第二日

計算量 (完全問題と帰着)

令和8年5月20日

定義 (再)

機械 M が**多項式時間**であるとは　或る多項式 $p\colon \mathbf{N} \to \mathbf{N}$ が存在し
任意の長さ n の任意の入力に対する M の計算が
時間 $p(n)$ 以内に終る (遷移 $p(n)$ 回以下で停止する) ことをいう

言語 A を認識する多項式時間の機械 M が存在するとき
A は**多項式時間認識可能**である ($A \in \mathbf{P}$) という

チャーチとチューリングの定立 (続)

「現実的な手間で計算できる」 ＝ (チューリング機械で) 多項式時間認識可能

今後は一々チューリング機械を考えず　計算手順が解り易いように説明する

時間＝「基本的な操作の回数」と大雑把に考えてよい
- 四則演算やビットの操作など
- 機械語での命令数

定義（再）

機械 M が**多項式空間**であるとは　或る多項式 $p\colon \mathbf{N} \to \mathbf{N}$ が存在し
任意の長さ n の任意の入力に対する M の計算が　停止し
空間 $p(n)$ 以内である（テープ上で初めの欄の左右 $p(n)$ 個の範囲に留まる）ことをいう
言語 A を認識する多項式空間の機械 M が存在するとき
A は**多項式空間認識可能**である（$A \in \mathbf{PSPACE}$）という

定理（再）

$\mathbf{P} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$

定理（再）

PRIME, HAMILTON, SAT $\in \mathbf{PSPACE}$

しかし「指数個の可能性を調べる」と言っても
$\overline{\text{PRIME}}$ や HAMILTON や SAT は　調べた指数個の候補のうち
「適するものが存在するか」を問うという特殊な形の要求である

定義

言語 A が **NP** に属するとは
或る多項式時間機械 M と多項式 $p\colon \mathbf{N} \to \mathbf{N}$ が存在し
任意の入力 x に対し
■ もし $x \in A$ ならば　長さ $p(|x|)$ の文字列 r が存在して $M(x;r)$ は受理
■ もし $x \notin A$ ならば　長さ $p(|x|)$ の任意の文字列 r で $M(x;r)$ は不受理

明らかに $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$

定理

$\overline{\text{PRIME}}$, HAMILTON, SAT $\in \mathbf{NP}$

問題 SR

入力　書換え規則の集合 R と文字列 w

答　R による書換えを次々と w に施して ε にできるか

問題 $SR_{\geq}$

入力　書換え規則の集合 R と文字列 w
但し R の各規則 $u \to v$ において (u の長さ) $\geq$ (v の長さ)

答　R による書換えを次々と w に施して ε にできるか

問題 $SR_{>}$

入力　書換え規則の集合 R と文字列 w
但し R の各規則 $u \to v$ において (u の長さ) $>$ (v の長さ)

答　R による書換えを次々と w に施して ε にできるか

問題 $SR^{1}_{>}$

入力　書換え規則の集合 R と文字列 w
但し R の各規則 $u \to v$ において (u の長さ) $>$ (v の長さ)

答　R による書換えを次々と w に施すと
可能な書換え方は毎回一通りしかなく　やがて ε に達する

困難そう（一般的な入力）

容易そう（特殊な入力）

問題 SR

入力 (R, w)

答 $w \Rightarrow_R^* \varepsilon$ か

定理（昨日）

SR $\in$ **CE**（だが $\overline{\text{SR}} \notin$ **CE**）

問題 $\text{SR}_{\geq}$

SR を各規則が
(旧長さ) ≥ (新長さ)
の場合に制限したもの

? （次頁）

問題 $\text{SR}_{>}$

更に各規則が
(旧長さ) > (新長さ)
の場合に制限したもの

定理

$\text{SR}_{>} \in$ **NP**

書換えの列（出発文字列の長さ以内と保証されている）の存在を問う問題だから

問題 $\text{SR}^1_{>}$

$\text{SR}_{>}$を更に
可能な書換え方が毎回一通り
な場合に制限したもの

定理

$\text{SR}^1_{>} \in$ **P**

書換えを実際に順次（機械のテープ上で）行ってみればよい

定理

$\mathrm{SR}_{\geq} \in \mathbf{PSPACE}$

※ $\Sigma = \{a, b\}$ の場合を考える（他のときも同様）

問題 $\mathrm{SR}_{\geq}$

入力 (R, w) 但し各規則が（旧長さ）≥（新長さ）

問 $w \Rightarrow_R^* \varepsilon$ か

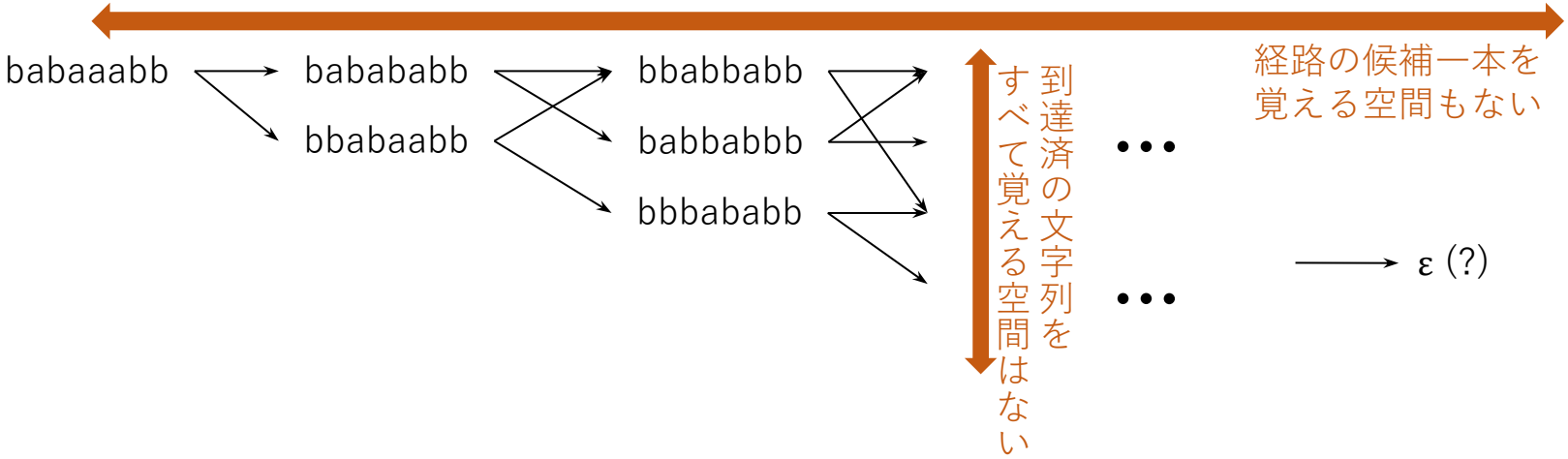
入力の長さが n である（w の長さ $< n$）とき

w から書換えにより生じ得る文字列も長さ $< n$ であり　その個数は $< 2^n$

したがって $w \Rightarrow_R^{\leq 2^n} \varepsilon$ かどうか調べればよい

書換え 2^n 回以内で w から ε が得られる という意味

しかし　単純に長さ $\leq 2^n$ の経路すべてを調べることはできない



定理

$\mathrm{SR}_{\geq} \in \mathbf{PSPACE}$

問題 $\mathrm{SR}_{\geq}$

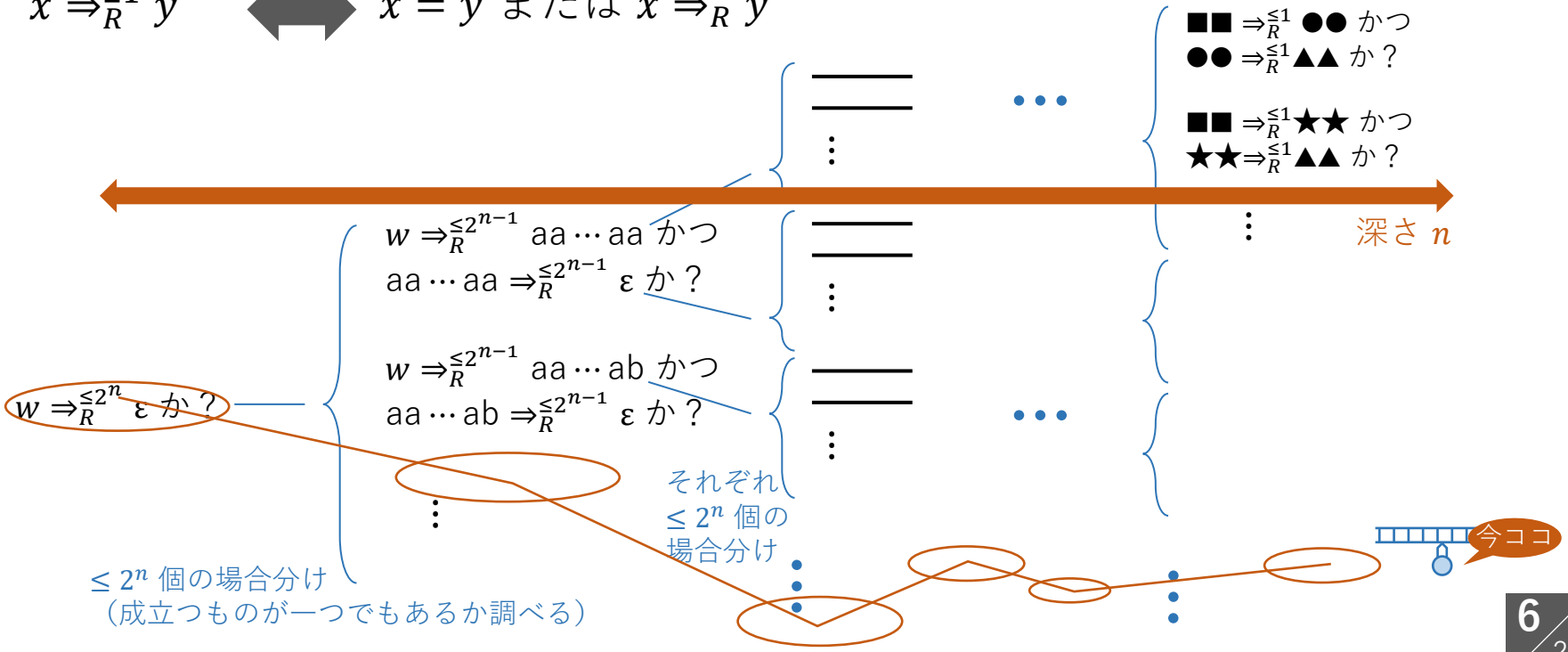
入力 (R, w) 但し各規則が（旧長さ）≥（新長さ）

問 $w \Rightarrow_R^* \varepsilon$ か

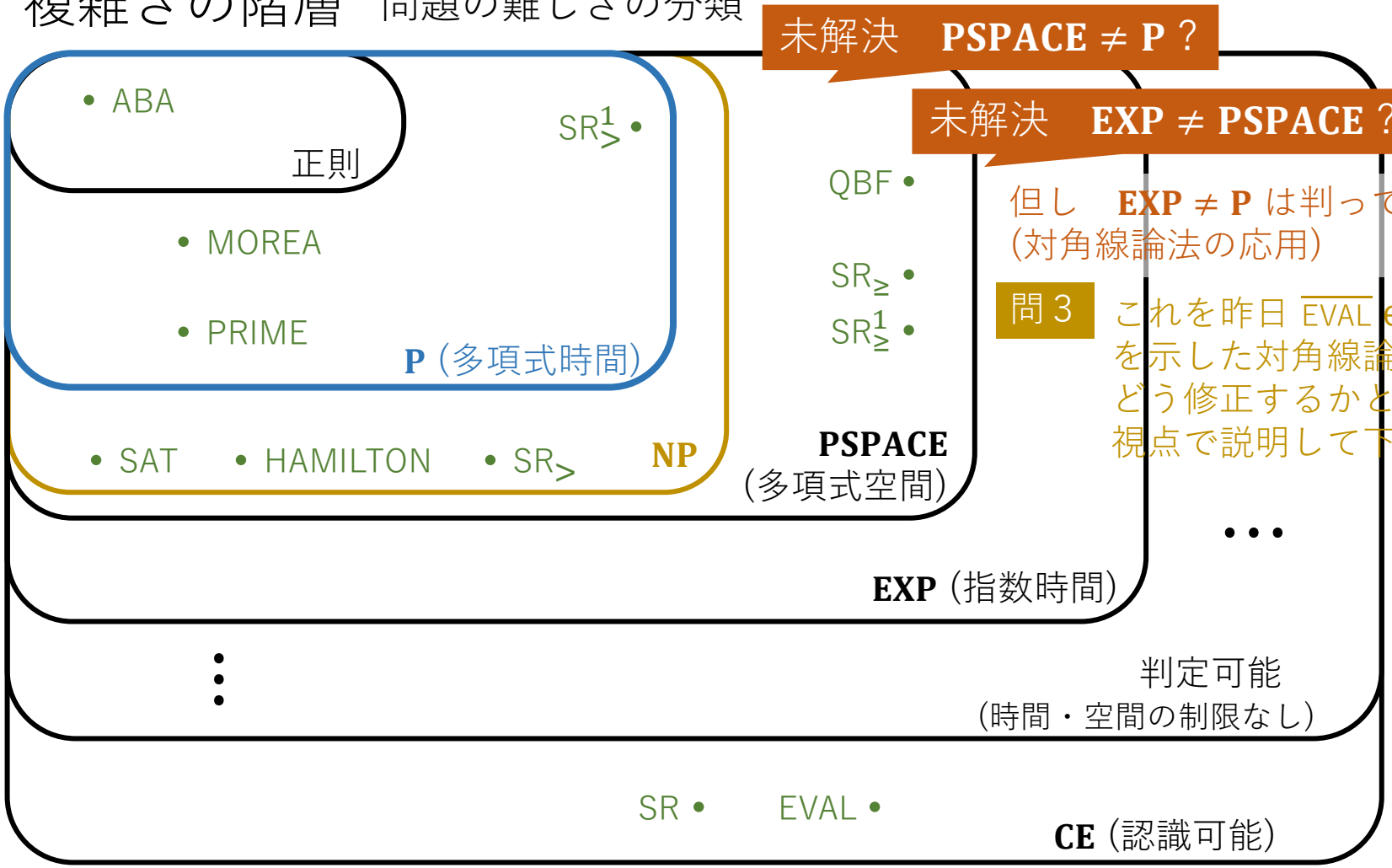
$w \Rightarrow_R^{\leq 2^n} \varepsilon$ かどうか　次の関係を再帰的に用いて調べればよい

$x \Rightarrow_R^{\leq 2^{i+1}} y$ ⟺ 或る $z \in \Sigma^{<n}$ が存在して $x \Rightarrow_R^{\leq 2^i} z$ かつ $z \Rightarrow_R^{\leq 2^i} y$

$x \Rightarrow_R^{\leq 1} y$ ⟺ $x = y$ または $x \Rightarrow_R y$



複雑さの階層　問題の難しさの分類



但し　EXP ≠ P は判っている
(対角線論法の応用)

問 3　これを昨日 $\overline{\text{EVAL}} \in$ CE を示した対角線論法をどう修正するかという視点で説明して下さい

乱択

乱数を利用した算法

例 与えられた（多変数の）整数係数多項式が零か判定

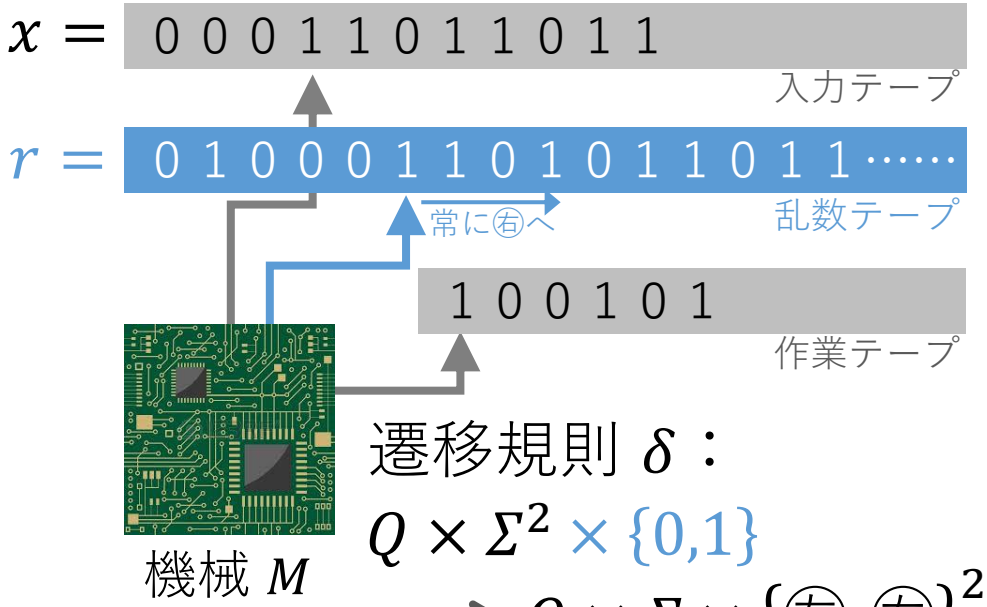
$$
\begin{aligned}
&(z-x)(x+y)(yy+zz)-xxyz\\
&+xy(z+x)(y-z)\\
&-(x-z)(xx+xy-yz)(x+y-z)\\
&+xyyz+(x+y)y(z-x+y)(x+y-z)\\
&-(x-z)(xy+xz+yz)(x+y-z)\\
&+(x-y)(x+z)(y+z)(y-z)+yyzz
\end{aligned}
$$

文字式のまま全部展開して計算 ➡ 大変（多項式時間でない）

適当に数値を代入して計算し
0 になるか調べる
$(x,y,z)=(1,2,3)$ とか

➡ ラク（高速・単純）
高確率で正解

非決定性（＝乱択）機械



遷移規則 δ：

$Q \times \Sigma^2 \times \{0,1\}$

$\longrightarrow Q \times \Sigma \times \{㊧, ㊨\}^2$

計算結果

$M(x; r)$ 受理か不受理

入力　乱数　に依存

次の遷移を二つの分岐から
非決定的に（等確率で）選ぶ

＝

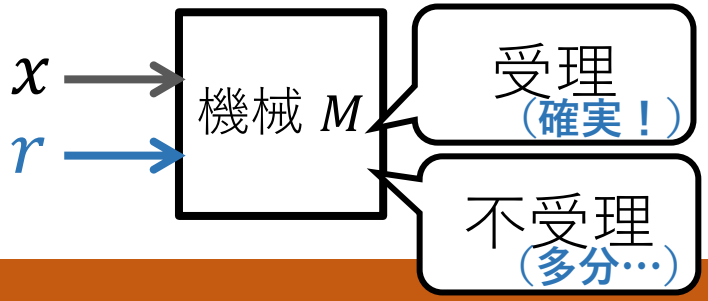
初めに「乱数テープ」上に
乱数列が ~~無限に~~ 供給される

$p(|x|)$ ビットで十分

時間量が $p: \mathbf{N} \to \mathbf{N}$ とは
任意の x と**任意の r について**
$p(|x|)$ 時間以内で停止すること

nondeterministic の N

NP の定義の言い換え



定義

言語 A が級 **NP** に属するとは
或る多項式時間 (非決定性) 機械 M が存在し
任意の入力 x に対し

$x \in A$ のとき $M(x;r)$ は**或る r で**受理

$x \notin A$ のとき $M(x;r)$ は**すべての r で**不受理

注意

NP (や次頁の RP) の定義について

理くつ上はこの動詞の方を
(「ヨ 認識する」とかに？) 変えるべきだが
「NP は非決定性多項式時間で解ける問題」
のように言ってしまうことも多い

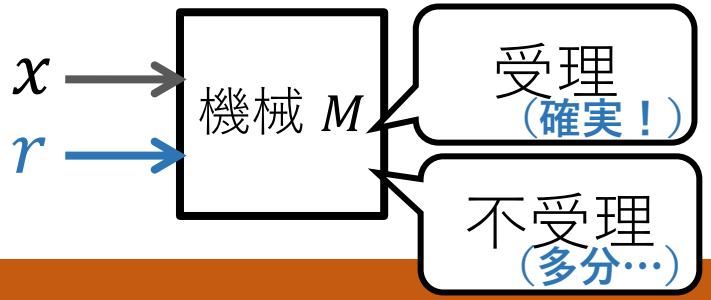
定義	定義	
機械 M が**多項式時間**であるとは…	機械 M が言語 A を**認識**するとは…	**P** 多項式時間限定な機械によって認識される言語の全体
非決定性機械 M が**多項式時間**であるとは…	機械 M が言語 A を●●するとは…	**NP** 多項式時間限定な機械によって●●される言語の全体
非決定性機械 M が**多項式時間**であるとは…	機械 M が言語 A を▲▲するとは…	**RP** 多項式時間限定な機械によって▲▲される言語の全体

↑ **こっちは共通**
(**PSPACE** や **EXP** の定義では
これを変える)

↑ **ココを変えた！**

randomized の R

RP



片側誤り
誤受理なし

定義

言語 A が級 **RP** に属するとは
或る多項式時間 (非決定性) 機械 M が存在し
任意の入力 x に対し

$x \in A$ のとき $M(x;r)$ は**確率** $> \frac{1}{2}$ **で**受理

$x \notin A$ のとき $M(x;r)$ は**必ず**不受理

明らかに
$\mathbf{P \subseteq RP \subseteq NP}$

$\frac{1}{2}$ の代りに $\frac{99}{100}$ にしたければ…
乱数を独立に 7 回取って $r_1, \cdots, r_7$ とし
$M(x,r_1), \cdots, M(x,r_7)$ どれかが受理したら受理

先程の例

問題 与えられた整数係数多項式 $p(X_1, \dots, X_m)$ が**非零**であるか判定せよ

この問題が **P** に属するかは未解決だが
次の算法により **RP** には属する（多項式**零**判定問題が **coRP** に属する）

d は p の全次数

算法 数 $r_1, \dots, r_m \in \{1, \dots, 2d\}$ を一様独立に乱択し
$p(r_1, \dots, r_m) \neq 0$ ならば受理

➡ p が零なら確実に不受理
非零なら次の補題により確率 $\geq \frac{1}{2}$ で受理

補題

全次数 d の非零多項式 p について

$$\Pr[\,p(r_1, r_2, \dots, r_m) \neq 0\,] \geq 1 - \frac{d}{B}$$

但し Pr は
$r_1, r_2, \dots, r_m \in \{1, \dots, B\}$ を
一様独立に選ぶときの確率

証明

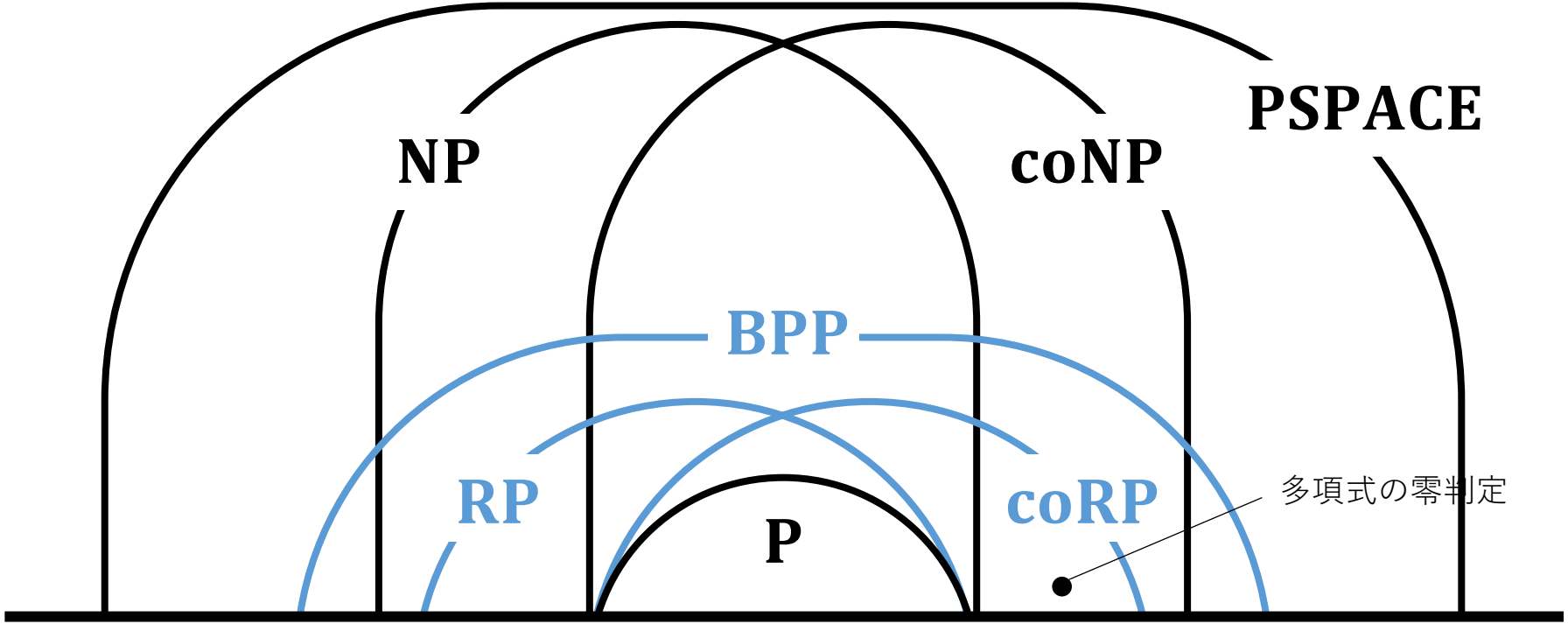
m に関する帰納法　$m = 0$ のとき自明　$m > 0$ とする
X_1 の最高次数を i として　次のように書く

$$p(X_1, X_2, \dots, X_m) = X_1^i \cdot q(X_2, \dots, X_m) + (X_1\text{の低次の項})$$

帰納法の仮定より $\Pr[\,q(r_2, \dots, r_m) \neq 0\,] \geq 1 - \frac{d-i}{B}$

もし□ならば $p(X_1, r_2, \dots, r_m)$ は X_1 に関する i 次の非零な多項式で
その根は高々 i 個　故に

$$\Pr[\square] \geq \Pr[\square] \cdot \left(1 - \frac{i}{B}\right) \geq \left(1 - \frac{d-i}{B}\right)\left(1 - \frac{i}{B}\right) \geq 1 - \frac{d}{B}$$



「含まれる」の関係を図示したが
等しいか否かは証明されていない
（図中の級がすべて等しいかもしれない）

未解決 $\mathbf{BPP} \stackrel{?}{=} \mathbf{P}$

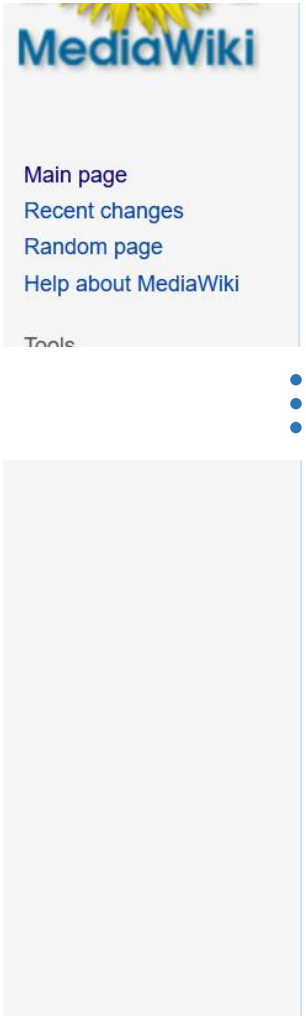
（等しいと予想する人が多い）

様々な計算量級（クラス）

本講義で扱った
- 正則（REG）
- 多項式時間認識可能（P）
- 多項式空間認識可能（PSPACE）
- 判定可能（R または Decidable）
- 認識可能（RE または CE）

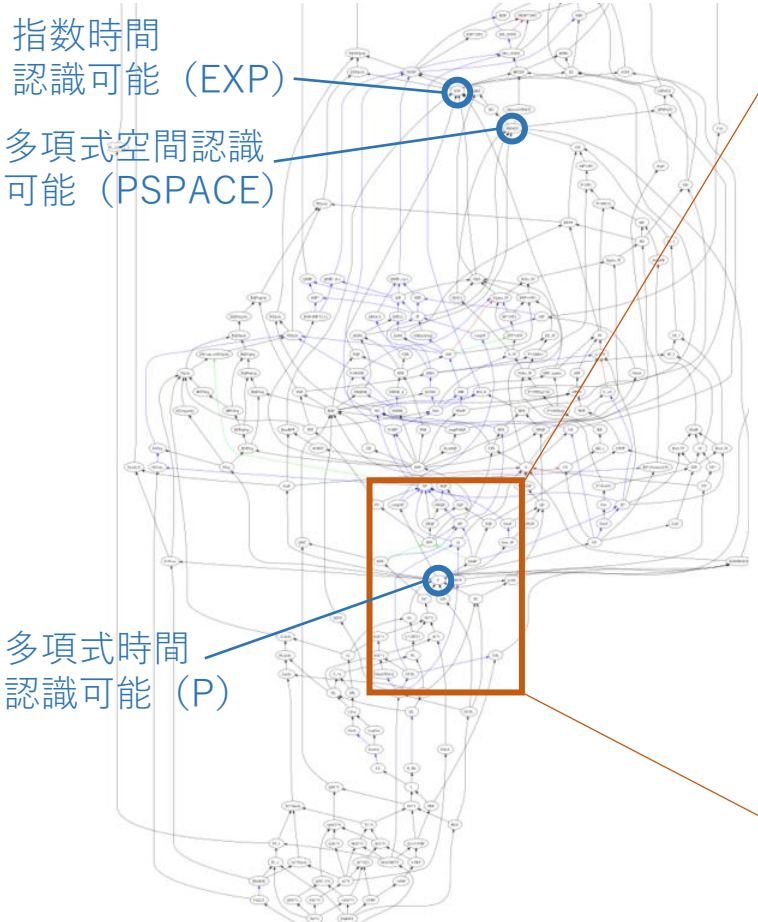
以外にも問題の複雑さの度合を色々な側面から捉えた基準が数多く定義され研究されている

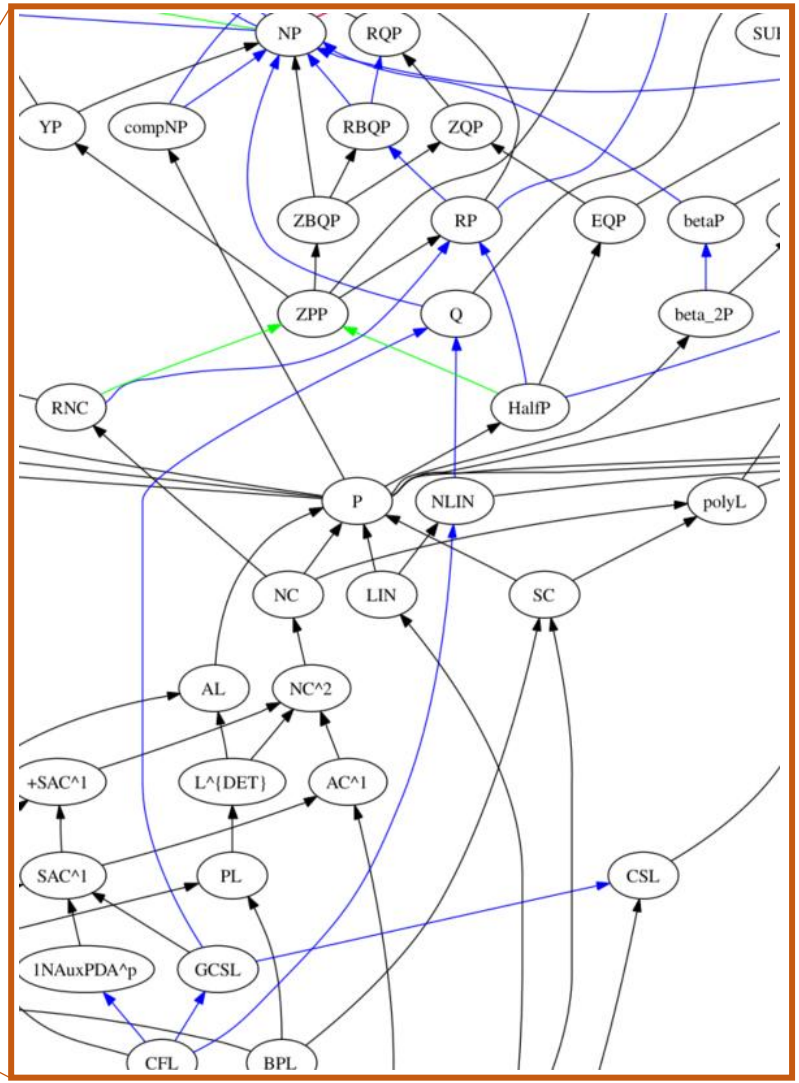
ウェブサイト「Complexity Zoo」 500以上の計算量級を紹介
https://complexityzoo.net/Complexity_Zoo



様々な計算量クラス

「Complexity Zoology」
https://www.math.ucdavis.edu/~greg/zoology/





定義

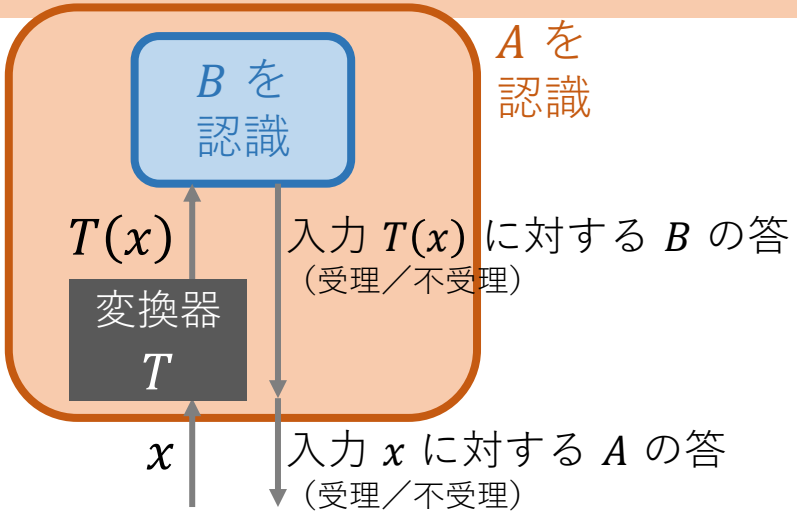
言語 A が言語 B に**多項式時間帰着**するとは
多項式時間機械 T が存在して
任意の入力 x に対する T の出力 $T(x)$ が
$x \in A \Leftrightarrow T(x) \in B$ を満すことをいう

停止したときの
テープ上の文字列

$A \leq^{\mathrm{P}} B$ と書く

向きに注意

「A の難しさは B 以下」という
気持なのでこの不等号で表す



「B を使うと A が計算できる」

「$B \in \mathbf{P}$ なら $A \in \mathbf{P}$」を成立たすには
必ずしも問題 A の入力 x を問題 B の入力 $T(x)$ に対応させるという上記の形でなくてもよい
（例えば $x \in A$ かどうか知るために幾つかの文字列 $y_1, y_2, \cdots$ が B に属するか否かを使うとか）
つまり上記の定義は「帰着」のうち特殊な形のものである
しかし以下の話にはこの形の帰着で十分なので　以下ではこれを単に「帰着」という

$A \leq^{\mathrm{P}} B$ であるとすると…

もし $B \in \mathbf{P}$ ならば $A \in \mathbf{P}$

もし $B \in \mathbf{PSPACE}$ ならば $A \in \mathbf{PSPACE}$

もし B が判定可能ならば A も判定可能

問題どうしの**相対的な困難さの比較**ができる

問題 EVAL	入力	(M, x)
	答	M は x を受理するか

問題 SR	入力	(R, w)
	答	$w \Rightarrow_R^* \varepsilon$ か

昨日は EVAL $\leq^{\mathrm{P}}$ SR を示すことで
EVAL の判定不能性から SR の判定不能性を導いた
（その時点では帰着が多項式時間どうかは気にしていなかったが）

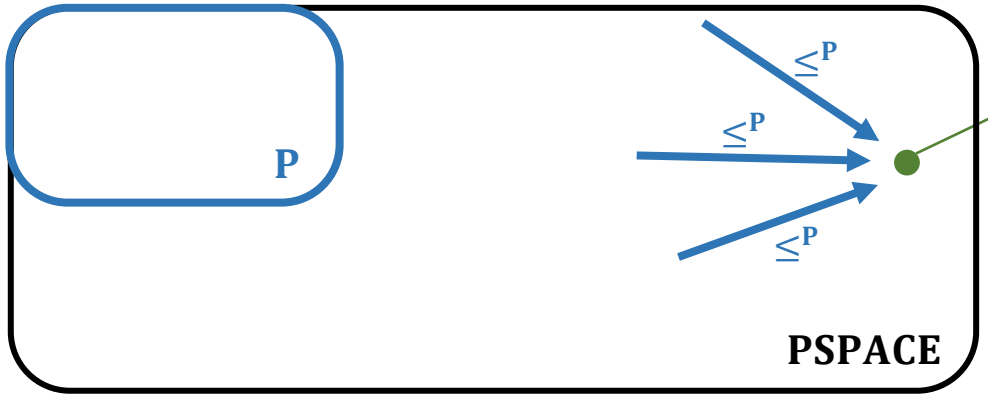
定義

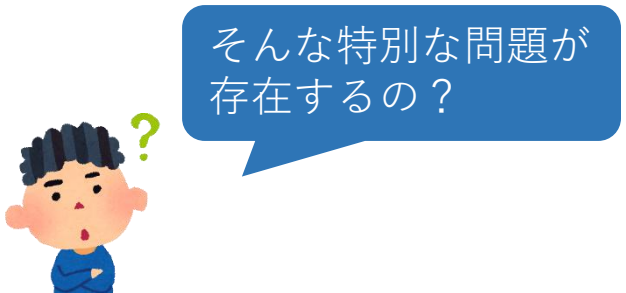
言語 $B \in \mathbf{PSPACE}$ が **PSPACE 完全**であるとは
任意の言語 $A \in \mathbf{PSPACE}$ が B に
多項式時間帰着する $(A \leq^{\mathrm{P}} B)$ ことをいう

「B は **PSPACE** な問題のうち最難」

「**PSPACE** $=$ **P** でない限り B は **P** に属しない」

「**PSPACE** とは　複雑さが問題 B 以下であること」





問題 EVAL	入力	文字列の組 (M, x)
	答	機械 M は入力 x を受理するか
問題 $\mathrm{EVAL}_{\mathbf{space}}$	入力	文字列の組 $(M, x, 0^s)$
	答	機械 M は入力 x を空間 s 以下で受理するか

定理

$\mathrm{EVAL}_{\mathbf{space}}$ は **PSPACE** 完全

∵ 言語 $A \in \mathbf{PSPACE}$ を考える
A を認識する多項式空間の機械 M が存在
多項式 p が存在し
任意の長さ n の入力 x に対して M は空間 $\leq p(n)$ で停止する
そこで $T(x) = \left(M, x, 0^{p(n)}\right)$ とすると T は A から $\mathrm{EVAL}_{\mathbf{space}}$ への帰着

問題 EVAL

入力 文字列の組 (M, x)

答 機械 M は入力 x を受理するか

問題 $\mathrm{EVAL}_{\mathrm{space}}$

入力 文字列の組 $(M, x, 0^s)$

答 機械 M は入力 x を空間 s 以下で受理するか

定理

$\mathrm{EVAL}_{\mathrm{space}}$ は **PSPACE** 完全

問題 $\mathrm{EVAL}_{\mathrm{ntime}}$

入力 文字列の組 $(M, x, 0^t)$

答 或る文字列 r が存在して
機械 M は入力 (x, r) を時間 t 以下で受理するか

定理

$\mathrm{EVAL}_{\mathrm{ntime}}$ は **NP** 完全

∵ 同様

定理

$\mathrm{SR}_{\geq}$ は **PSPACE** 完全　　（同じ証明により $\mathrm{SR}^{1}_{\geq}$ も）

昨日の $\mathrm{EVAL} \leq^{\mathrm{P}} \mathrm{SR}'' \leq^{\mathrm{P}} \mathrm{SR}$ と同様に
$\mathrm{EVAL}_{\mathbf{space}} \leq^{\mathrm{P}} \mathrm{SR}''_{\geq} \leq^{\mathrm{P}} \mathrm{SR}_{\geq}$ を示す

問題 $\mathrm{SR}_{\geq}$
入力 (R, w) 但し各規則が（旧長さ）≥（新長さ）
問 $w \Rightarrow_R^* \varepsilon$ か

帰着（略）

問題 $\mathrm{SR}''_{\geq}$
入力 (R, w, w'') 但し各規則が（旧長さ）≥（新長さ）
問 $w \Rightarrow_R^* uw''v$ なる文字列 u, v は存在するか

問題 $\mathrm{EVAL}_{\mathbf{space}}$ → 帰着

$(M, x, 0^s)$ ⋯⋯▶ $(R, \sqcup^s q_{始} x \sqcup^s, 受)$

R は　M の各状態 $q \in Q$ と文字 $\sigma \in \Gamma$ について次の規則を加えて作る

- もし $\delta(q, \sigma) = (q', \sigma', 左)$ ならば　各 $\tau \in \Gamma$ に対し規則 $\tau q \sigma \to q' \tau \sigma'$
- もし $\delta(q, \sigma) = (q', \sigma', 右)$ ならば　規則 $q\sigma \to \sigma' q'$
- もし $\delta(q, \sigma) =$ 止 かつ $q \in Q_{受理}$ ならば　規則 $q\sigma \to 受$

（$\mathrm{EVAL} \leq^{\mathrm{P}} \mathrm{SR}''$ のときに設けた「空白文字を端に付け足す規則」 ▸ → ▸␣ および ◂ → ␣◂ は設けない）

すると

M は x を空間 $\leq s$ で受理	⇔	R の規則に従って $\sqcup^s q_{始} x \sqcup^s$ から 受 を含む文字列に到達できる

が成立

定理

$SR_>$ は **NP** 完全

定理

SAT は **NP** 完全

定理

HAMILTON は **NP** 完全

∵　頑張ると多項式時間帰着が作れる

問題
QBF

与えられた量化命題論理式

$$Q_n X_n . Q_{n-1} X_{n-1} \cdots Q_1 X_1 . \varphi(X_1, \ldots, X_n)$$

の真偽を判定せよ

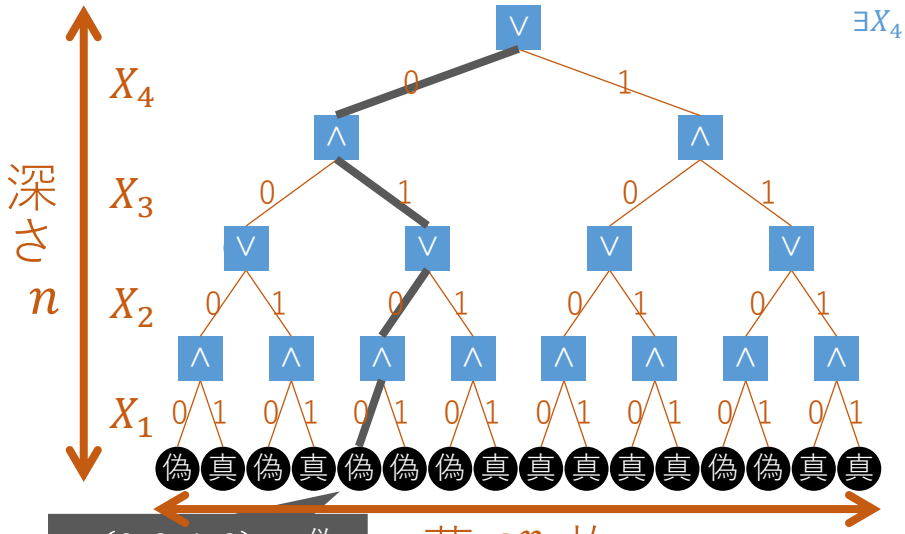
命題変数
真（1）か偽（0）の値をとる

量化子（∀ か ∃）

命題論理式
命題変数から論理記号
∧（かつ）
∨（または）
¬（否定）
を使って作られる式

「二人ゲームでどちらが必勝か判定する問題」

例 $\exists X_4 . \forall X_3 . \exists X_2 . \forall X_1 . (X_2 \vee \neg X_3) \wedge (X_1 \vee X_4)$



$\varphi(0,0,1,0) =$ 偽
容易に計算できる

葉 2^n 枚

この例では
$n = 4$

$\exists X_4 . \forall X_3 . \exists X_2 . \forall X_1 . \varphi(X_1, X_2, X_3, X_4)$

$\forall X_3 . \exists X_2 . \forall X_1 . \varphi(X_1, X_2, X_3, X_4)$

$\exists X_2 . \forall X$

$\forall X$

$$\mathrm{QBF}(\forall x . \psi(x)) = \mathrm{QBF}(\psi(0)) \wedge \mathrm{QBF}(\psi(1))$$

$$\mathrm{QBF}(\exists x . \psi(x)) = \mathrm{QBF}(\psi(0)) \vee \mathrm{QBF}(\psi(1))$$

深さ優先探索
➡ 多項式空間認識可能

定理

QBF は **PSPACE** 完全

問題 $SR_{\geq}$

入力 (R, w) 但し各規則が (旧長さ) ≥ (新長さ)

問 $w \Rightarrow_R^* \varepsilon$ か

$w \Rightarrow_R^{\leq 2^n} \varepsilon$ かどうかを調べれば良い（昨日）

$SR_{\geq} \leq^P$ QBF を示す

昨日 $SR_{\geq}$ の多項式空間認識可能性を示したときと同じ次の関係を用いる

$x \Rightarrow_R^{\leq 2^{i+1}} y \iff \exists\, z\ (x \Rightarrow_R^{\leq 2^i} z$ かつ $z \Rightarrow_R^{\leq 2^i} y)$

「∃ w」は「或る $w \in \Sigma^{<n}$ が存在して」
「∀ w」は「任意の $w \in \Sigma^{<n}$ に対し」の意

$\iff \exists\, z\ \forall\, u, v$

$(\text{もし } (u, v) = (x, z) \text{ または } = (z, y) \text{ ならば } u \Rightarrow_R^{\leq 2^i} v)$

これを再帰的に用いて

$w \Rightarrow_R^{\leq 2^n} \varepsilon \iff$

$\exists\, z_n\ \forall\, u_n, v_n\ \ \exists\, z_{n-1}\ \forall\, u_{n-1}, v_{n-1}\ \cdots\cdots\ \exists\, z_1\ \forall\, u_1, v_1$

もし $(u_n, v_n) = (w, z_n)$ または $= (z_n, \varepsilon)$ ならば

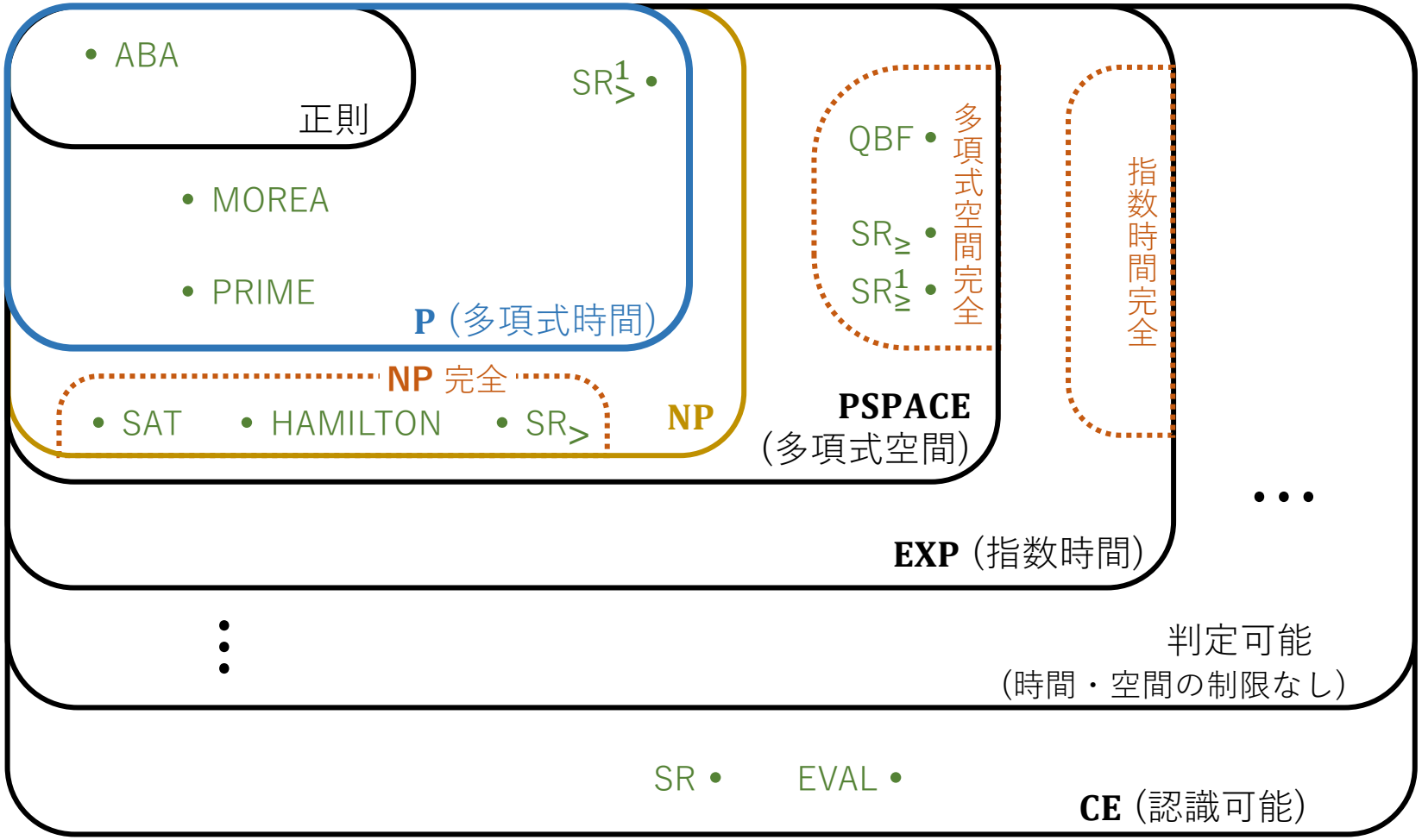
もし $(u_{n-1}, v_{n-1}) = (u_n, z_{n-1})$ または $= (z_{n-1}, v_n)$ ならば

……

もし $(u_1, v_1) = (u_2, z_1)$ または $= (z_1, v_2)$ ならば $u_1 \Rightarrow_R^{\leq 1} v_1)$

これを量化命題論理式として書くことができる

複雑さの階層と完全問題



Applying type-two complexity to computation over the reals
Part 1

Akitoshi Kawamura (Kyoto U)

Type-two theory of effectivity (TTE)

- Theory of computation for higher types
  - Type 0: finite objects (e.g., strings, integers, graphs, …)
  - Type 1: sets/functions of type-0 objects
    (e.g., sequences of integers, sets of strings, real numbers, …)
  - Type 2: sets/functions of type-1 objects (e.g., real functions, …)
  - Type 3 and higher? – Not in this talk
- Foundation of Computable Analysis

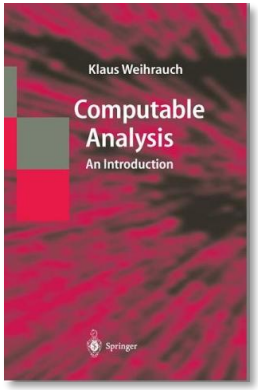
Computable Analysis

Computability and Complexity in Analysis

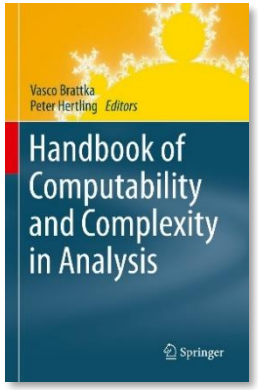
http://cca-net.de/

Workshop (1995–)

Conference (2005–)



K. Weihrauch, *Computable Analysis*, Springer, 2000.

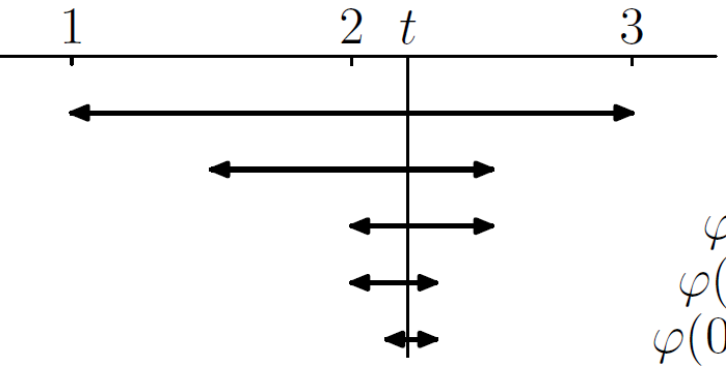


V. Brattka and P. Hertling, Eds., *Handbook of Computability and Complexity in Analysis*, Springer, 2021

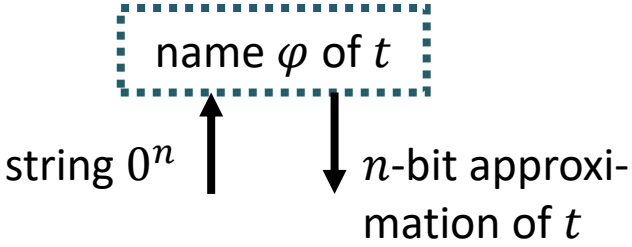
Related areas: Constructive mathematics
Reverse mathematics
Validated numerics

Definition

We say that φ: $\Sigma^* \to \Sigma^*$ is a name of $t \in \mathbf{R}$ if for each $n \in \mathbf{N}$,
$\varphi(0^n)$ is (the binary expansion of) t rounded up or down at the nth bit.



$$\varphi(\varepsilon) = +10.$$
$$\varphi(0) = +10.0$$
$$\varphi(00) = +10.01$$
$$\varphi(000) = +10.001$$
$$\varphi(0000) = +10.0011$$

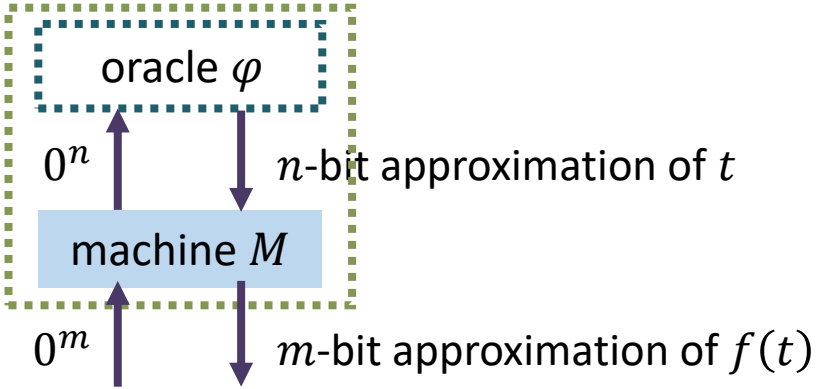


Thus, we view real numbers as Cauchy sequences (guaranteed to converge fast).

Rational numbers, algebraic numbers, as well as well-known numbers like π and e, are polynomial-time computable (i.e., have polynomial-time computable names), though of course almost all reals are non-computable.

Definition

Machine M computes $f\colon [0,1] \to \mathbf{R}$ if, for any name φ of any $t \in [0,1]$, M^{φ} computes a name of $f(t)$.



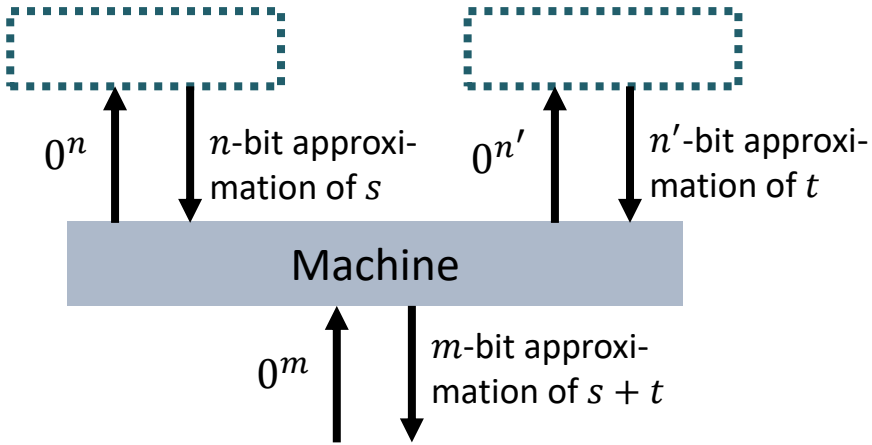
Thus, given access to approximations of t to any precision, the machine must output $f(t)$ to any precision.

Polynomial-time machine M runs in time poly(m).
We write $\mathbf{P}$ also for the class of real functions computed by such M.

Example To compute $f\colon \mathbf{R} \to \mathbf{R}$:
$t \mapsto 3 \cdot t$

1. The user inputs 0^m.
2. The machine queries 0^n for $n = m + 3$.
3. The oracle (or the user) gives a rational r (satisfying $|r - t| < 2^{-n}$).
4. The machine outputs $3 \cdot r$ rounded to the nearest multiple of 2^{-m}.

Example Addition $+\colon [0,1]^2 \to \mathbf{R}$ is in $\mathbf{P}$



$\because$ $n = n' = m + 2$ suffice.

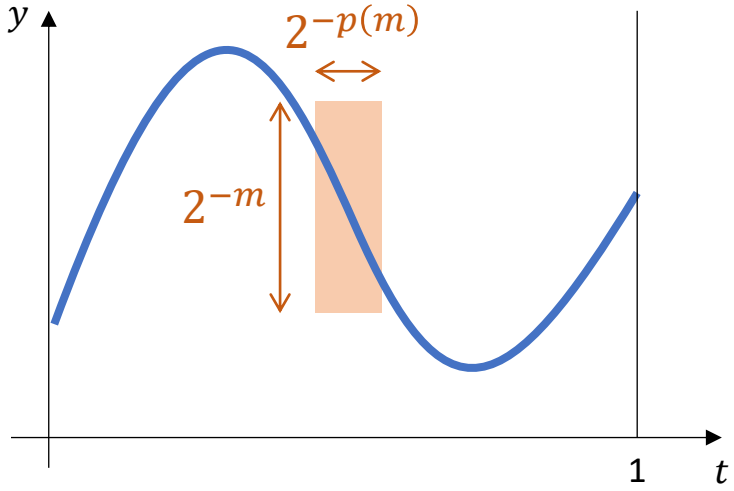
Example $\sin\colon [0,1] \to \mathbf{R}$ is in $\mathbf{P}$

$$\sin t = \frac{t}{1!} - \frac{t^3}{3!} + \frac{t^5}{5!} - \frac{t^7}{7!} + \cdots$$

$\because$ Since sin has slope ≤ 1,
the value $\sin t$ can be approximated
by the value $\sin t'$ at a nearby point t'.

The series converges fast enough,
so that an m-bit approximation is
obtained using the first $O(m)$ terms.

Computable ⟹ Continuous



Theorem

- A computable real function is continuous.
- A real function $f\colon [0,1] \to \mathbf{R}$ in $\mathbf{P}$ has a polynomial modulus of continuity, i.e., a polynomial $p\colon \mathbf{N} \to \mathbf{N}$ satisfying $|t - t'| < 2^{-p(m)} \implies |f(t) - f(t')| < 2^{-m}$

Theorem

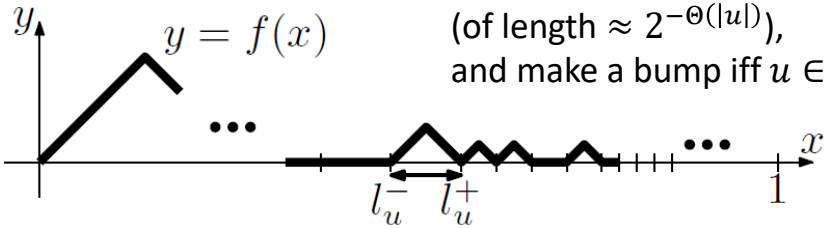
A real function $f\colon [0,1] \to \mathbf{R}$ is in $\mathbf{P}$ if and only if

- f has a polynomial modulus of continuity, and
- there is a function $\varphi\colon \Sigma^* \to \Sigma^*$ in $\mathbf{P}$ (in the usual type-1 sense) such that $|\varphi(u, 0^n) - f(u)| \leq 2^{-n}$ for each $u \in \mathbf{Q}$ and $n \in \mathbf{N}$.

Example

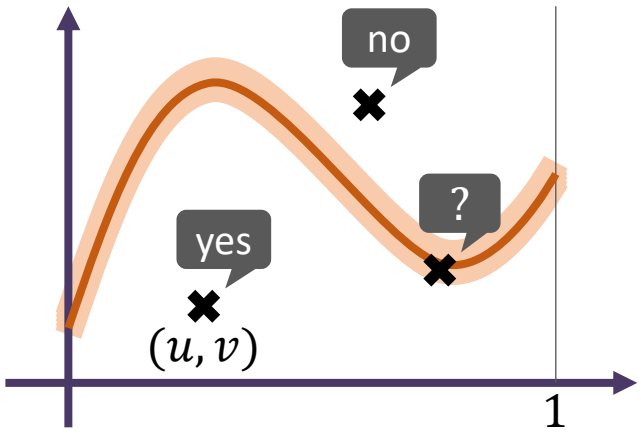
For $A \subseteq \{0,1\}^*$, define $f_A\colon [0,1] \to \mathbf{R}$ like this:

Then $f_A \in \mathbf{P}$ iff $A \in \mathbf{P}$.



Assign to each $u \in \{0,1\}^*$ an interval $[l_u^-, l_u^+]$ (of length $\approx 2^{-\Theta(|u|)}$), and make a bump iff $u \in A$.

Computable ⟹ Continuous



Theorem

A real function $f\colon [0,1] \to \mathbf{R}$ is in $\mathbf{P}$ if and only if

- f has a polynomial modulus of continuity, and
- there is a set $\mathrm{Below}_f \subseteq \Sigma^*$ in $\mathbf{P}$ (in the usual type-1 sense) such that for all $u \in \mathbf{Q} \cap [0,1]$, $v \in \mathbf{Q}$ and $n \in \mathbf{N}$ with $|v - f(u)| > 2^{-n}$, we have

$$\mathrm{Below}_f(u, v, 0^n) = 1 \Leftrightarrow v < f(u).$$

We can determine in polynomial time
whether a given point (u, v) is below the graph of f,
except when the point is very close (within 2^{-n}) to the graph.

We look at various constructions (operators) on real functions in mathematics and ask: Does this construction preserve polynomial-time?

Maximization

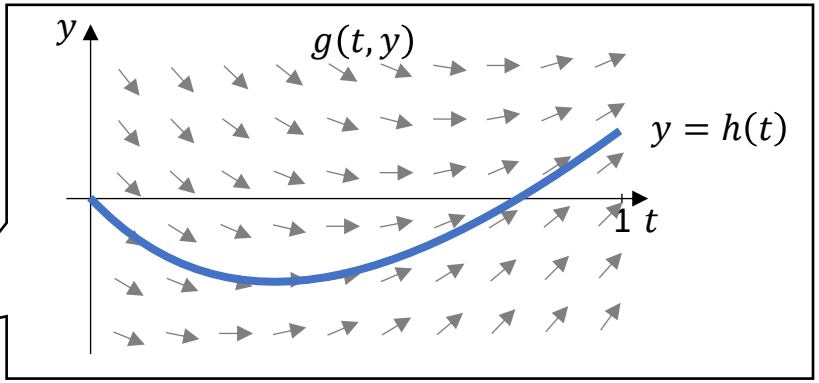
$$h(t) = \max_{0 \le \tau \le t} g(\tau)$$

Integration

$$h(0) = 0, \quad h'(t) = g(t)$$

Solving ODE

$$h(0) = 0, \quad h'(t) = g\big(t, h(t)\big)$$



Question

g is in $\mathbf{P} \implies h$ is in $\mathbf{P}$?

Answer: Yes iff $\mathbf{P} = \mathbf{NP}$ (for maximization)

$\#\mathbf{P}$ (for integration)

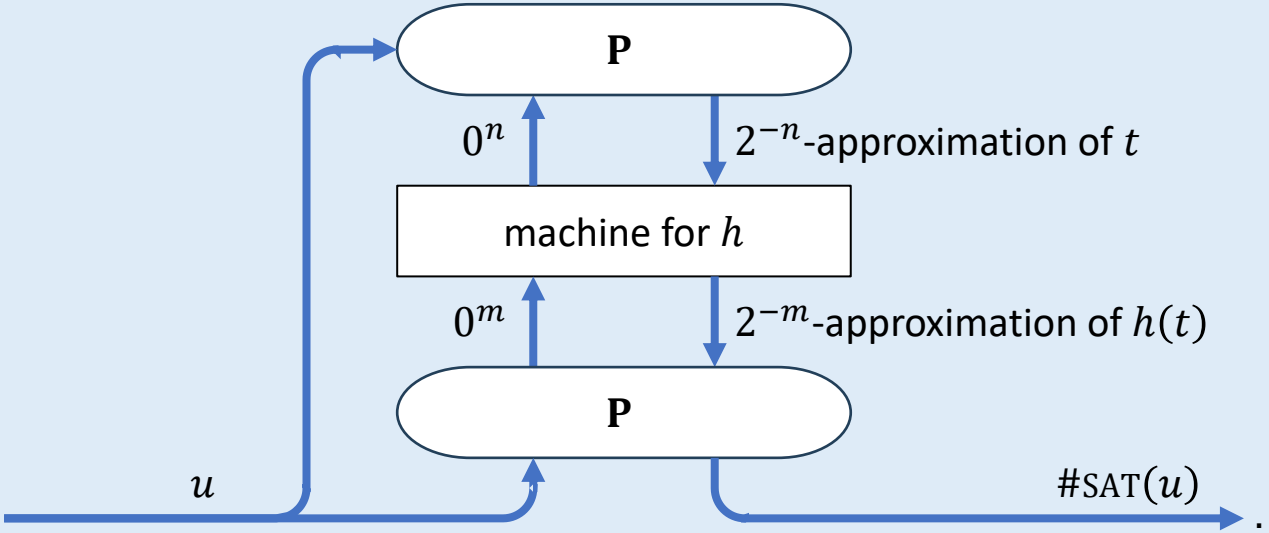
$\mathbf{PSPACE}$ (for solving Lipschitz continuous ODE)

Complexity of integration

Theorem (essentially [Fri84])

There are $g\colon [0,1] \to \mathbf{R}$ and $h\colon [0,1] \to \mathbf{R}$ such that

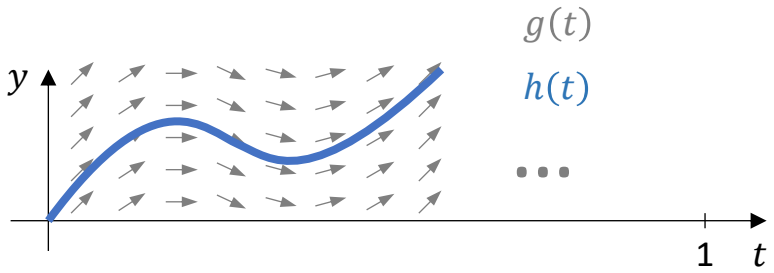
- g is in $\mathbf{P}$;
- $h(0) = 0,\ h'(t) = g(t)$;
- h is $\#\mathbf{P}$-hard, in the sense that



$\#\mathrm{SAT}(u)$ = number of satisfying assignments of formula u

[Fri84] H. Friedman. The Computational Complexity of Maximization and Integration. Advances in Mathematics 53, 80–98, 1984.

Reducing #SAT to integration



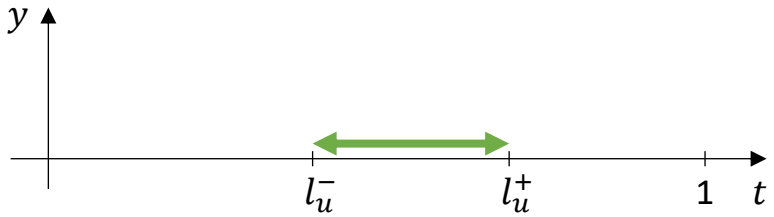
What we want:

- g is easy (in $\mathbf{P}$);
- h is hard (#SAT reduces to it).

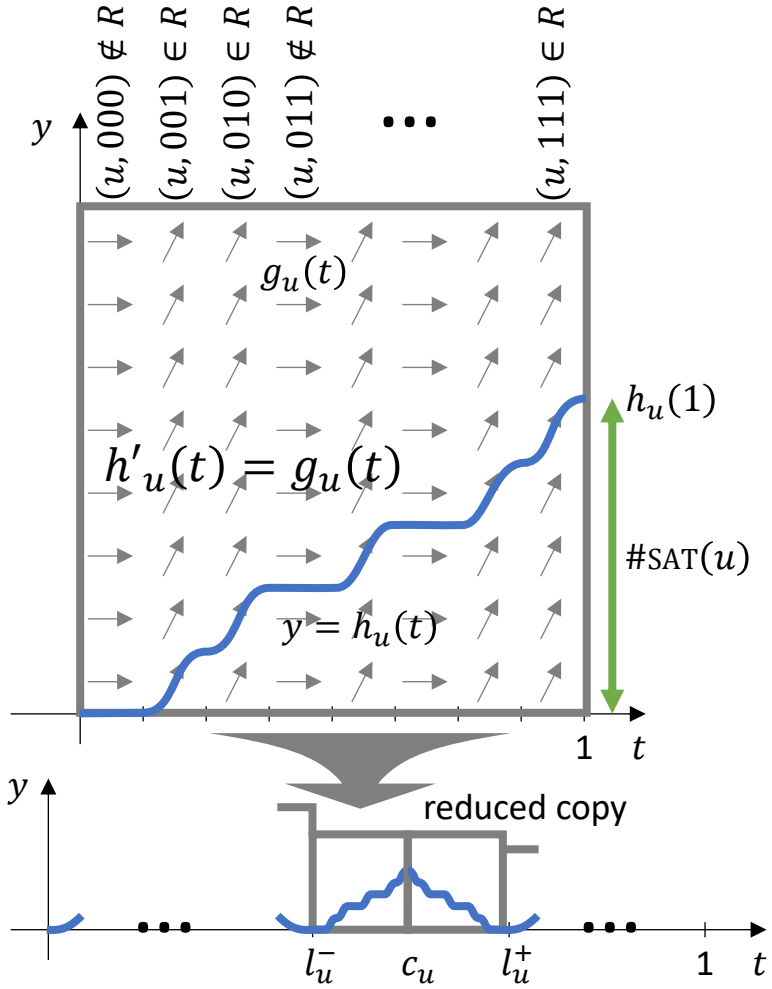
Reducing #SAT to integration

$\#\mathrm{SAT}(u) =$
number of v with $(u, v) \in R$
(for the satisfaction relation $R \in \mathbf{P}$).

For each $u \in \Sigma^*$,
assign interval $[l_u^-,\ l_u^+]$



Reducing #SAT to integration



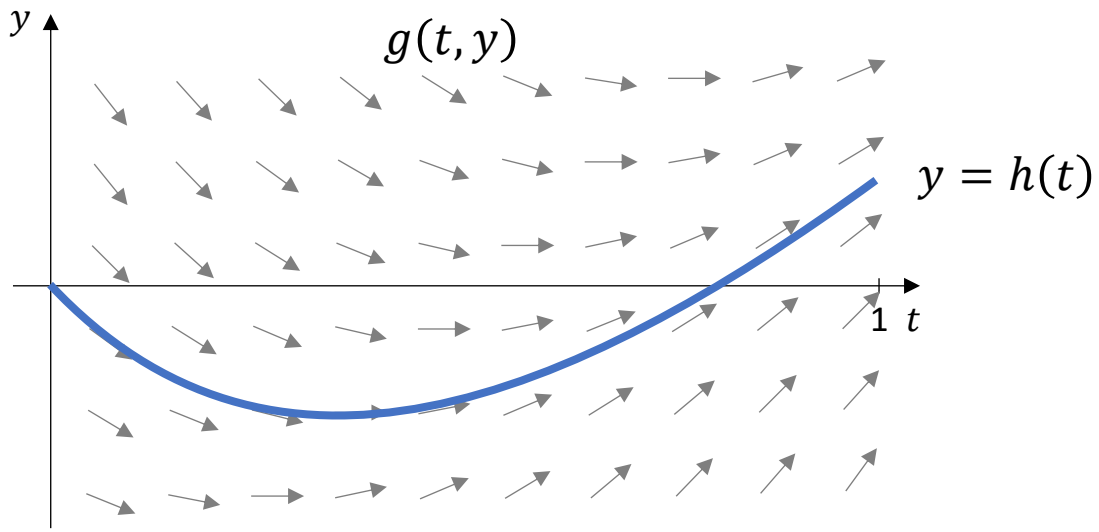
$\#\mathrm{SAT}(u) =$
number of v with $(u, v) \in R$
(for the satisfaction relation $R \in \mathbf{P}$).

For each $u \in \Sigma^*$,
assign interval $[l_u^-,\ l_u^+]$.
Put there a pair of
reduced copies of g_u.

$\#\mathrm{SAT}(u)$ is encoded in
$h_u(1)$, or in $h(c_u)$.

The Lipschitz condition

$$h(0) = 0, \quad h'(t) = g\big(t, h(t)\big)$$



A sufficient condition to have a unique solution h is that g be **Lipschitz continuous**: there is some constant L such that

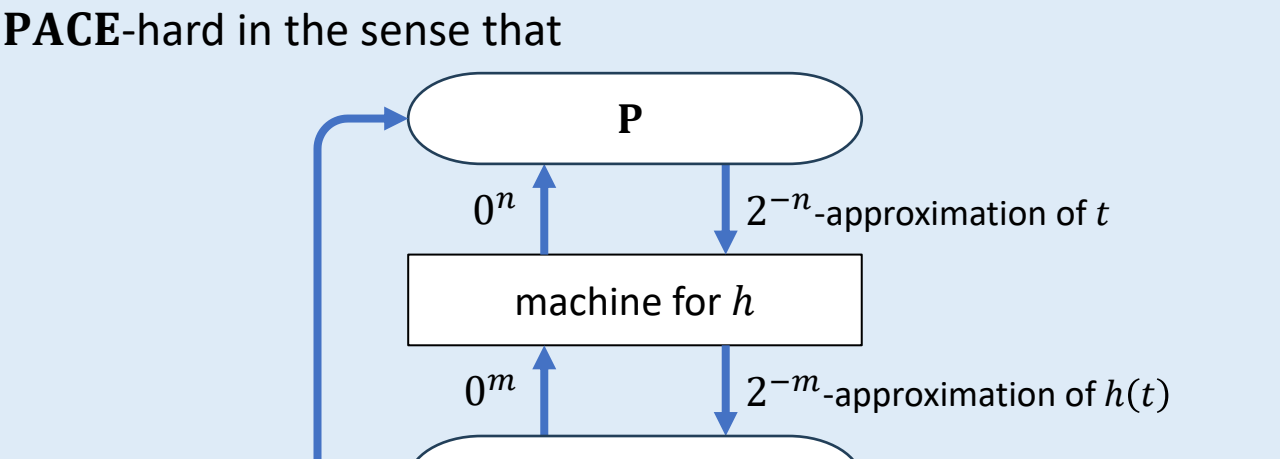
$$|g(t, y_0) - g(t, y_1)| \leq L|y_0 - y_1|.$$

Complexity of Lipschitz ODEs

Theorem [Kaw10]

There are $g\colon [0,1]\times[-1,1]\to \mathbf{R}$ and $h\colon [0,1]\to[-1,1]$ such that

- g is in $\mathbf{P}$ and Lipschitz continuous;
- $h(0)=0,\ h'(t)=g\big(t,h(t)\big)$;
- h is $\mathbf{PSPACE}$-hard in the sense that



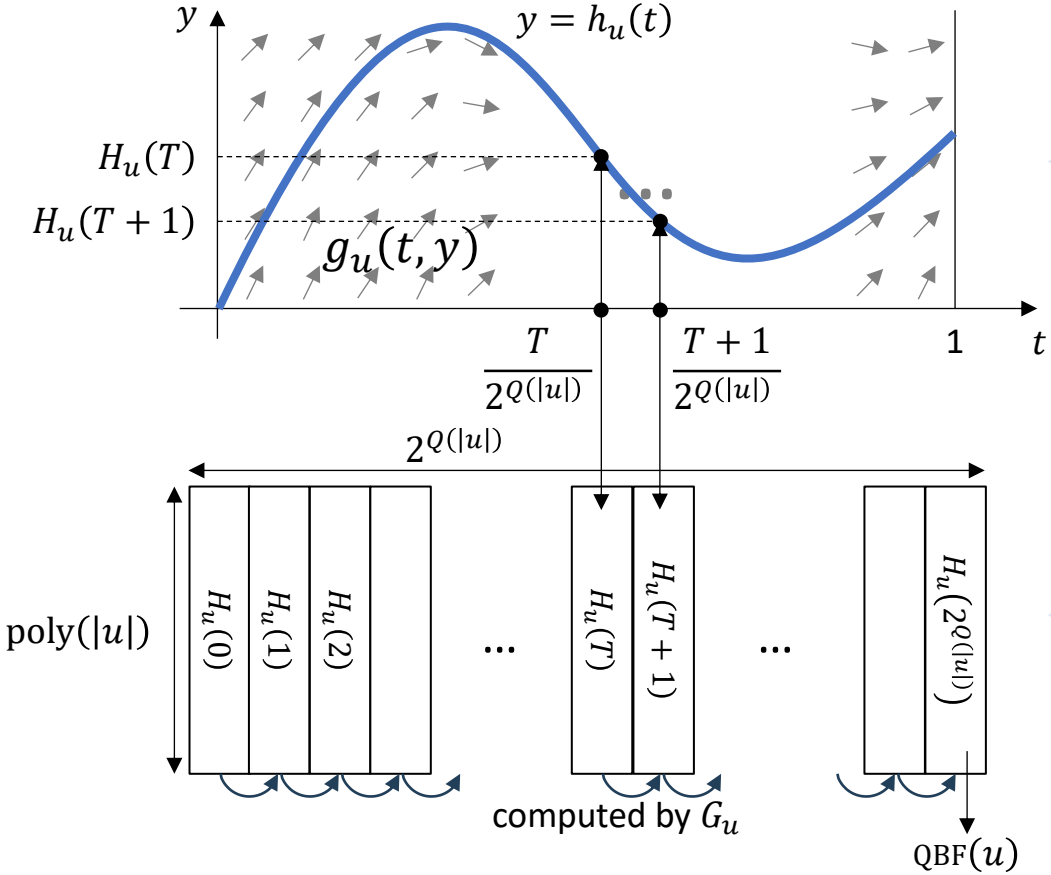
Cf. Upper bound: h is in $\mathbf{PSPACE}$ by the Euler method [Ko83].

[Kaw10] A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. Computational Complexity 19, 305–332, 2010.

[Ko83] K. Ko. On the computational complexity of ordinary differential equations. Information and Control 58, 157–194, 1983.

Proof (1/4): An attempt to reduce PSPACE to ODE

As before, we need blocks g_u such that $h_u(1) \Leftrightarrow u \in \mathrm{QBF}$.



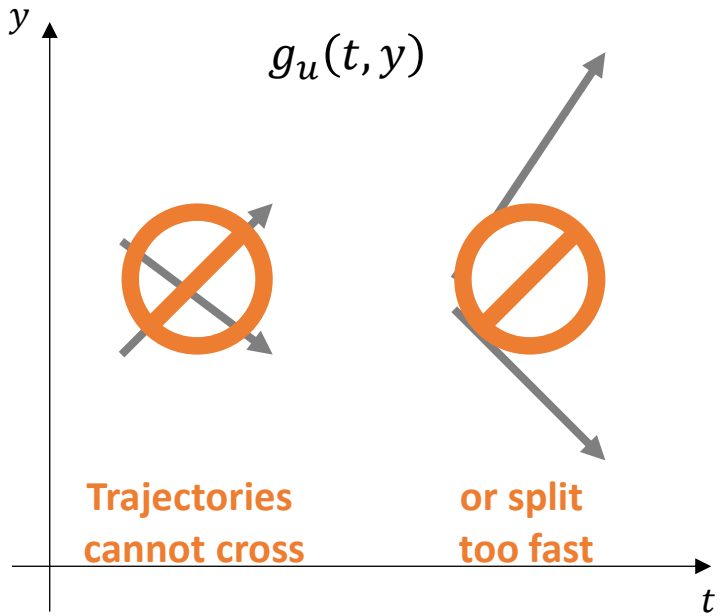
Differential equation:
$h'_u(t) = g_u\big(t, h(t)\big)$

Description of a **PSPACE** machine on input u:
$H_u(T+1) = G_u\big(T, H_u(T)\big)$

Q: polynomial

Proof (2/4): Why this attempt does not work

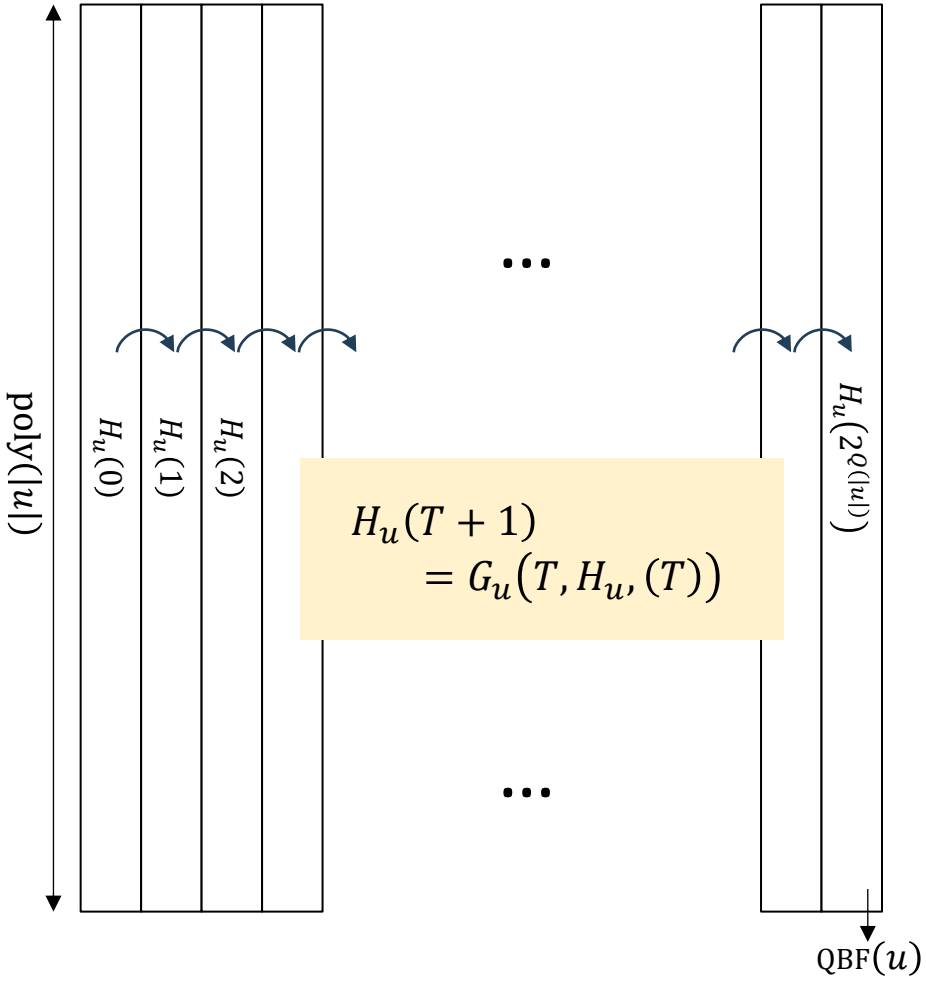
Not all G_u can be translated to G_u.



By the Lipschitz condition,
g_u cannot change quickly, i.e.,
the flows must be nearly parallel.

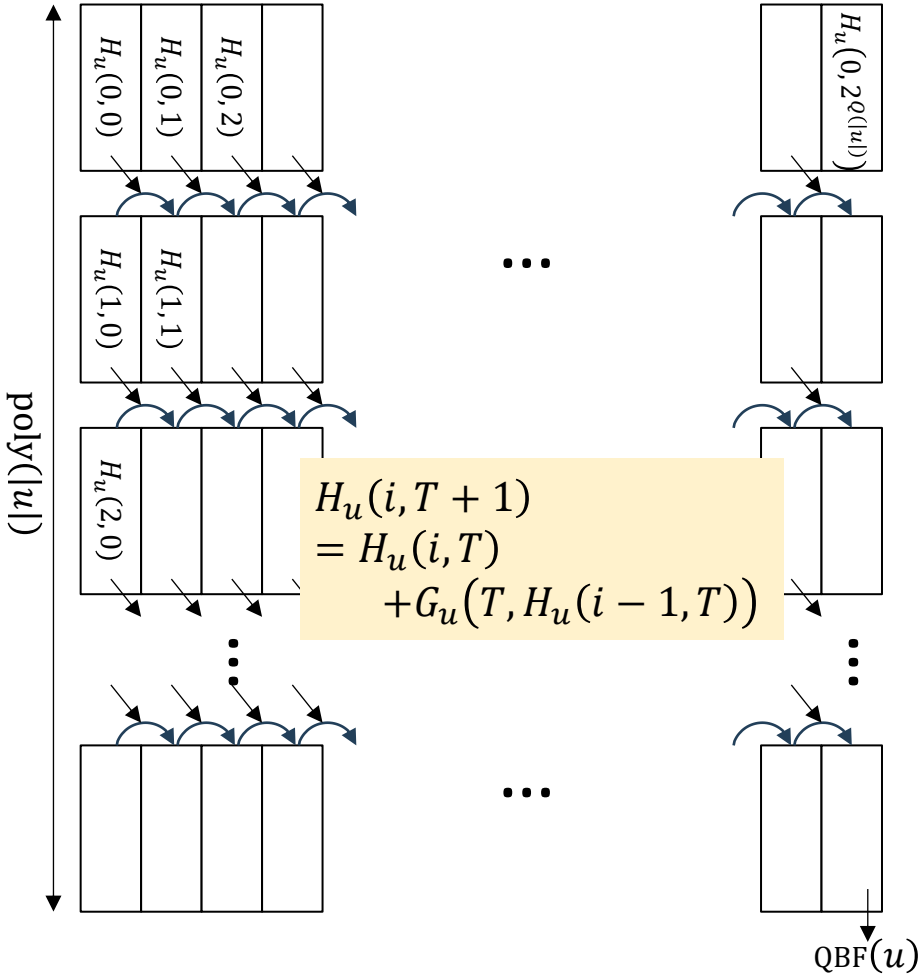
Thus, the 'feedback' (on h_u) described by the equation
$h'_u(t) = g_u\big(t, h_u(t)\big)$ is very weak.

Proof (3/4): Layered PSPACE tables



Differential equations cannot simulate general **PSPACE** computation tables.

Proof (3/4): Layered PSPACE tables

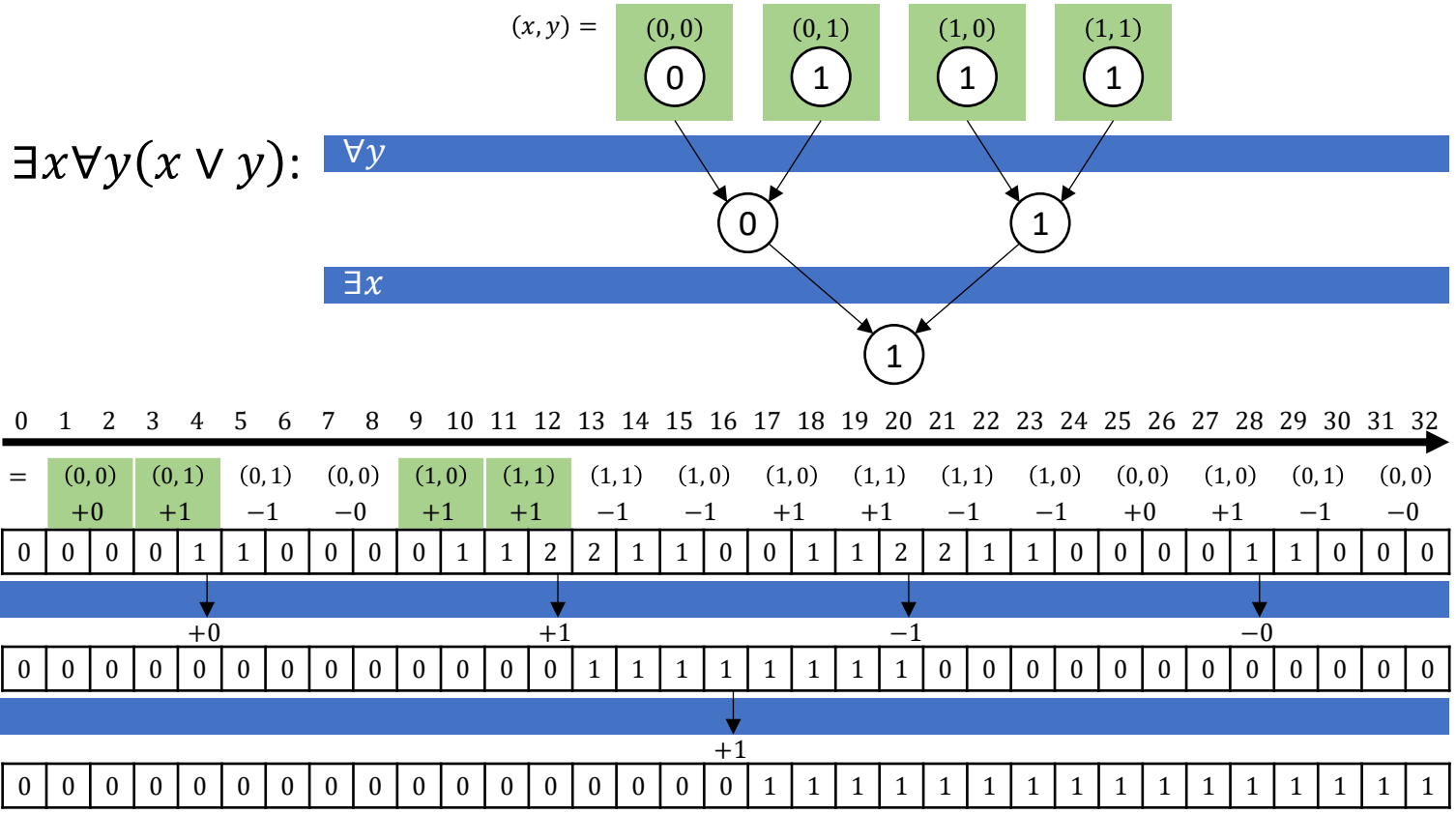


Differential equations cannot simulate general **PSPACE** computation tables.

But it can simulate **layered tables** where each cell is split into parts and each part only affects parts below itself.

Proof (4/4): Layered tables are still hard

Despite the restriction, the layered table is **PSPACE**-complete,
by a reduction from QBF.



Computational complexity of ODE

$$h(0) = 0, \qquad h'(t) = g(t, h(t))$$

Assuming g is in **P**, how hard can h be?

Assumptions	Upper bound	Lower bound
None	—	Can be all non-computable [PR79]
Solution h is unique	Computable	Can take arbitrarily long time [Mil70]
g is "locally Lipschitz"	In **EXPSPACE** [Ko 1992]	Can be **EXPSPACE**-hard [Kaw10]
g is Lipschitz	In **PSPACE** [Ko83]	Can be **PSPACE**-hard [Kaw10]
g is of class C^1		Can be **PSPACE**-hard [KORZ15]
g is of class C^k		Can be **CH**-hard [KORZ15]
g is analytic	In **P** [Mül87]	—

[Ko92] K. Ko. On the computational complexity of integral equations. Annals of Pure and Applied Logic 58, 201–228, 1992.
[KORZ15] A. Kawamura, H. Ota, C. Rösnick and Martin Ziegler. Computational complexity of smooth differential equations. Logical Methods in Computer Science, 10, 2014.
[Mil70] W. Miller. Recursive function theory and numerical analysis. Journal of Computer and System Sciences 4, 465–472, 1970.
[Mül87] N. Müller. Uniform computational complexity of Taylor series. Automata, Languages and Programming, 435–444, 1987.
[PR79] M. B. Pour-El and I. Richards. A computable ordinary differential equation which possesses no computable solution. Annals of Mathematical Logic 17, 61–90, 1979

Applying type-two complexity to computation over the reals
Part 2

Akitoshi Kawamura (Kyoto U)

In Part 1:

- Representation of real numbers
- Computability of real functions in the type-two model (oracle machine)
- Which operators preserve polynomial-time computability?
  - Maximization, integration, solving differential equations, …
  - Hardness for type-1 classes

Theorem

If a Lipschitz continuous $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ is in $\mathbf{P}$, so is the unique solution $h\colon [0,1]\to[-1,1]$ of

$$h(0)=0,\qquad h'(t)=g\big(t,h(t)\big).$$

$\Leftrightarrow \mathbf{P}=\mathbf{PSPACE}$

Let's look at these arguments in a more general framework.

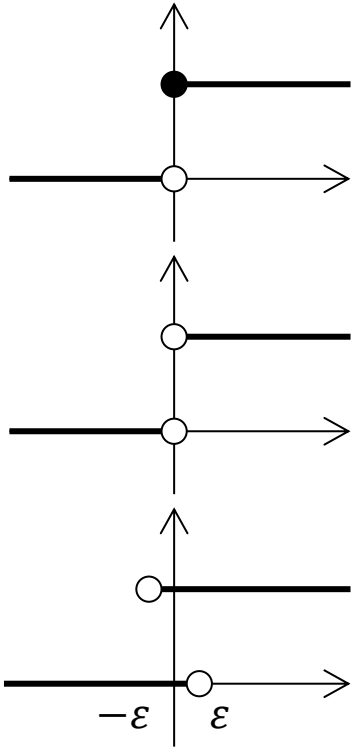
Computable functions are continuous.

Exact comparison of real numbers is not possible.

This test for $x \geq 0$ is discontinuous (and thus non-computable).

The partial test diverging at $x = 0$ is computable.

The multi-valued function that is allowed to err for $|x| < \varepsilon$ is also computable.



Partial and multi-valued functions are needed (in more essential ways than for type-1 computation).

Definition

A problem (multi-valued partial function) $A \colon \subseteq X \rightrightarrows Y$ consists of

- a set $\operatorname{dom} A \subseteq X$, and
- for each $x \in \operatorname{dom} A$, a set $A[x] \subseteq Y$.

This is regarded as the problem: "Given $x \in \operatorname{dom} A$, output *any* element of $A[x]$".

The problem A is

- total if $\operatorname{dom} A = X$,
- single-valued if each $A[x]$ is a singleton, and
- a function if it is both total and single-valued.

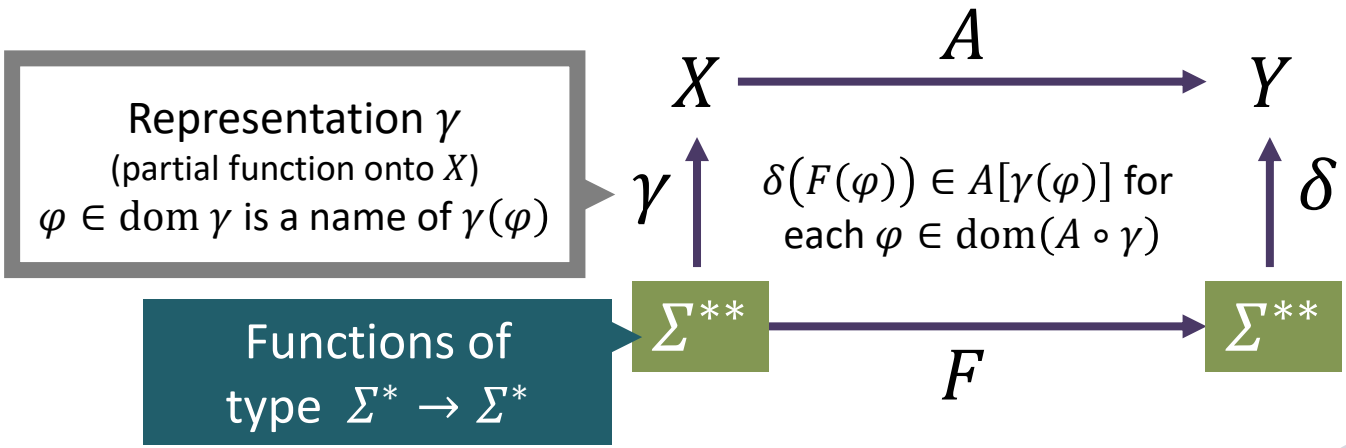
Thus a problem B is "easier" than A if: $\operatorname{dom} A \supseteq \operatorname{dom} B$ and
$A[x] \subseteq B[x]$ for each $x \in \operatorname{dom} B$.

The composition $A \circ B$ of problems A and B is defined by

$$\operatorname{dom}(A \circ B) = \{\, x \in \operatorname{dom} B : B[x] \subseteq \operatorname{dom} A \,\},$$

$$(A \circ B)[x] = \bigcup_{y \in B[x]} A[y]\,.$$

Solving a type-two problem $A :\subseteq X \rightrightarrows Y$
wrt representations $\gamma :\subseteq \Sigma^{**} \to X$ and $\delta :\subseteq \Sigma^{**} \to Y$:



"F realizes A wrt γ, δ"

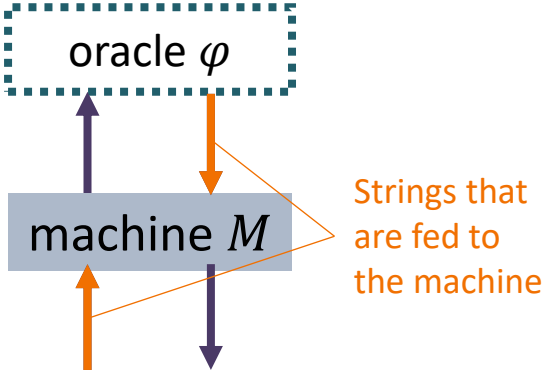
Yesterday:
$X = [0,1]$
$Y = \mathbf{R}$
$\gamma = \rho_{\mathrm{Cauchy}}^{[0,1]}$
$\delta = \rho_{\mathrm{Cauchy}}$

To discuss the complexity of A, we should define

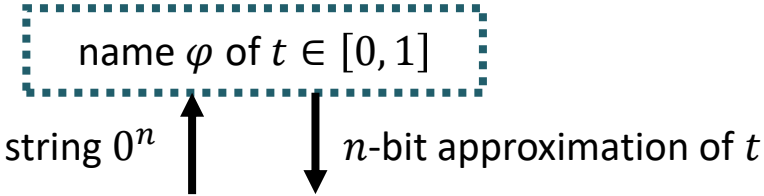
- what it means to compute $F :\subseteq \Sigma^{**} \to \Sigma^{**}$, and
- the representations γ and δ.

Type-two complexity

What does it mean for an oracle machine to
"run in polynomial time" (in the size of the inputs)?



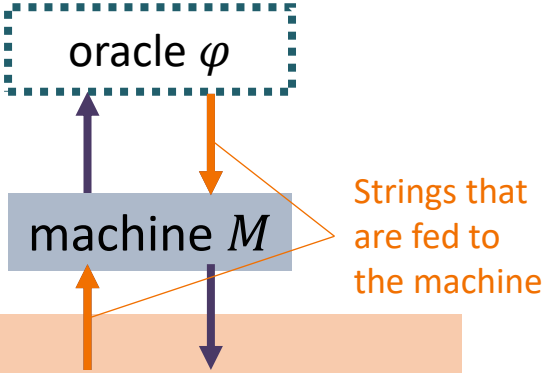
Yesterday, when we the oracle was of the form



we did not have to worry about the oracle answer being too long,
so we simply measured complexity in the size of the string input to M.

Type-two complexity

What does it mean for an oracle machine to
"run in polynomial time" (in the size of the inputs)?



Definition [Meh76 / KC96 / KC12]

- We write Σ^{**} for the set of functions $\varphi\colon \Sigma^* \to \Sigma^*$ such that
$$|x| \le |y| \Rightarrow |\varphi(x)| \le |\varphi(y)|.$$
- The length $|\varphi|\colon \mathbf{N} \to \mathbf{N}$ of $\varphi \in \Sigma^{**}$ is defined by $|\varphi|(|x|) = |\varphi(x)|$.
- An oracle TM M runs in polynomial time if for each $\varphi \in \Sigma^{**}$ and $x \in \Sigma^*$, the number of steps in computation $M^{\varphi}(x)$ is bounded by a second-order polynomial $P(|\varphi|)(|x|)$.

an expression built by $+$, $\times$, and application of $|\varphi|$
e.g., $|\varphi|(4|\varphi|(2|x|^3)^2 + 5) + |\varphi|(|x|^4)$

$P\colon (\mathbf{N} \to \mathbf{N}) \to (\mathbf{N} \to \mathbf{N})$

P: the class of type-two problems that are polynomial-time computable

Preserves (the type-one) class **P**

PSPACE: analogously defined

[KC96] B. M. Kapron and S. A. Cook. A new characterization of type-2 feasibility. SIAM Journal Computing 25(1):117–132, 1996.
[KC12] A. Kawamura and S. Cook. Complexity theory for operators in analysis. ACM Transactions on Computation Theory 4, Article 5, 2012.
[Meh76] K. Mehlhorn. Polynomial and abstract subrecursive classes. Journal of Computer and System Sciences 12, 147–178, 1976.

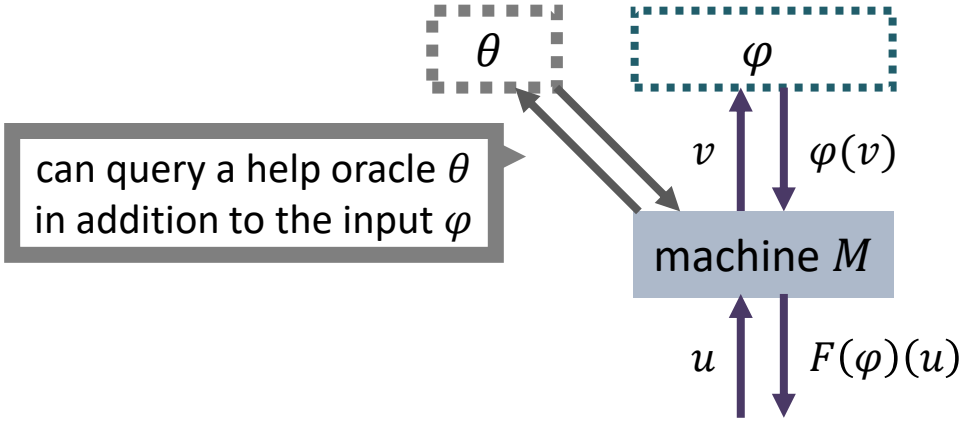
Type-two classes $\mathbf{P} \subsetneq \mathbf{PSPACE} \subsetneq \mathbf{Computable} \subsetneq \mathbf{Continuous}$

On the topological space Σ^{**} where
sets of the form $\{ \varphi \in \Sigma^{**} : \varphi(u) = v \}$ are open

Continuous functions on Σ^{**} are those where finite information about the output is determined by finite information of the input.

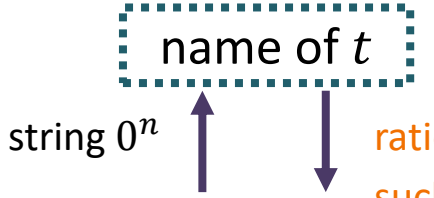
Theorem

Continuous $\Leftrightarrow$ Computable with the help of some oracle



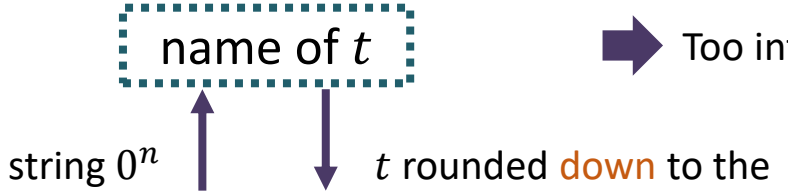
Bad representations of real numbers

Naïve converging sequences

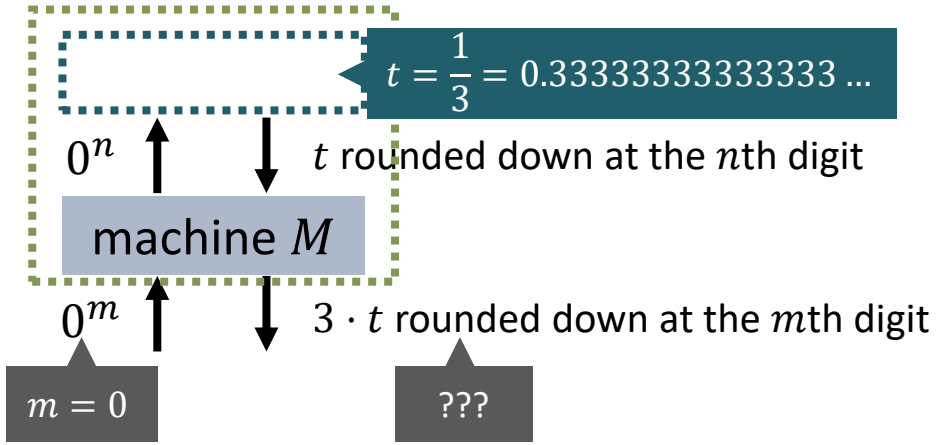


➡ Finitely many terms t_n are useless
(they do not give any information about t)

Naïve binary expansion



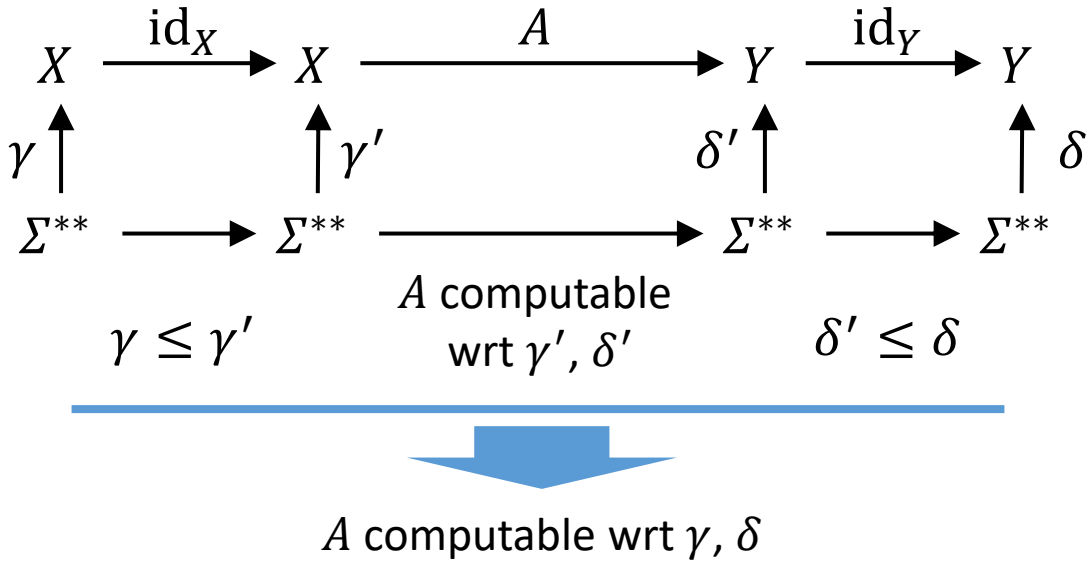
➡ Too informative, making the "× 3" function non-computable



Definition

For representations $\gamma, \gamma' :\subseteq \Sigma^{**} \to X$,
a **translation** of γ to γ' is a realization of the identity function id_X wrt γ, γ'.

The existence of a computable translation from γ to γ' (written $\gamma \leq \gamma'$) means that γ is "richer", "more informative" than γ'.



Which equivalence class of representations is the "good" one?

Definition

A representation $\gamma :\subseteq \Sigma^{**} \to X$ of a (second-countable T0) topological space X is **admissible** if

- γ is continuous, and
- every continuous representation γ' of X has a continuous translation to γ.

Theorem

For admissible representations $\gamma :\subseteq \Sigma^{*} \to X$ and $\delta :\subseteq \Sigma^{*} \to Y$,
a function $f : X \to Y$ is continuous if and only if it has a continuous realizer wrt γ, δ.

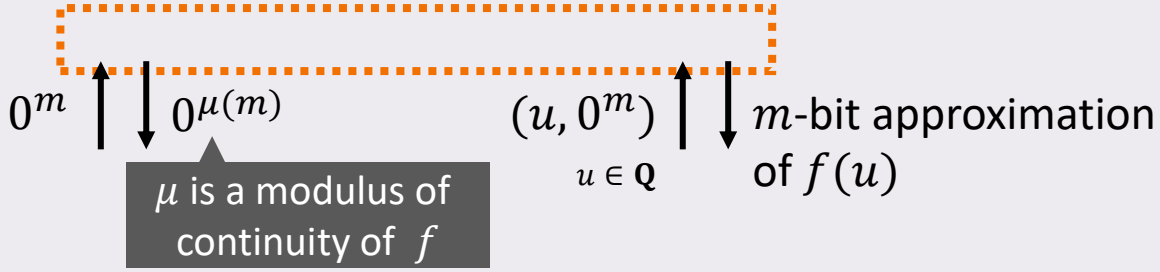
For example, in $\mathbf{R}$,

- The Cauchy representation is admissible.
- The naïve converging sequence is not continuous.
- The naïve binary expansion is continuous but too strong.

［KW85］ C. Kreitz and K. Weihrauch. Theory of representations. Theoretical Computer Science 38, 35–53, 1985.

Representation of continuous real functions

A $\delta_{\square}$-name of $f \in \mathrm{C}[0,1]$ is:



All $f \in \mathrm{C}[0,1]$ has a $\delta_{\square}$-name.

It has a polynomial-time $\delta_{\square}$-name iff f is in $\mathbf{P}$ wrt $\rho_{\mathrm{Cauchy}}^{[0,1]}$ and ρ_{Cauchy}.

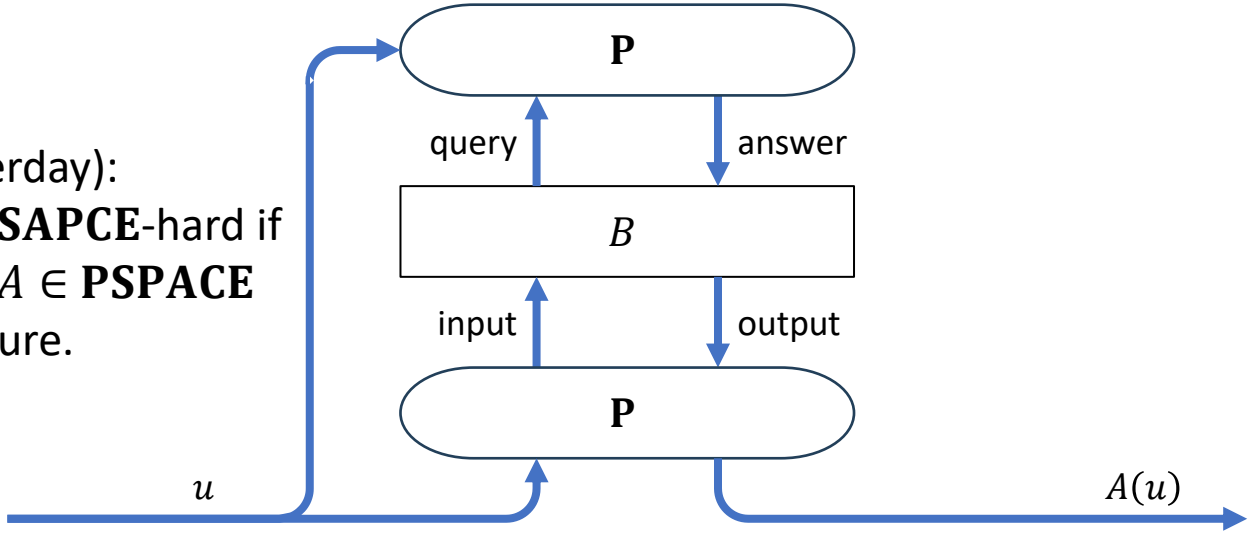
Theorem

- Function evaluation APPLY: $\mathrm{C}[0,1] \times [0,1] \to \mathbf{R}$ is in $\mathbf{P}$ wrt $\delta_{\square}$, $\rho_{\mathrm{Cauchy}}^{[0,1]}$ and ρ_{Cauchy}.
- $\delta_{\square}$ is the weakest such.

Likewise, in many cases we have a “natural” representation for the spaces we consider.

Hardness

PSPACE-hardness (yesterday):
Type-two problem B is **PSAPCE**-hard if every type-one problem $A \in$ **PSPACE** reduces to it as in this figure.

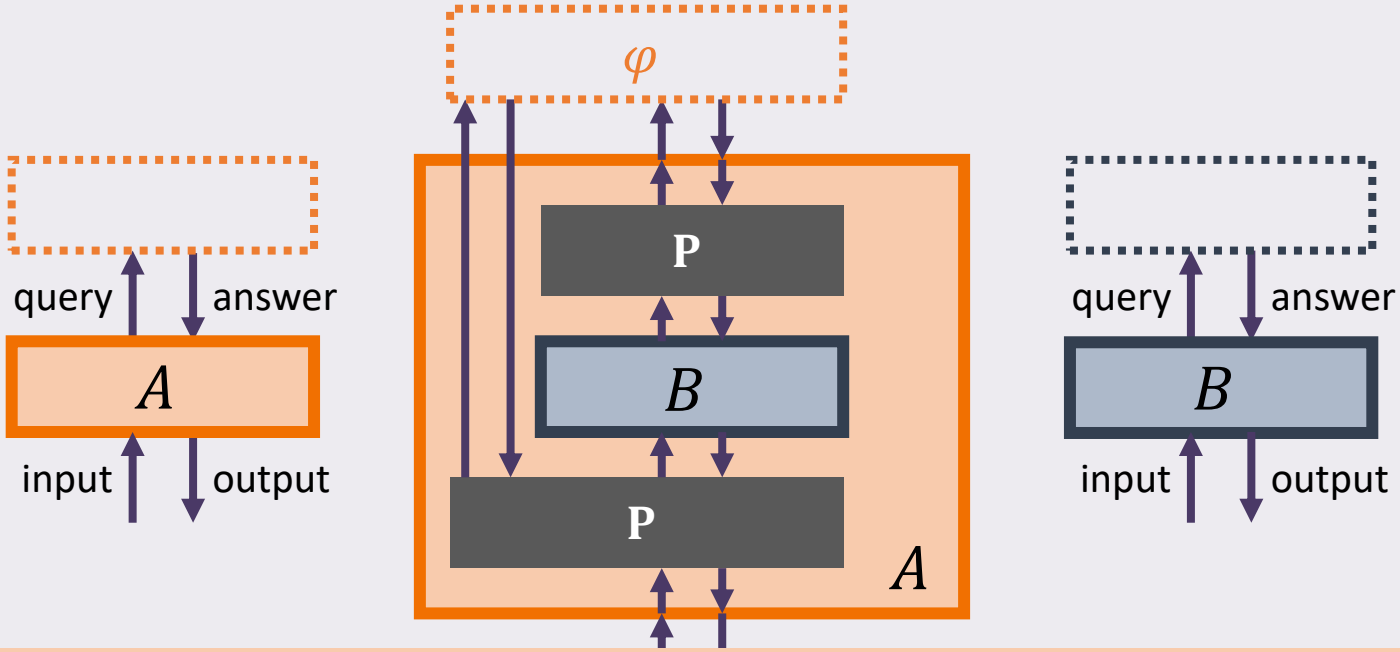


Cf. Papadimitriou's classes **PLS**, **PPA**, **PPAD**, … ⊆ **TFNP** (of type-1 problems) [Pap94]

More generally, we can consider reductions between type-two problems.

[Pap94] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. Journal of Computer and System Sciences 48, 498–532, 1994.

Weihrauch reducibility



Definition

Let A, B be type-two problems.
We say A reduces to B ($A \leq^{\mathbf{P}} B$) if there are functions $r, s \in \mathbf{P}$ such that
for all $\varphi \in \operatorname{dom} A$, we have $s(\varphi) \in \operatorname{dom} B$ and
for all $\theta \in B[s(\varphi)]$, we have $r(\langle \varphi, \theta \rangle) \in A[u]$.

Theorem

The following problems are (type-two) **PSPACE**-complete.

SPACE^2

Given oracle	triple $\langle M, \mu, \varphi \rangle$
Given string	u
Answer	The result of the computation $M^{\varphi}(u)$, if the computation halts within space $\mu(\lvert u \rvert)$

- machine M
- function $\mu\colon \mathbf{N} \to \mathbf{N}$ (represented as $0^n \mapsto 0^{\mu(n)}$)
- $\varphi \in \Sigma^{**}$

POW^2

Given oracle	Length-preserving function $f\colon \Sigma^* \to \Sigma^*$ ($\lvert f(x) \rvert \leq \lvert x \rvert$)
Given string	u
Answer	$f^{2^{\lvert u \rvert}}(u)$

QBF^2

Given oracle	$p\colon \{0,1\}^* \to \{0,1\}$
Given string	Quantified Boolean formula with a function symbol □
Answer	The value of the formula when □ is assigned p

E.g. $\forall X_2.\, \exists X_1.\, \square(X_1) \vee (\neg X_2 \wedge \square(X_2, \neg X_1))$

Theorem

The operator $solve$ that maps each Lipschitz continuous $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ to the unique solution $h\colon [0,1]\to[-1,1]$ of

$$h(0)=0,\qquad h'(t)=g\big(t,h(t)\big)$$

is **PSPACE**-complete wrt representations $\delta_{\square\mathrm{L}}$ and $\delta_{\square}$.

A $\delta_{\square\mathrm{L}}$-name of g consists of,
in addition to a $\delta_{\square}$-name of g,
the Lipschitz constant L of g written in unary

Corollary

Theorem (Yesterday)

If a Lipschitz continuous $g\colon [0,1]\times[-1,1]\to\mathbf{R}$ is in $\mathbf{P}$,
so is the unique solution $h\colon [0,1]\to[-1,1]$ of

$$h(0)=0,\qquad h'(t)=g\big(t,h(t)\big).$$

$\Leftrightarrow \mathbf{P}=\mathbf{PSPACE}$

Existence theorems for ODE at various complexity levels

$$h(0) = 0, \qquad h'(t) = g(t, h(t))$$

Cauchy–Peano Theorem

For each g, there is solution h (near the origin)

(Failure of) Computable Cauchy–Peano Theorem

$g \mapsto h$ wrt representation δ is non-computable

Cauchy–Lipschitz Theorem

For each Lipschitz g, there is solution h

Polynomial-space Cauchy–Lipschitz Theorem

$g \mapsto h$ wrt representation $\delta_{\square L}$ is in **PSPACE**

Cauchy–Kovalevskaya Thm

For each analytic g, there is analytic h

Polynomial-time Cauchy–Kovalevskaya Theorem

$g \mapsto h$ wrt representation δ_{analytic} is in **P**

Computational content of various mathematical theorems

Complexity-theoretic explanation of hardness of numerical problems

まとめ　帰着と完全問題

- 時間計算量は「入力長 n のとき時間 $T(n)$ 以内で計算できる」という形で測る (空間計算量も同様)
- 多項式時間 (**P** や **BPP**)　≒　現実的な手間での計算
- 多項式空間 (**PSPACE**)　指数個の調べ尽しができる
- **NP**　指数個の候補中の「存在」判定型の問題
- しかし本当に $\mathbf{PSPACE} \neq \mathbf{P}$ かは未解決 (不可能性の証明は難しい)
- 帰着により問題の難しさを比べる
- ○○完全　＝　「○○なうちで最難」の問題
- 多様なジャンルの○○完全問題
  見た目は違えど困難さの核心は同じ