出力を決めていけばよい

$\rightsquigarrow$ 必要な作業領域量 $= O(|C|)$



# 方針の得失



**方針：$\mathcal{C} \subseteq \mathcal{D}$ を証明する方針 1**

$\mathcal{C}$ に属する任意の問題が $\mathcal{D}$ に入ることを証明する

メリット：　他のクラスも同様に証明できる

デメリット：抽象的で，分かりにくい

**方針：$\mathcal{C} \subseteq \mathcal{D}$ を証明する方針 2**

$\mathcal{C}$ 困難問題 1 つが $\mathcal{D}$ に入ることを証明する

メリット：　具体的で，分かりやすい

デメリット：困難性 (完全性) を別に証明する必要がある

# EXPTIME ⊆ EXPSPACE：方針 1

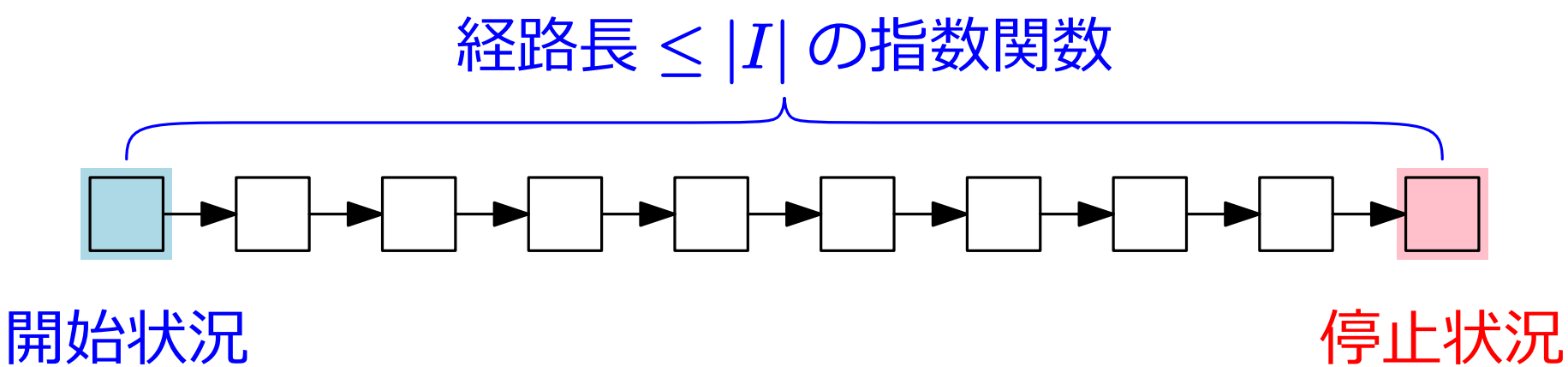


**性質**

EXPTIME ⊆ EXPSPACE

略証：P ⊆ PSPACE の証明 (方針 1) において

「多項式」を「指数 (関数)」に置き換えればよい　□



# NP ⊆ NPSPACE：方針 1

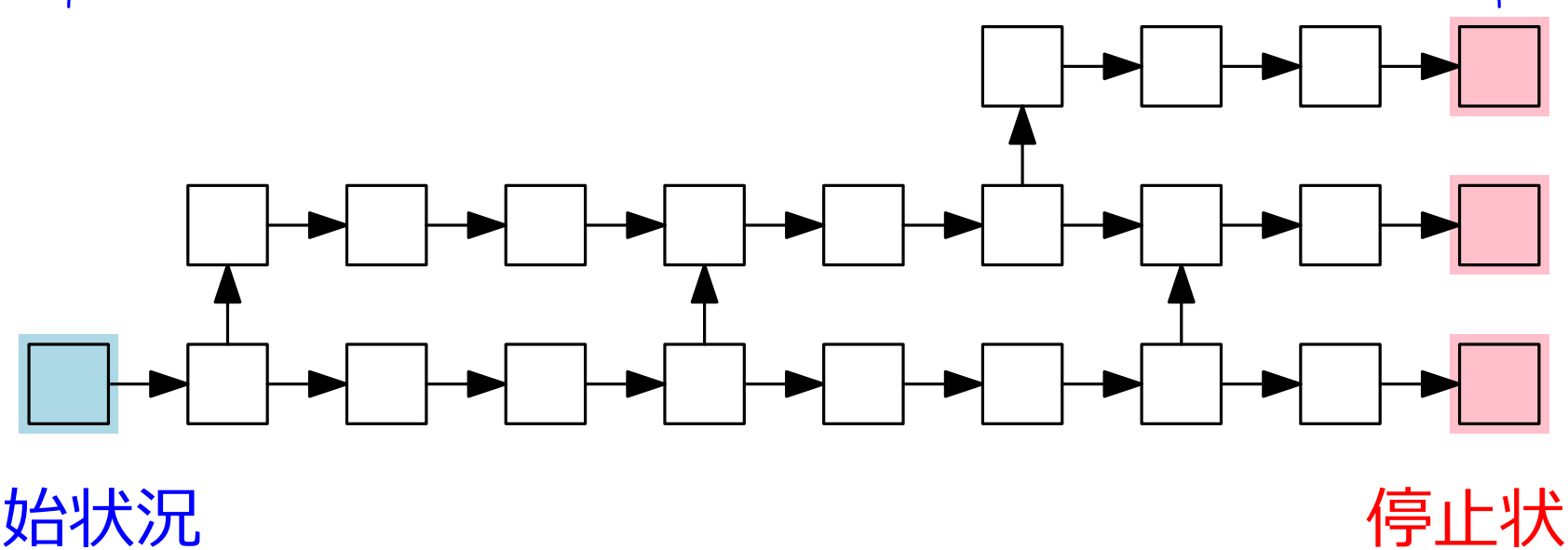


**性質**

NP ⊆ NPSPACE

略証：P ⊆ PSPACE の証明 (方針 1) の証明で

アルゴリズムを非決定性に変えればよい □



# 本日の内容



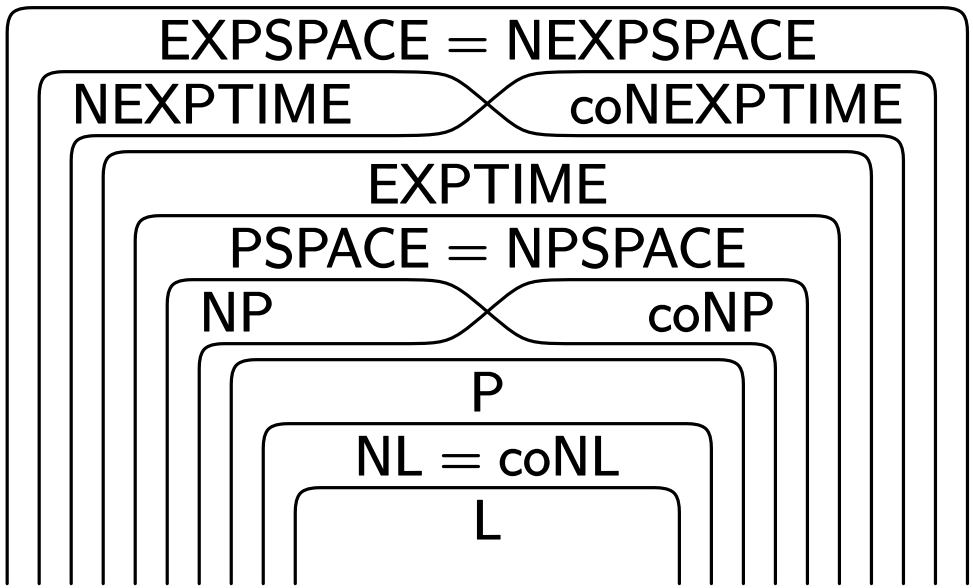
1. $\mathsf{P} \subseteq \mathsf{PSPACE}$
2. $\mathsf{NP} \subseteq \mathsf{PSPACE}$
3. $\mathsf{NL} \subseteq \mathsf{P}$
4. $\mathsf{P} = \mathsf{NP} \Rightarrow \mathsf{EXPTIME} = \mathsf{EXPSPACE}$

# NP ⊆ PSPACE



## 性質

NP ⊆ PSPACE




実際は，PSPACE = NPSPACE なので，

NP ⊆ NPSPACE = PSPACE であるとすぐ分かるが，

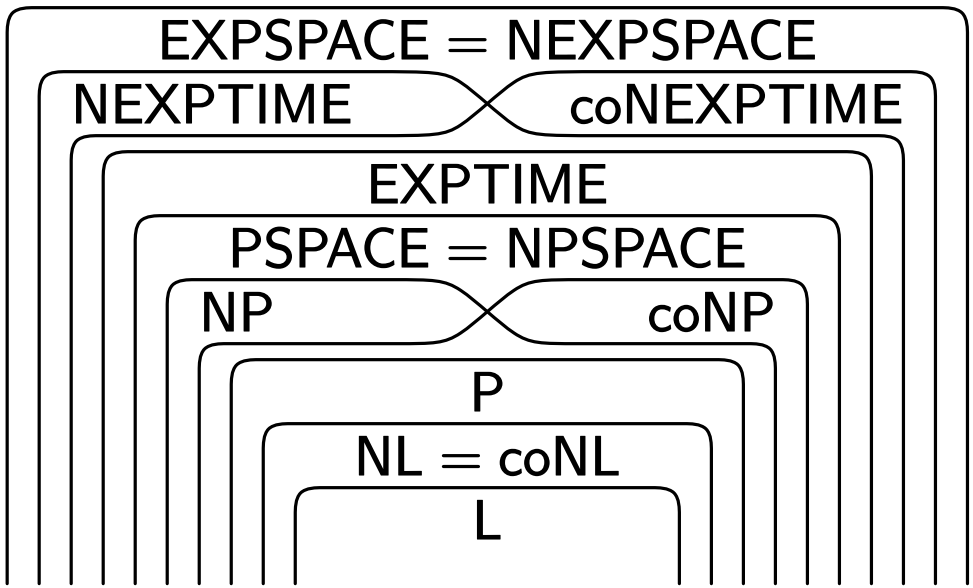
PSPACE = NPSPACE を知らなくても，証明できる

# NP ⊆ PSPACE



## 性質

NP ⊆ PSPACE




## 方針：$\mathcal{C} \subseteq \mathcal{D}$ を証明するには？

次のどちらかを行う

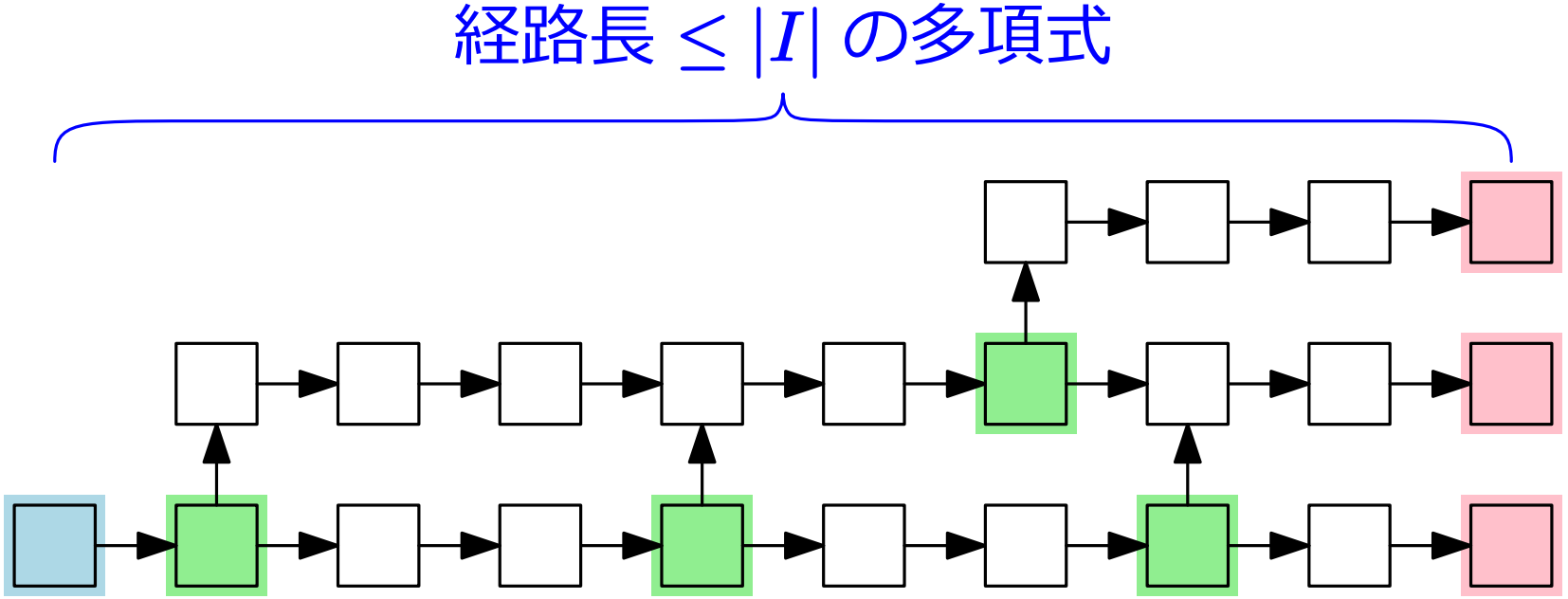
1. $\mathcal{C}$ に属する任意の問題が $\mathcal{D}$ に入ることを証明する
2. $\mathcal{C}$ 困難問題 1 つが $\mathcal{D}$ に入ることを証明する

# NP ⊆ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

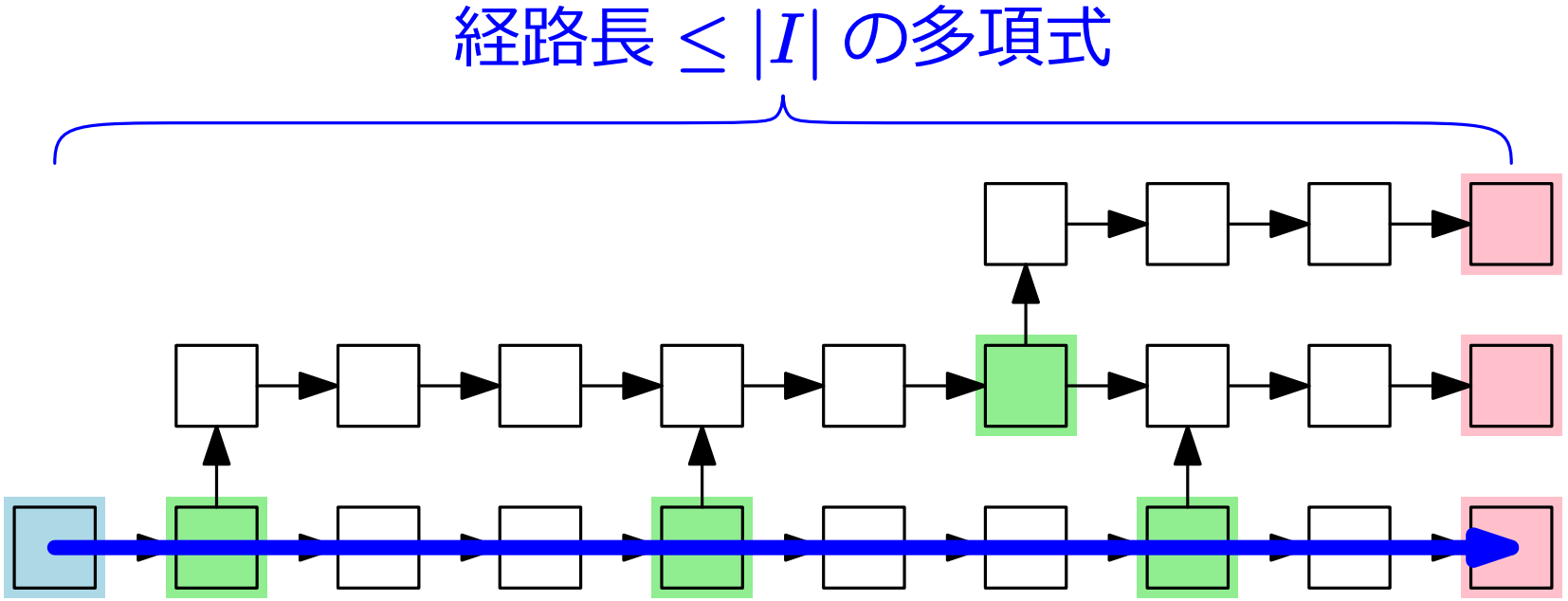


# NP ⊆ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

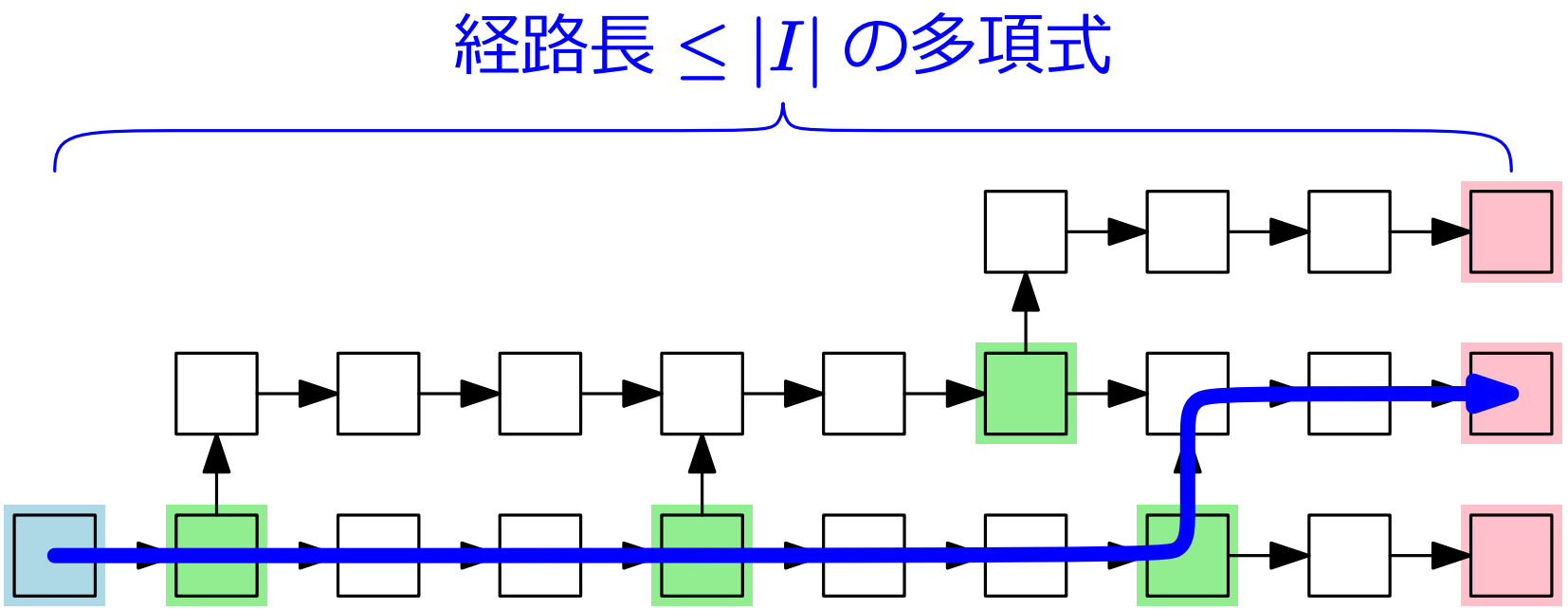


# NP $\subseteq$ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

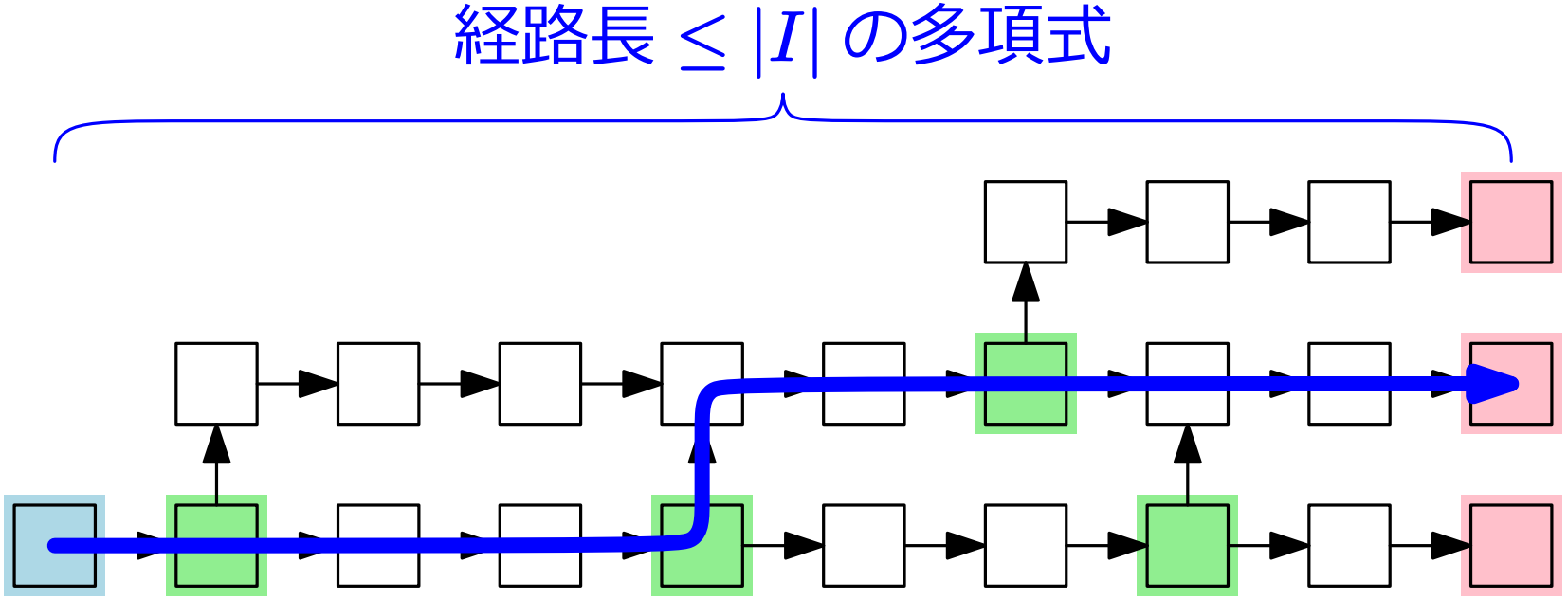


# NP $\subseteq$ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

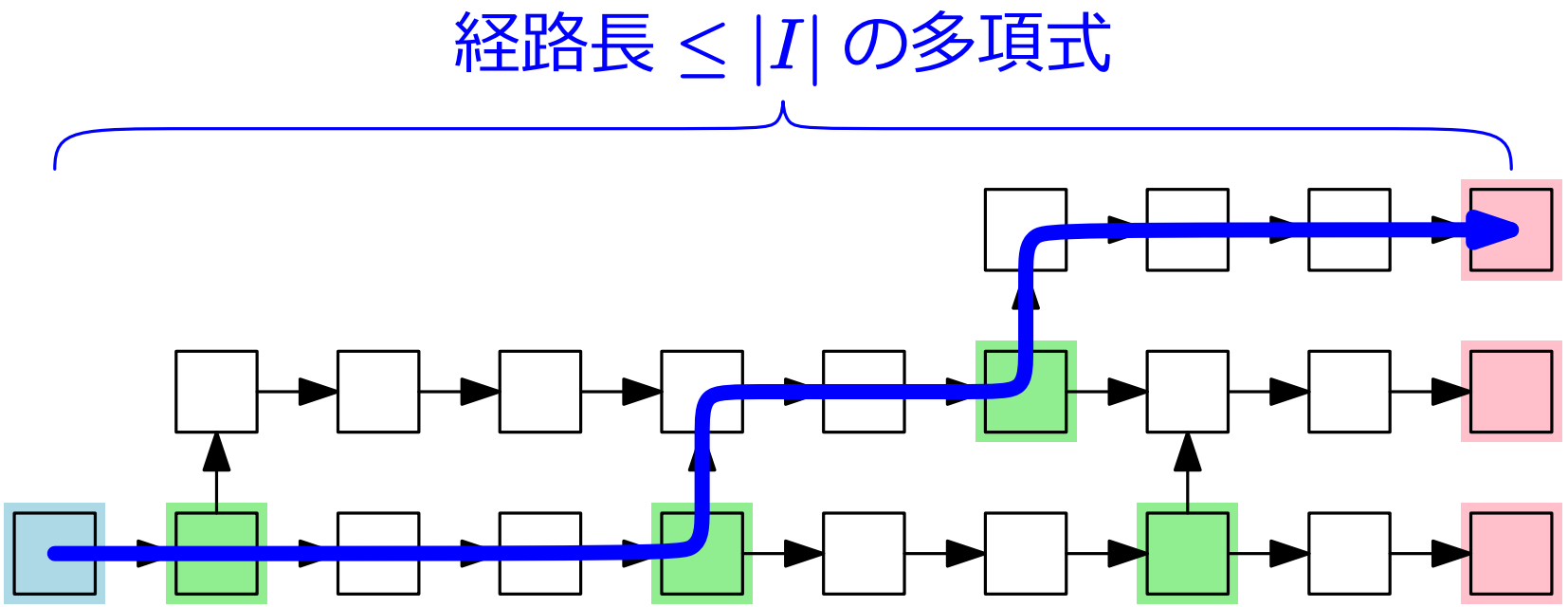


# NP ⊆ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

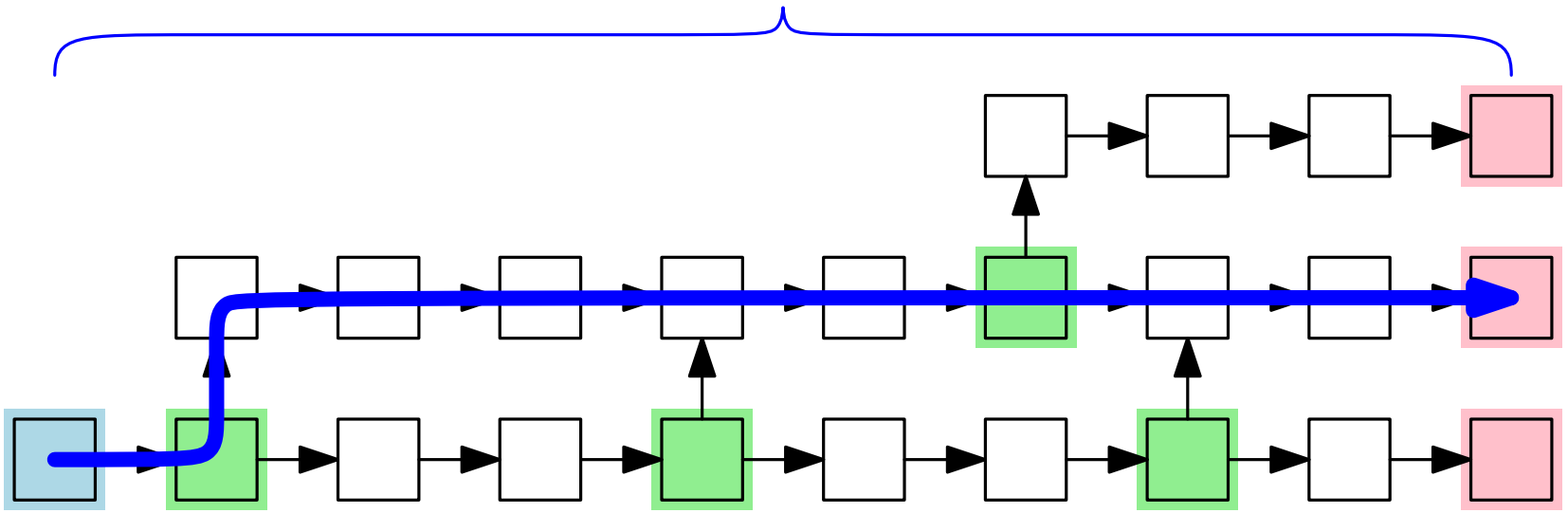


# NP $\subseteq$ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □

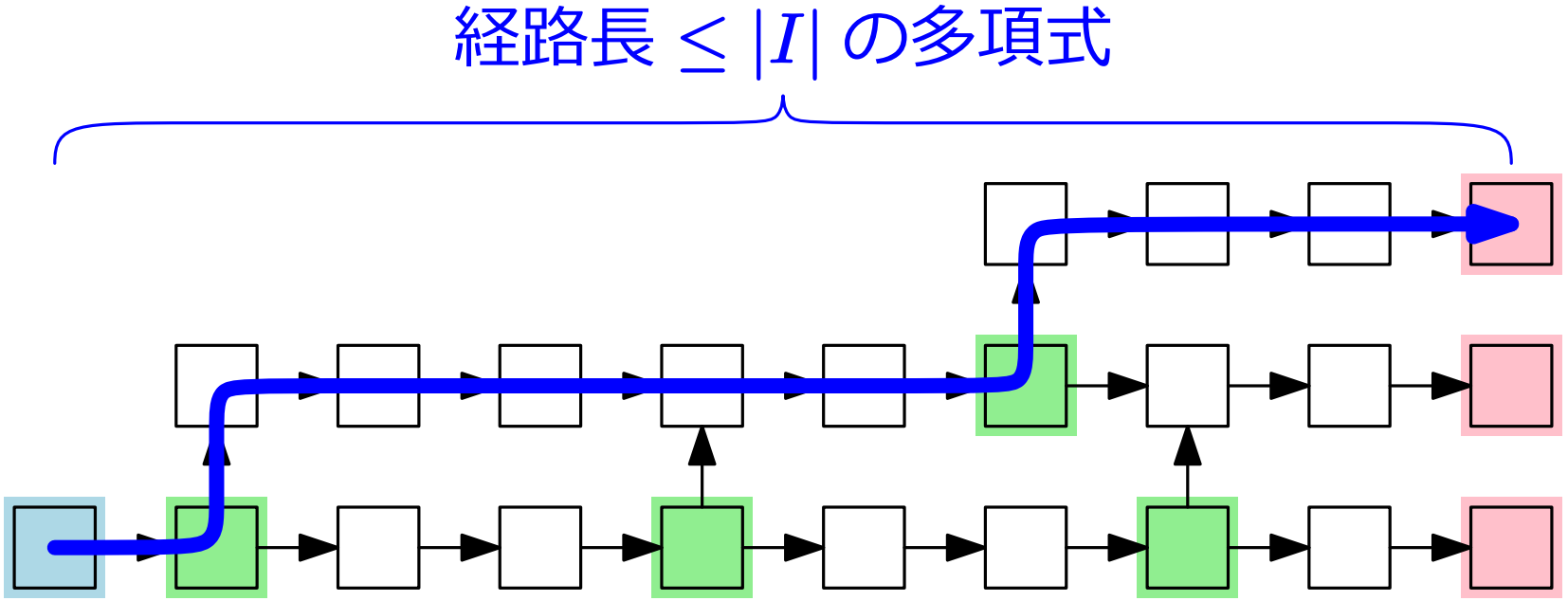


# NP ⊆ PSPACE：方針 1



$P \in$ NP として,
$A$ は $P$ を解く非決定性多項式時間アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ が行う各 guess の候補を覚えれば,
  すべての経路に対する計算を
  非決定性のないアルゴリズムで行うことができる
- 作業領域量 $\leq |I|$の多項式 + guess の回数 $= |I|$の多項式
- $\therefore P \in$ PSPACE □



# NP $\subseteq$ PSPACE：方針 2

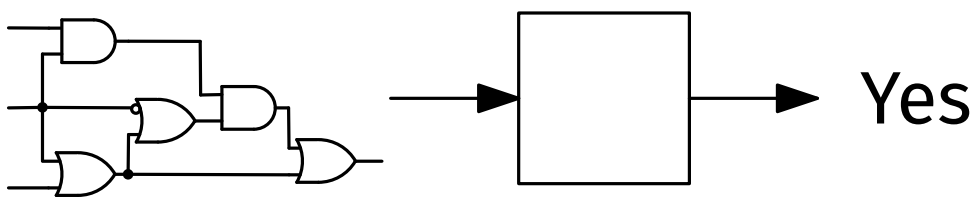


次の NP 完全問題を考える

**論理回路の充足可能性問題 (CIRCUIT-SAT)**

**入力：** 論理回路 $C$
**出力：** $C$ が充足割当を持つ $\Rightarrow$ Yes
$C$ が充足割当を持たない $\Rightarrow$ No

充足可能性問題 $=$ Satisfiability Problem



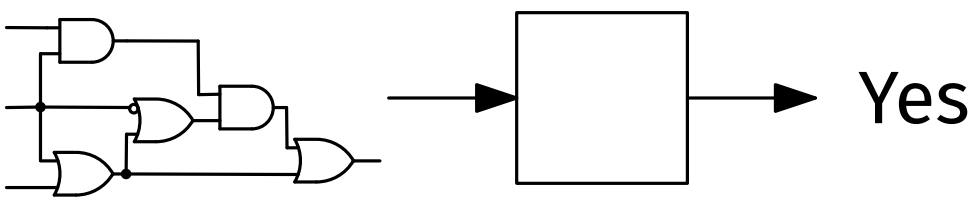

**定義：充足割当 (satisfying assignment)**

論理回路 $C(x)$ に対する割当 $\alpha$ が **充足割当** であるとは
$C(\alpha) = 1$ であること

# 充足可能性問題に対するアルゴリズム



1: すべての可能な割当 $\alpha$ に対して次を行う
 1-1: $C(\alpha)$ を計算 ← 多項式時間でできる
 1-2: $C(\alpha) = 1$ ならば，Yes を出力して終了
2: ここまで到達したら，No を出力して終了




作業領域量 $= O(|C|) + |\alpha| = O(|C|)$

$\therefore$ CIRCUIT-SAT $\in$ PSPACE $\square$

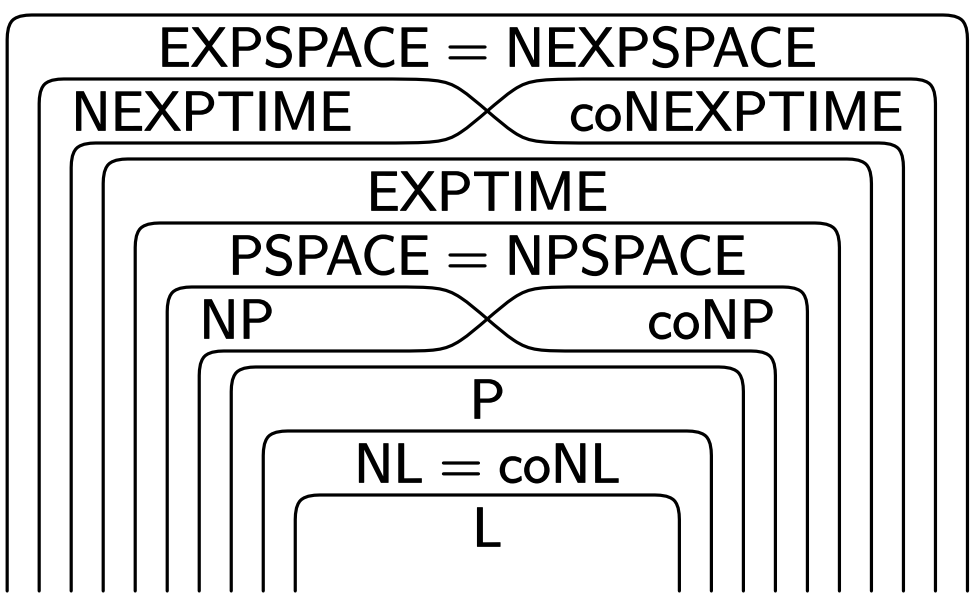
# NEXPTIME ⊆ EXPTIME など



方針 1 に従って，次も同様に証明できる

**性質**

- NEXPTIME ⊆ EXPSPACE
- coNP ⊆ PSPACE
- coNEXPTIME ⊆ EXPSPACE



# 本日の内容



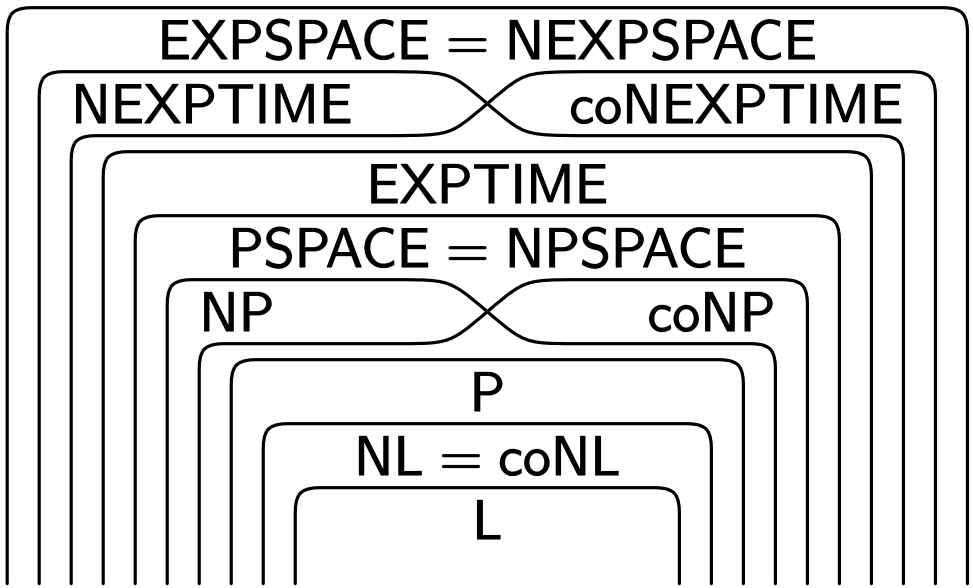
1. $\mathrm{P} \subseteq \mathrm{PSPACE}$
2. $\mathrm{NP} \subseteq \mathrm{PSPACE}$
3. $\mathrm{NL} \subseteq \mathrm{P}$
4. $\mathrm{P} = \mathrm{NP} \Rightarrow \mathrm{EXPTIME} = \mathrm{EXPSPACE}$

# NL ⊆ P



## 性質

NL ⊆ P




## 方針：$\mathcal{C} \subseteq \mathcal{D}$ を証明するには？

次のどちらかを行う

1. $\mathcal{C}$ に属する任意の問題が $\mathcal{D}$ に入ることを証明する
2. $\mathcal{C}$ 困難問題 1 つが $\mathcal{D}$ に入ることを証明する

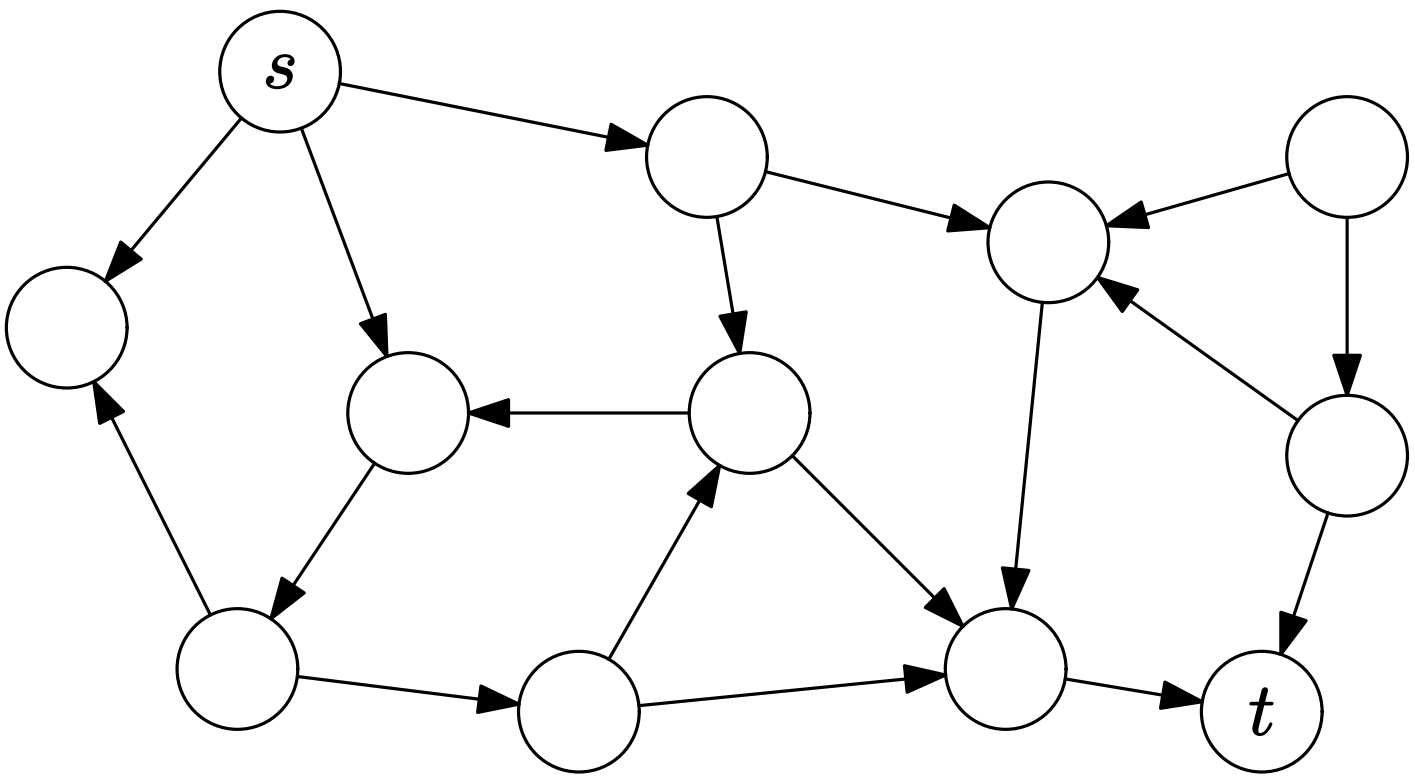
# STCON：典型的な NL 完全問題



## 有向グラフ連結性判定問題 (STCON)

**入力：** 有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$
**出力：** $s$ から $t$ へ至る経路が存在する　$\Rightarrow$ Yes
$s$ から $t$ へ至る経路が存在しない $\Rightarrow$ No

STCON は PATH とも呼ばれる



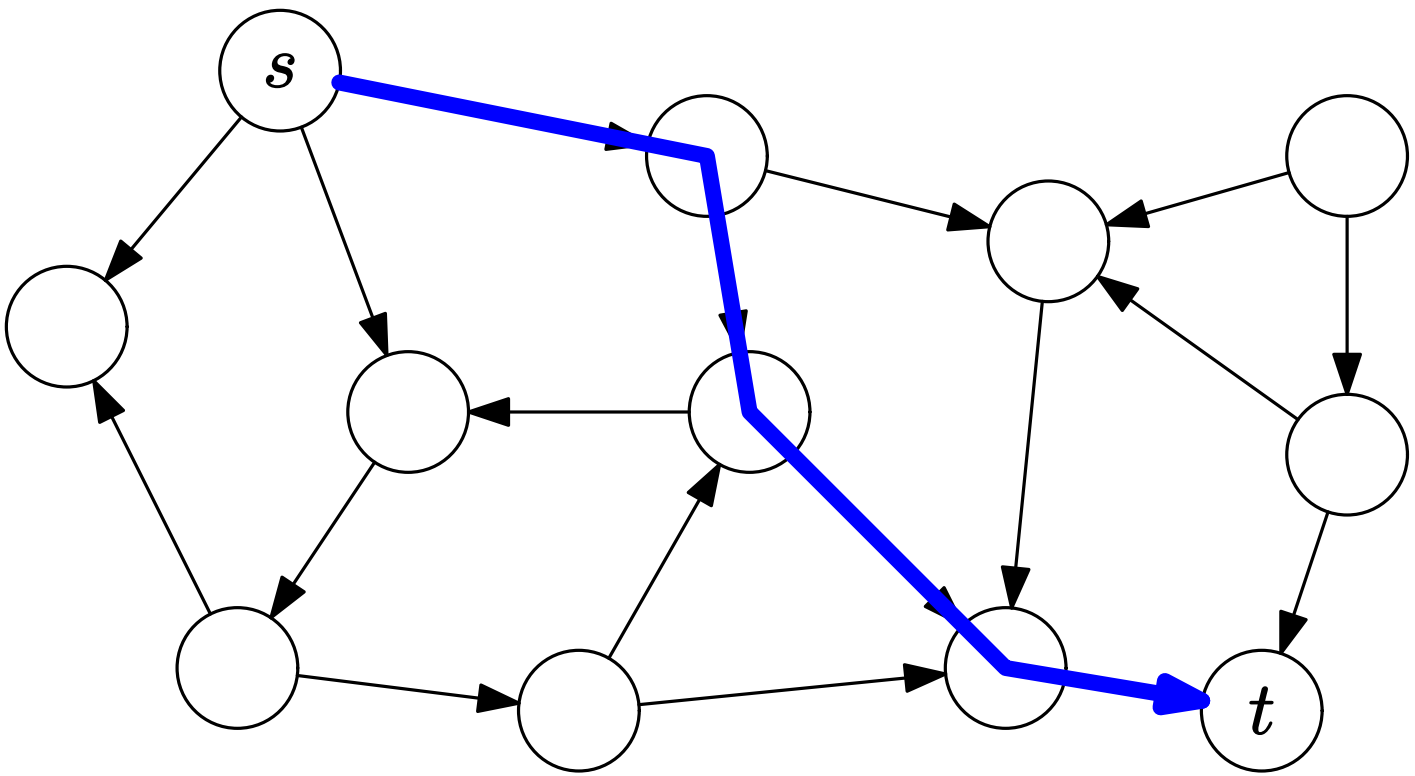
# STCON：典型的な NL 完全問題



## 有向グラフ連結性判定問題 (STCON)

**入力：** 有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$
**出力：** $s$ から $t$ へ至る経路が存在する　⇒ Yes
$s$ から $t$ へ至る経路が存在しない ⇒ No

STCON は PATH とも呼ばれる




Yes

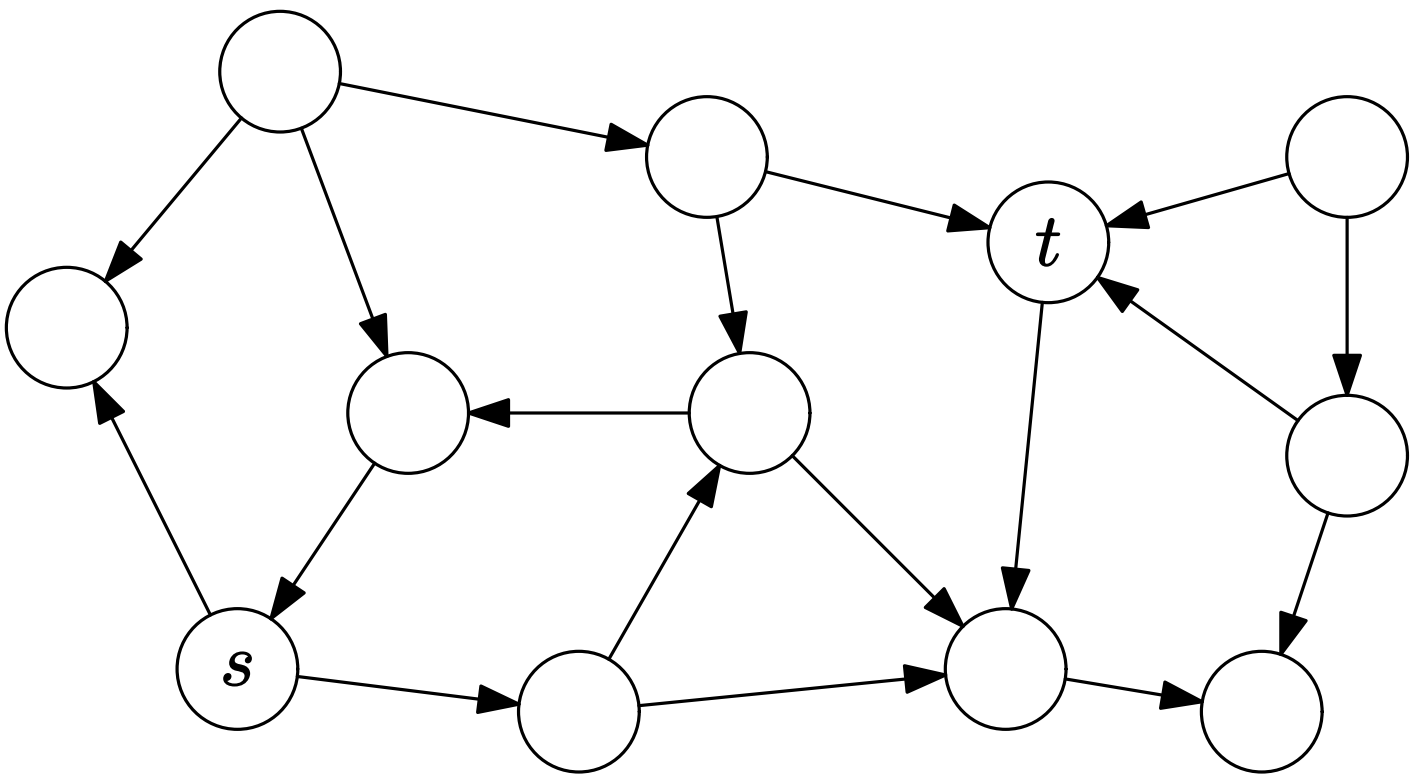
# STCON：典型的な NL 完全問題



## 有向グラフ連結性判定問題 (STCON)

**入力：** 有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$
**出力：** $s$ から $t$ へ至る経路が存在する　$\Rightarrow$ Yes
$s$ から $t$ へ至る経路が存在しない $\Rightarrow$ No

STCON は PATH とも呼ばれる



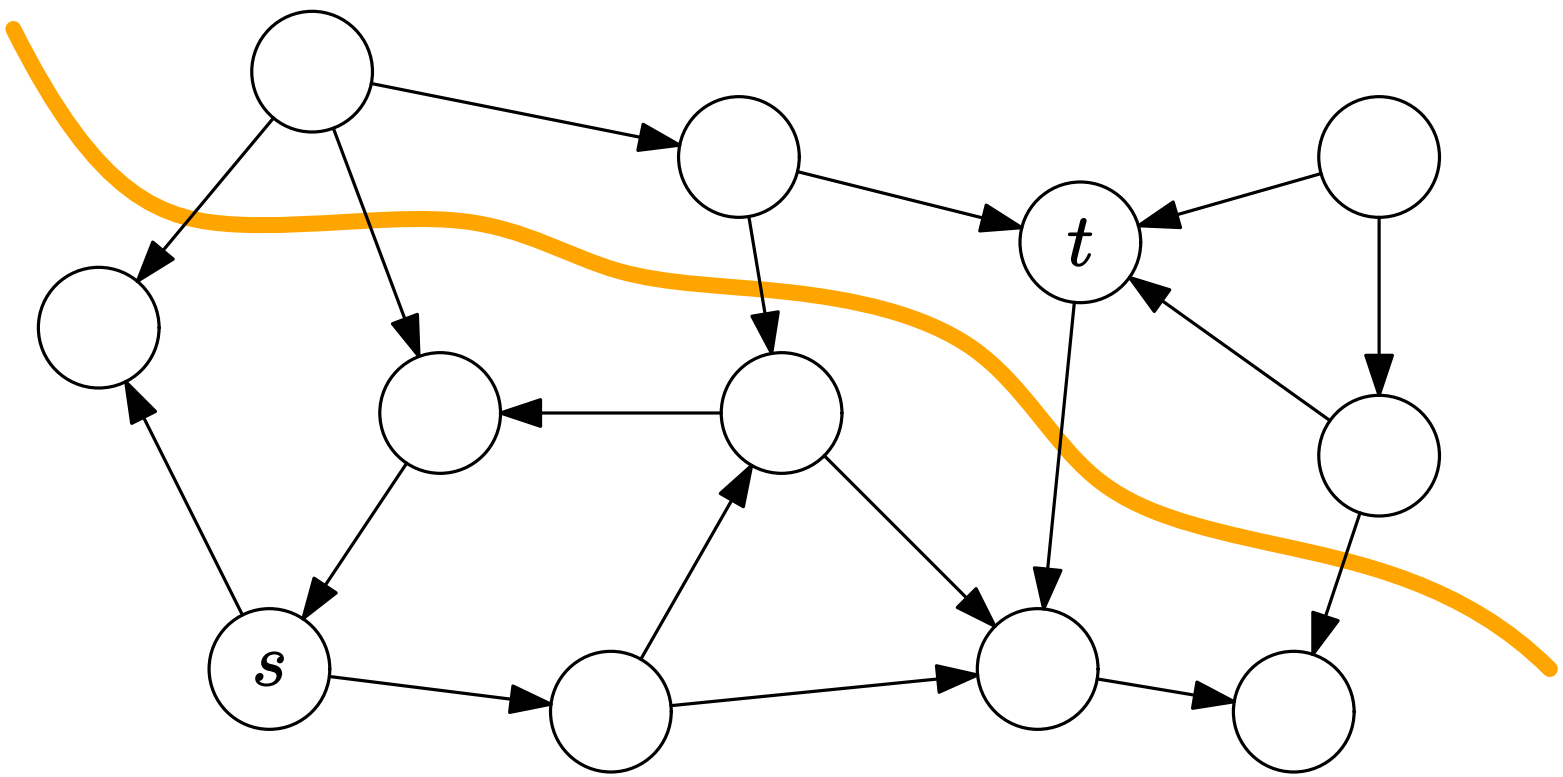
# STCON：典型的な NL 完全問題



## 有向グラフ連結性判定問題 (STCON)

**入力：** 有向グラフ $G = (V, A)$，2 頂点 $s, t \in V$
**出力：** $s$ から $t$ へ至る経路が存在する　$\Rightarrow$ Yes
$s$ から $t$ へ至る経路が存在しない $\Rightarrow$ No

STCON は PATH とも呼ばれる



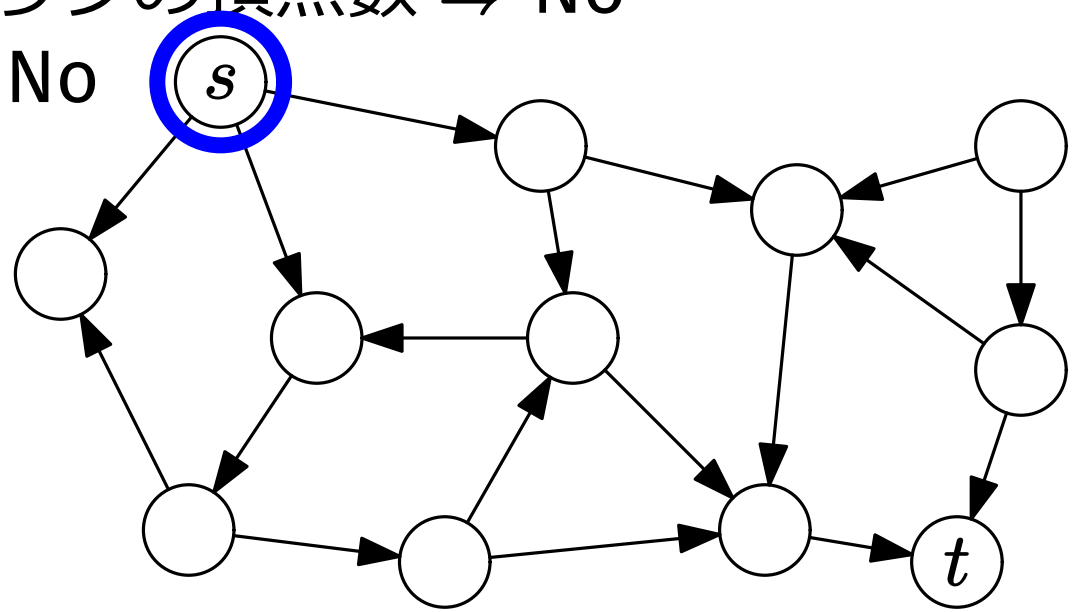

No

## 性質 (証明しない)

STCON は NL 完全

STCON ∈ NL は次のように分かる

- 変数：現在の頂点，訪れた頂点の数
- アルゴリズム：経路上の次の頂点を guess
  $t$ に到達 ⇒ Yes
  訪れた頂点の数 > グラフの頂点数 ⇒ No
  移動できなくなる ⇒ No

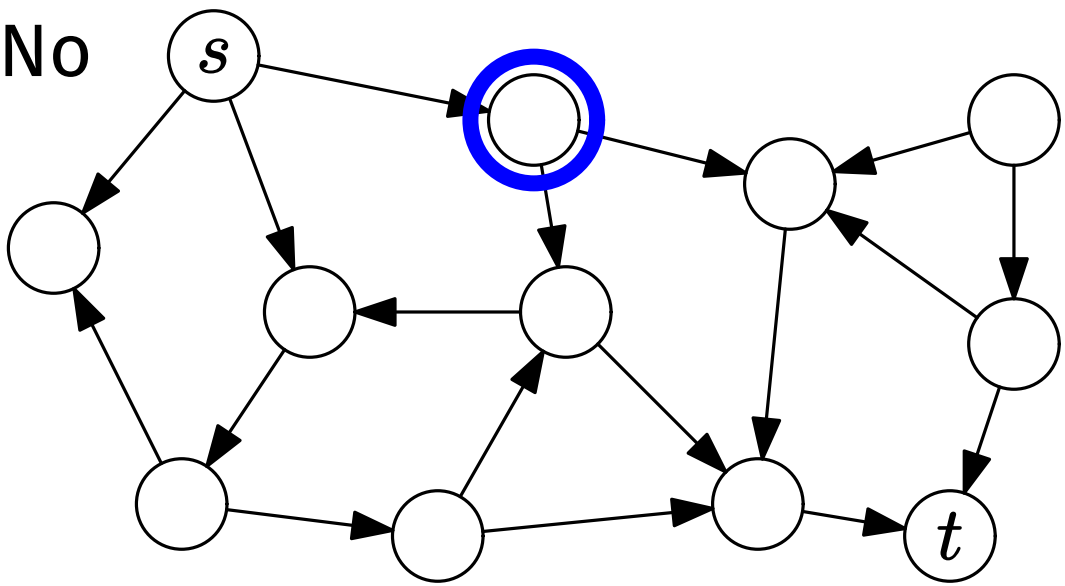


## 性質 (証明しない)

STCON は NL 完全

STCON ∈ NL は次のように分かる

- 変数：現在の頂点，訪れた頂点の数
- アルゴリズム：経路上の次の頂点を guess
  $t$ に到達 ⇒ Yes
  訪れた頂点の数 > グラフの頂点数 ⇒ No
  移動できなくなる ⇒ No

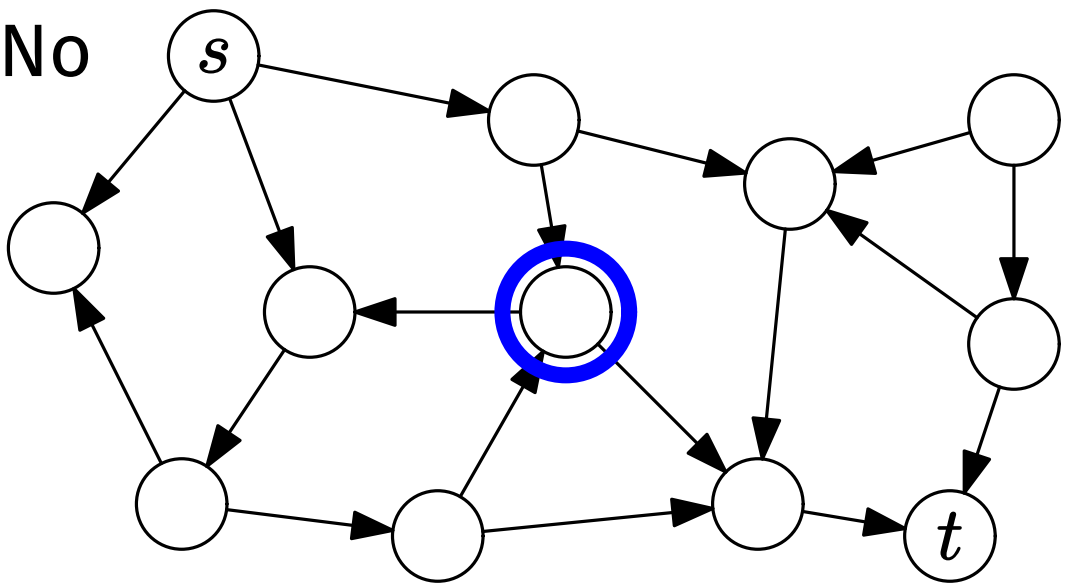


## 性質 (証明しない)

STCON は NL 完全

STCON $\in$ NL は次のように分かる

- 変数：現在の頂点，訪れた頂点の数
- アルゴリズム：経路上の次の頂点を guess
  $t$ に到達 $\Rightarrow$ Yes
  訪れた頂点の数 $>$ グラフの頂点数 $\Rightarrow$ No
  移動できなくなる $\Rightarrow$ No

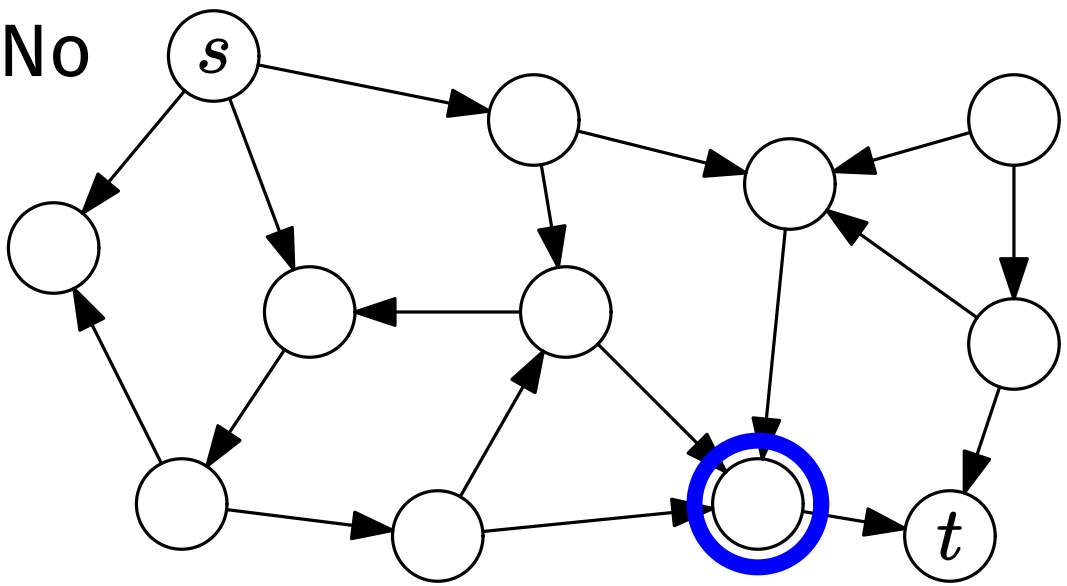


## 性質 (証明しない)

STCON は NL 完全

STCON $\in$ NL は次のように分かる

- 変数：現在の頂点，訪れた頂点の数
- アルゴリズム：経路上の次の頂点を guess
  $t$ に到達 $\Rightarrow$ Yes
  訪れた頂点の数 $>$ グラフの頂点数 $\Rightarrow$ No
  移動できなくなる $\Rightarrow$ No

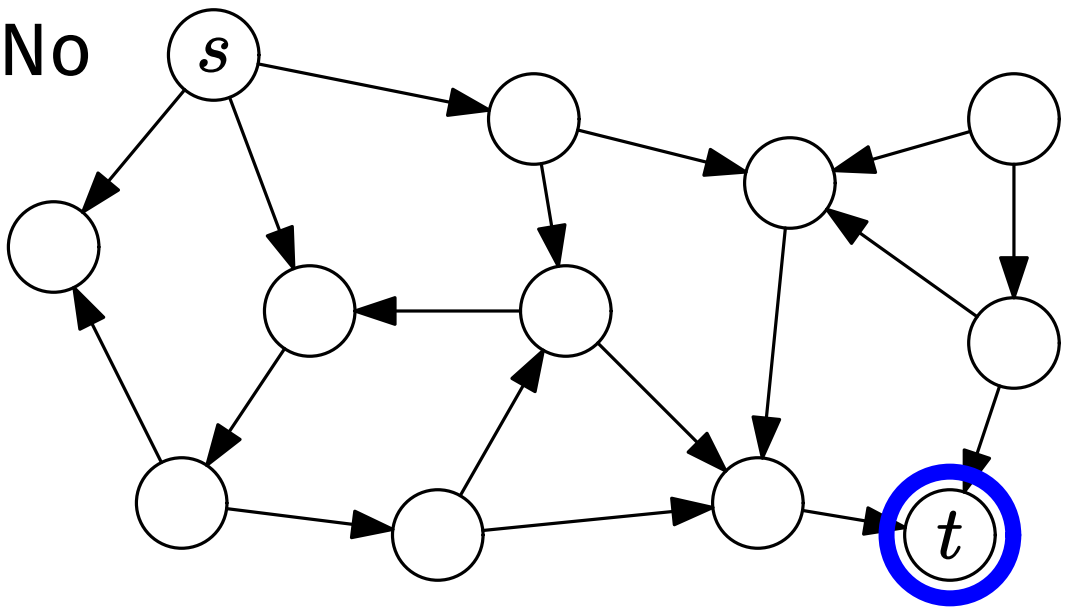


## 性質 (証明しない)

STCON は NL 完全

STCON ∈ NL は次のように分かる

- 変数：現在の頂点，訪れた頂点の数
- アルゴリズム：経路上の次の頂点を guess
  $t$ に到達 ⇒ Yes
  訪れた頂点の数 > グラフの頂点数 ⇒ No
  移動できなくなる ⇒ No

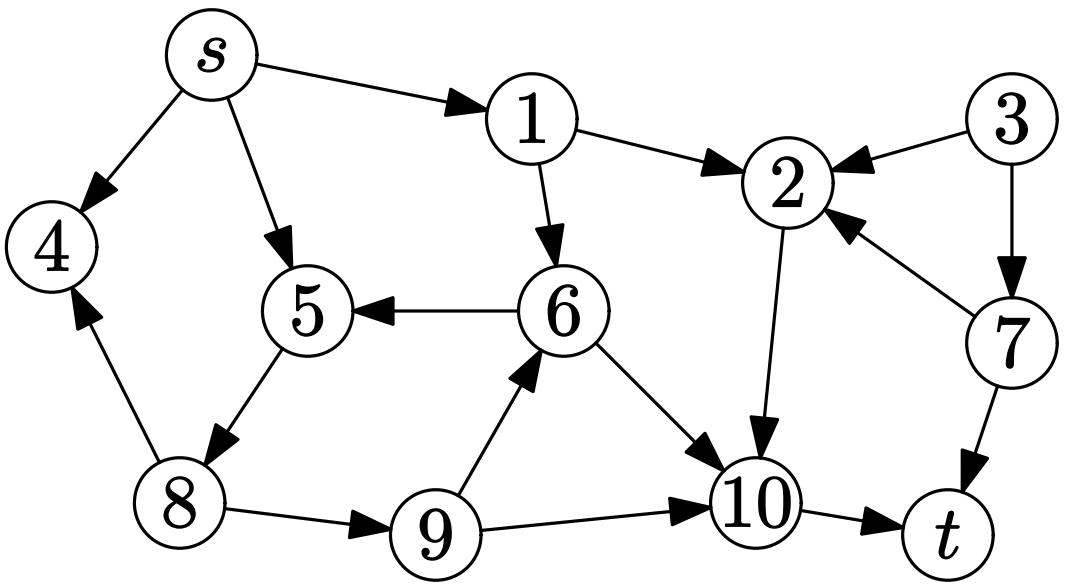


**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム



**性質**

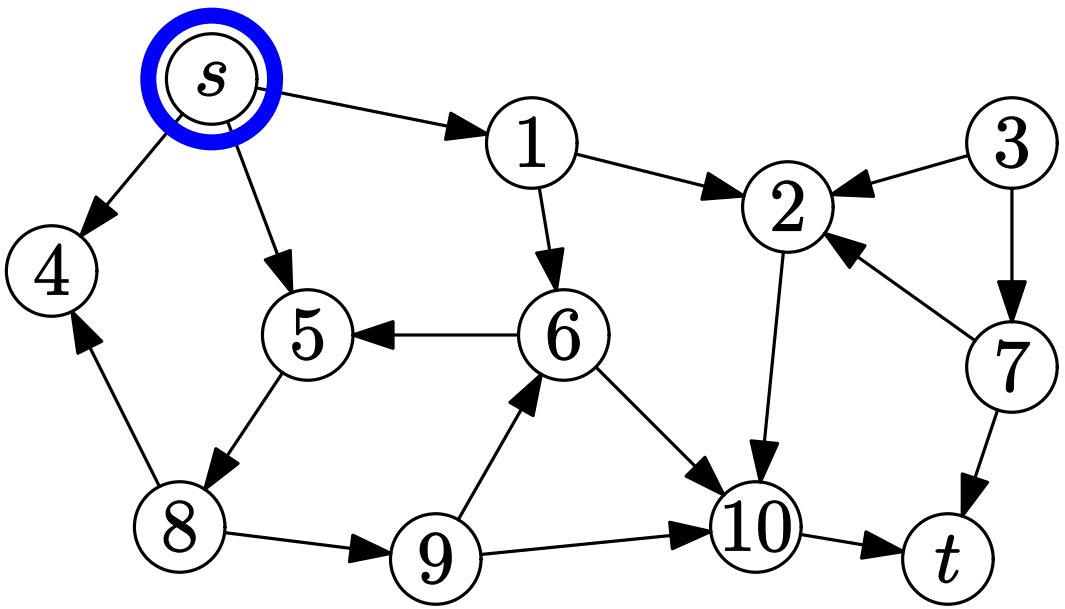
STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

$s$

スタック



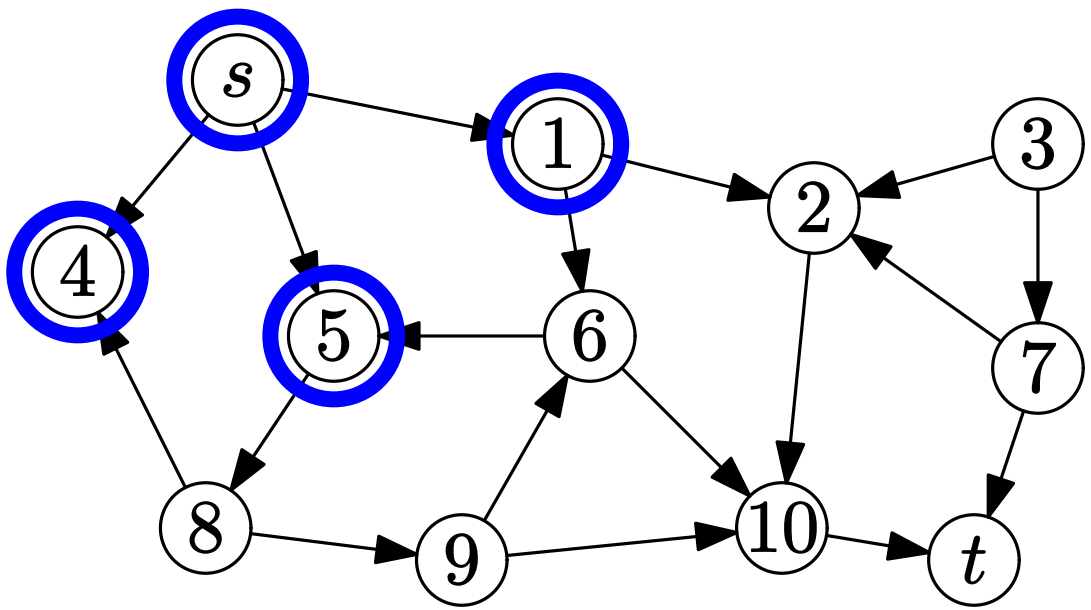
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

1
5
4
スタック



**性質**

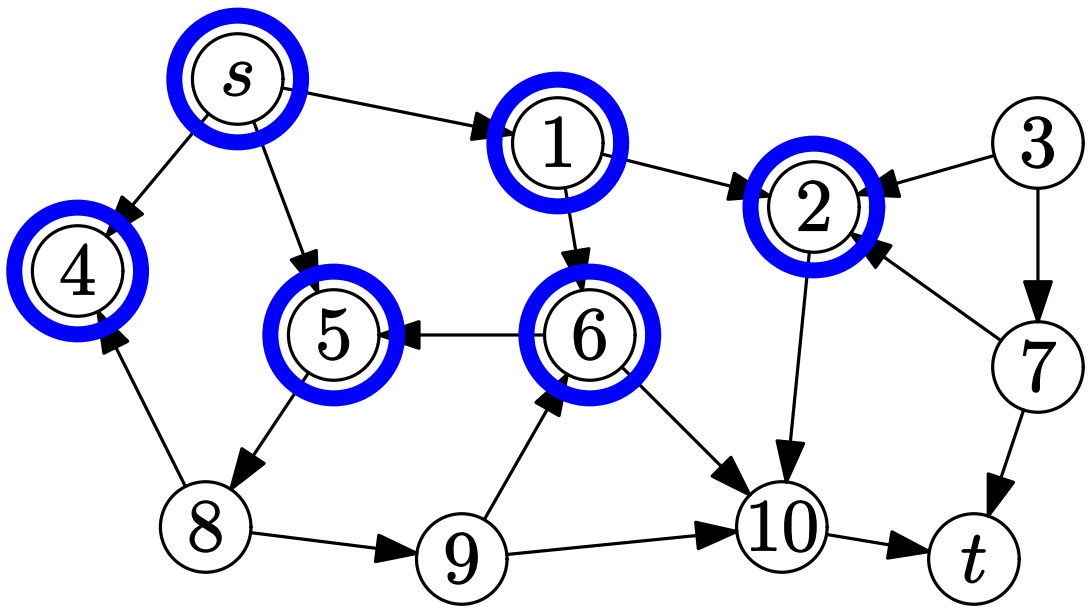
STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

2
6
5
4

スタック



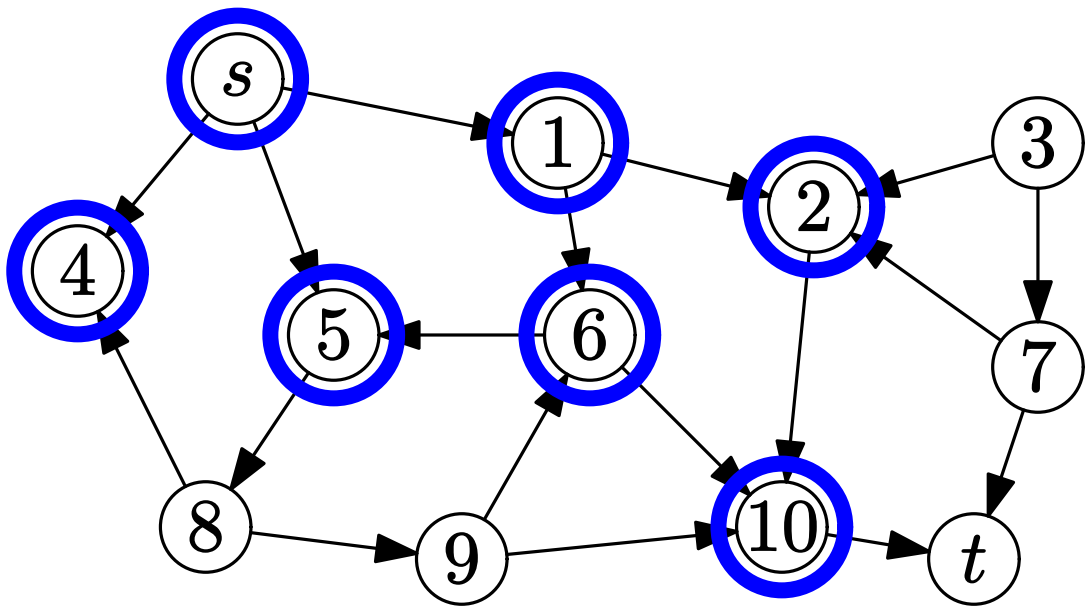
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

10
6
5
4
スタック



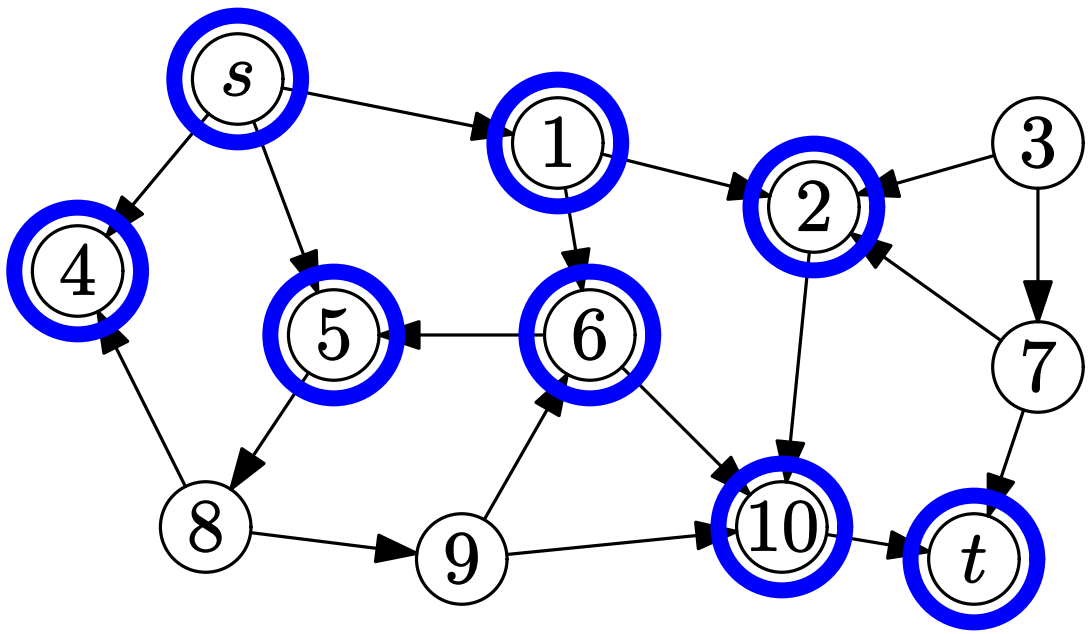
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

$t$
6
5
4
スタック



# STCON ∈ P

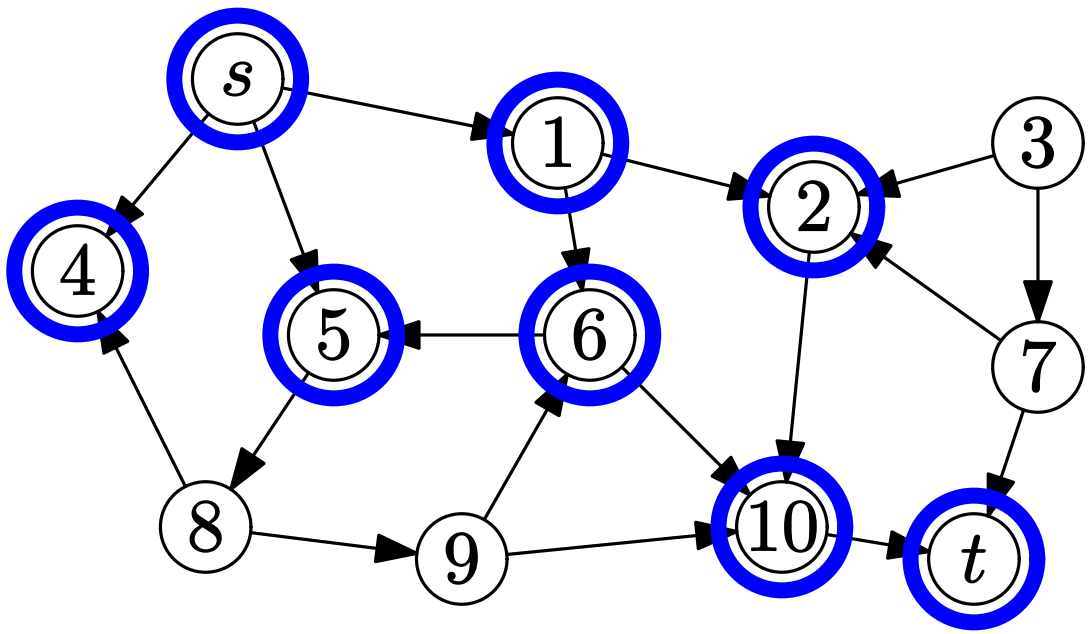
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

6
5
4
スタック



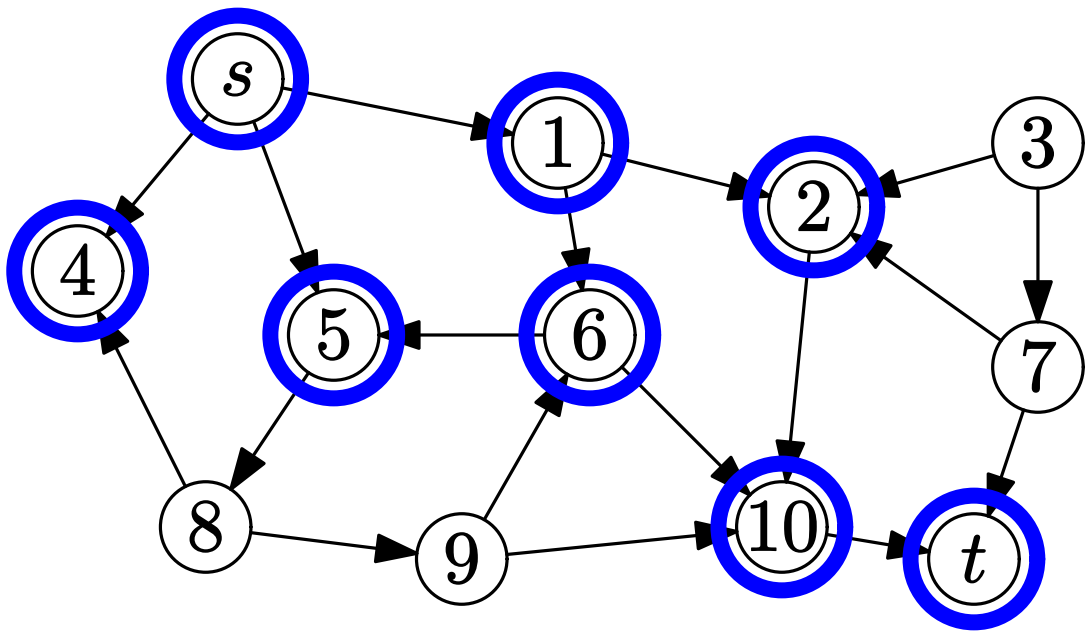
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

5
4
スタック



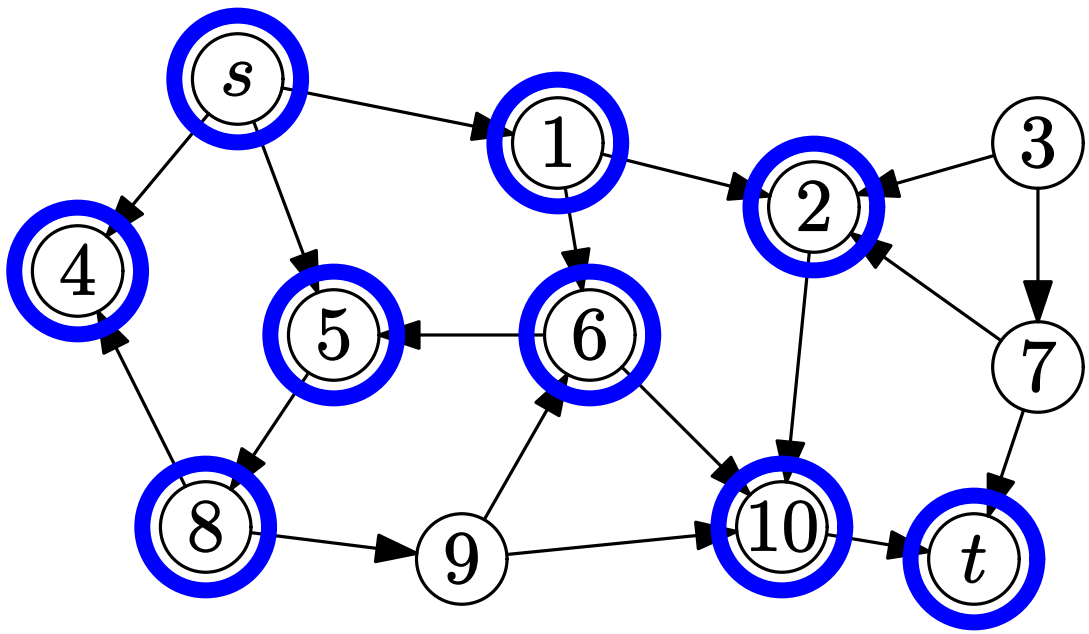
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

8
4
スタック



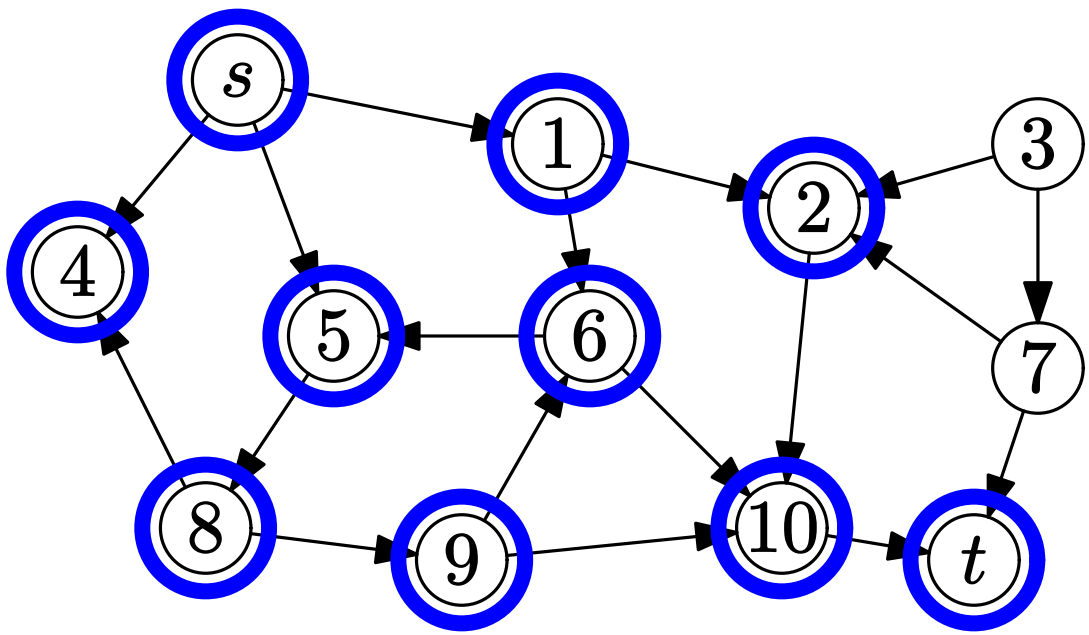
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

$$\begin{array}{c} 9 \\ 4 \\ \hline \text{スタック} \end{array}$$

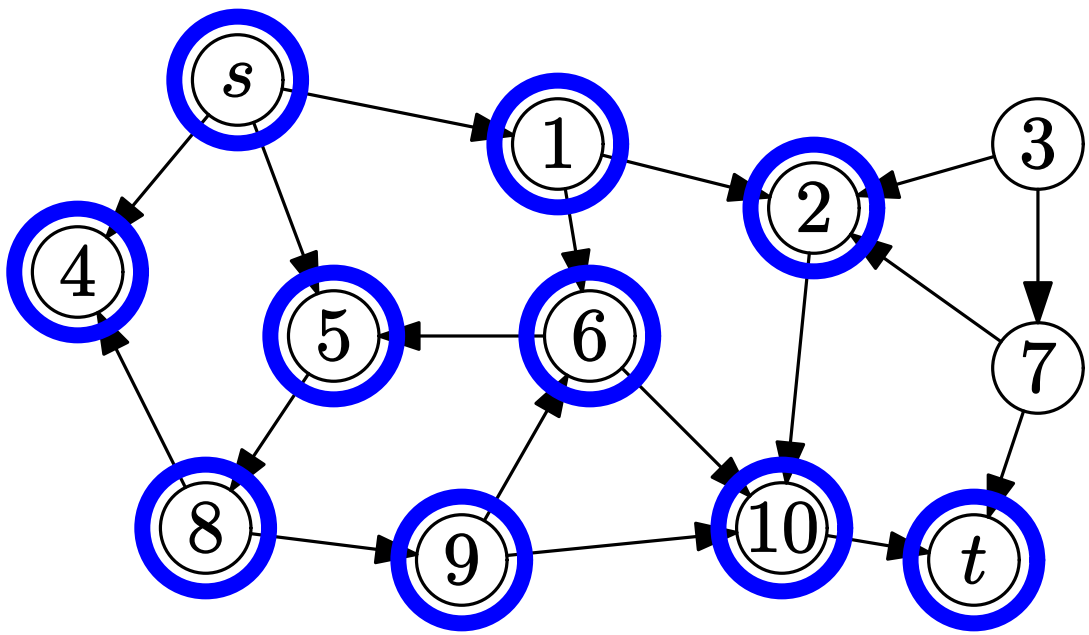


**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム




4

スタック

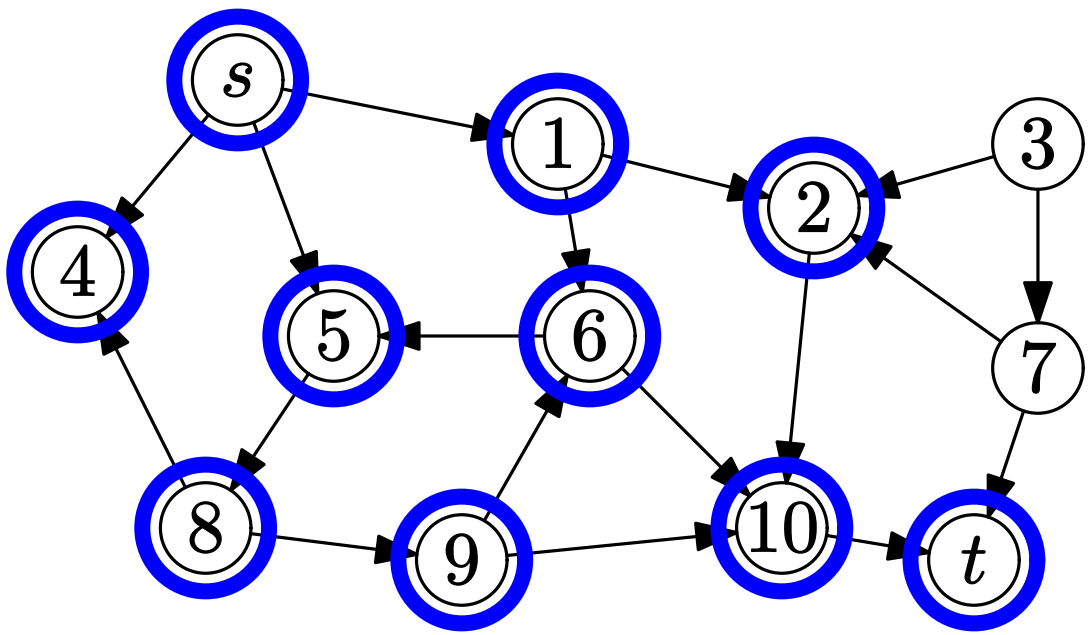
**性質**

STCON は多項式時間で解ける (つまり，STCON ∈ P)

証明：深さ優先探索を行えばよい □

$s$ から到達できる頂点をすべて求めるアルゴリズム

スタック

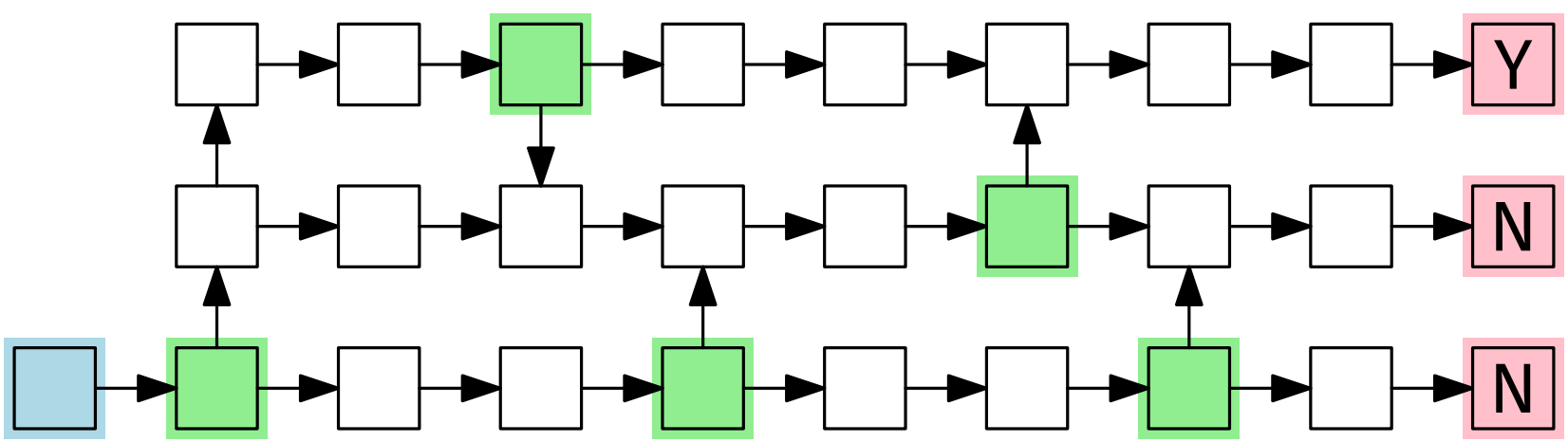


# NL ⊆ P：方針 1



$P \in$ NL として,
$A$ は $P$ を解く非決定性対数領域アルゴリズムとする

- 任意の入力 $I$ を考える
- $A(I)$ の実行における時点状況の総数
  $= O$(作業領域の取りうる状態の総数)
  $= 2^{O(\log |I|)} = O($|$I$|の多項式)



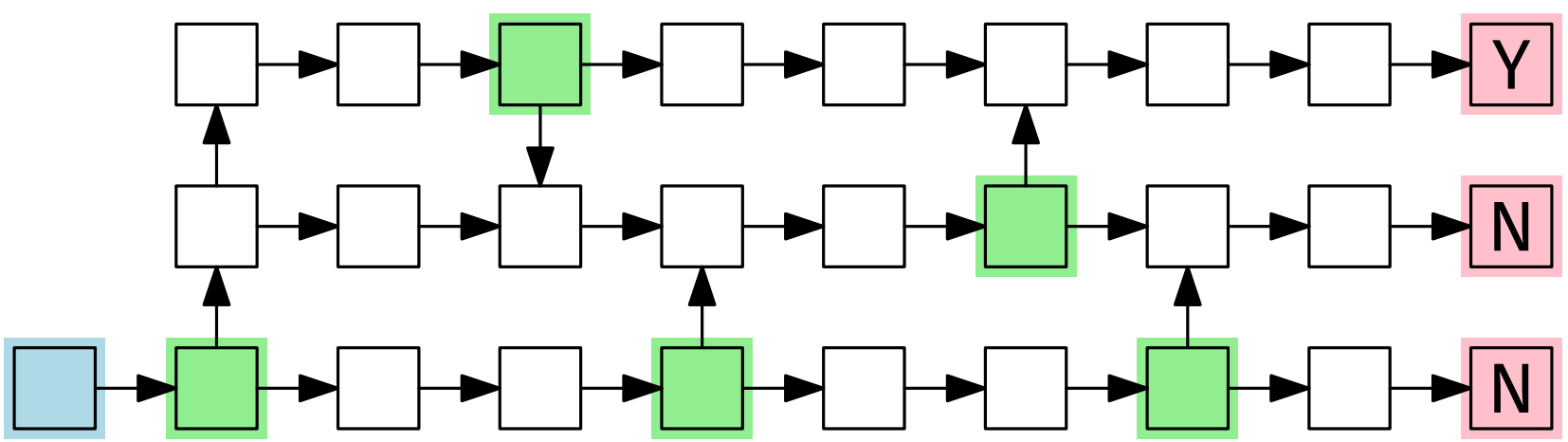
# NL ⊆ P：方針 1 (続)



- ∴ 次のアルゴリズムは $P$ を解く多項式時間アルゴリズム

  $O(\log |I|)$ 領域を用いる時点状況をすべて列挙し，
  グラフを作り，深さ優先探索を行う

- ∴ $P \in \mathsf{P}$ □



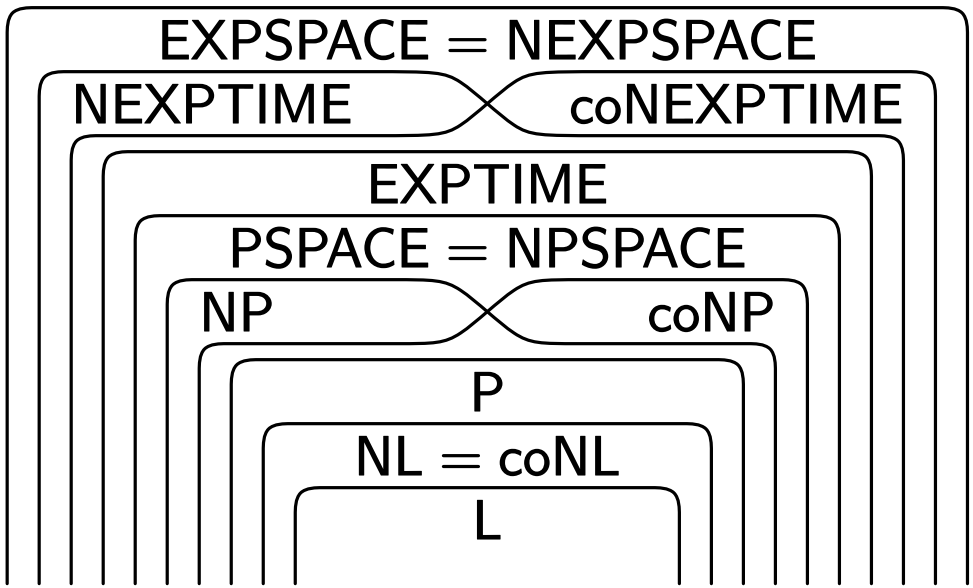
# NPSPACE $\subseteq$ EXPTIME



次も同様に証明できる

**性質**

NPSPACE $\subseteq$ EXPTIME




この図における「=」以外の証明ができるようになった

# 本日の内容



1. $\mathrm{P} \subseteq \mathrm{PSPACE}$
2. $\mathrm{NP} \subseteq \mathrm{PSPACE}$
3. $\mathrm{NL} \subseteq \mathrm{P}$
4. $\mathrm{P} = \mathrm{NP} \Rightarrow \mathrm{EXPTIME} = \mathrm{EXPSPACE}$

# 水増し論法の典型例



**性質：水増し論法の典型例**

$P = NP \quad \Rightarrow \quad EXPTIME = NEXPTIME$

注意：次はどちらも未解決

- $P = NP$
- $EXPTIME = NEXPTIME$

帰結

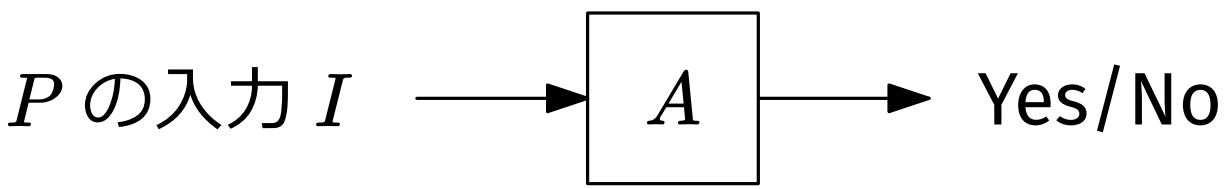
- $P = NP$ だと分かれば，すぐに $EXPTIME = NEXPTIME$ だと分かる
- 「$P = NP$ かつ $EXPTIME \neq NEXPTIME$」の可能性はない
- 「$P = NP$ の証明」は「$EXPTIME = NEXPTIME$ の証明」より簡単ではない

# 水増し論法による証明 (1)



証明：P = NP ⇒ NEXPTIME ⊆ EXPTIME を示せば十分

- $P \in$ NEXPTIME とする
- $A$ は $P$ を解く非決定性指数時間アルゴリズムだとする
- $A$ の時間計算量を $O(2^{|I|^k})$ とする ($k$ は定数)

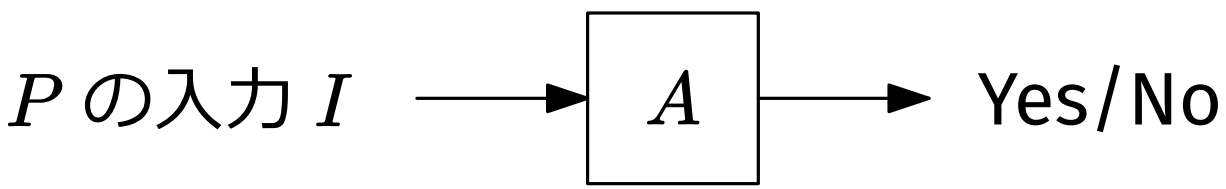


# 水増し論法による証明 (1)



証明：P = NP ⇒ NEXPTIME ⊆ EXPTIME を示せば十分

- $P \in$ NEXPTIME とする
- $A$ は $P$ を解く非決定性指数時間アルゴリズムだとする
- $A$ の時間計算量を $O(2^{|I|^k})$ とする ($k$ は定数)




**次の問題「水増し $P$」を考える**

**入力：** $P$ の入力 $I$，長さ $2^{|I|^k}$ のビット列 $b$
**出力：** $I$ が $P$ の Yes インスタンス ⇒ Yes
$I$ が $P$ の No インスタンス ⇒ No

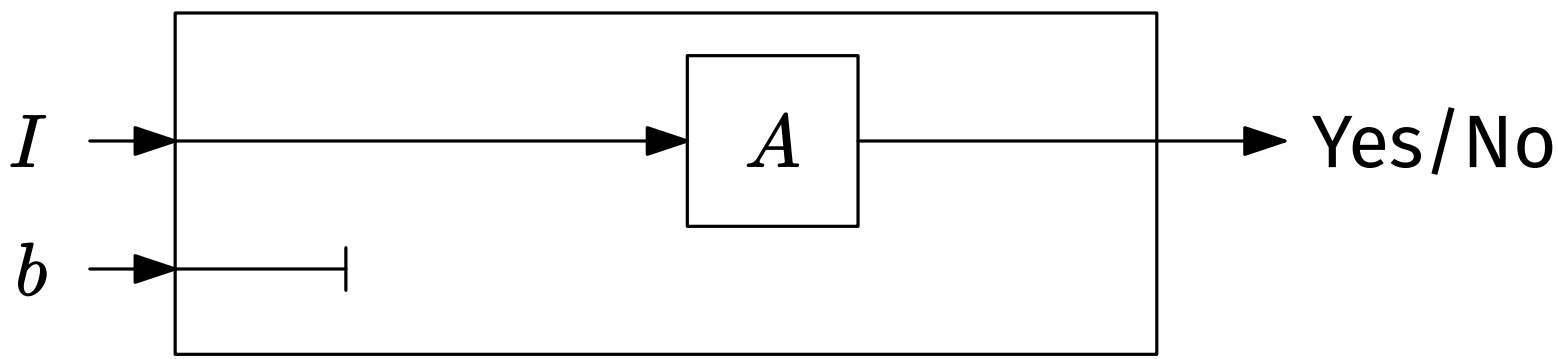
$P$ の水増し (padding) と呼ぶ

# 水増し論法による証明 (2)



- $A$ を使って，「水増し $P$」を解くアルゴリズム $A'$ が次のように作れる

1: $b$ を無視して，$I$ だけを取り出す
2: $A(I)$ を実行して，その出力をそのまま出力する



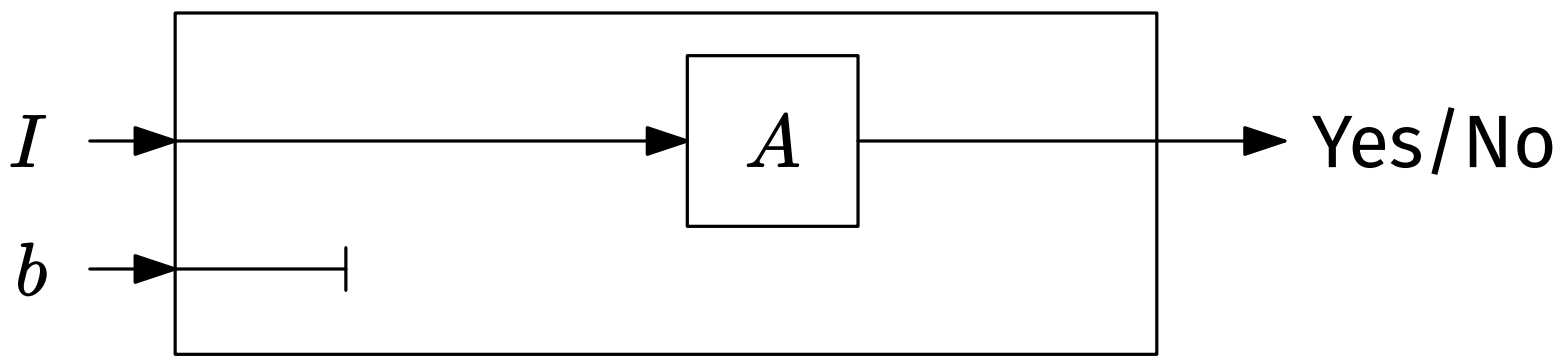

水増し $P$ を解くアルゴリズム $A'$

# 水増し論法による証明 (2)



- $A$ を使って，「水増し $P$」を解くアルゴリズム $A'$ が次のように作れる

1: $b$ を無視して，$I$ だけを取り出す
2: $A(I)$ を実行して，その出力をそのまま出力する




水増し $P$ を解くアルゴリズム $A'$

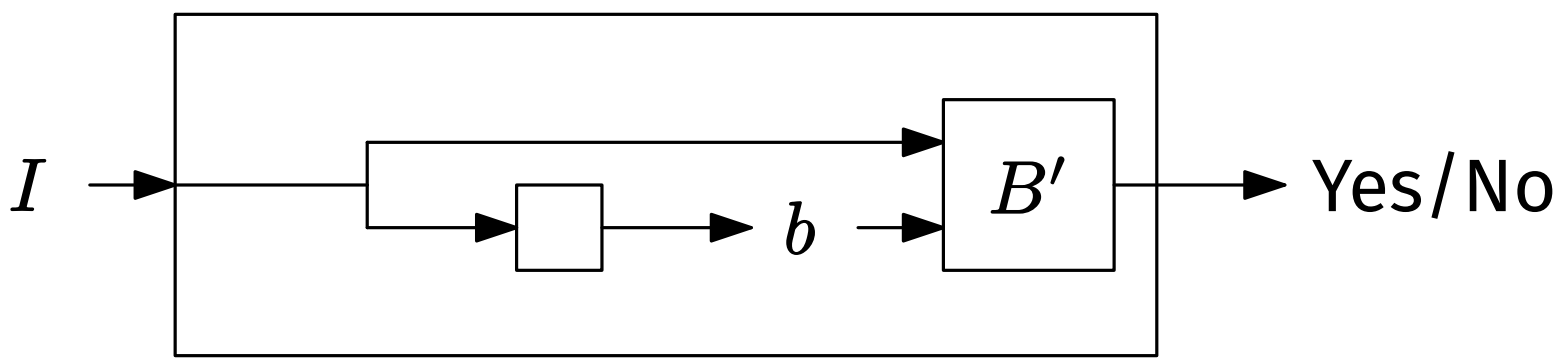
- 入力の符号長 $n = O(|I| + |b|) = O(|I| + 2^{|I|^k}) = O(2^{|I|^k})$
- $A$ の時間計算量 $= O(2^{|I|^k}) = O(n)$
- $\therefore A'$ は非決定性多項式時間アルゴリズム
- $\therefore$ 水増し $P \in$ NP

# 水増し論法による証明 (3)



- 仮定 P = NP より，水増し $P \in \mathrm{P}$
- $\therefore$ 水増し $P$ を解く多項式時間アルゴリズム $B'$ が存在
- $B'$ を使って，元の問題 $P$ を解くアルゴリズム $B$ を作る

1: 入力 $I$ から $2^{|I|^k}$ を計算する
2: 長さ $2^{|I|^k}$ のビット列 $b$ を作る
3: $B'(I, b)$ を実行し，その出力をそのまま出力とする



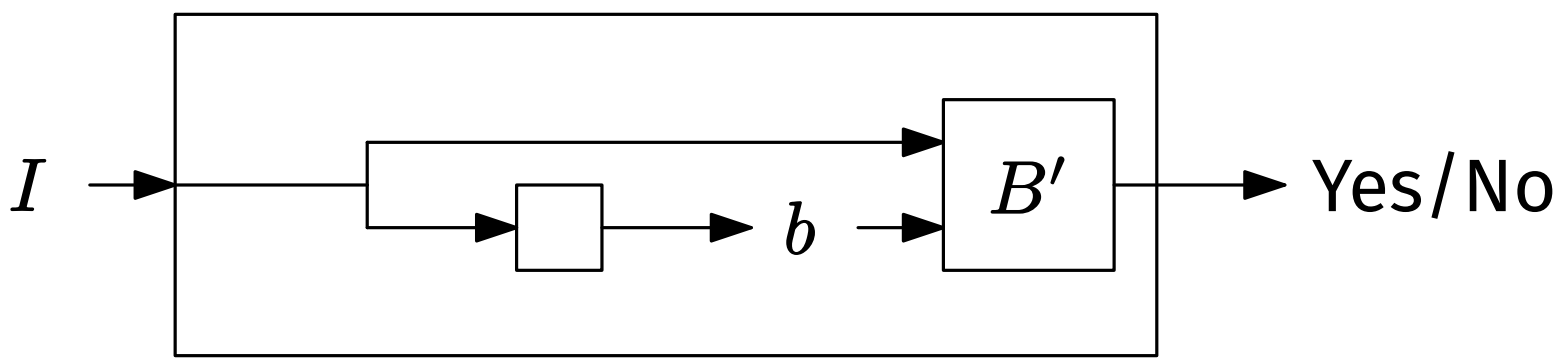

$P$ を解くアルゴリズム $B$

- 注：ステップ 1, 2 は $O(2^{|I|^k})$ 時間で実行可能

# 水増し論法による証明 (4)



- $B'$ の時間計算量 $= O(|I'|^{k'})$ とすると
  $B$ の時間計算量 $= O((2^{|I|^k})^{k'}) = O(2^{k'|I|^k})$
- $\therefore P \in$ EXPTIME
- $\therefore$ NEXPTIME $\subseteq$ EXPTIME $\square$




$P$ を解くアルゴリズム $B$

# 水増し論法



## 水増し論法 (padding argument) とは？

入力に水増しを行うことで，
小さな複雑性に関する命題を 大きな複雑性に関する命題
に変換する論法

いま紹介した例

- $\mathrm{P} = \mathrm{NP} \Rightarrow \mathrm{EXPTIME} = \mathrm{NEXPTIME}$

他の例

- $\mathrm{NP} = \mathrm{coNP} \Rightarrow \mathrm{NEXPTIME} = \mathrm{coNEXPTIME}$
- $\mathrm{PSPACE} = \mathrm{NPSPACE} \Rightarrow \mathrm{EXPSPACE} = \mathrm{NEXPSPACE}$

## 注意：次は未解決

$\mathrm{EXPTIME} = \mathrm{NEXPTIME} \Rightarrow \mathrm{P} = \mathrm{NP}$

# 本日のまとめ



## 目標

次の証明を通して，計算複雑性に関する基本的な論法が使えるようになる

- $P \subseteq PSPACE$
  - $EXPTIME \subseteq EXPSPACE$，$NP \subseteq NPSPACE$，...
- $NP \subseteq PSPACE$
  - $NEXPTIME \subseteq EXPSPACE$
- $NL \subseteq P$
  - $NPSPACE \subseteq EXPTIME$
- $P = NP \Rightarrow EXPTIME = NEXPTIME$
  証明法：水増し論法

# 次回の予告



## 次回

次を証明する

- P $\neq$ EXPTIME
  - 証明法：対角線論法

関連する「階層定理」を紹介する

# クイズ



Q

1.

2.

3.

4.