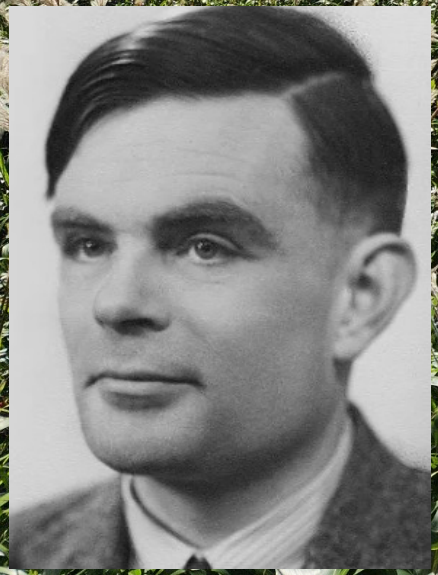
計算複雑性理論入門

Turingマシンから量子コンピュータまで



計算複雑性理論入門
Turingマシンから量子コンピュータまで

Part 1

計算可能性理論と計算複雑性理論

Part 2

複雑性クラスの基本階層

Part 3

チューリングマシンの拡大と計算複雑性

計算複雑性理論入門
Turingマシンから量子コンピュータまで

Part 1　計算可能性理論と計算複雑性理論

- はじめに
- 計算可能性理論とChurch-Turing Thesis
- 計算複雑性理論とは何か

Part 2　複雑性クラスの基本階層

- 「時間計算量」と「空間計算量」
- NP 問題
- NP-完全問題

計算複雑性理論入門
Turingマシンから量子コンピュータまで

Part 3 チューリングマシンの拡大と計算複雑性

- 非決定性チューリングマシンとNPクラス
- 確率的チューリングマシンとBPPクラス
- 量子チューリングマシンとBQPクラス
- 主要な計算複雑性クラスと対応するチューリングマシン
- Church-Turing Thesisの変化と「量子情報理論」

Part 1

計算可能性理論と計算複雑性理論

- はじめに
- 計算可能性理論とChurch-Turing Thesis
- 計算複雑性理論とは何か

はじめに

はじめに

今回のセミナーは、前回のセミナーと基本的な議論は連続しています。

それは、コンピュータとAI技術の次の未来を展望するうえで、人間の二つの基本的能力である「言語能力」と「数学的能力」の「統合」の課題が重要だという議論です。

そういう立場から、前回のセミナー「コンピュータと数学 -- TuringからVoevodskyまで」では、コンピュータにはそもそも「数学的能力」が備わっているのではという問題意識を述べてきました。

そこでは、コンピュータの持つ数学的能力の(再)発見の突破口となった「計算 ＝ 証明 ＝ プログラミング」という認識の成立の場所である「型の理論」を取り上げ、実際にコンピュータによる証明の例を紹介してきました。

コンピュータと数学

2024年8月12日　2024年9月2日　MaruyamaFujio

https://www.marulabo.net/docs/computer-math/

目次

https://www.marulabo.net/docs/computer-math/

今回のセミナー「計算複雑性理論入門　-- Turingマシンから量子コンピュータまで」では、こうした論点をさらに深める上で、計算科学の代表的な数学理論である「計算複雑性理論」に注目しようと思います。

「不完全性定理」と「計算可能性理論」

少し歴史を遡ってみましょう。

現実にコンピュータが登場するはるか以前の1930年代に、Gödelの「不完全性定理」の強い衝撃の中で、「数学的に計算できるものとは何か？」という問いの中から、Gödel - Church - Turing らによって、「計算可能性理論」が生まれました。

「計算可能性理論」は、計算可能なものと計算不可能なものとを区別する基準を数学的に定義することに成功します。それは、ゲーデルの「不完全性定理」の背後にある構造の理解を可能とする「計算不可能性理論」と言ってもいいものです。

そのエッセンスは、「計算可能な計算はすべて、ある帰納的チューリングマシンによって実行される」という「チャーチ＝チューリングのテーゼ」です。

彼らが切り開いたのは、数学的能力をはじめとして人間の知性と認識能力自身の限界を数学的分析の対象とするという、これまでの数学にはなかった新しい探究の道でした。彼らが始めた「計算可能性理論」が「計算複雑性理論」の第一世代だと考えていいと思います。

計算複雑性理論の登場

ただ、それで計算可能なものの境界が明確になったかというと、必ずしもそうではありませんでした。「計算可能性理論」では理論的には「計算可能」とされる計算でも、実際には計算できそうもないものがあることは、数学的には知られていました。

1950年代にはコンピュータが登場します。1960年代に入ってその利用は飛躍的に拡大を始めます。そうした背景の中で、1970年代になって、「計算複雑性理論」が登場します。

それは、先に述べた「理論的には計算可能だが、現実には手に負えない」領域も視野に入れて、計算可能なものの世界の内部により実際的でより現実的な分類、簡単に計算可能な問題から計算が難しい問題の階層を新たに導入しようという理論です。こうして導入された分類を、「計算複雑性」といいます。

よく混同されるのですが、それはフラクタルやカオス等を対象とした、いわゆる「複雑系の理論」とは異なるものです。

今、考えるべきこと

では、なぜ「生成AI」が大ブレイクしている今、1930年代や1970年代に起源を持つ「計算可能性理論」や「計算複雑性理論」また、1980年代に起源を持つ「型の理論」といった「古い理論」に注目するのでしょう？

それは、冒頭に述べた「コンピュータとAI技術の次の未来を展望する」という問題意識から発するものです。

私たちは、「機械の知能は、どこまで発展しうるのか？」という問題を考える必要があります。

重要なことは、その問題は、「我々人間の認識は、どこまで発展しうるのか？」という問題と同時に提起されているということです。

楽観論者も悲観論者も入り混じっているのですが、我々がそうした問題の前に立たされていることは、多くの人が、感じていることだと思います。

なぜ今「計算複雑性理論」なのか？

「計算複雑性理論」は、人間の数学的な認識能力の拡大とその限界を「現実的」に境界付けようという理論なのですが、その理論は、基本的に、「チャーチ＝チューリング・テーゼ」に基づいて計算という人間の数学的能力に「チューリング・マシン」という機械的なモデルを与えることで展開します。

それは、人間が可能な計算の現実的な限界の理論であると同時に、そのモデルである機械が可能な計算の限界の理論でもあるのです。

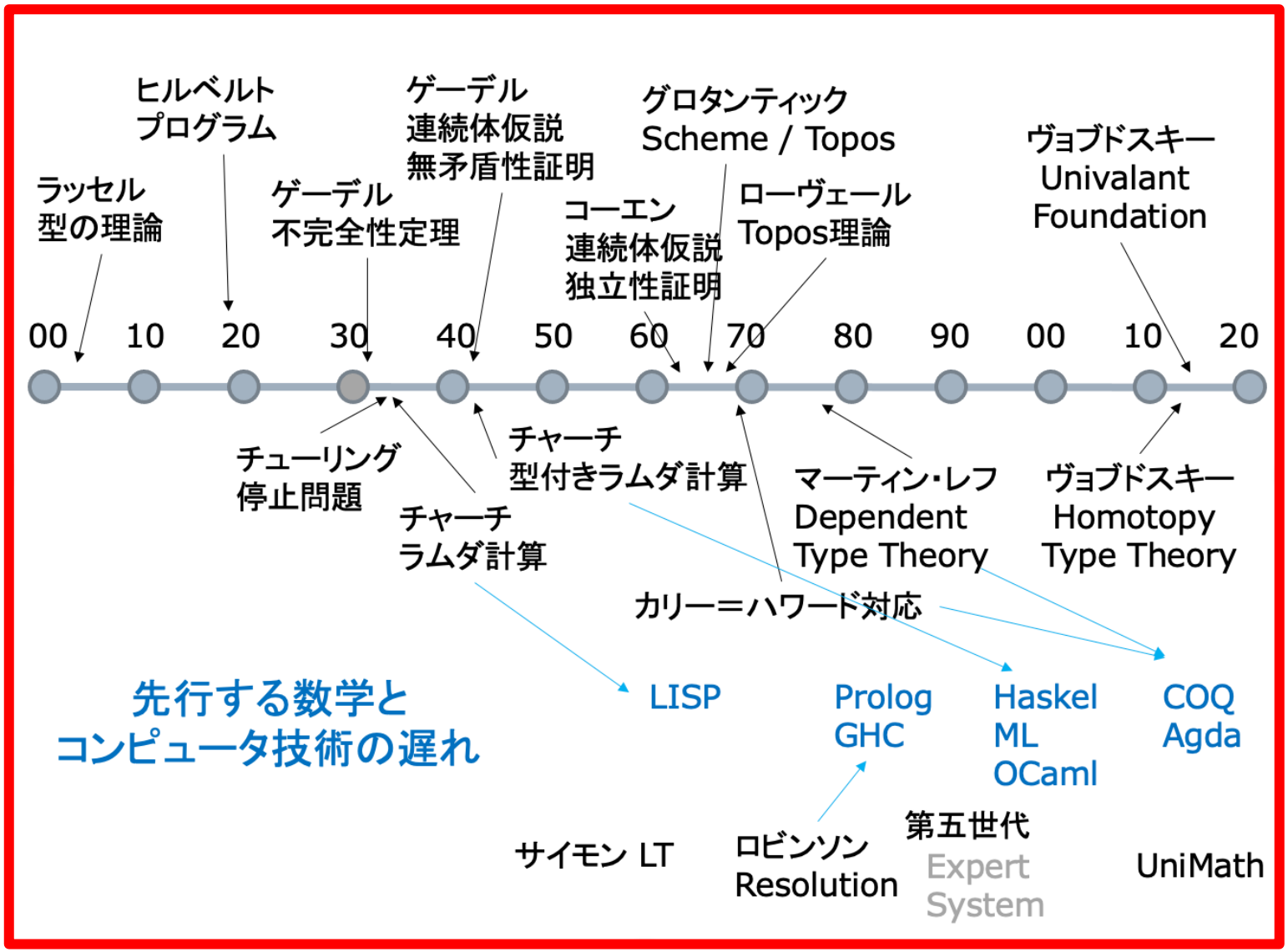
それは、人間の認識の限界の理論であると同時に、機械の認識の限界の理論でもあるのです。

「計算複雑性理論」が示すそうした視点は、我々が今考えるべき、機械と人間の認識能力の発展の可能性について、また、機械と人間の未来での関係について、多くの示唆を与えると僕は考えています。

「計算複雑性理論」を学ぼう

「計算複雑性理論」は、けっして古い理論ではありません。それは、「人工知能論」の基本的な数学理論として、十分に有効だと思います。

この間の経験に照らせば、先行した数学的理論が、コンピュータの世界で技術的な応用を獲得するまで、数十年の時間を要することは、珍しいことではありません。


ラッセル
型の理論
ヒルベルト
プログラム
ゲーデル
不完全性定理
ゲーデル
連続体仮説
無矛盾性証明
コーエン
連続体仮説
独立性証明
グロタンティック
Scheme / Topos
ローヴェール
Topos理論
ヴョブドスキー
Univalant
Foundation
00
10
20
30
40
50
60
70
80
90
00
10
20
チューリング
停止問題
チャーチ
ラムダ計算
チャーチ
型付きラムダ計算
マーティン・レフ
Dependent
Type Theory
ヴョブドスキー
Homotopy
Type Theory
カリー＝ハワード対応
先行する数学と
コンピュータ技術の遅れ
LISP
Prolog
GHC
Haskel
ML
OCaml
COQ
Agda
サイモン LT
ロビンソン
Resolution
第五世代
Expert
System
UniMath

今回のセミナーでは、前半で「多項式時間と指数関数的時間」「P問題ととNP問題」「NP完全問題」といった「計算複雑性理論」の基本的な概念を紹介しようと思います。

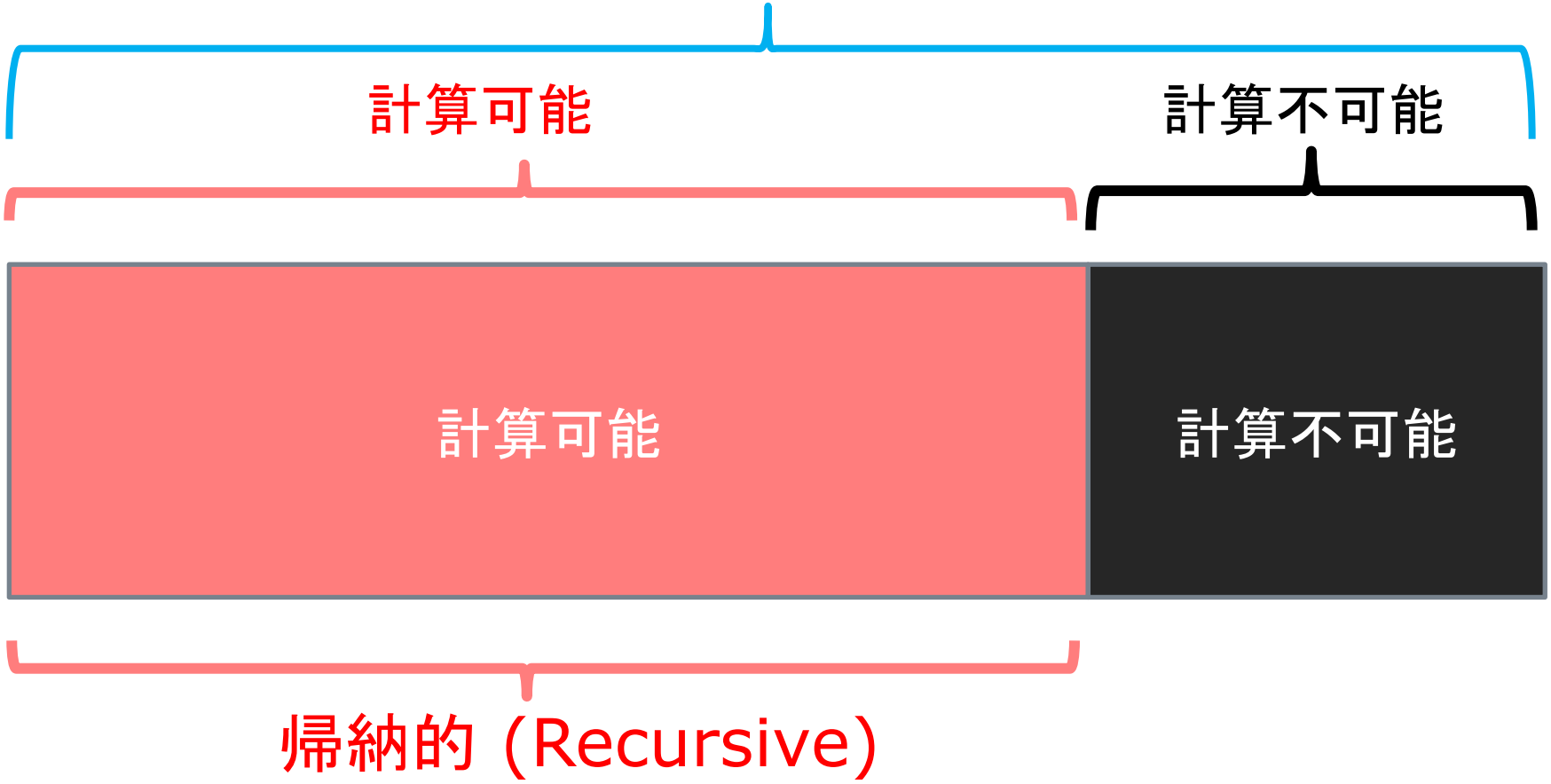
セミナーの後半では、チューリング・マシンの拡大に伴って、対応する複雑性のクラスが拡大し、「拡大されたチャーチ＝チューリング・テーゼ」のもとで、そうした複雑性クラスの拡大が、人間＝機械の能力の拡大として解釈されうることを見ていきたいと思います。

こうしてコンテキストのもとで、「量子コンピュータ」の位置付けを明確にすることができると思います。


計算可能性理論とChurch-Turing Thesis

計算可能性理論

-- Church-Turing Thesis --



計算可能性理論と
チャーチ・チューリングのテーゼ

今回のセミナーの目標は、コンピュータと数学の関係を、「計算複雑性理論」から考えることです。

今回のセッションでは、コンピュータの登場以前に成立し、後の「計算複雑性理論」をはじめとして現代の計算科学の母胎となった「計算可能性理論」と、そのエッセンスの定式化である「チャーチ＝チューリングのテーゼ」を取り上げたいと思います。

「チャーチ・チューリングのテーゼ」は、計算という人間の数学的能力にチューリング・マシンという機械的なモデルを与えるものです。それは、逆にいうと、機械的モデル自身が数学的能力を持つということ表現していると、僕は解釈しています。

すこし議論を先取りすると、チューリング・マシンの機能を拡大することで、拡大されたチューリング・マシンの能力に対応する、複雑性のクラスを構成することができます。その複雑性のクラスは、我々人間の数学的な認識能力の「現実的」な限界を特徴づけるものとして解釈可能です。

「複雑性理論」は、人間の数学的な認識能力の拡大とその限界を「現実的」に境界付けようという理論なのですが、その理論展開に、チューリング・マシンの拡大を利用することができます。

興味深いことは、それは人間だけの限界ではなく、機械の認識の限界でもあるということです。その意味で、「複雑性理論」は、「人工知能論」の最も基礎的な土台を提供する理論だと、僕は考えています。

ゲーデルの不完全性定理

“On Formally Undecidable Propositions of Principia Mathematica and Related Systems”

Kurt Gödel 英訳 by Bernard Meltzer

https://goo.gl/PMMqYr

ヒルベルト・プログラムと「有限の立場」

1920年、ヒルベルトは、全ての数学に確実な基礎を提供するという壮大な計画を提出する。その主要な内容は、

- 数学は、形式言語で、明確なルールに従って記述されること。
- 数学的に正しい命題は、全て、この形式的システムの中で証明されること。
- この数学の形式化が矛盾を含まないことが証明されること。
  その無矛盾性の証明は、有限の数学的対象に対する「有限の立場」からの証明であるべきこと。
- 「実在的な対象」について、「理想的な対象」（例えば、「非可算集合」のような）を用いて行われた証明は、それらを用いなくても証明されるべきこと。
- 全ての数学的命題の真または偽を決定する方法が存在すること。

1930年　ゲーデルの不完全性定理 (1)

ゲーデルの不完全性定理とは、次のような定理である。

「ある形式的体系Sを作ったとしよう。その体系では、初等数論を定義でき、矛盾がないとする。その時、Sに属する命題Gで、Sの中では証明も反証もできないような命題Gが存在する。」

例えば、次のような命題Gを考えてみよう。

G:「Gは証明できない。」

この命題Gが証明できたとする。それは、Gは証明できないことを証明したことになる。

この命題Gの否定が証明できたとする。それはGは証明できること、すなわち、Gは証明できないことを証明したことになる。

1930年　ゲーデルの不完全性定理 (2)

それだけだと、そんなものかと思われるかもしれないが、ゲーデルの不完全性定理が、数学の世界に強い衝撃を与えたのは、不完全性定理の次の形である。

Fを、初等数論を含む、無矛盾の形式的体系としよう。この時、この形式的体系Fの無矛盾性は、この体系Fの中では証明出来ない。

この結果は、ヒルベルト・プログラムが、実行不可能であることを意味していた。

「計算とはなにか？」「証明とは何か？」より基本的な「問い」の始まり

ゲーデルの不完全性定理は、「証明不可能性」「形式的体系の無矛盾性」についてのショッキングな定理なのだが、いかなる計算、いかなる証明が「可能」であるかについては、あまり、多くのことを我々に教えてはくれないのだ。

歴史的に見ると、不完全性定理は、証明あるいは計算の性質についての探求の中で生まれた「最後の結論」ではないのだ。事実は、その逆である。1930年代に、この分野は、不完全性定理に強烈な刺激を受けて、「証明とは何か？」「計算とは何か？」という基本的な「問い」に突き進むことで発展する。

様々な「計算可能性」へのアプローチと その同値性の認識

ゲーデルの結果を受けて、「計算可能性」についての探求が一斉に始まる。

- ゲーデル＝エルブランの「帰納関数論」
- チャーチの「ラムダ・カリキュラス」
- チューリング＝ポストの「チューリング・マシン」

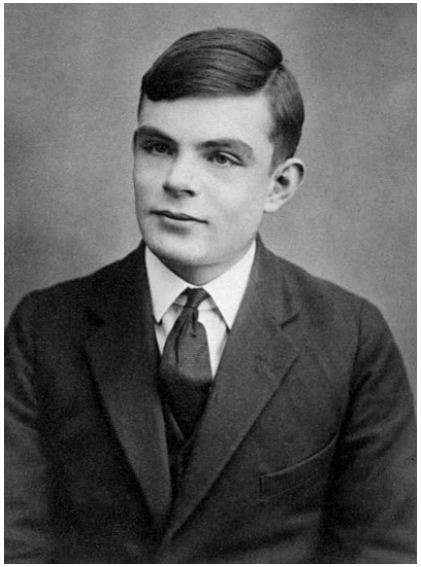
が続々と登場する。

この時代の白眉は、「計算可能性」についての見かけは全く異なるこれらのアプローチが（もちろん、それは、ゲーデルの結果を再現できるものだ）、次々と、「同値」であることが証明されていくことだ！

様々な「計算可能性」へのアプローチと
その同値性の認識

Kurt Gödel
1906-1978

Alonzo Church
1903-1995



Alan Turing
1912-1954

帰納関数論

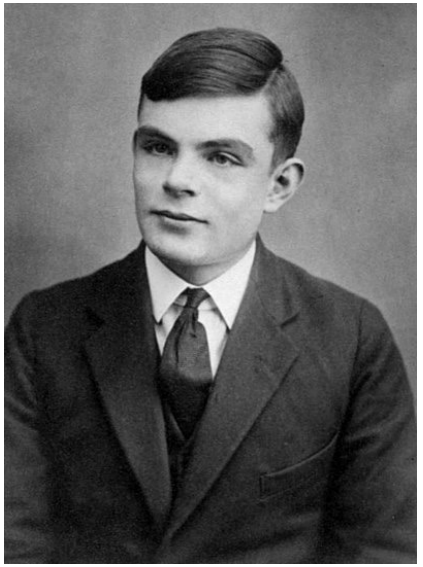
ラムダ計算

チューリング
マシン

様々な「計算可能性」へのアプローチと
その同値性の認識

Kurt Gödel
1906-1978

Alonzo Church
1903-1995



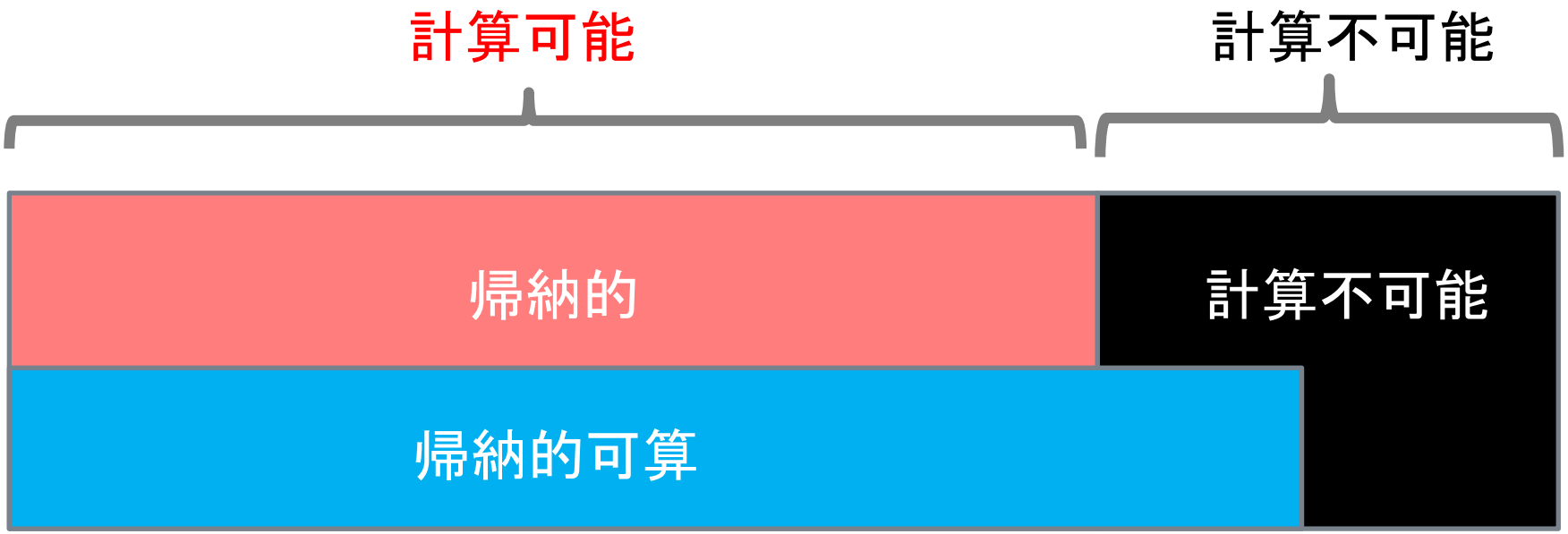
Alan Turing
1912-1954

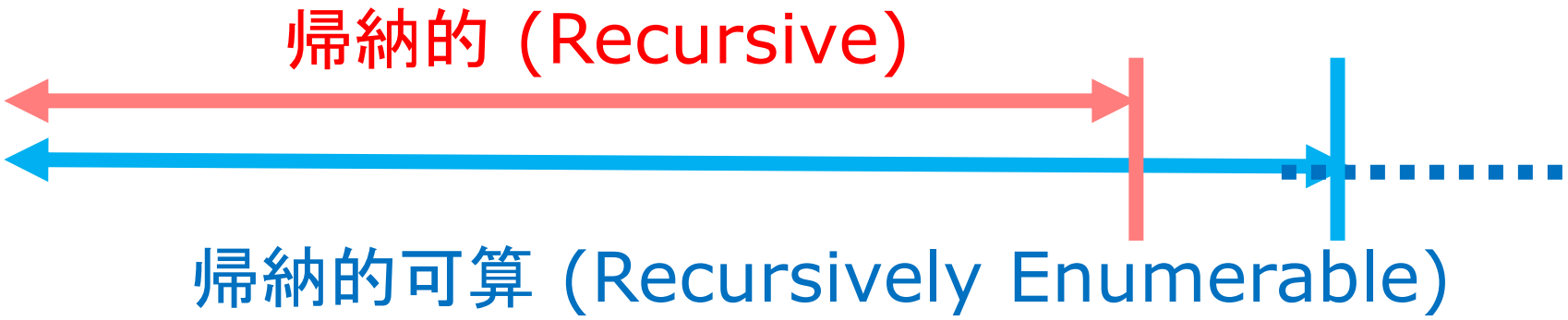
帰納関数論 ＝ ラムダ計算 ＝ チューリングマシン

計算可能性の定式化
-- 帰納的と帰納的可算 --

計算可能性の定式化

-- 帰納的と帰納的可算 --





帰納的可算
recursively enumerable

ある自然数の集合Sに対して、チューリング・マシン Mが存在して、次の条件を満たす時、Sは「帰納的可算 recursively enumerable 」であるという。

- n ∈ S なら、Mは停止して 1 を出力する。

この定義でポイントが一つある。
Mは、Sに属するメンバーに対しては、停止して1を出力するのだが、Sに属さないメンバーに対しては、何も語っていない。
recursively enumerableなチューリング・マシン Mでは、全ての入力に対してMが停止するとは限らないということである。

Sに属さないメンバーに対しても、チューリング・マシン Mが停止することを要求するのが、次の「帰納的 recursive」の定義である。

帰納的
recursive

自然数の集合Sは、次の条件を満たすチューリング・マシンMが存在する時、「帰納的 recursiveな集合」と呼ばれる。
与えられた自然数nについて、

- $n \in S$ なら、Mは停止して 1 を出力する。
- $n \notin S$ なら、Mは停止して 0 を出力する。

自然数から自然数への関数 f は、次の条件を満たすチューリング・マシンが存在する時、「帰納的な関数」と呼ばれる。

- 全ての入力 n に対して、Mは停止して、出力 f(n)を返す。

計算可能な集合 / 関数は、帰納的であるという主張が、次のChurch-Turing Thesisである。

チャーチ＝チューリングのテーゼ

1940年代には、今日、「チャーチ＝チューリングのテーゼ」と呼ばれる「計算可能性」についての認識は、確立されることになる。

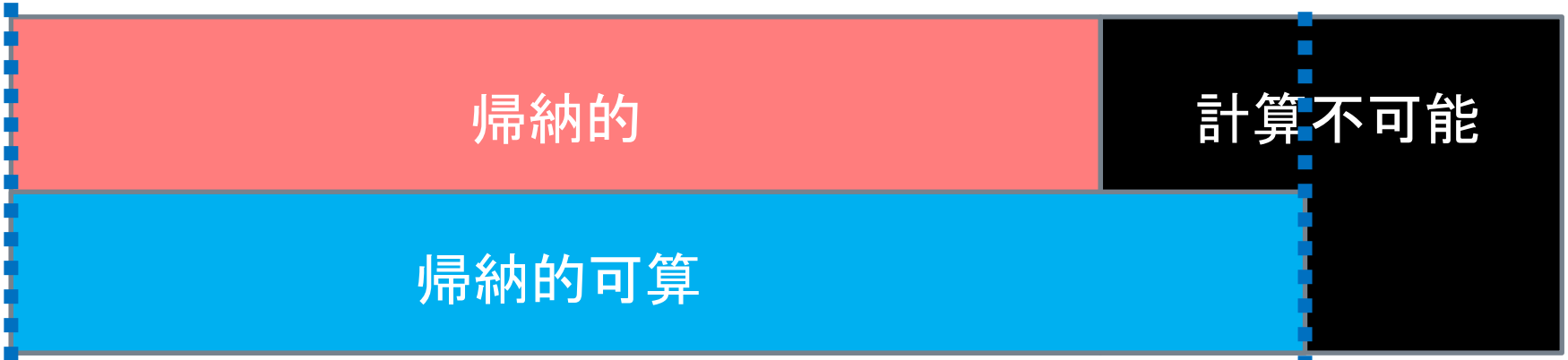
「チャーチ＝チューリングのテーゼ」というのは、次のような提案である。これは帰納関数論的特徴づけである。

すべての実効的に計算可能な関数（実効的に決定可能な述語）は、一般帰納的である。

チューリング・マシンによって特徴づければ、次のようになる。
計算可能な計算はすべて、ある帰納的Turingマシンによって実行される。

帰納的可算

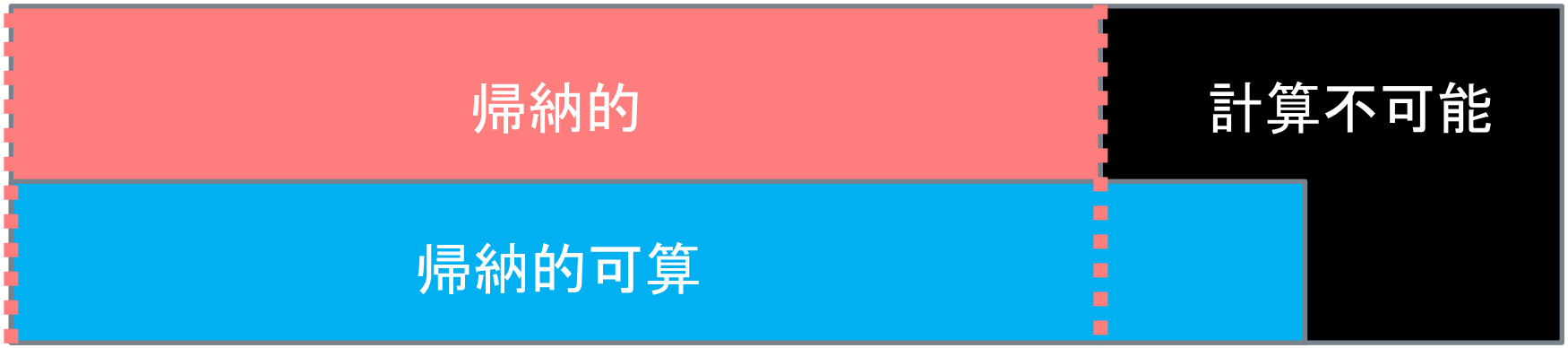
自然数の集合Sは、次の条件を満たす
チューリング・マシンMが存在する時、
「帰納的可算」と呼ばれる



- n ∈ S なら、Mは停止して 1 を出力する。

帰納的

自然数の集合Sは、次の条件を満たす
チューリング・マシンMが存在する時、
「帰納的」と呼ばれる



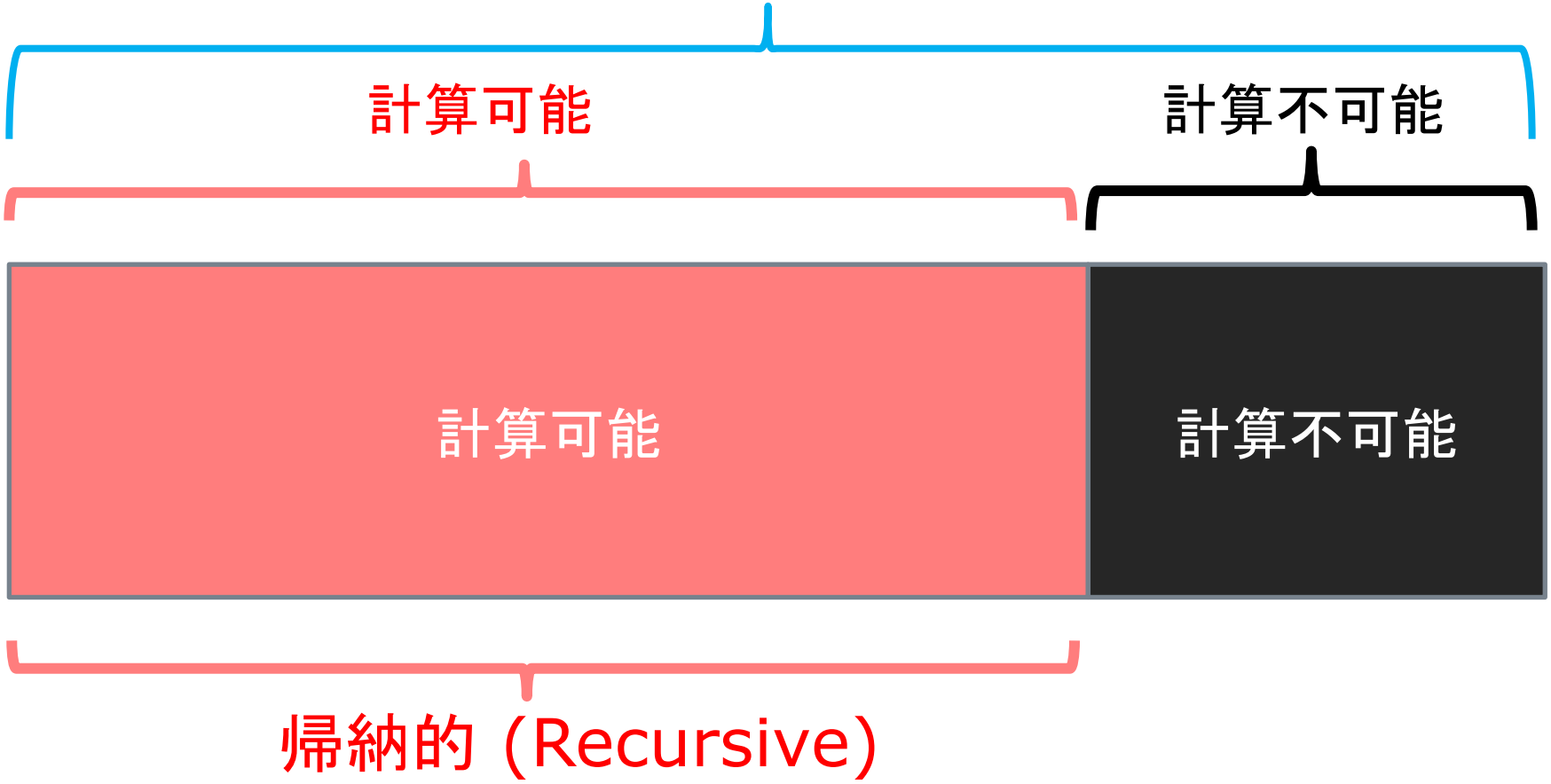
- $n \in S$ なら、Mは停止して 1 を出力する。
- $n \notin S$ なら、Mは停止して 0 を出力する。

Church-Turing Thesis

計算可能なものは帰納的である

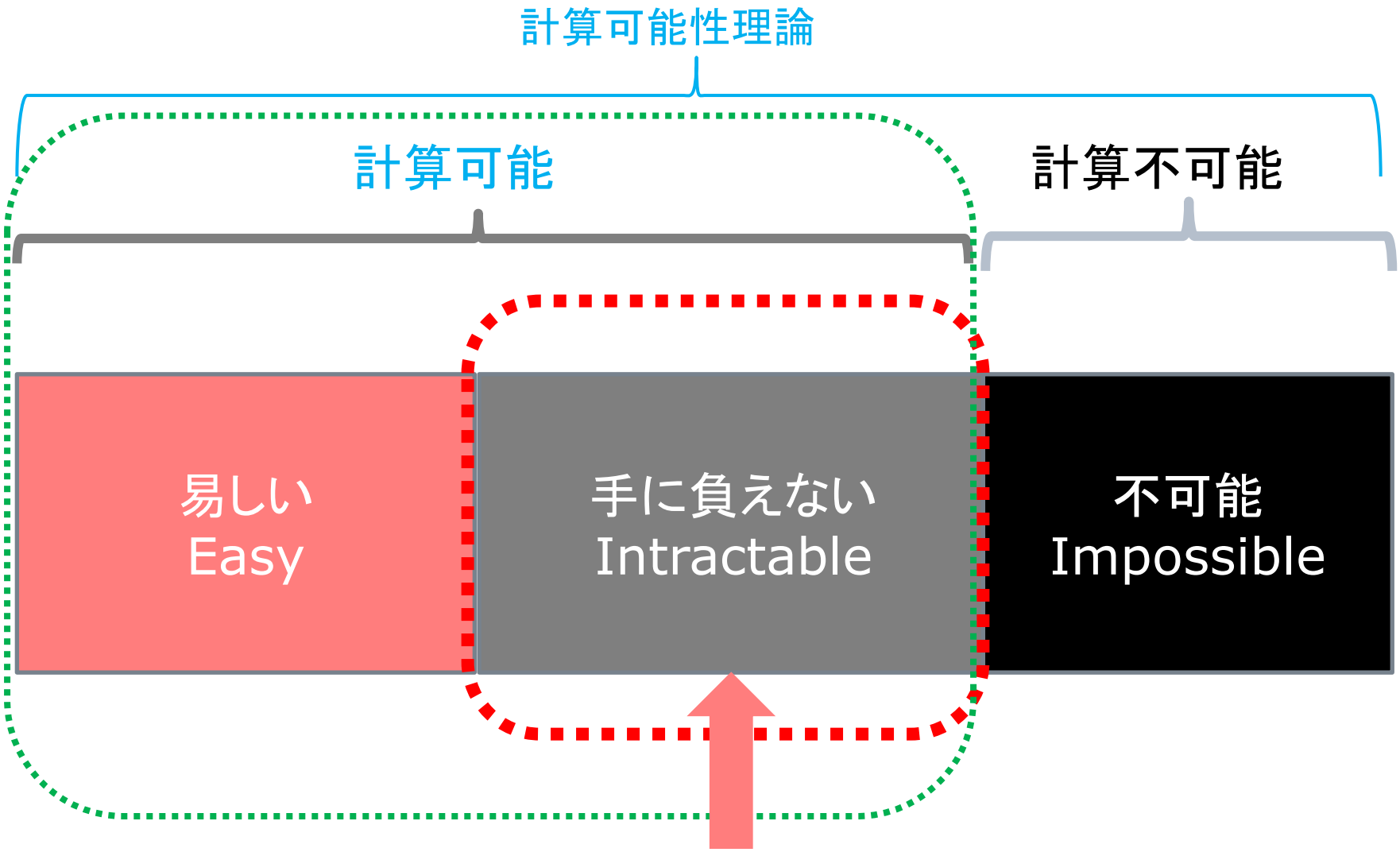
計算可能性理論

-- Church-Turing Thesis --




計算複雑性理論とは何か

計算複雑性理論



計算可能だが計算が手に負えないものがある

計算複雑性理論とは何か

「計算可能性理論」は、計算可能なものと計算不可能なものとを区別する基準を数学的に定義することに成功します。それは、ゲーデルの「不完全性定理」の背後にある構造の理解を可能とする「計算不可能性理論」と言っていいものです。

そのエッセンスは、「計算可能な計算はすべて、ある帰納的チューリングマシンによって実行される」という「チャーチ＝チューリングのテーゼ」です。

ただ、それで計算可能なものの境界が明確になったかというと、必ずしもそうではありませんでした。

といいますのも、「計算可能性理論」では、理論的には計算可能とされるものの中には、実際には計算できそうもないものが含まれていることに気づき始めたからです。

「理論的には計算可能だが、実際には計算できそうもない」とはどういうことなのでしょう？

それは、計算をどう実行すればいいかはわかるのだが、実際に計算を始めると手に負えなくなり、実際には、計算結果を得ることができない計算です。

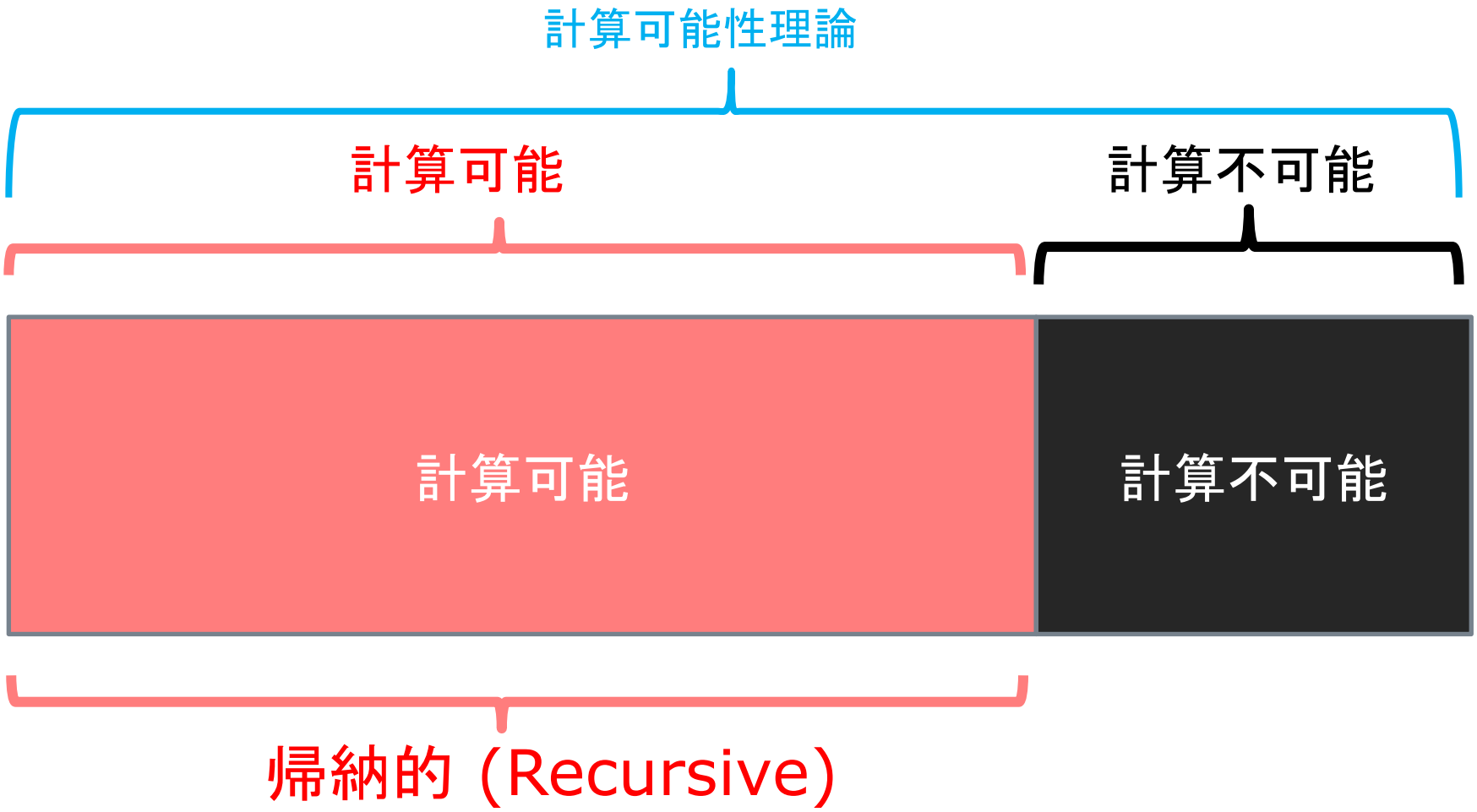
セッションの後半では、こうした「手に負えない計算」の例を、いくつか紹介しようと思います。

「計算複雑性理論」は、まさにこの「理論的には計算可能だが、現実には手に負えない」領域も視野に入れて、計算可能なものの世界の内部に、より実際的でより現実的な分類を新たに導入しようという理論です。

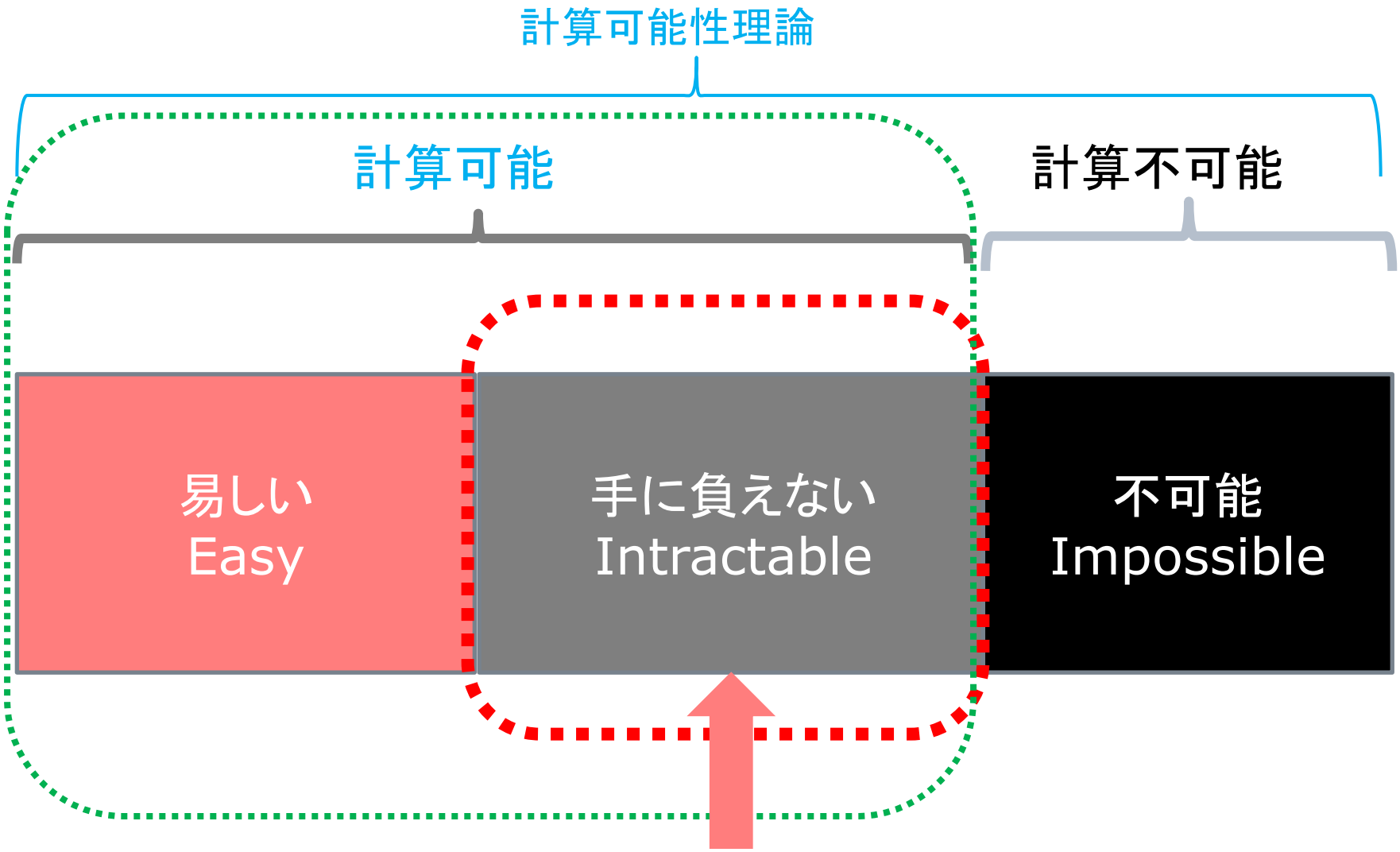
こうして導入された分類を、「計算複雑性」といいます。

計算可能性理論と、
計算複雑性理論とを比較する

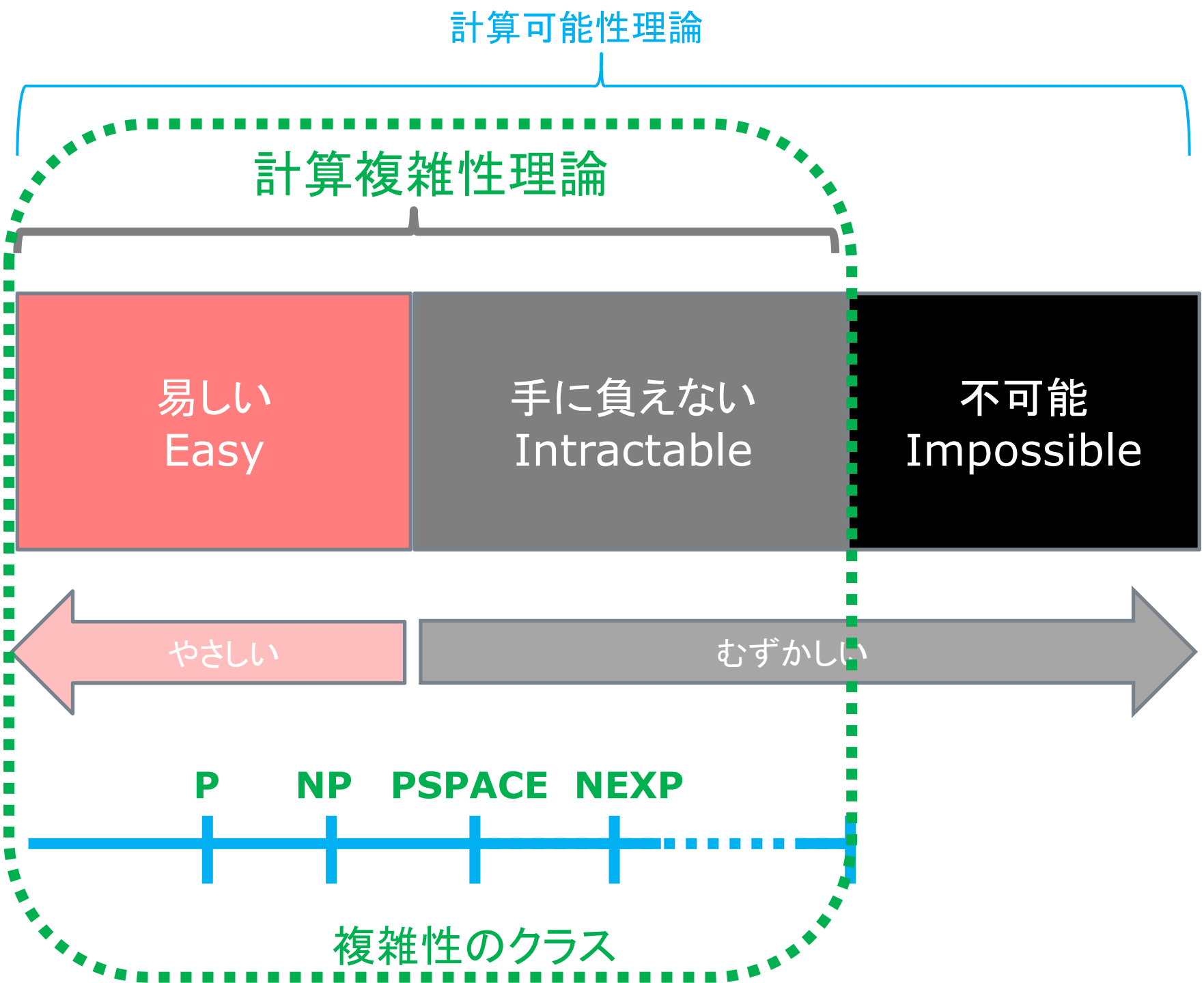
計算可能性理論



計算複雑性理論



計算可能だが計算が手に負えないものがある


計算可能性理論
計算複雑性理論
易しい
Easy
手に負えない
Intractable
不可能
Impossible
やさしい
むずかしい
P
NP
PSPACE
NEXP
複雑性のクラス

計算可能だが
計算が手に負えないものの例

計算可能だが、計算が手に負えないものの例

ここでは、計算可能だが、計算が手に負えないものの例として、次の例を挙げています。

- 一般帰納関数として計算可能なアッカーマン関数
- Nashの「誰にも破れない暗号」
- Busy Beaver 問題

一般帰納関数の計算可能性
アッカーマン関数

次のような関数を考えます。

$$\mathrm{Ack}(m,n) = \begin{cases} n+1, & \text{if } m = 0 \\ \mathrm{Ack}(m-1,1), & \text{if } n = 0 \\ \mathrm{Ack}(m-1,\mathrm{Ack}(m,n-1)), & \text{otherwise} \end{cases}$$

この関数は、一般帰納関数として定義されているので、チャーチ＝チューリングのテーゼの条件を満たしています。

ただ、m,nの値が、少しでも大きくなると、この関数の値は、爆発的に増大して、普通のコンピュータでは、事実上、計算が不可能になります。

次ページに、Ack(m,n)の値を示します。

A(*m*, *n*) の値

*m**n*	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2=2+(n+3)-3$
2	3	5	7	9	11	$2n+3=2\times(n+3)-3$
3	5	13	29	61	125	$2^{n+3}-3$
4	13	65533	$2^{65536}-3$	$2^{2^{65536}}-3$	A(3, A(4, 3)) = $2^{2^{2^{65536}}}-3$	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3\text{ twos}}-3$
5	65533	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{65536\text{ twos}}-3$	A(4, A(5, 1))	A(4, A(5, 2))	A(4, A(5, 3))	$A(4,A(5,n-1))$
6	A(5, 1)	A(5, A(6, 0))	A(5, A(6, 1))	A(5, *A*(6, 2))	A(5, A(6, 3))	$A(5,A(6,n-1))$

$$A(5,1)=\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{65536\text{ twos}}-3$$

誰にも破れない暗号　Nash

1950年代に入ると、時代の先鋭な知性たちは、今日の複雑性理論の原型とも言える認識に到達し始める。

スコット・アーロンソンは、複雑性理論の到達点を概観した論文 "N = NP?" https://goo.gl/TZUkgh の冒頭に、ジョン・ナッシュの 1955年のNSA（スノーデンがいた、あのSNAだ！）宛の手紙をあげている。かつては「機密文書」とされていて、いつかの時点で情報公開されたものらしい。

誰にも破れない暗号を作るためには、指数関数的な計算が必要な暗号を作ればいいことを述べている。また、そうすれば、それまでの職人技的な暗号破り（多分、チューリングやファインマンがしていたこと）は「過去」のものになると、明確に自覚している！

NashからNSAへの天紙

現時点での、私の一般的な予想は次のようなものです。:

ほとんどすべての十分に複雑なタイプの暗号化にとって、特に、鍵の異なる部分によって与えられた命令が、相互に複雑に相互作用して、それが暗号化の最終的な決定において影響を与えている場合、鍵の計算の平均的な長さは、鍵の長さ、すなわち、鍵の持つ情報の内容に関して、指数関数的に増加します。

もし、この予想が正しいと仮定すれば、この一般的な予想の重要性を理解するのは容易です。それは、実際的には誰も破れない暗号を設計することを、極めて簡単に実行できることを意味します。暗号が洗練されたものになるにつれて、熟練したチームなどによる暗号破りのゲームは、「過去」のものになっていくはずです。

この予想の性質は、たとえ特別な種類の暗号をとっても、私にはそれを証明できないようなものです。また、私は、それが証明されることも期待していません。」

---

https://www.nsa.gov/news-features/declassified-documents/nash-letters/assets/files/nash_letters1.pdf

原文は、こちら。

Now my general conjecture is as follows: For almost all sufficiently complex types of enciphering, especially where the ~~information~~ instructions given by different portions of the key interact complexly with each other in the determination of their ultimate effects on the ~~cipher~~ enciphering, the ~~number of~~ key computation length increases exponentially with the length of the key, or in other words, with the information content of the key.

The significance of this general conjecture, assuming its truth, is easy to see. It means that it is quite feasible to design ciphers that are effectively unbreakable. As ciphers become more sophisticated, the game of cipher breaking by skilled teams, etc., should become a thing of the past.

The nature of this conjecture is such that I cannot prove it, even for a special type of cipher. Nor do I expect it to be proven. But this

Busy Beaver 問題

Turingマシンの「停止性問題」は、あるTuringマシンが答えを出して停止するのか否かを問題にするのだが、この「忙しいビーバー」問題は、Turingマシンが停止するまでのステップ数の最大値を求めるという問題である。

Turingマシンは、内部状態を持つ。状態Sでヘッドのあるマス目の値が、'0'か'1'かに応じて、状態をSからS'に変更し、'0'か'1'を書き込んで、ヘッドを右（R）あるいは左(L)に移動する。こうした命令の集まりが、Turingマシンの「プログラム」になる。

Turingマシンのプログラムの例を示す。この表の、例えば、最初の State Aの行で、On '0' の欄の 'B1R' は、「状態Aで、ヘッドが'0'の上にあるなら、状態をBに変えて、1を書き込んで、ヘッドを右（R）に移動する」という命令を表す。

State	on 0	on 1	on 0			on 1		
			Print	Move	Goto	Print	Move	Goto
A	B1R	D0L	1	right	B	0	left	D
B	C1R	F0R	1	right	C	0	right	F
C	C1L	A1L	1	left	C	1	left	A
D	E0L	H1L	0	left	E	1	left	H
E	A1L	B0R	1	left	A	0	right	B

On '1' の欄の 'D0L' は、「状態Aで、ヘッドが'1'の上にあるなら、状態をDに変えて、0を書き込んで、ヘッドを左（L）に移動する」という命令を表している。

この表のプログラムで定義されるTuringマシンは、A, B, C, D, E, F の6つの状態を持っている。このサンプルの、プログラム自体は、単純である。もちろん、マシンは停止状態HALTを持たないといけないのだが、ここでは、HALT状態Hは、数にカウントしていない。Dの行に出てくる 'H1L' が停止の命令になっている。）

「忙しいビーバー」問題は、状態の数がN個のTuringマシンを全部考えて、全て0で埋まっているテープから初めて、一番長いステップで停止するTuringマシンとそのステップ数を求めよという問題である。その最大値をBB(N)で表すことにすると、次の結果が知られている。

BB(1) = 1.
BB(2) = 6.
BB(3) = 21
BB(4) = 107

実は、まだ、誰も、BB(5)の値を知らない。

BB(5) ≥ 47,176,870

というのはわかっている。（誰かが、状態の数が5つで、47,176,870ステップで停止するTuringマシンを作ったということ。ただ、それが「最大」だということはわかっていない。）

そんなものかと思われるかもしれないが、雲行きが怪しくなるのは、BB(6)あたりから。

BB(6) ≥ 7.412 x (10の36,534乗）

だという。36,534桁の数字になる。先のBaezのblogに、「具体的に」この数字が紹介されている。延々と数字が続く。3万6千桁以上の数字が。

よくそんなことわかるなと思われるかもしれないが、それは、先にも述べたように、実際にそういう6状態のTuringマシンを構成して見せたからだ。実は、先に紹介した、6個の状態と12個の命令からなるTuringマシンが、そういう動き方をするのだ。

「プログラム自体は、単純である。」と先に書いたのだが、正しくないかもしれない。このプログラムは、えんえんと走り続け、そして、最後に停止する。（停止すること自体、不思議な気がする）

先に、BB(6)が、36,534桁の数字より大きいという話をした。
2014年に、Wythagoras（怪しい名前だ）は、BB(7)について、次のような評価を示した。右側の数は、とても巨大な数だが、「有限」な数字である。

$$\mathrm{BB}(7) > 10^{10^{10^{10^{10^7}}}}$$

今年（2016年）の4月に、Adam Yedidia と Scott Aaronson は、BB(7910)が、普通の集合論の枠組みの中では決定できないことを証明した。

"A Relatively Small Turing Machine Whose Behavior Is Independent of Set Theory"
http://arxiv.org/pdf/1605.04343v1.pdf (Scottの次のblogが参考になる。
http://www.scottaaronson.com/blog/?p=2725)

それ以来、BB(N)が計算不能になる、より小さいNの探索が、インターネット上で活発に行われ、この "logic hack" で、つい二週間ほど前に、BB(1919)も計算不能であることが示されたのである！

確かに、僕が生きているうちは、BB(10)の値も決まらないだろうと思って、ほぼ、間違いないので、BB(7910) とか BB(1919)とかは、とんでもなく大きな数である。ただ、これらの数字は「有限」で、しかも、BB(N)自体は、とても明確に定義されたものである。

要するに、「有限」の値を持つのだが、どういう数学的な手段を持ちいても、我々が知り得ない数字があるということ。さらに、その限界が、7910とか1919といった、比較的小さな数字で特徴付けられるというのが、驚きである。

Part 2

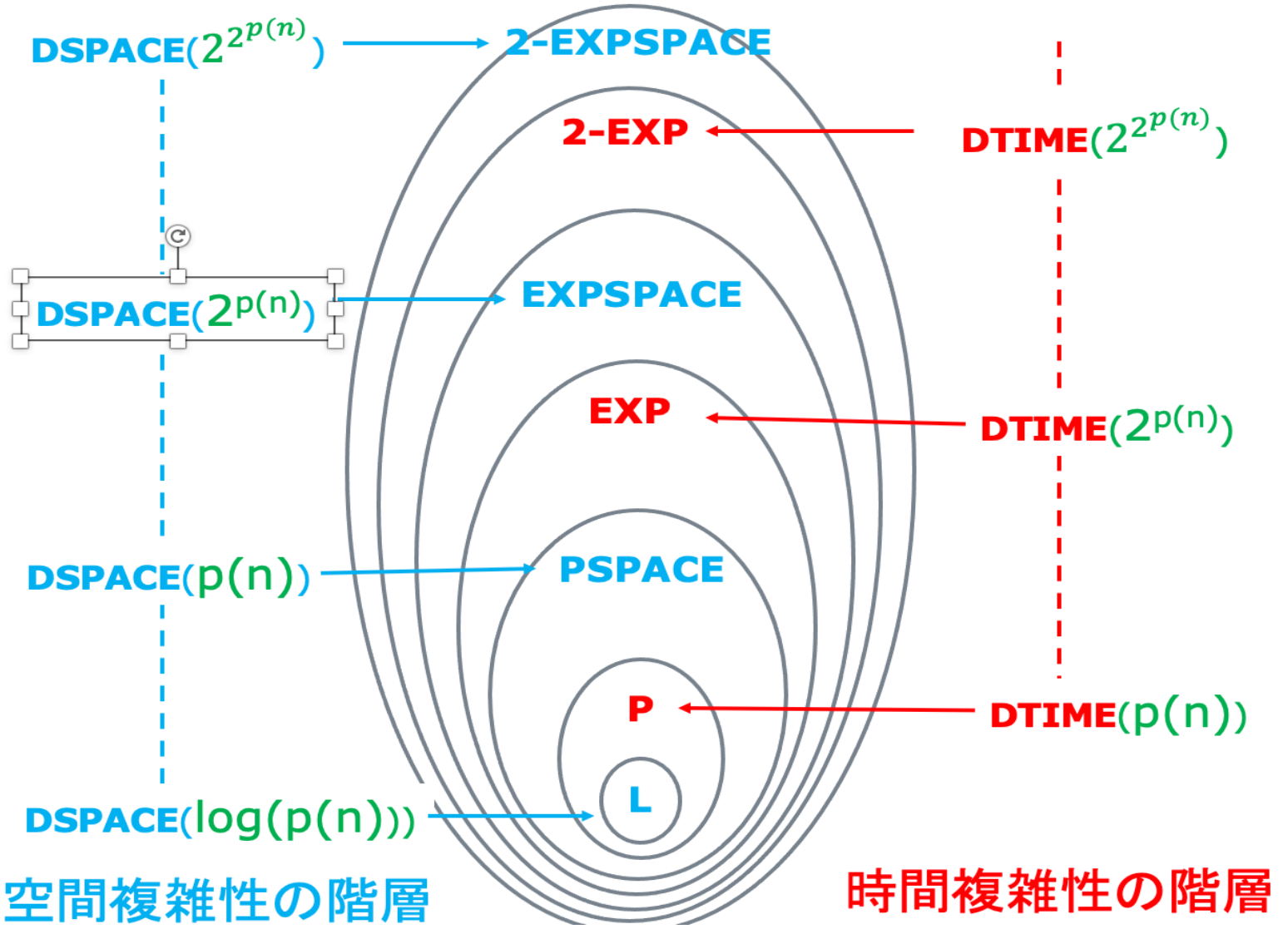
複雑性クラスの基本階層

- 「時間計算量」と「空間計算量」
- NP 問題
- NP-完全問題


「時間計算量」と「空間計算量」

計算複雑性の基本的階層

空間複雑性と時間複雑性



複雑性理論の背景を考える

複雑さへのアプローチ

私たちの認識は単純なものから複雑なものに進みます。世界は複雑なものであふれているので、我々の認識もどんどん複雑になっていきます。ただ、こうした私たちの認識の前進がいつまでも続くとは限りません。私たちの認識は、しばしば、複雑さの前に立ちすくみます。複雑なものを理解するのには、長い時間と膨大な知識の集積が必要であることは、科学の歴史を見れば分かります。

僕は、対象の複雑さと認識の複雑さは同じものだと考えています。そうした認識は「複雑さ」そのものを対象とした認識が可能だということを意味します。現代の計算科学のもっとも活発な研究分野はの一つは「複雑性理論」なのですが、そうした理論が登場したことの意味は、そう考えれば、よく理解できると思います。

今回のセミナーでは、様々なチューリングマシンの拡大と複雑性のクラスの対応を紹介したいと思います。次のようなチューリングマシンを取りあげます。

- 決定性チューリングマシン
- 非決定性チューリングマシン
- 確率性チューリングマシン
- 量子チューリングマシン

先に、「複雑なものを理解するのには、長い時間と膨大な知識の集積が必要である」と書きましたが、このチューリングマシンによる複雑さのモデルでは、複雑なものを理解するのに必要な「長い時間」がチューリングマシンの「計算時間」に、「膨大な知識の集積」がチューリングマシンの「テープの長さ」に対応すると思って構いません。

「そんな単純化で大丈夫か？」

確かに。

もちろん、科学の歴史をチューリングマシンに喩えようと思っているわけではありません。

ただ、こうした単純化（「抽象化」は、単純化に他なりません）された切り口が、「認識の有限性と対象の無限性」「古典論と量子論」「決定論と確率論」といった認識の大きな問題にアプローチする一つの有効な手段を提供するという話ができればと思っています。

今回のセッションでは、複雑性の階層について考えます。

無限も有限も複雑であること

無限について
無限は複雑である

無限は、古来から、多くの哲学的・数学的な関心を引き付けてきました。

時間や空間が連続的で無限であると仮定すると引き起こされる「ゼノンの逆理（アキレスはかめに追いつけない）」は有名ですね。

ユークリッドは、素数が無限にあることを証明したのですが、その証明では、慎重に、「無限」という言葉を使うことを避けました。アリストテレスは、もっと踏み込んで、無限について語っています。彼は、無限を「実無限」と「可能的無限」に分けました。

数学では、連続と無限の問題は、ニュートンやライブニッツが微積分法を始めた時にも、「無限大」「無限小」の概念を巡って大きな議論が巻き起こりました。現代の数学の出発点の一つであるカントールの集合論は、無限についての数学といってもいいのですが、ここでも多くの論争がありました。

20世紀になっても、超限解析(Non-Standard Analysis)の登場や、連続体仮説の独立性証明といったトピックでも、議論は尽きません。

物理学についても、現代の物理学の歴史を「連続的なものとの「格闘」という視点から振り返った数理物理学者のJohn Baezは、次のように述べています。

「すべての主要な物理学の理論は、時空が連続的なものであるという想定から生まれる数学的問題に対する挑戦であることを、この一連の投稿でみてきた。連続的なものは、無限によって、我々をおびやかす！　これらの無限は、これらの理論から予測を行う我々の能力を脅かすのだろうか？　あるいは、これらの無限は、こうした理論を正確な形で定式化する我々の能力さえも脅かしているのであろうか？」

"Struggles with the Continuum"

https://johncarlosbaez.wordpress.com/2016/09/08/struggles-with-the-continuum-part-1/

こうした無限と連続をめぐる議論は、とても興味深いものです。

と言いますのは、今回フォーカスしたいのは、複雑性理論での無限と有限の捉え方だからです。

誤解を恐れず単純化していうと、複雑性理論は無限を直接の対象とはしません。複雑性理論がまず関心を持つのは、有限の捉え方です。

ただ、有限という概念は、決して単純ではないのです。

有限は複雑である

有限の中に、どのように量的階層を持ち込むか？
なぜ、「多項式」 vs. 「指数関数式」に注目するのか？

「有限」の「限界」を考える

有限と無限は対の概念です。英語でも、"finite"と "infinite" と対になっていますね。ただ、有限と無限の対比の仕方は、日本語と英語とでは、少し違います。日本語では、有限は「限界があるもの」で、無限は「限界がないもの」ですが、英語では、「無限」は「有限でないもの」です。日本語の方が、定義が踏み込んでいます。

ここでは、「限界がある」という有限の定義で、有限には、どんな限界があるのかを考えてみましょう。

有限の身近なモデルとして、自然数を考えましょう。ある具体的な自然数nを考えて、nは有限であると考えることは自然なことのように思われます。

具体的な自然数を有限のモデルだとして、今度は、有限の「限界」を、どんな自然数nに対しても、n ＜ N なるNが存在することだと考えましょう。有限な自然数には、大きさで超えられない壁があって、それが有限の「限界」だと考えるということです。

ただ、この有限の「限界」の特徴づけは、いくつか奇妙なところがあります。

限界Nが存在して、それが自然数だとします。その時、N＋1も自然数で、N ＜ N＋1 ですので、限界はNではなく、N＋1になります。これは矛盾です。ですから限界Nが、自然数だと考えることはできません。これは、素直に考えれば、最大の自然数は存在しないということです。また、それは、具体的に与えられた自然数nより大きい自然数は「無限」に存在することを意味します。

先に、ユークリッドが素数が無限に存在することの証明を、無限という言葉を使わずに行ったと書きましたが、彼が行ったのは、次のような証明です。基本的には、上と同じ論理です。

最大の素数pが存在するとする。p以下の全ての素数の積に1を加えた数をPとする。Pは、全ての素数で割っても、1が余って割り切れない。よってPは素数で、p ＜ P で、pは最大の素数ではない。これは矛盾。よって、最大の素数は存在しない。

有限の「限界」1
自然数の「上限」ω を考える

色々奇妙なところもあるのですが、「具体的に与えられた自然数＝有限」というモデルを維持することにこだわりましょう。このモデルは、自然なものにおもえますので。

それでは、この有限のモデルで、その「限界」は、どう考えればいいのでしょう？

次のようなωを考えます（先に見たように、ωは自然数ではありません）。

「全ての自然数nについて、n ＜ ω」

そう考えれば、「自然数＝有限」というモデルの「有限の限界」は、ωで与えられることになります。

この解釈も、いささか奇妙なものです。ωは、無限個の自然数より「大きく」、かつ自然数ではないのですから、このモデルでは「有限」のものとは考えにくいのです。この解釈によれば、有限なものの「限界」は、「有限ではない」ものによって規定されることになります。n < ω という式は、「有限は、無限より小さいもの」とも解釈できます。"finite" で "infinite" を定義するのとは、まったく逆です。

「そもそも、n < ω というようなωが存在することが具体的にイメージできない」

そうかもしれません。ただ、イメージだけなら、ωをイメージするいい方法があります。

0以外の自然数nを1/nに対応させます。1は1/1に、2は1/2に、3は1/3に、4は1/4 .... nは1/n.の点に移ります。

このnと1/nとの一対一の対応は強力です。無限に広がった無限個の自然数nは、全て0と1の間の1/nの形の有理数に閉じ込められます。nが無限に大きくなるにつれ、1/nは0に近づくのですが、この対応づけで、ωは0に対応します。

0の周辺には、無数の1/nの形の有理数が集まっています。こういう点を「集積点」と呼びます。

少し慣れれば、nと1/nの対応を意識しなくても、有限な自然数全体の「集積点」としてωをイメージすることができるようになると思います。

こうした構成は、カントールの「順序数」の構成の最初の一歩をなぞったものです。数学的には、ωのような無限を「可算無限」と呼びます。それについては、また別に説明したいと思います。

有限の「限界」 2
我々が具体的に構成できるもの「帰納的可算」

ただ、有限なものの限界をωで抑えるのは、ある意味では本質的ですが、あまりに広すぎます。

「具体的に与えられた自然数 n」という言い方をしてきましたが、我々は、少なくとも、可算無限に属する自然数の集合に関して言えば、その集合を全て定義し計算できるとは限らないのです（もちろん、有限個の自然数からなる集合なら、それを計算することはできるのですが）。我々が具体的に構成できるωの部分集合を「帰納的可算」と言います。

計算可能であるという条件を付ければ、有限の「限界」の第二の候補として、「可算無限」ではなくこの「帰納的可算」が浮上します。この「帰納的可算」を与えるのが、実は、チューリングマシンなのです。この認識は、とても大事なことです。我々がチューリングマシンを学ぶべき最大の理由の一つは、ここにあります。

有限の「限界」 3
多項式で表現できる「有限」

ただ、悲しいことに、こうして有限の「限界」を狭めていっても、「帰納的可算」も広すぎて、我々の手には余ります。我々が現実的・具体的に計算できる有限な量の特徴を把握するのには、有限の「限界」の別の特徴づけが必要になります。

それには、どんなnに対しても、xが大きくなると、x^nより、n^xの方がずっと大きくなるという「多項式関数 ＜ 指数関数」という関係を利用します。x^{10000}は大きな数ですが、xが大きくなると、2^xの方がずっと大きくなります。

もっと一般的には、多項式関数 ＜ 指数関数 ＜ 二重指数関数 ＜ 三重指数関数 ＜ … 。p(n) ＜ $2^{p(n)} < 2^{2^{p(n)}}$ ＜ …という関係を複雑性の階層として利用できます

多項式時間へ

「多項式関数 ＜ 指数関数 ＜ 二重指数関数 ＜ 三重指数関数 ＜ …」という関係は、先に見た「n ＜ ω」のような「有限 ＜ 無限」という関係ではありません。どちらも有限です。ただ、「多項式関数＝有限で人間にとって扱いやすいもの」で「指数関数＝有限だが、人間には扱いにくいもの」を代表していると考えればいいと思います。

こうして、有限の「限界」の第三の候補として、「チューリングマシンで定義できて（「帰納的可算」に属するということです）、かつ、その計算が「多項式」で表される時間で 可能なもの」という「限界」が、複雑性の最初の階層として経験的に生まれることになります。

時間・空間リソースによる複雑性の階層

時間・空間リソースによる複雑性の階層

先に、次のように書きました。「複雑なものを理解するのには、長い時間と膨大な知識の集積が必要である」と。

それは、人間の認識の歴史について語ったものでしたが、具体的な計算の複雑さの尺度にも、この「長い時間と膨大な知識の集積」に対応する量があります。

それは、「その計算にどれぐらいの時間が必要か？」という時間の尺度と、「その計算にどのくらいのメモリー空間が必要か？」という空間の尺度です。

前者を「時間計算量」、後者を「空間計算量」と言います。

ある時間T_0では解けないけれど、T_0 より大きな時間 T_1では解ける 問題、あるいは、ある量のメモリー空間S_0では解けないけれど、S_0 より大きなメモリー空間 S_1では解ける 問題を考えて、多くの計算リソース（時間、空間）を要する問題を「複雑な問題」と考えることは自然なことです。

また、多くの計算リソースを要する問題を「むずかしい問題」、より少ない計算リソースを要する問題を「やさしい問題」と考えることができます。

こうした「時間複雑性」や「空間複雑性」といった階層が存在することは、ある問題をできるだけ効率的に、短いステップで少ないメモリーで書こうとするプログラマーが、日常的に意識していることです。

「時間計算量」と「空間計算量」を チューリングマシンで定義する

複雑性理論で、その計算モデルにチューリングマシンを用いる大きな理由の一つは、複雑性の尺度である「時間計算量」と「空間計算量」に、キチンとした定義を与えることができるからです。

「時間計算量」はある計算に必要なチューリングマシンの命令の「ステップ数」に、「空間計算量」はある計算に必要なチューリングマシンの「テープの長さ」だと考えることができます。

DTIME(f(x))とDSPACE(f(x))

「決定性チューリングマシン」で、O(f(x))で表される時間計算量の複雑性クラスをDTIME(f(x))で表し、同じく空間計算量の複雑性クラスをDSPACE(f(x))で表します。

「決定性チューリングマシン」がどのようなものかは、第二部で紹介します。

DTIME(f(x)),DSPACE(f(x))のままでもいいのですが、代表的な複雑性クラスについては、次のような別名を付けます。ここでp(n)は、入力の大きさnについての多項式です。

DTIME(f(x))とDSPACE(f(x))の別名

時間複雑性については、

P=DTIME(p(n)) ;「多項式時間」です。

EXP=DTIME($2^{p(n)}$) ;「指数関数的時間」です。

2-EXP=DTIME($2^{2^{p(n)}}$) ;「二重指数関数的時間」です。

空間複雑性については、

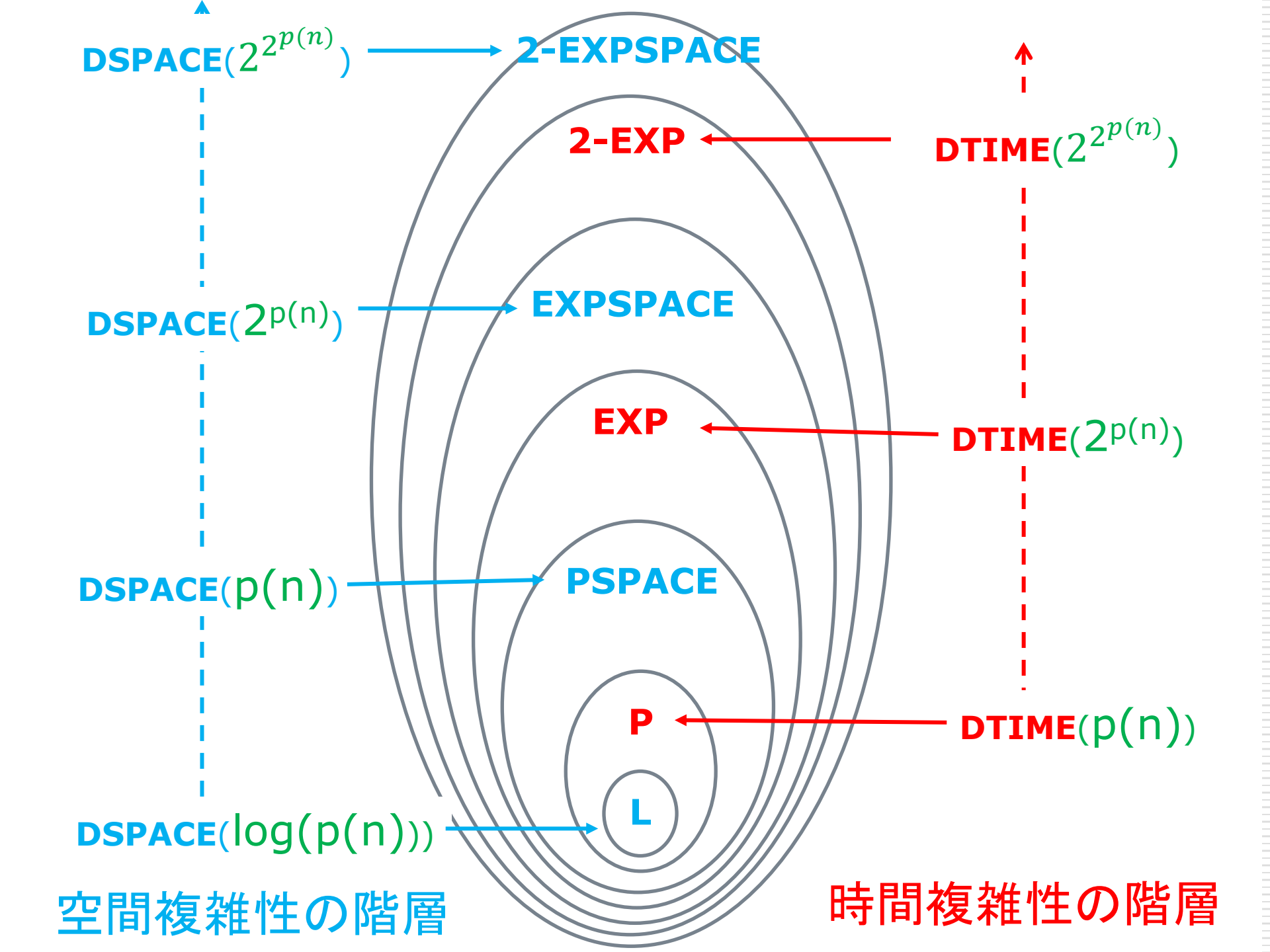
PSPACE=DSPACE(p(n)) ;「多項式空間」です。

EXPSPACE=DSPACE($2^{p(n)}$);「指数関数的空間」です。

2-EXPSPACE=DSPACE($2^{2^{p(n)}}$);「二重指数関数的空間」です。

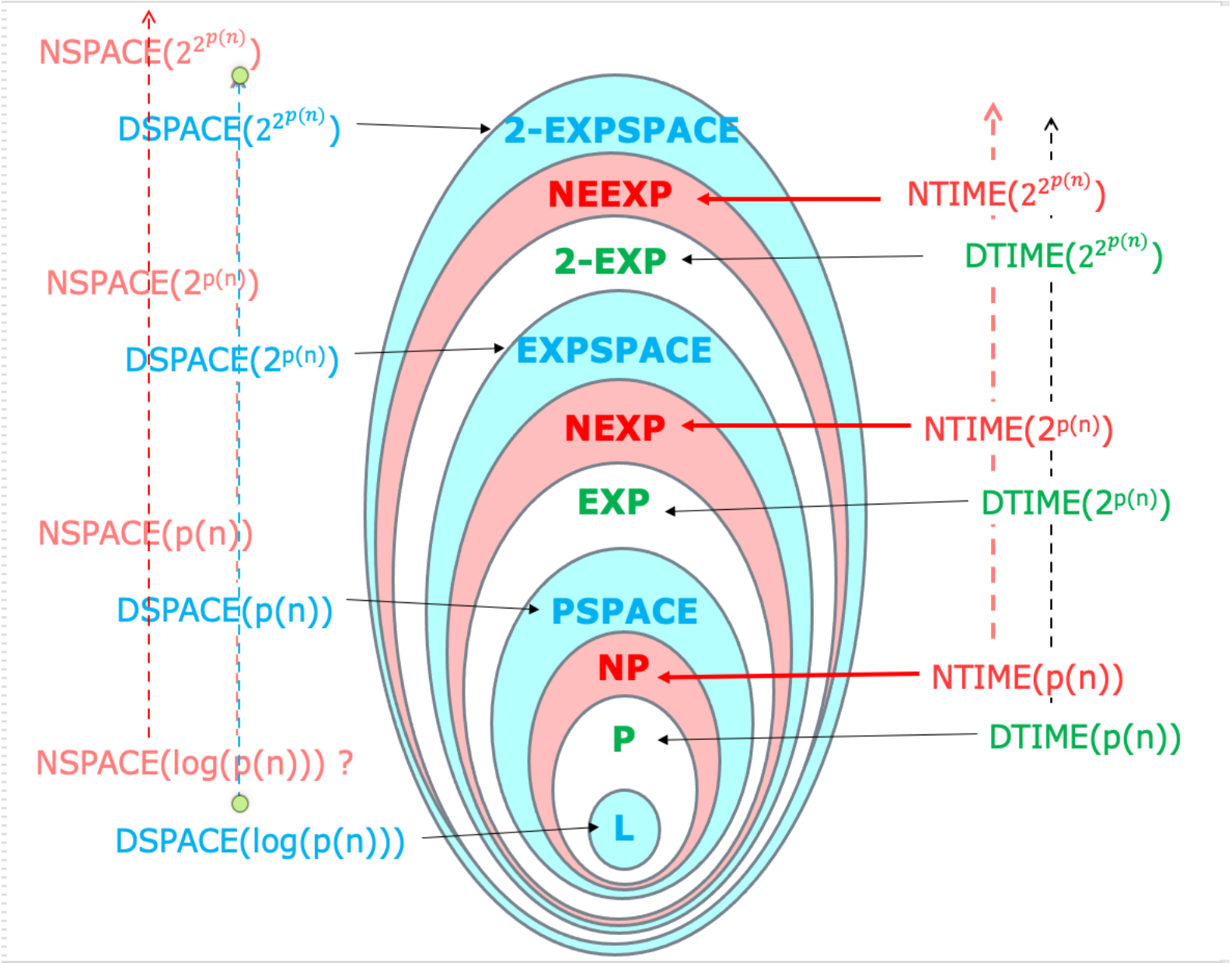
ここで、$p(n) < 2^{p(n)} < 2^{2^{p(n)}}$であることに注意してください。

これらの複雑性の関係を図にしました。


DSPACE($2^{2^{p(n)}}$)
2-EXPSPACE
2-EXP
DTIME($2^{2^{p(n)}}$)
DSPACE($2^{p(n)}$)
EXPSPACE
EXP
DTIME($2^{p(n)}$)
DSPACE(p(n))
PSPACE
P
DTIME(p(n))
L
DSPACE(log(p(n)))
空間複雑性の階層
時間複雑性の階層


NP 問題

NP問題


NSPACE($2^{2^{p(n)}}$)
DSPACE($2^{2^{p(n)}}$)
2-EXPSPACE
NEEXP
NTIME($2^{2^{p(n)}}$)
2-EXP
DTIME($2^{2^{p(n)}}$)
NSPACE($2^{p(n)}$)
DSPACE($2^{p(n)}$)
EXPSPACE
NEXP
NTIME($2^{p(n)}$)
EXP
DTIME($2^{p(n)}$)
NSPACE(p(n))
DSPACE(p(n))
PSPACE
NP
NTIME(p(n))
P
DTIME(p(n))
NSPACE(log(p(n))) ?
DSPACE(log(p(n)))
L

NPというコンセプト

歴史的には、現代の「複雑性理論」は、NPという複雑性のクラスと「NP-完全」という概念の発見によって始まります。

NP問題というのは、その証明は「むずかしい」としても、その答えが与えられた時、その答えが「正しい」ことは、多項式時間で「検証」できる時間複雑性のクラスです。

この定義の双対で、その答えが「間違っている」ことを多項式時間で「検証」できるクラスをcoNPと呼びます。

SORTには、Bubble-SortやQuick-Sortのようにさまざまのアルゴリズムが存在します。問題を NP問題として捉えると、どのアルゴリズムを採用したとしてもその並べ替えが正しいことの検証は、n個のもの全てが順番に並んでいるかの n-1回のチェックで可能です。

このn-1という値は下限です。最悪ケースを考えれば、これ以下にはできないことがわかります。それは、どのSORTアルゴリズムを採用するかには依存しません。

少しでも速いSORTアルゴリズムの開発に命をかけている人には、当たり前すぎてつまらないことのように思うかもしれませんが、n-1回のチェックで、ソートの正しさが検証できるというのは、ある意味、SORT問題の本質を捉えています。

「やさしい問題」と「難しい問題」

前に見た「時間複雑性」や「空間複雑性」の階層では、「やさしい問題」と「むずかしい問題」の区別は、必要な計算リソース（時間、空間）の量的区別に帰着します。

しかし、NPというクラスが複雑性の階層に持ち込むものは、それとは違っています。それは、ある問題に答えを与えることと、その答えを検証することとは違うものだという質的区別です。

ある問題に答えを与えるのは「むずかしい問題」だとしても、その答えが正しいかをチェックするのは、それよりも「やさしい問題」になります。数学の証明を行うのは一般に「むずかしい」のですが、その証明が正しいかを検証するのは、証明することと比べれば「やさしい」のです。

「証明は「むずかしい」が検証は「やさしい」！」

このイメージはとても大事です。

もちろん、このことは、NP問題自身が、「やさしい問題」であることを意味しません。それは、問題自体の証明と比較して、その検証が「やさしい」ことを意味しているだけです。

NP問題というクラスは、とても豊かなものです。

例えば、全ての数学的証明は、NP問題です。

なぜなら、その証明が「正しい」ことを、我々が理解できるものでなければ、証明としての意味を持たないからです。それは、その証明の各ステップを、我々が有限の時間の中で後追いでチェックできるということを意味します。ここでは、我々人間にとって有限の時間とは、指数関数ではなく多項式で表される時間だと考えています。

計算複雑性理論の始まり

計算複雑性理論の始まり
P=NP?

ある数学的命題Sの真偽を決定する「純粋に機械的手続き」は存在しないにせよ、そのSが、ある制限長さn以下の証明を持つかどうかを決定する手続きは存在する。単純に、長さn以下の証明を全て枚挙して、それが命題Sの証明になっているかをチェックすればいい。

しかし、この方法は指数関数的な時間を必要とする。P=NP?問題は、こうした問題を解く「高速」なアルゴリズムがあるかを問うている。その意味では、P=NP?問題は、ヒルベルトの問題提起の現代版だと考えることができる。

この問題が、明確に提起されたのは、1970年代の初めのことで、CookとLevinの仕事である。現在のようなスタイルで、N ＝ NP?問題を最初に定式化したのは、バークレーのStephen Cookだという。1967年のことだ。計算複雑性理論の始まりである。

ただ、こうした問題が存在することは、50年代に、ゲーデルもナッシュも気がついていた。

Gödel から von Neumannへの手紙
1956年

僕は、ナッシュのNSAへの手紙は知らなかったのだが、有名なのは、1956年にゲーデルが、フォン・ノイマンに宛てた、次の手紙である。ゲーデルが、ほとんど完全に、問題の所在を把握していることがわかる。

我々が、一階の述語論理の全ての式Fに対して、全ての自然数について、長さn（長さ＝記号の数）のFの証明があるかどうかを決定することを可能にするチューリング機械を、簡単に構築することができるのは明らかです。

機械がこれを決定するのに必要とするステップの数をψ(F, n)とし、[特定のFについて] φ(n)＝max ψ(F, n)としましょう。問題は、最適なマシンで、φ(n)がどのくらい急速に大きなものになるかです。

我々は、[nとは独立な定数Kについて] φ(n) ≥ K･n　であることを示すことができます。[φ(n)は、nの一次式より大きいということ。]　実際に、実行時間が Knに比例する（またはKn^2に比例する）マシンが実際にあったとした場合 [「論理式Fに対して長さnのFの証明があるかどうか？」という問題が、nの一次式または二次式に比例する時間で解けるなら]　、これは最大級に重要な結果をもたらすことになります。

つまり、「決定問題」の解決不可能性にもかかわらず、YesかNoかの質問に答えるタイプの問題の場合、数学者の知的努力は完全に機械に置き換えることが可能であることを、明確に示すことができるでしょう。

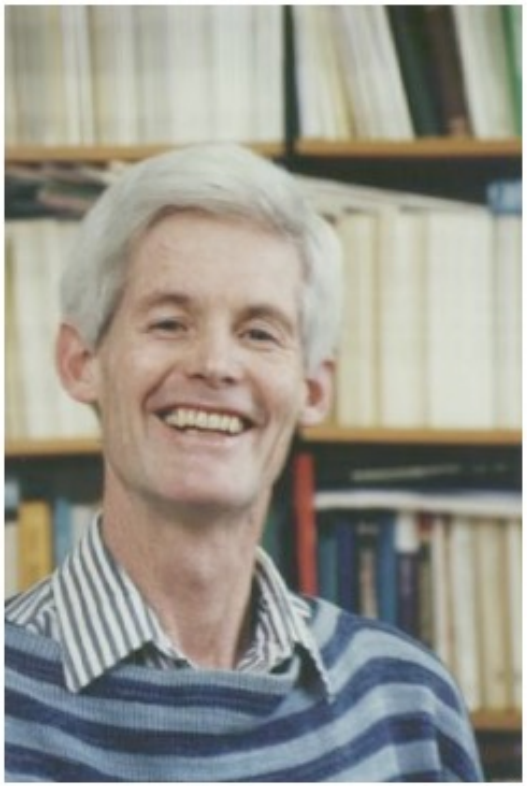
我々が　実際に行うべきことは、単純に数nを選ぶだけです。そのnが非常に大きくても、もしも、マシンが [「長さnのFの証明があるかどうか？]という質問に] "No"という結果を出した場合には、さらに問題を考える理由はなくなります。

“Gödel’s lost Letter and P=NP”

https://rjlipton.wordpress.com/the-gdel-letter/

もっとも、これらの50年代半ばの認識は、広く共有されたものではなかったかもしれない。（論文として公開されたものではなく「手紙」だった！）

Cook-Levin Theorem
SATはNP完全である

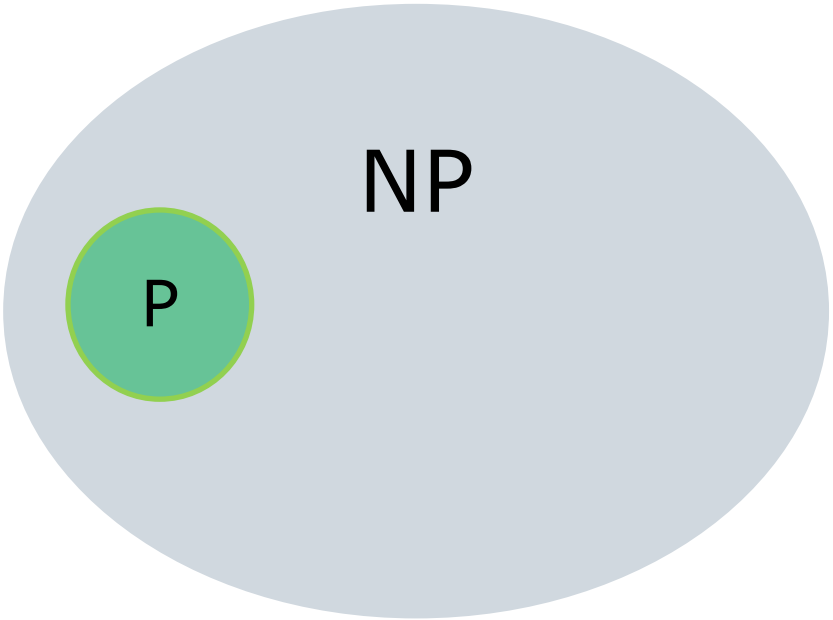


NP-完全というコンセプト

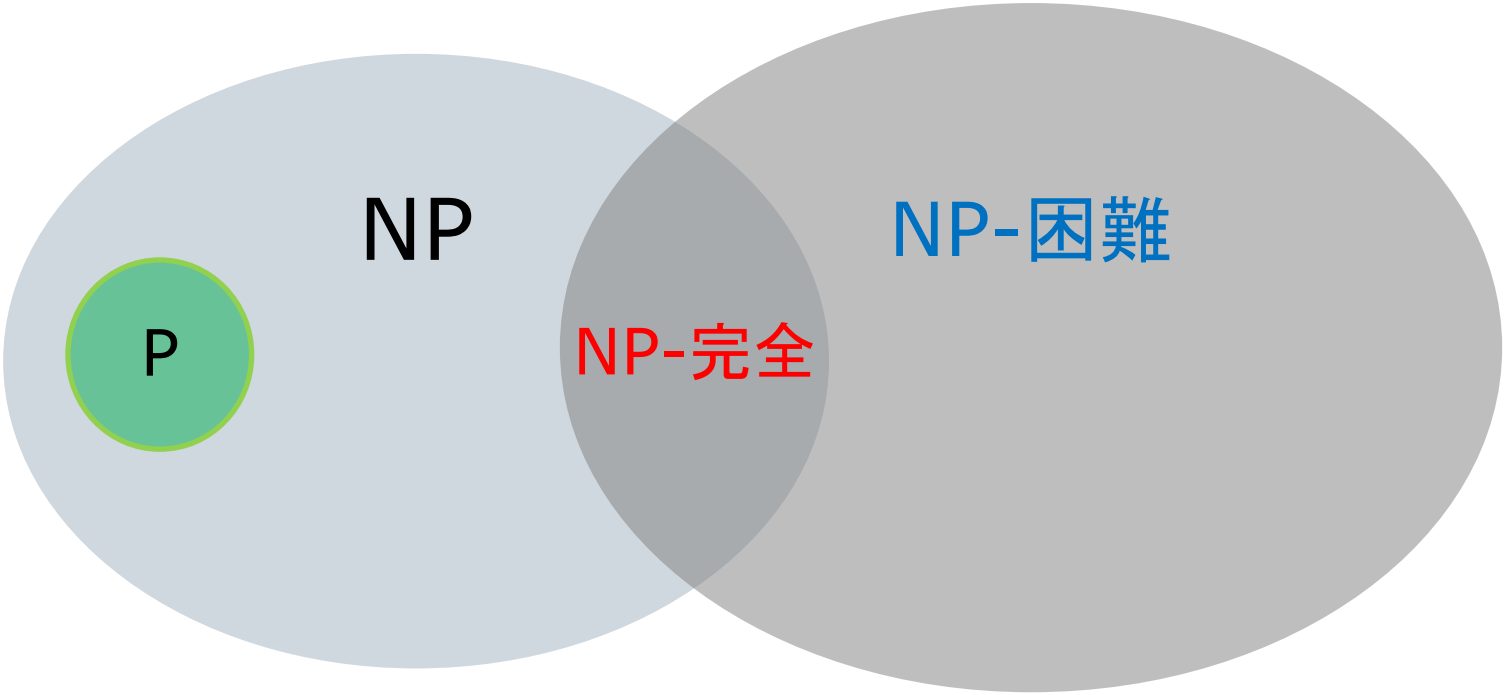
1971年、Cookは、充足可能性問題が、NP-完全であることを証明した。ここに初めて、NP-完全というコンセプトが登場する。

同じ頃、ソヴィエトの数学者Levinも同じ結果を得ている。

現代の複雑性理論は、この発見から始まる。



P:　多項式時間で解ける問題
NP: 多項式時間で解けるとは限らないが、
　　答えが正しいかは多項式時間でチェックできる問題



P:　多項式時間で解ける問題
NP: 多項式時間で解けるとは限らないが、
　　答えが正しいかは多項式時間でチェックできる問題

NP-困難: 全てのNP問題が、このクラスの問題に帰着される問題
　　（この帰着は多項式時間で行われなければいけない）
NP-完全: NP問題で、かつ、NP-困難な問題

SAT問題（充足可能性問題）

$$
\begin{aligned}
&(l_1 \vee l_2 \vee x_2) \wedge \\
&(\neg x_2 \vee l_3 \vee x_3) \wedge \\
&(\neg x_3 \vee l_4 \vee x_4) \wedge \cdots \wedge \\
&(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge \\
&(\neg x_{n-2} \vee l_{n-1} \vee l_n)
\end{aligned}
$$

3-SAT

充足可能性

全ての変数に、ある論理値T(真)あるいはF(偽)を割り当てた時、その式を真にできるような割り当てが存在する時、その式は「充足可能」という。

$x_1 \vee x_2$は、充足可能である。

$x_1 = T, x_2 = T$とすればいい。 $x_1 = T, x_2 = F$ でもいい。

$x_2 \wedge \sim x_3$は充足可能である。

$x_2 = T, x_3 = F$とすればいい。

$(x_1 \vee x_2 \vee \sim x_3) \wedge (\sim x_4 \vee x_5 \vee \sim x_6)$ は、充足可能である。

ある論理式Fが与えられた時、Fを真とする変数への真理値の割り当てが存在するか否かを決定する問題を充足可能性問題という。

N-SAT問題

ある式Aが、CNFの形をしているとする。Aを構成する節が、高々N個のリテラルしか含まない時、Aの充足問題をN-SAT問題という。

$x_1 \vee x_2$の充足問題は、$2SAT$問題である。

$x_2 \wedge \sim x_3$の充足問題は、$1SAT$問題である。

$(x_1 \vee x_2 \vee \sim x_3) \wedge (\sim x_4 \vee x_5 \vee \sim x_6)$
の充足問題は、$3SAT$問題である。

Karpの「還元」の手法

1972年、Karpは、ある問題がNP-完全であることを証明する標準的な「還元」の手法を開発し、多くの問題がNP-完全であることを示した。

"Reducibility Among Combinatorial Problems”

Richard M Karp 1972

https://people.eecs.berkeley.edu/~luca/cs172/karp.pdf

還元可能性 $\leq_p$ とNP-完全

- LとMを二つの言語（問題のこと）とする。LがMに、「還元可能」というのは、多項式時間で計算可能な関数 fが存在して、

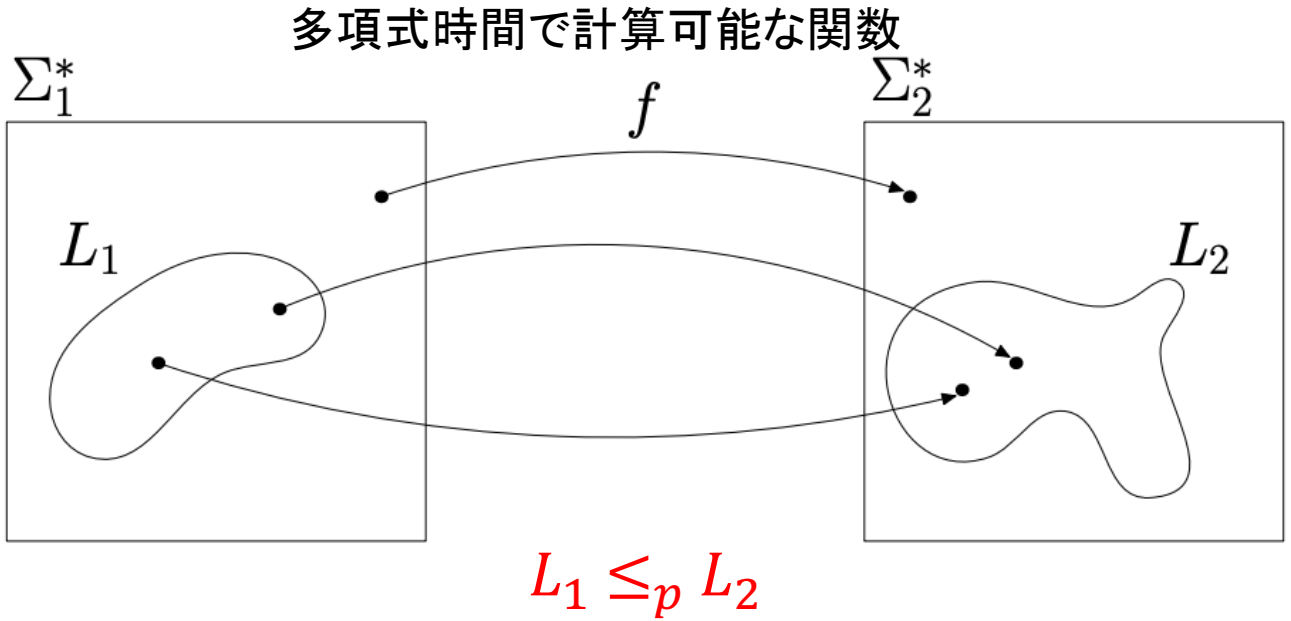
  $$x \in L \Leftrightarrow f(x) \in M$$

  が成り立つことをいう。この関係を $L \leq_p M$と表す。
- 関係 $\leq_p$は、対称的で推移的である。
  また、

  $$L \leq_p M\text{で、かつ、}M \in P \Longrightarrow L \in P$$

  $$L \leq_p M\text{で、かつ、}M \in NP \Longrightarrow L \in NP$$
- LがNP-完全であるのは、次の二つの条件が成り立つ時
  1. $L \in NP$
  2. $L' \in NP$である全てのL'について$L' \leq_p L$（LはNP-Hard）

$L_1 \leq_p L_2$



- x ∈ L1 ⇒ f(x) ∈ L2
- x ∉ L1 ⇒ f(x) ∉ L2
- f computable in polynomial time

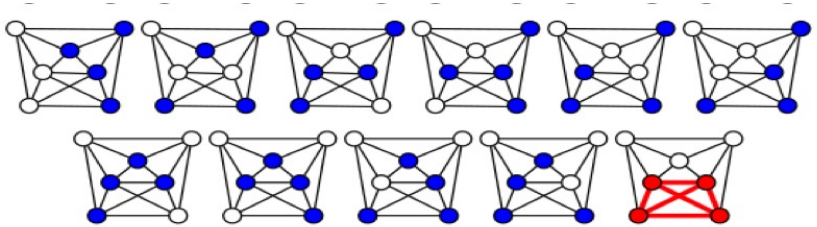
Karpの21の NP-完全問題

- **Satisfiability**: the boolean satisfiability problem for formulas in conjunctive normal form (often referred to as SAT)
  - **0–1 integer programming** (A variation in which only the restrictions must be satisfied, with no optimization)
  - **Clique** (see also independent set problem)
    - **Set packing**
    - **Vertex cover**
      - **Set covering**
      - **Feedback node set**
      - **Feedback arc set**
      - **Directed Hamilton circuit** (Karp's name, now usually called **Directed Hamiltonian cycle**)
        - **Undirected Hamilton circuit** (Karp's name, now usually called **Undirected Hamiltonian cycle**)

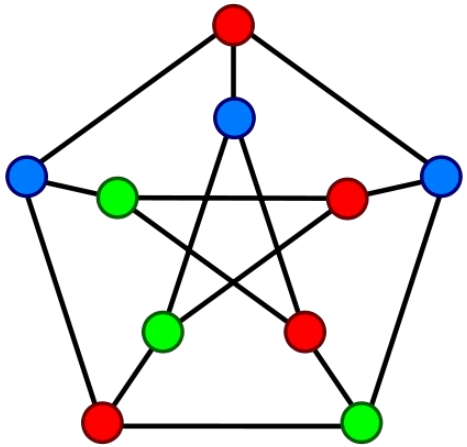
https://goo.gl/56qLWk

- **Satisfiability with at most 3 literals per clause** (equivalent to 3-SAT)
  - **Chromatic number** (also called the Graph Coloring Problem)
    - **Clique cover**
    - **Exact cover**
      - **Hitting set**
      - **Steiner tree**
      - **3-dimensional matching**
      - **Knapsack** (Karp's definition of Knapsack is closer to Subset sum)
        - **Job sequencing**
        - **Partition**
          - **Max cut**

https://goo.gl/56qLWk



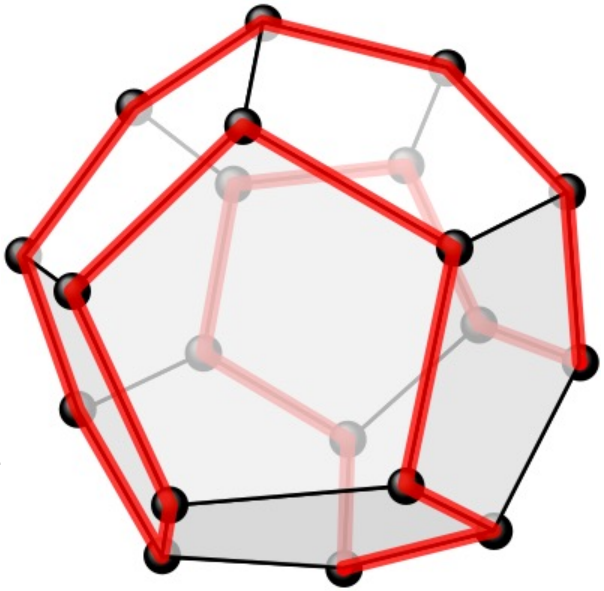
Clique



Graph Coloring

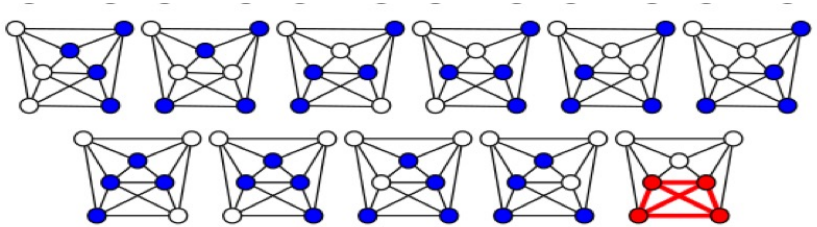
$$(l_1 \vee l_2 \vee x_2) \wedge$$
$$(\neg x_2 \vee l_3 \vee x_3) \wedge$$
$$(\neg x_3 \vee l_4 \vee x_4) \wedge \cdots \wedge$$
$$(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge$$
$$(\neg x_{n-2} \vee l_{n-1} \vee l_n)$$

3-SAT

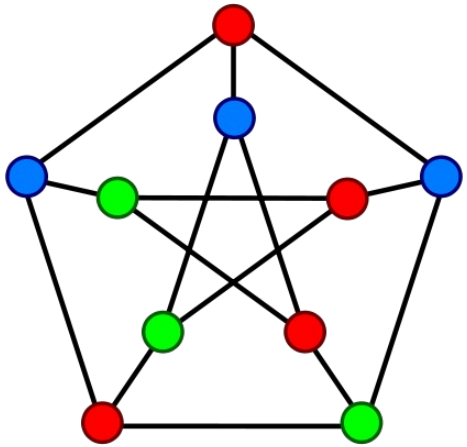


Hamilton Path

様々なNP-完全問題



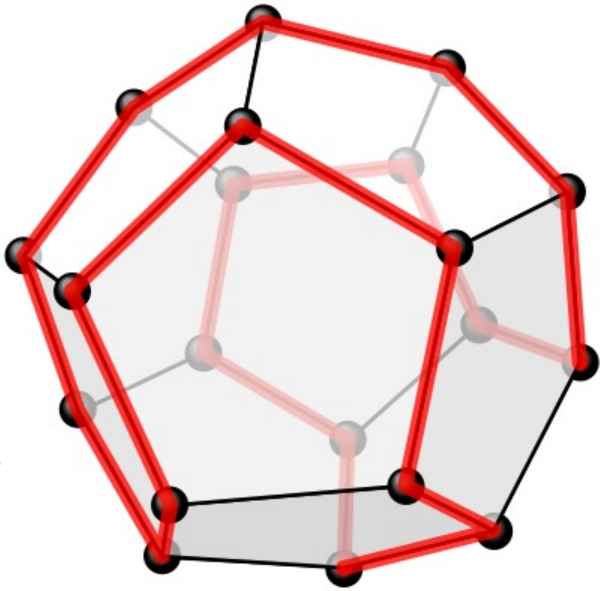
Clique



Graph Coloring

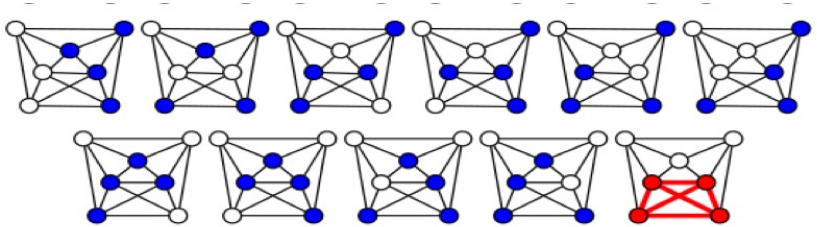
$$(l_1 \vee l_2 \vee x_2) \wedge$$
$$(\neg x_2 \vee l_3 \vee x_3) \wedge$$
$$(\neg x_3 \vee l_4 \vee x_4) \wedge \cdots \wedge$$
$$(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge$$
$$(\neg x_{n-2} \vee l_{n-1} \vee l_n)$$

3-SAT

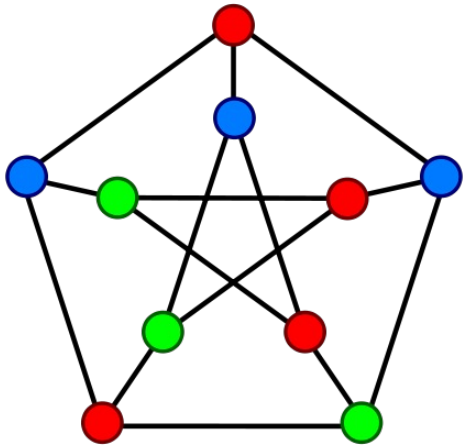


Hamilton Path

これらの問題は、解くのが難しい。もちろん、
力まかせに、総当たりでやれば解けるのだが、
それには指数関数的な膨大な時間がかかる。



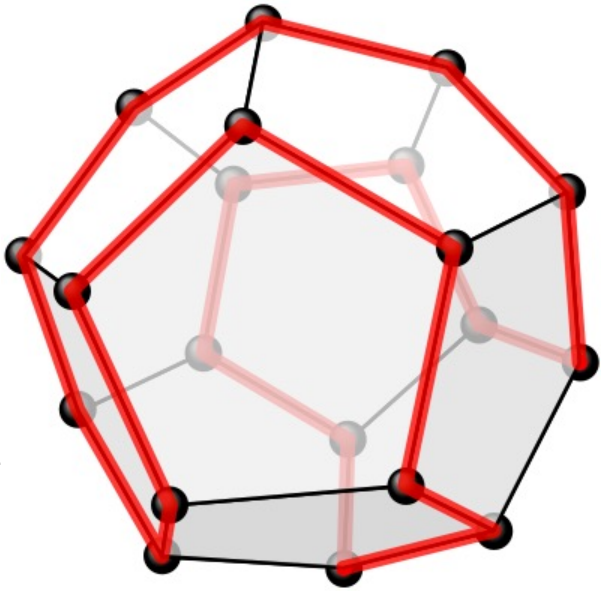
Clique



Graph Coloring

$$(l_1 \vee l_2 \vee x_2) \wedge$$
$$(\neg x_2 \vee l_3 \vee x_3) \wedge$$
$$(\neg x_3 \vee l_4 \vee x_4) \wedge \cdots \wedge$$
$$(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge$$
$$(\neg x_{n-2} \vee l_{n-1} \vee l_n)$$

3-SAT



Hamilton Path

しかも、この見かけは全く異なる、「現実的には
手に負えない問題」は、すべて同一のアルゴリズム
で、解くことができるのだ！

このNP-完全問題のリストは、現在では数百の問題をカバーして拡大している。（例えば、”List of NP-complete problems” https://goo.gl/XUSvRz　）

これらの問題の「手に負えなさ: “Intractability ”」は、例外的なものではなく、一般的なものである。

Cookは1982年、Karpは1985年、Turing賞を受賞した。

NPのもう一つの定義
Nondeterministic Polynomial time

先に、「NPは、ある問題とその答えが与えられた時、その答えが「正しい」ことを多項式時間で「検証」できる時間複雑性のクラスです。」という定義を述べたのですが、NPには、もう一つの定義があります。

それは、「非決定性チューリングマシンで、多項式時間で計算できるもの」という定義です。「非決定性チューリングマシン」がどのようなものであるかは、次の第二部で紹介します。

NPは、"Not P (多項式時間ではない)"ではありません。それは、**Nondeterministic Polynomial time** の略です。

DTIME(f(x))とNTIME(f(x))

「決定性チューリングマシン」で、O(f(x))で表される時間計算量の複雑性クラスを**DTIME**(f(x))で表したように、

「非決定性チューリングマシン」で、O(f(x))で表される時間計算量の複雑性クラスを**NTIME**(f(x))で表します。

DTIMEの時と同じように、f(x)の形に応じて、別名を付けます。

NP=**NTIME**(p(n)) ;

NEXP=**NTIME**($2^{p(n)}$) ;

NEEXP=**NTIME**($2^{2^{p(n)}}$) ;

これらを先の図に付け加えてまとめてみました。次の図を見てください。

DSPACE(f(x))とNSPACE(f(x))

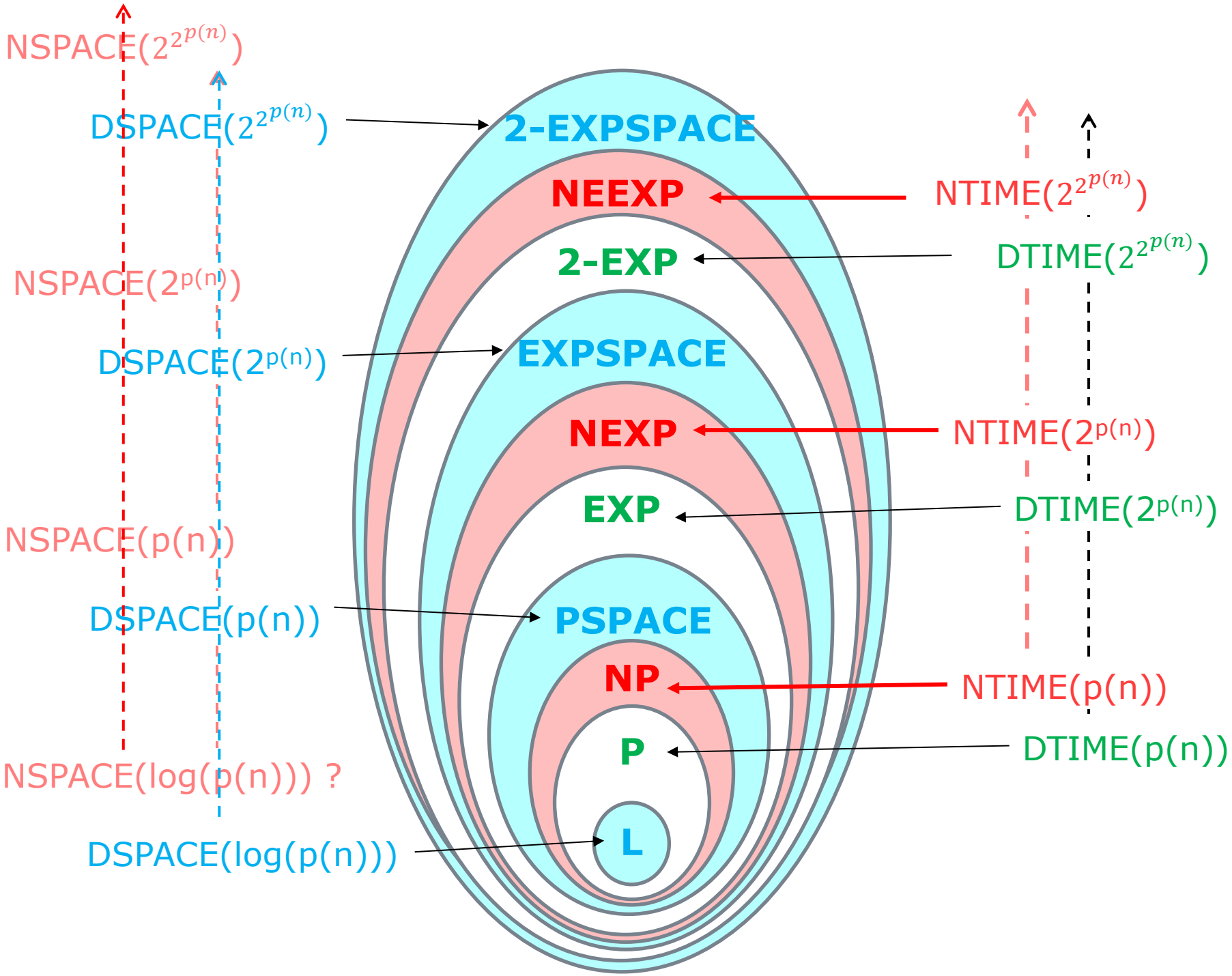
「決定性チューリングマシン」で、O(f(x))で表される空間計算量の複雑性クラスを**DSPACE**(f(x))で表したように、

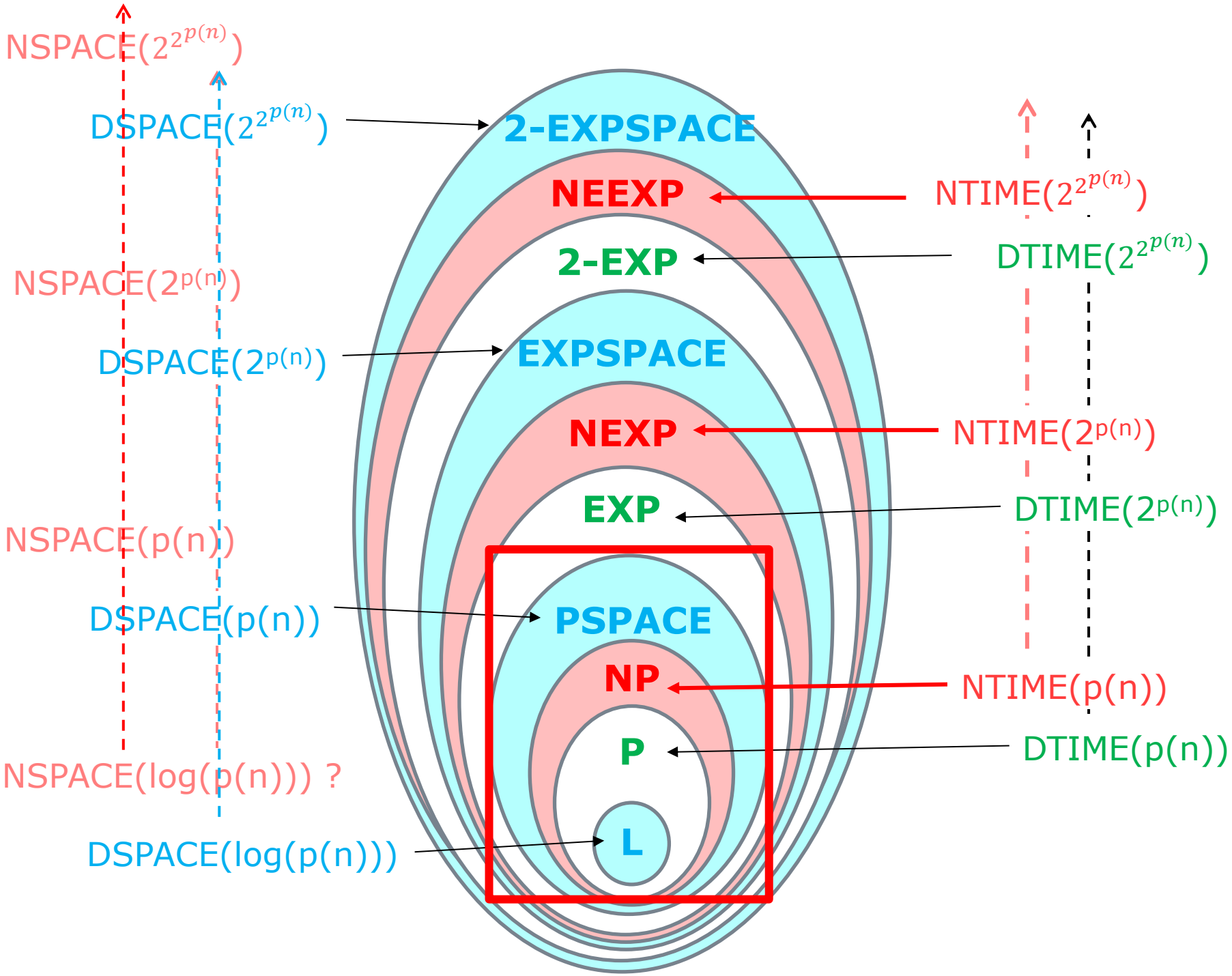
「非決定性チューリングマシン」で、O(f(x))で表される空間計算量の複雑性クラスを**NSPACE**(f(x))で表します。

DTIMEの時と同じように、f(x)の形に応じて、別名を付けます。

NSPACE=**NSPACE**(p(n)) ;
NEXPSPACE=**NSPACE**($2^{p(n)}$) ;
NEEXPSPACE=**NSPACE**($2^{2^{p(n)}}$) ;

これらを先の図に付け加えてまとめてみました。次の図を見てください。ただ、N*SPACEの領域は書き込んでいません。


NSPACE($2^{2^{p(n)}}$)
DSPACE($2^{2^{p(n)}}$)
2-EXPSPACE
NEEXP
NTIME($2^{2^{p(n)}}$)
2-EXP
DTIME($2^{2^{p(n)}}$)
NSPACE($2^{p(n)}$)
DSPACE($2^{p(n)}$)
EXPSPACE
NEXP
NTIME($2^{p(n)}$)
EXP
DTIME($2^{p(n)}$)
NSPACE(p(n))
DSPACE(p(n))
PSPACE
NP
NTIME(p(n))
P
DTIME(p(n))
NSPACE(log(p(n))) ?
DSPACE(log(p(n)))
L


NSPACE($2^{2^{p(n)}}$)
DSPACE($2^{2^{p(n)}}$)
2-EXPSPACE
NEEXP
NTIME($2^{2^{p(n)}}$)
2-EXP
DTIME($2^{2^{p(n)}}$)
NSPACE($2^{p(n)}$)
DSPACE($2^{p(n)}$)
EXPSPACE
NEXP
NTIME($2^{p(n)}$)
EXP
DTIME($2^{p(n)}$)
NSPACE(p(n))
DSPACE(p(n))
PSPACE
NP
NTIME(p(n))
P
DTIME(p(n))
NSPACE(log(p(n))) ?
DSPACE(log(p(n)))
L

複雑性クラスと問題の例

より難しい

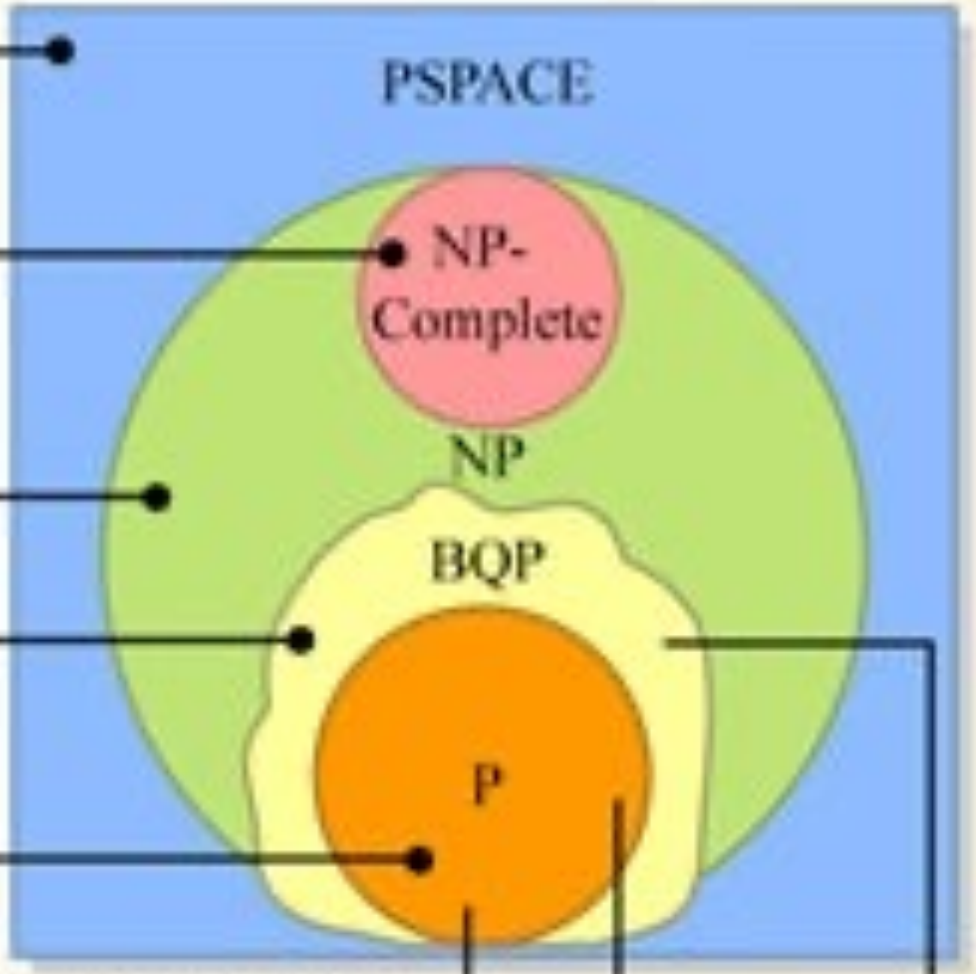
n x *n* チェス
n x *n* 碁

箱詰め問題
地図の塗り分け
トラベリング・セールスマン
n x *n* 数独

グラフ同型問題

素因数分解
離散対数

グラフの接続性
素数判定
マッチ・メーキング



古典コンピュータで
効率的に解ける問題

量子コンピュータで
効率的に解ける
問題

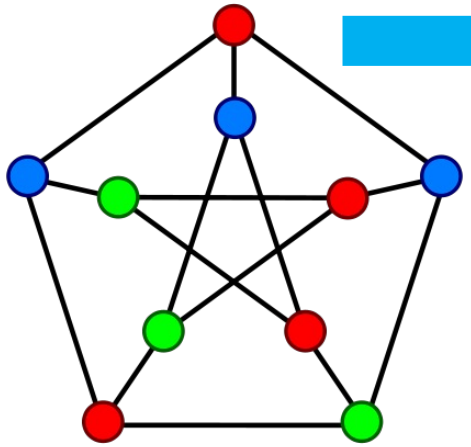

NP-完全問題
Cook-Levin定理とKarpの還元
今回は、この部分は割愛します。

NP-完全問題
Cook-Levin定理とKarpの還元（詳細編）

3-SAT

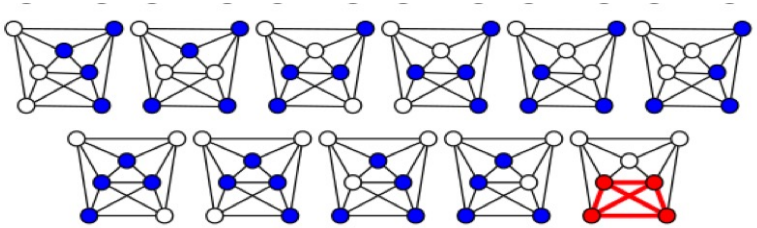
$$(l_1 \vee l_2 \vee x_2) \wedge$$
$$(\neg x_2 \vee l_3 \vee x_3) \wedge$$
$$(\neg x_3 \vee l_4 \vee x_4) \wedge \cdots \wedge$$
$$(\neg x_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge$$
$$(\neg x_{n-2} \vee l_{n-1} \vee l_n)$$

Graph Coloring



Hamilton Path

Clique



NP-完全問題

人工知能のもっとも基本的な問題の一つは、機械と人間にとっての「認識」の限界を正確に知ることであると僕は考えている。

今回は、改めて複雑性理論の中核である「NP-完全」問題という不思議なコンセプトを説明しようと思う。

「NP-完全」の発見なしには、複雑性理論は生まれなかったと言っていい。その理論的インパクトは、計算可能性理論の創出にゲーデルの不完全性定理が与えたインパクトと同じものだと思う。

「NP-完全」問題は、基本的に、力まかせの総当たりでやれば必ず解は見つかるはずというところが、どう頑張っても原理的に証明することが不可能な命題が存在することを主張する「不完全性定理」とは異なっている。

ただ、総当たりで問題を解くには、n個の条件が成り立つか否かというように問題を単純化しても、試すべき条件の組み合わせは、2のn乗個になる。nが小さいうちは（例えば、n＝10なら、1024個の場合をチェックすればいい）なんとかなるのだが、nが大きくなるにつれ、問題は、文字通り指数関数的に難しくなる。それは、総当たり法では、現実的には、全く手に負えない問題となる。

NP-完全問題を解くコンピュータ・プログラム？

「NP-完全問題というのは、コンピュータでは手も足も出ない問題なんじゃないの？」と考えている人も多いかもしれない。でも、それは正確とはいえない。

まず、NP-完全問題は、原理的には「計算可能」な問題である。

ただ、n (問題の大きさ＝入力の大きさをnとして) が大きくなるにつれて、その計算を実行することが現実的には不可能となるような問題なのである。逆に言えば、nが小さい時には、計算は可能である。

先に、「NP-完全問題を解くコンピュータ・プログラム」と言ったのは、こうしたプログラムのことである。

NP-完全問題への挑戦

いくつかのNP-完全問題は、それをを解くことに実践的に重要な意味がある。「最適化問題」の多くはNP-完全問題に属している。「NP-完全問題を解くコンピュータ・プログラム」のアルゴリズムの研究・改良は活発に行われている。3-SATについては、国際的な競技会が定期的に開催されているほどである。

いくつかのNP-完全問題の、最良のアルゴリズムの計算時間量を次に示す。

- 3-SAT　　O($2^{0.386n}$)
- cliques　　O(1.1888^n)
- 3-coloring　O(1.3289^n)
- Hamilton path　O(1.251^n)

量子コンピューティングへの期待

現在、NP-完全問題への挑戦として、もっとも期待を集めているのは、量子コンピュータである。

一時は、量子コンピュータがNP-完全問題を解くのではという期待があったのだが、それは現在では、理論的には否定されている。ただ、量子アニーリング系の量子コンピューティング技術が、近似的にNP-完全問題を解くうえで有効であるというのは確かである。

これらのトピックスについては、量子複雑性を扱うセッションで取り上げたいと思う。

証明を持つ
数学的命題のクラスは
NP完全である

古典的な証明のイメージ

命題：T

証明：$\pi_1, \pi_2, \pi_3, \ldots, \pi_n = T$

- π_iは、公理であるか、予め定義された論理的な演繹規則から導かれた命題である。（ここでは、ZFCの公理系を取ろう）
- 論理的な演繹規則は、一階の述語論理を利用することにしよう。
- π_iが、$\pi_1, \pi_2, \pi_3, \ldots, \pi_{i-1}$から導かれることは、簡単にチェックされなければならない。それが、我々が証明を理解できるということである。

チェックのアルゴリズムと数学的導出

命題：T

証明：$\pi_1, \pi_2, \pi_3, \ldots, \pi_n = T$

- 数学的証明のシステムを形式化するには、$\pi_1, \pi_2, \pi_3, \ldots, \pi_{i-1} \rightarrow \pi_i$ なる数学的導出の各ステップを形式化すれば十分である。それは、チェックのアルゴリズムを形式化することに等しい。
- 数学的導出のルール、すなわち、それぞれのチェック・アルゴリズムが、証明の中に現れる命題と証明全体を規定している。
- また、証明が有限の時間のうちに検証可能であるなら、それを構成する、それぞれのチェック・アルゴリズム、すなわち、数学的導出のルールは、時間的な制約条件を満たさなければならない。

数学的証明問題はNP完全問題である

数学の問題を解くことは難しいが、その証明を理解することは証明を行うより容易である。かつその証明の理解は、多項式時間で行われる。数学的証明は、NP問題である。

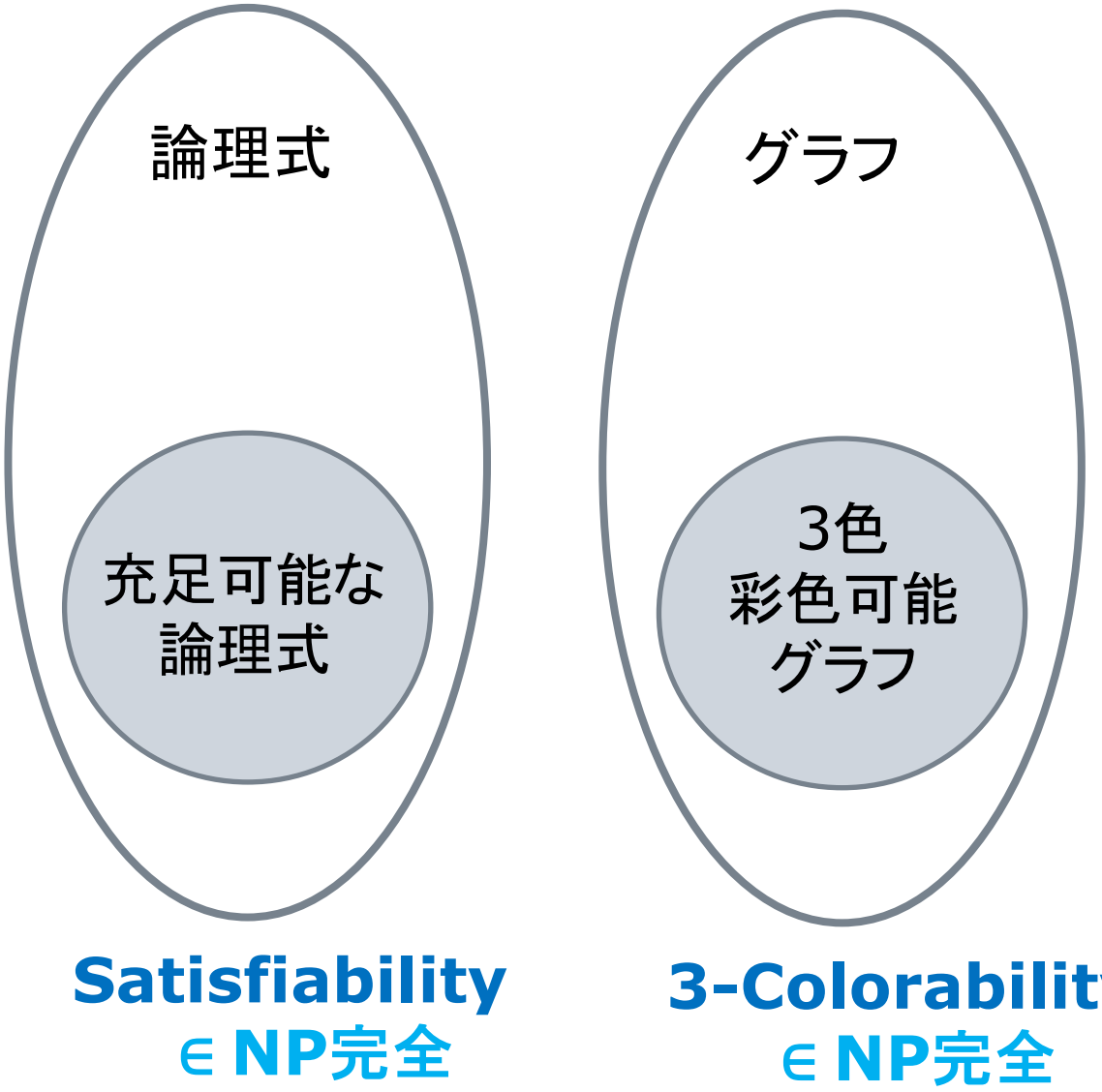
証明可能な数学的な命題が、NP-困難のクラスに属することは自明である。なぜなら、数学的に記述された全てのNP問題は、証明可能な数学的命題のクラスに還元できるからである。

よって、(実効的に)証明可能な数学的な命題のクラスは、NP完全である。

代表的なNP完全のクラス

Satisfiability
∈ NP完全

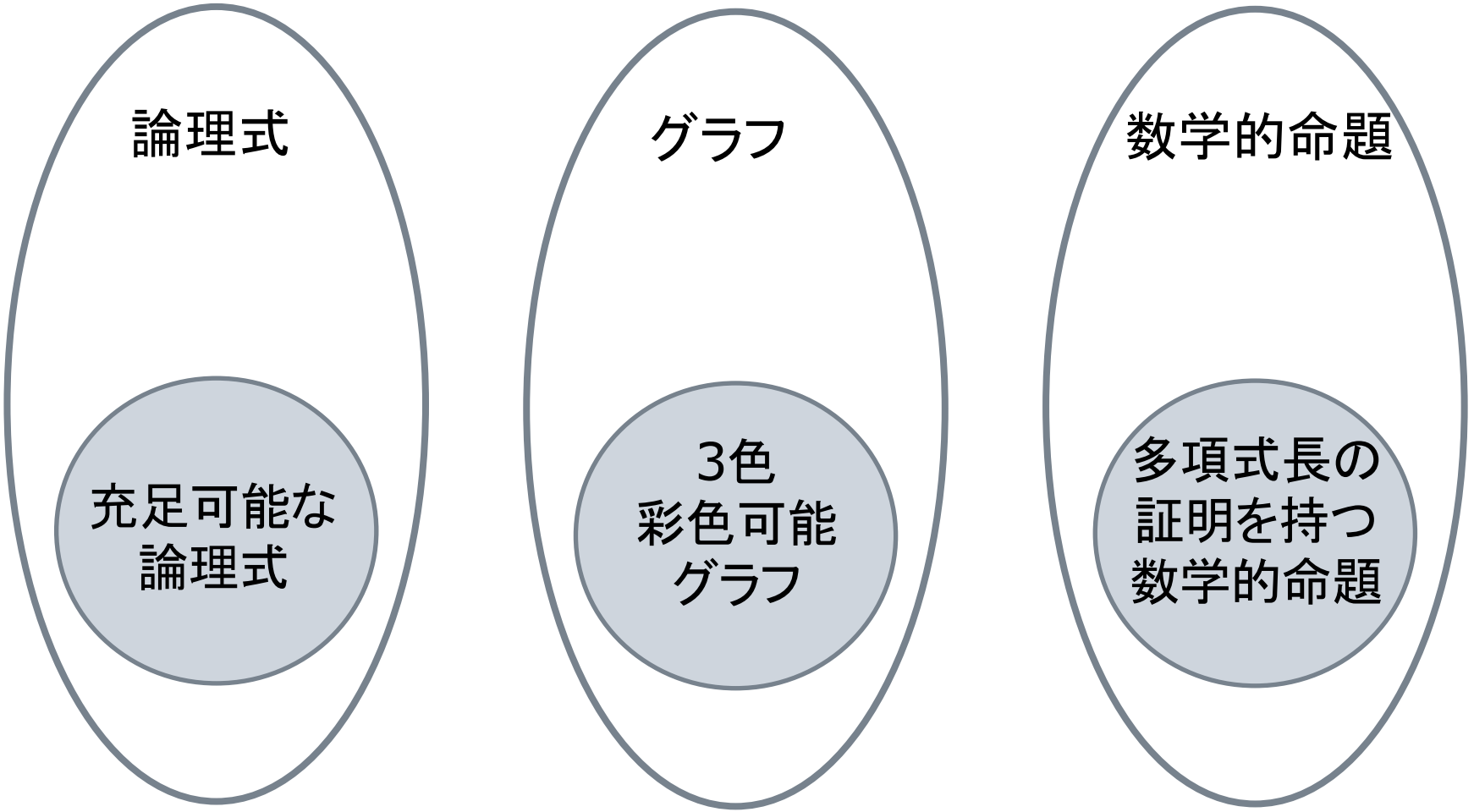
代表的なNP完全のクラス



Satisfiability
∈ NP完全

3-Colorability
∈ NP完全

代表的なNP完全のクラス

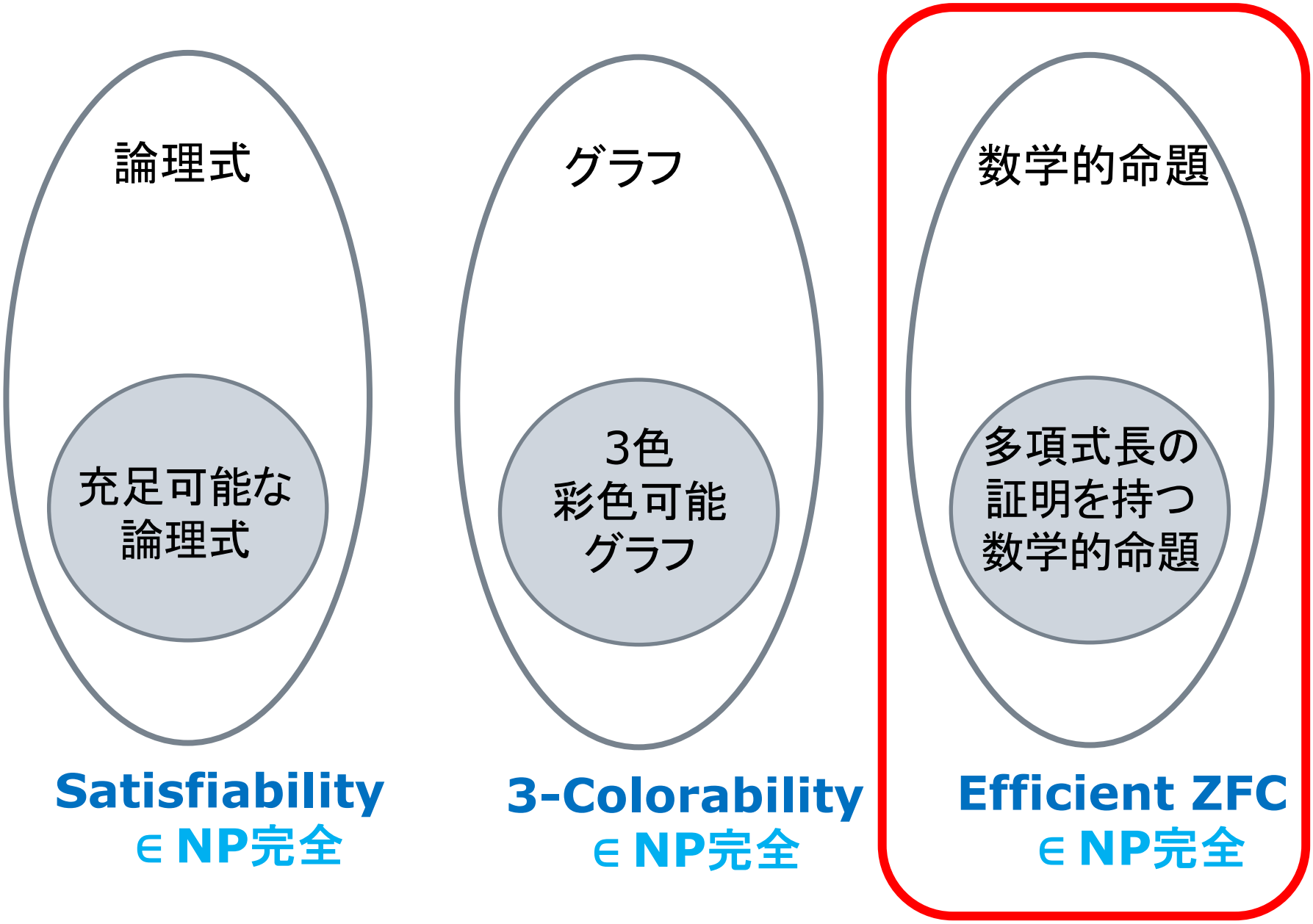


Satisfiability
∈ NP完全

3-Colorability
∈ NP完全

Efficient ZFC
∈ NP完全

代表的なNP完全のクラス



NP-完全問題
Cook-Levin定理とKarpの還元 (詳細編)

Agenda

- NP-完全問題 手におえない問題の存在
- Cook-Levin Theorem: SATはNP完全である
- Karp: 多項式時間で還元可能なNP完全問題
  - Clique 問題
  - Graph Coloring 問題
  - Hamiltonian Path 問題
- 実際に実装されているNP-完全問題のプログラム

今回は、この部分は割愛します。

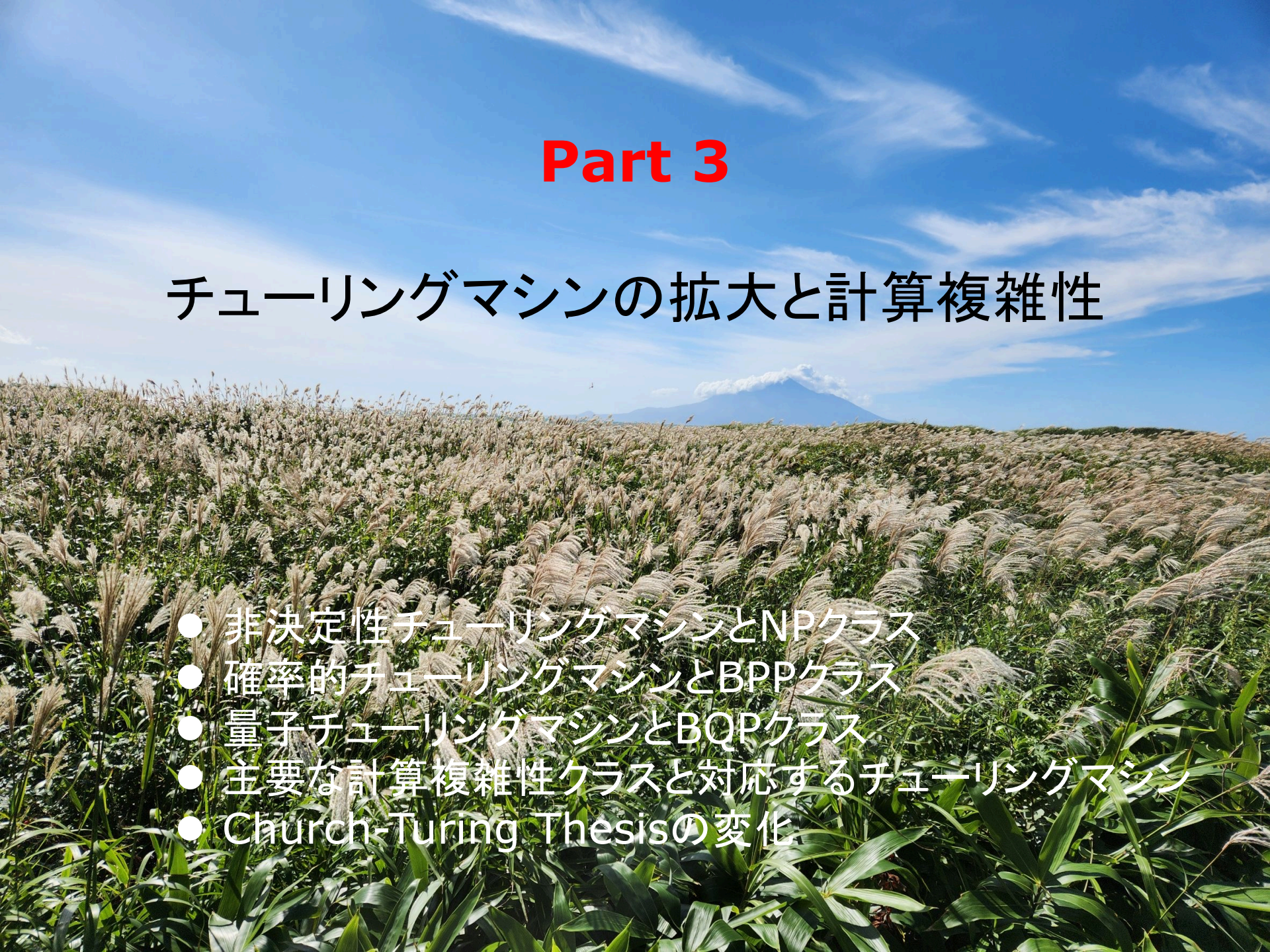
セッションの構成

このセッションでは、まず、3-SAT問題が NP-完全だという Cook-Levinの定理の証明を追いかけてみようと思う。

次に、Karpが、Clique 問題、Graph Coloring 問題、Hamiltonian Path 問題を、NP-完全問題だとした証明の概略を紹介する。これらの証明は、いずれもこれらの問題が 3-SAT 問題に多項式時間の計算で「還元」できるという形の証明になっている。

セッションの最後に、NP-完全問題を解くコンピュータ・プログラムをいくつかの問題について紹介しようと思う。

今回は、この部分は割愛します。



Part 3

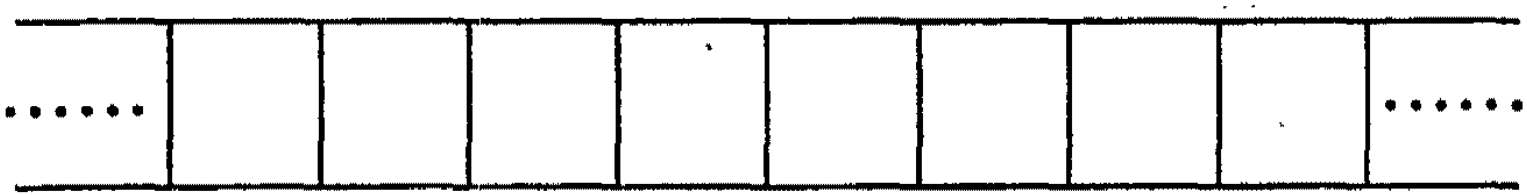
チューリングマシンの拡大と計算複雑性

- 非決定性チューリングマシンとNPクラス
- 確率的チューリングマシンとBPPクラス
- 量子チューリングマシンとBQPクラス
- 主要な計算複雑性クラスと対応するチューリングマシン
- Church-Turing Thesisの変化

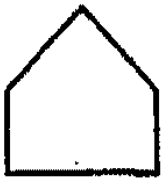

非決定性チューリングマシンとNPクラス

決定性チューリングマシン
（通常のチューリングマシン）

チューリング・マシンのメカニズム



テープ（マスで区切られている）



ヘッド

テープ：ひとマスずつに区切られており、左右に移動する。長い。

ヘッド：テープのひとマスの記号を読み取り、マシンの「状態」に応じて、次の動作をする。

可能な動作：

1. マス目に記号を書き込む（消す＋書く）。あるいは、そのままにして何も書き込まない。
2. マシンの状態を変える。あるいは状態を同じに保つ。
3. ヘッドを、右あるいは左に移動する。（テープが動く）

チューリング・マシンのプログラムの例

Turingマシンのプログラムの例を示す。この表の、例えば、最初の State Aの行で、On '0' の欄の 'B1R' は、「状態Aで、ヘッドが'0'の上にあるなら、状態をBに変えて、1を書き込んで、ヘッドを右（R）に移動する」という命令を表す。

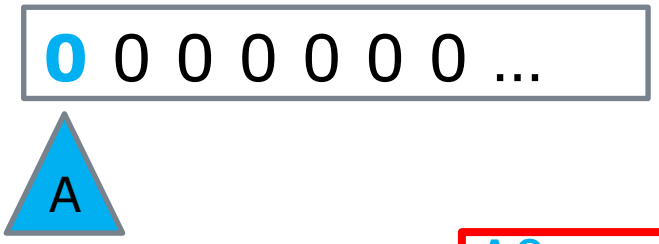
State	on 0	on 1	on 0			on 1		
			Print	Move	Goto	Print	Move	Goto
A	B1R	D0L	1	right	B	0	left	D
B	C1R	F0R	1	right	C	0	right	F
C	C1L	A1L	1	left	C	1	left	A
D	E0L	H1L	0	left	E	1	left	H
E	A1L	B0R	1	left	A	0	right	B

プログラム本体

プログラムの説明

Hは特別な「状態」で、この状態の時、マシンは「停止」する

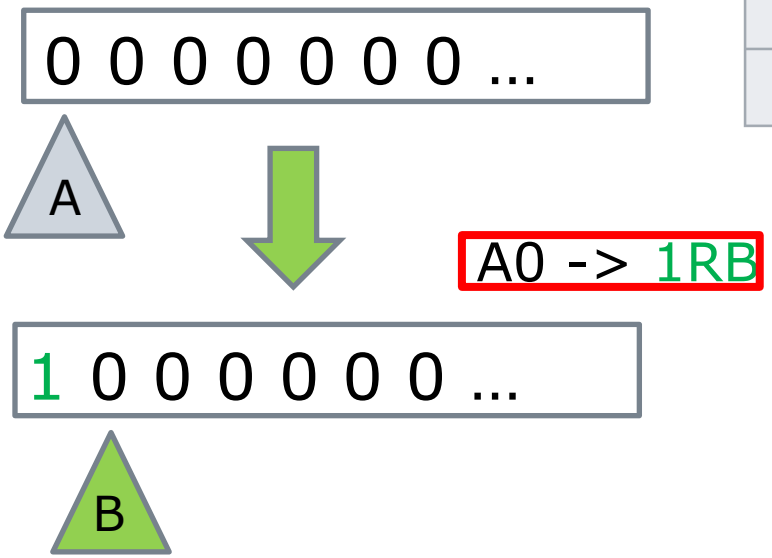
	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



A0 -> 1RB

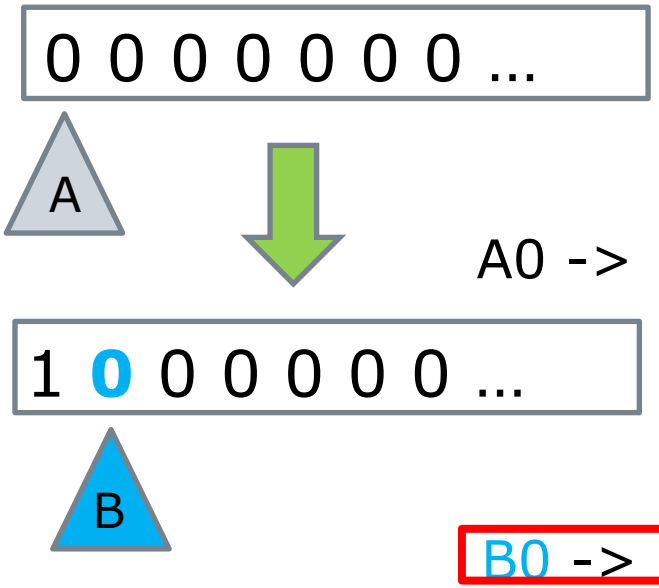
状態Aでヘッドが0の上にあるなら

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



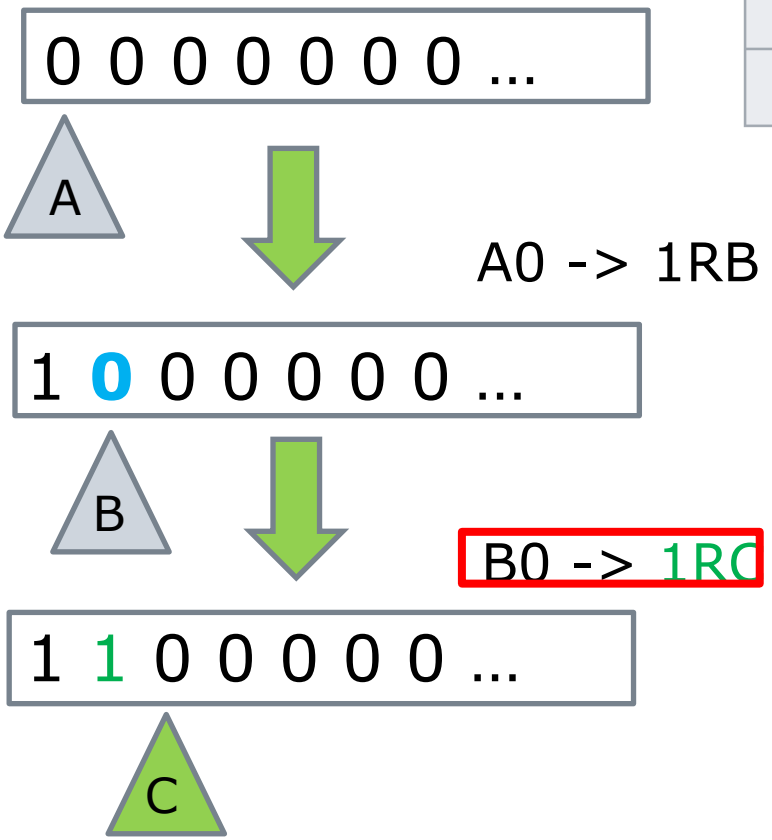
状態Aでヘッドが0の上にあるなら
1を書き込み、右にヘッドを移動し
状態をBに変える。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



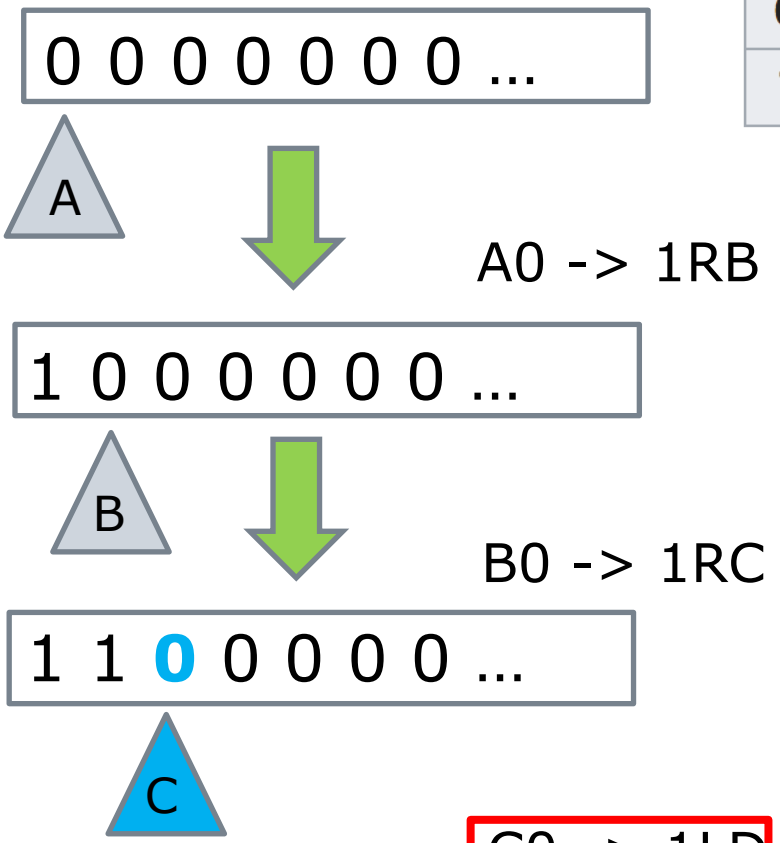
状態Bでヘッドが0の上にあるなら

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



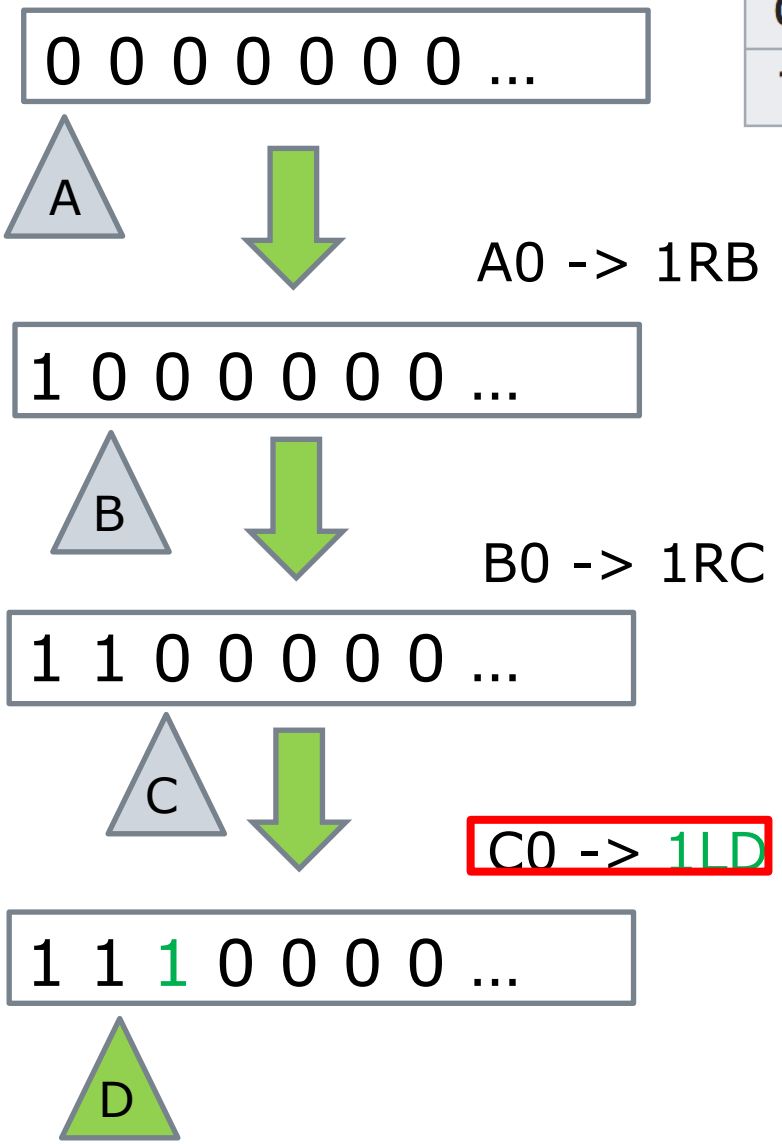
状態Bでヘッドが0の上にあるなら
1を書き込み、右にヘッドを移動し
状態をCに変える。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



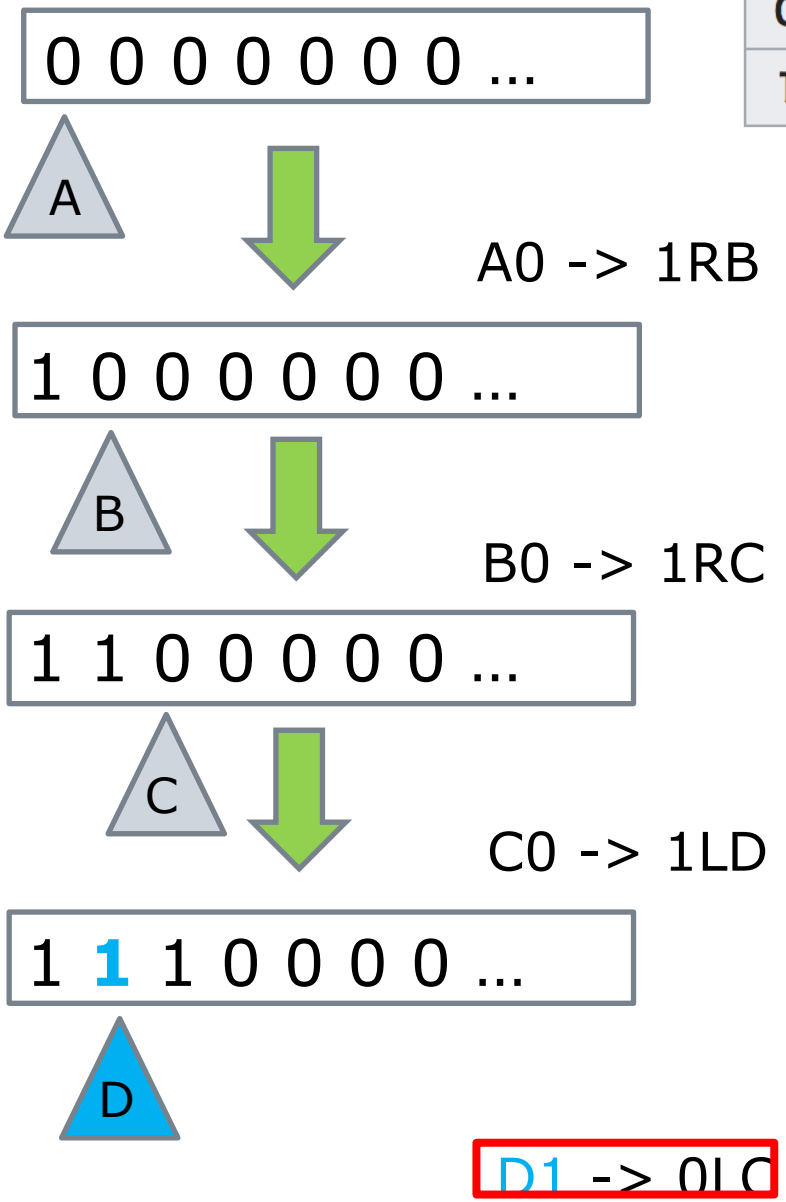
状態Cでヘッドが0の上にあるなら

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



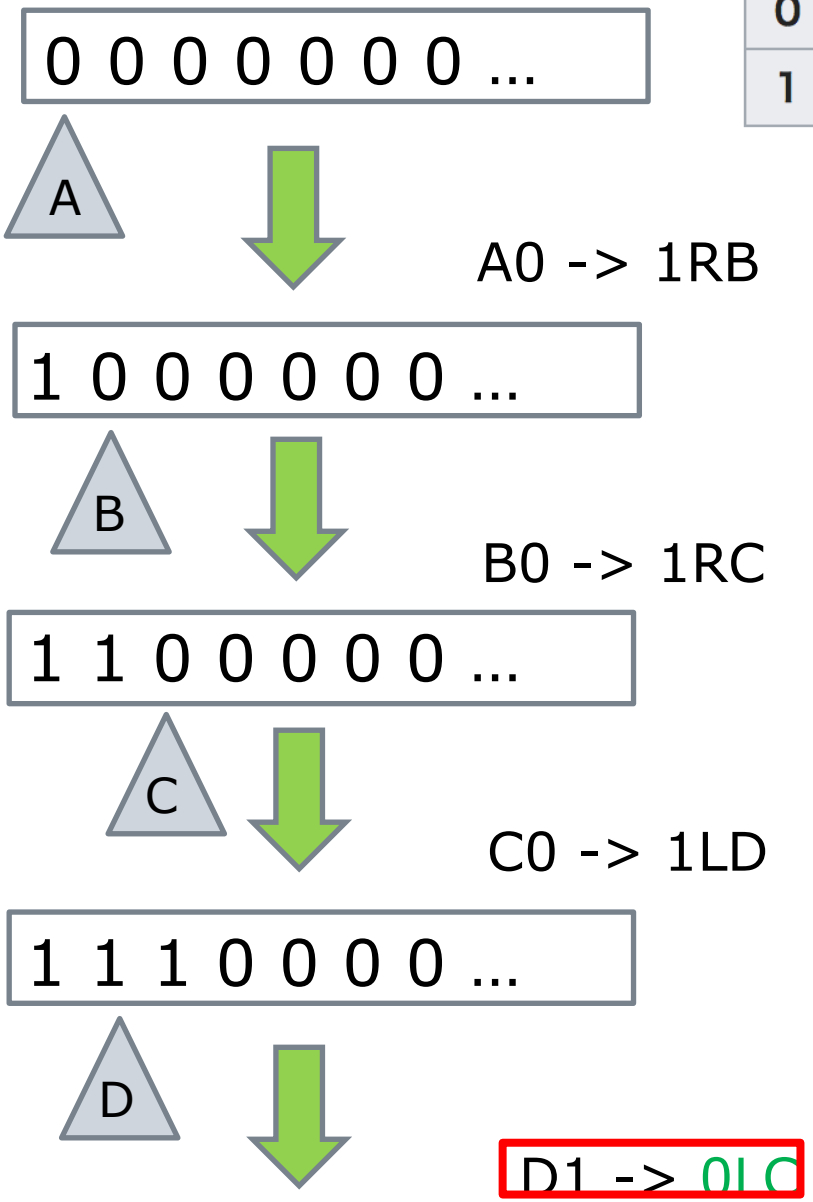
状態Cでヘッドが0の上にあるなら
1を書き込み、左にヘッドを移動し
状態をDに変える。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



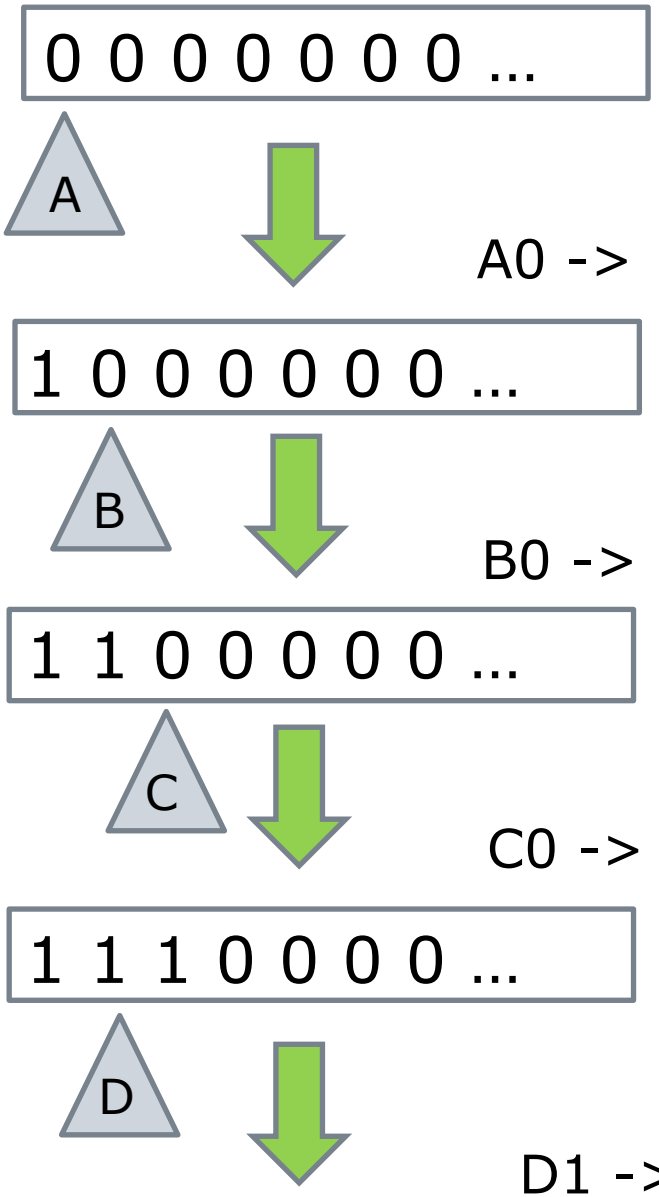
状態Dでヘッドが1の上にあるなら

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC



状態Dでヘッドが1の上にあるなら
0を書き込み、左にヘッドを移動し
状態をCに変える。

	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC

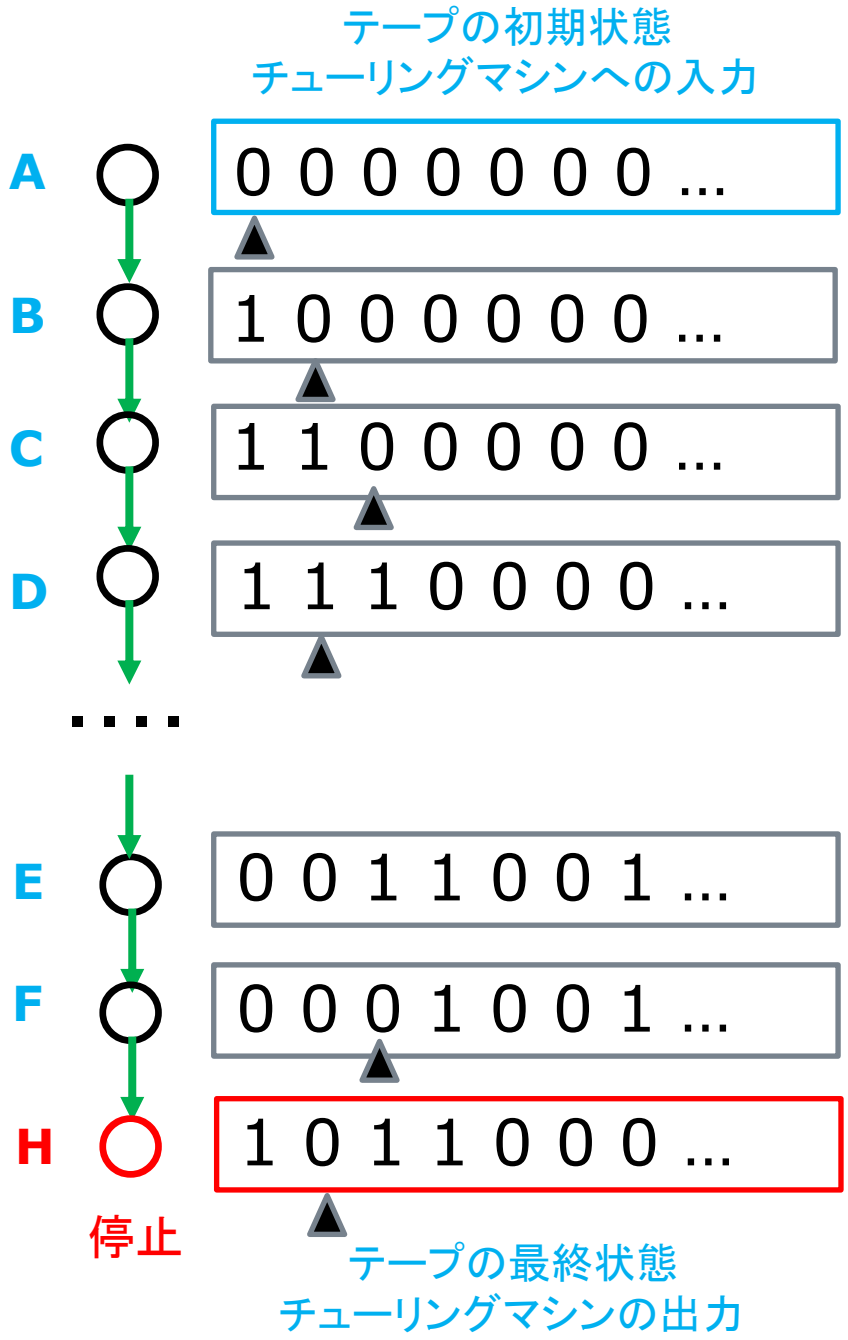


状態Aでヘッドが0の上にあるなら
1を書き込み、右にヘッドを移動し
状態をBに変える。

状態Bでヘッドが0の上にあるなら
1を書き込み、右にヘッドを移動し
状態をCに変える。

状態Cでヘッドが0の上にあるなら
1を書き込み、右にヘッドを移動し
状態をCに変える。

状態Dでヘッドが1の上にあるなら
0を書き込み、左にヘッドを移動し
状態をCに変える。



状態の遷移が、次の表で与えられた時のチューリングマシンの振る舞い

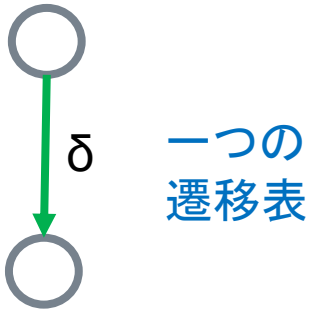
	A	B	C	D	E	F
0	1RB	1RC	1LD	1RE	1LA	1LH
1	1LE	1RF	0RB	0LC	0RD	1RC

決定性チューリングマシンは、一つの遷移表（命令セット）を持つ。

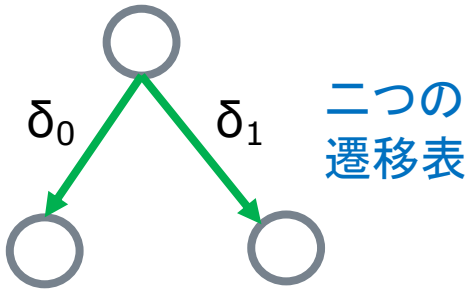
非決定性チューリングマシン

二つの遷移表(命令セット)を持つチューリングマシンを考える

- 通常のチューリングマシン(決定性チューリングマシン)は、一つの遷移表(命令セット)δを持つ。この時、マシンとテープの状態を一つのノードとみなしてマシンの実行を、ノードからノードへの遷移のグラフとして考えると、このグラフは、枝分かれのない、すべてのノードが一直線上に並んだものになる。

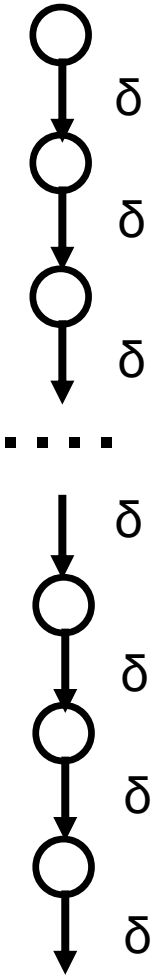


- 二つの遷移表(命令セット)δ_0とδ_1を持つチューリングマシンを考える。あるノード上で、δ_0とδ_1のどちらの命令セットを選ぶかは決まっていない。この時、このマシンで可能な実行のグラフは、木構造になる。こうしたマシンを非決定性チューリングマシンという。

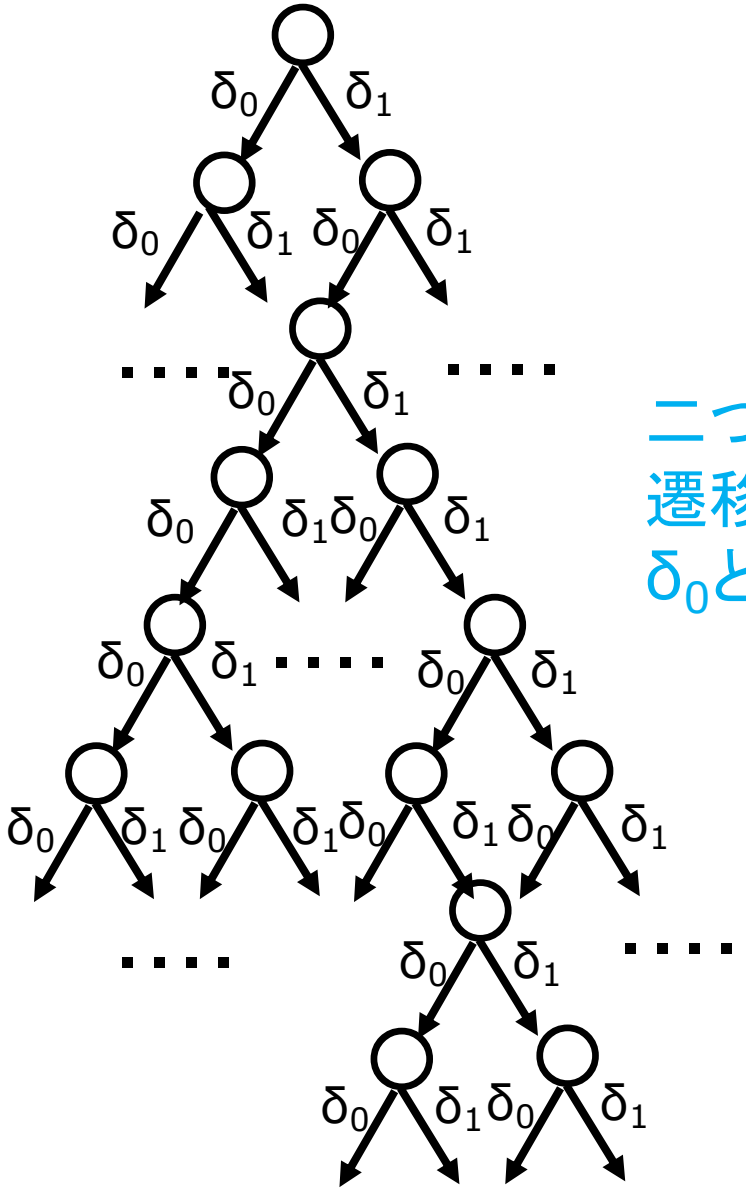


- 実は、遷移表は、二つ以上あってもいい。

直観的イメージ



一つの
遷移表δ



二つの
遷移表
δ_0とδ_1

決定性チューリングマシン

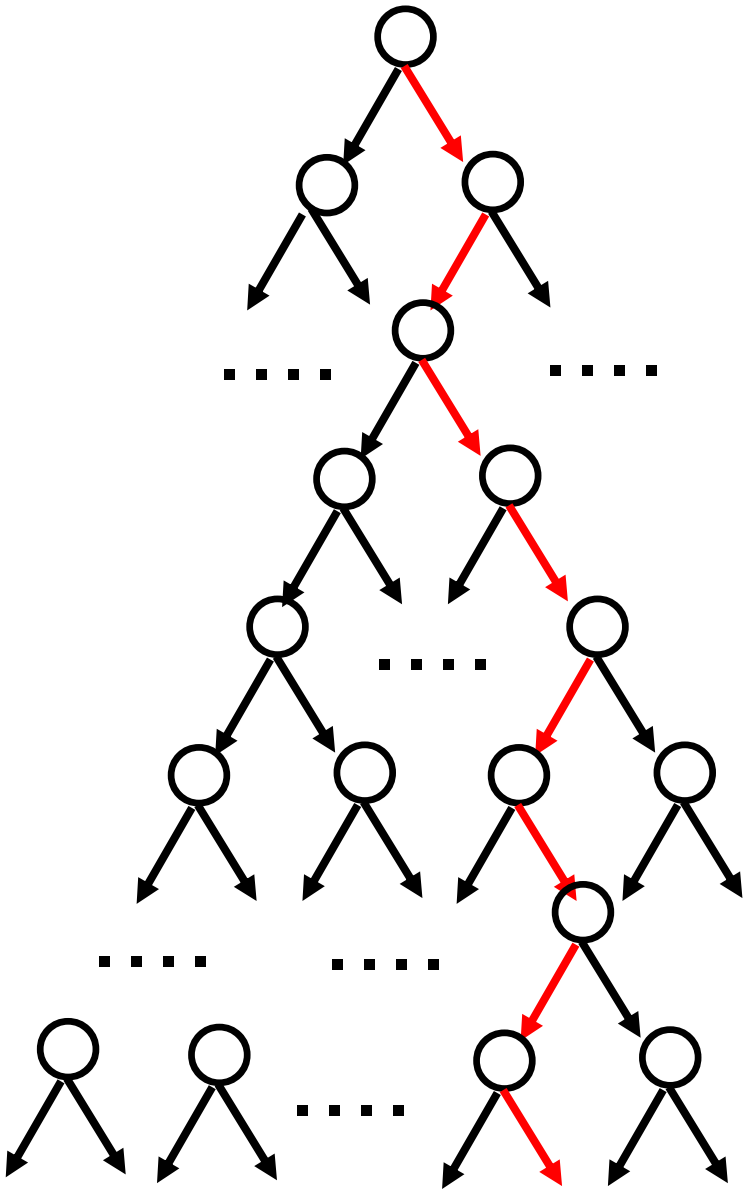
非決定性チューリングマシン

非決定性チューリングマシンでの受理と不受理

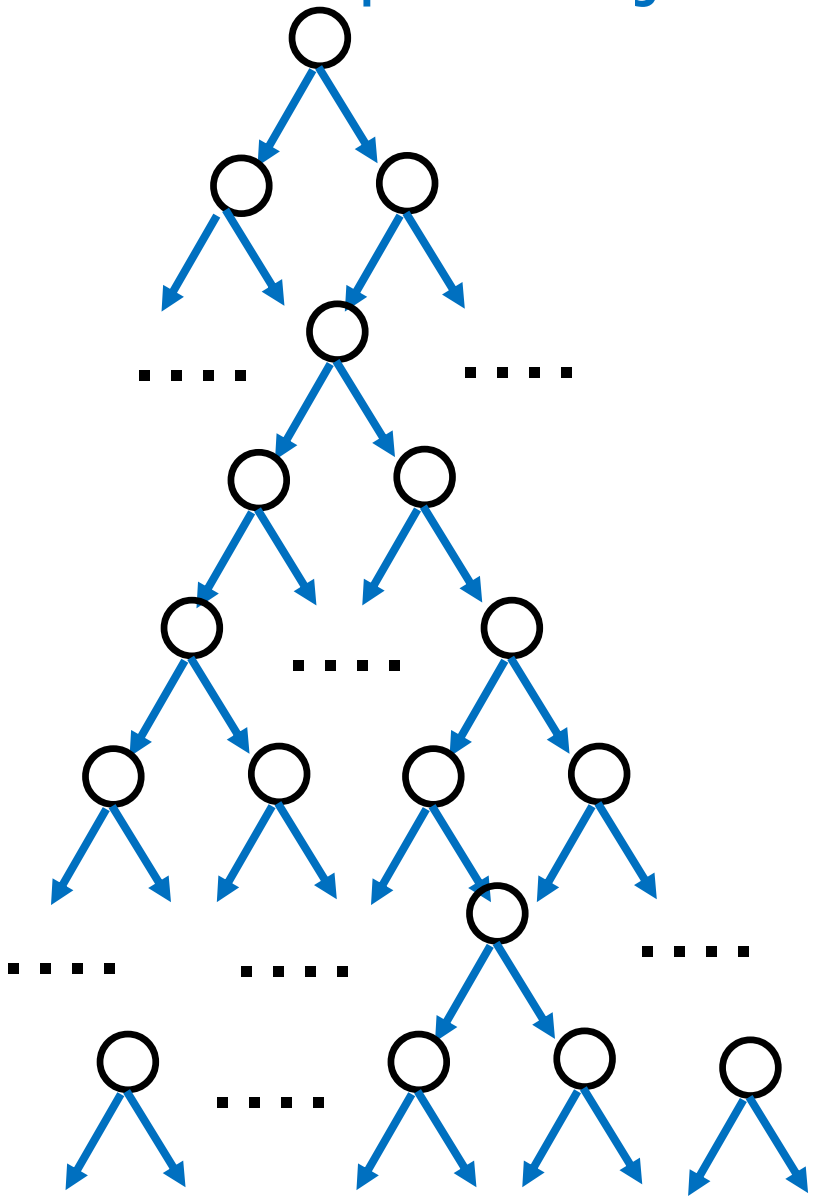
- 非決定性チューリングマシンNでの受理(accept)と不受理(reject)は、次のように定義される。

- Nがwを受理するのは、wを入力とするNの木構造で、acceptで終わるパスが一つでもある場合である。
- Nがwを不受理とするのは、wを入力とするNの木構造で、全てのパスが rejectで終わる場合である。

非決定性チューリングマシンのAccept と Reject



一つでも**accept**で終わるパスがある時

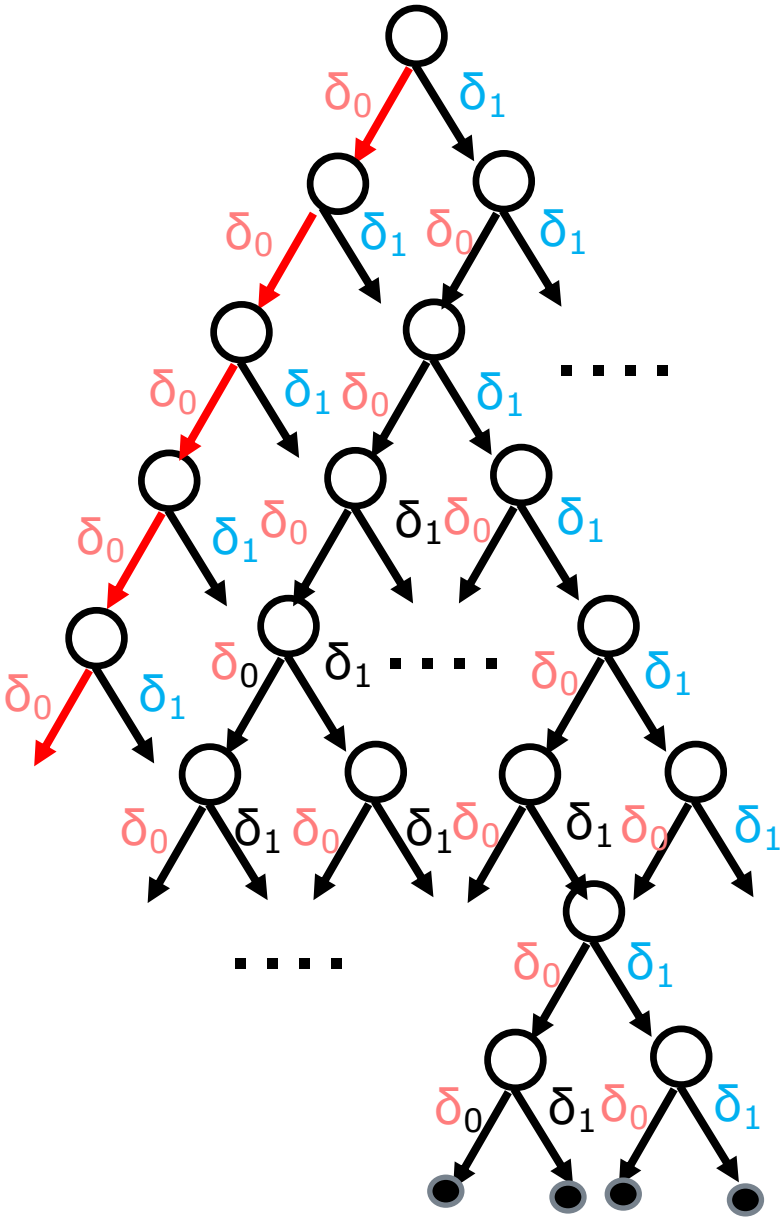


全てのパスが**reject** で終わる場合

非決定性チューリングマシンを決定性チューリングマシンでシミュレートする

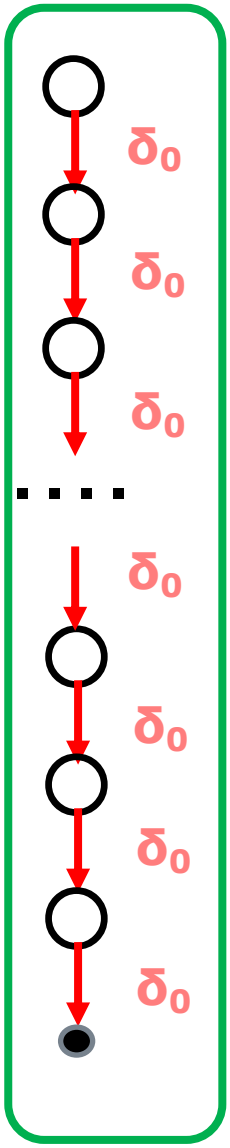
決定性チューリングマシン

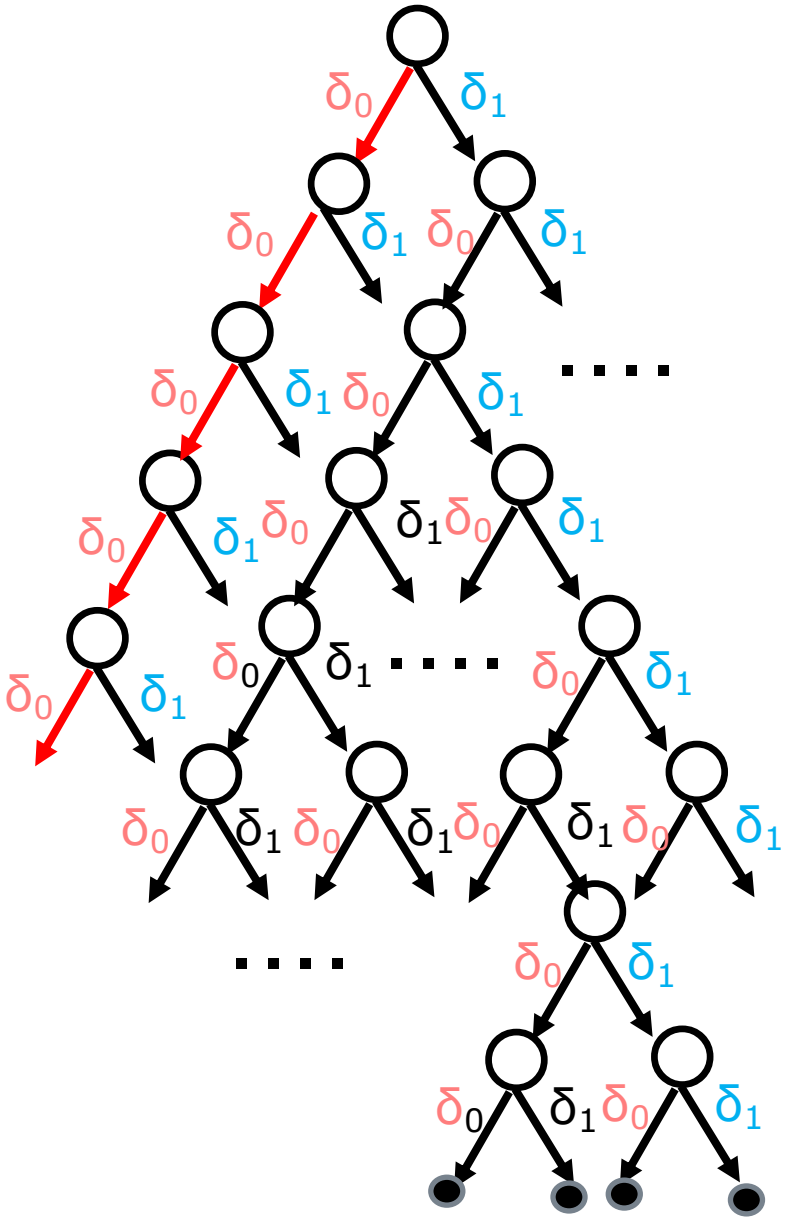
非決定性チューリングマシン



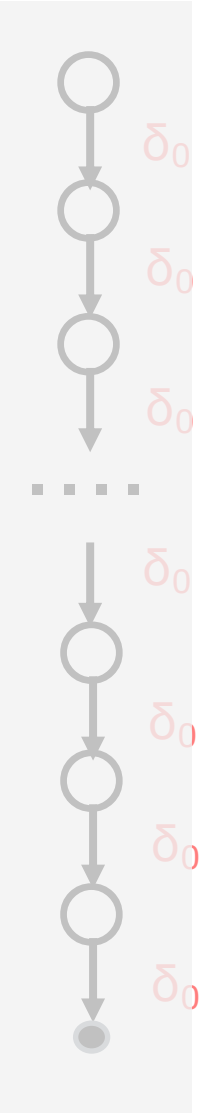
決定性チューリングマシン

非決定性チューリングマシン

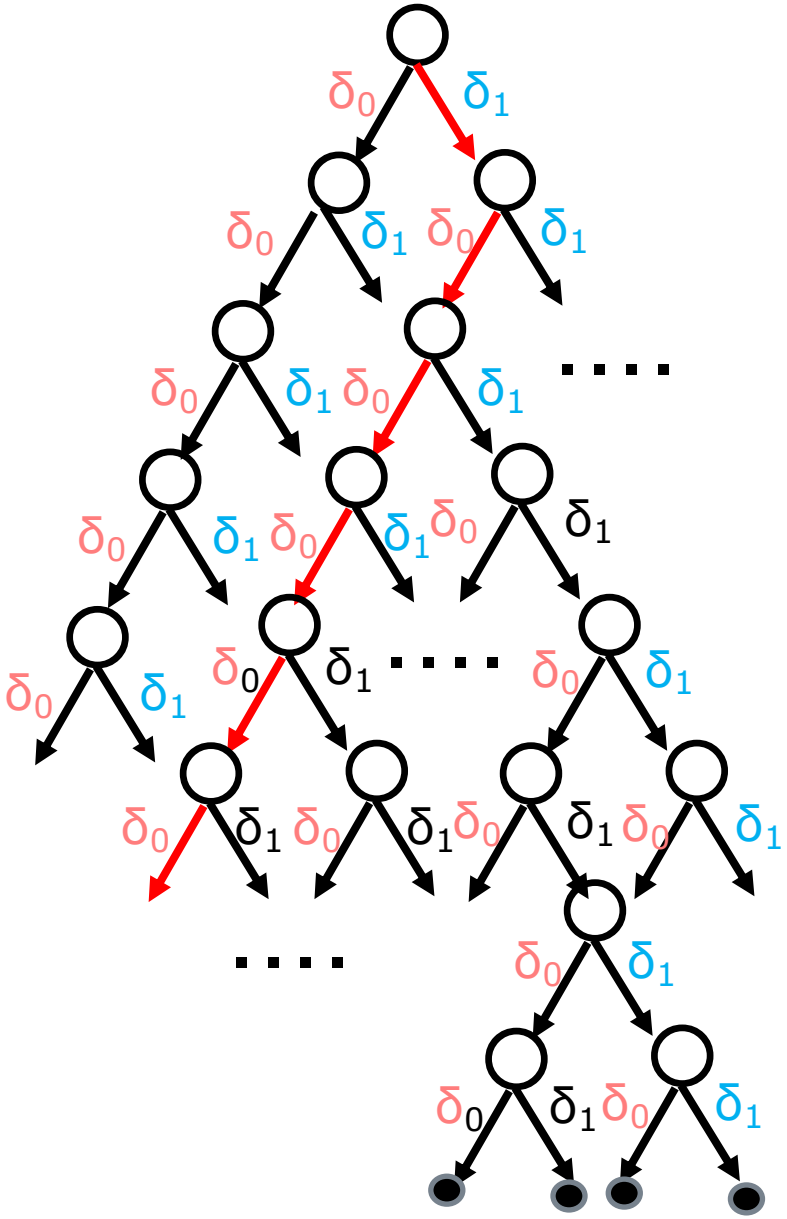




決定性チューリングマシン

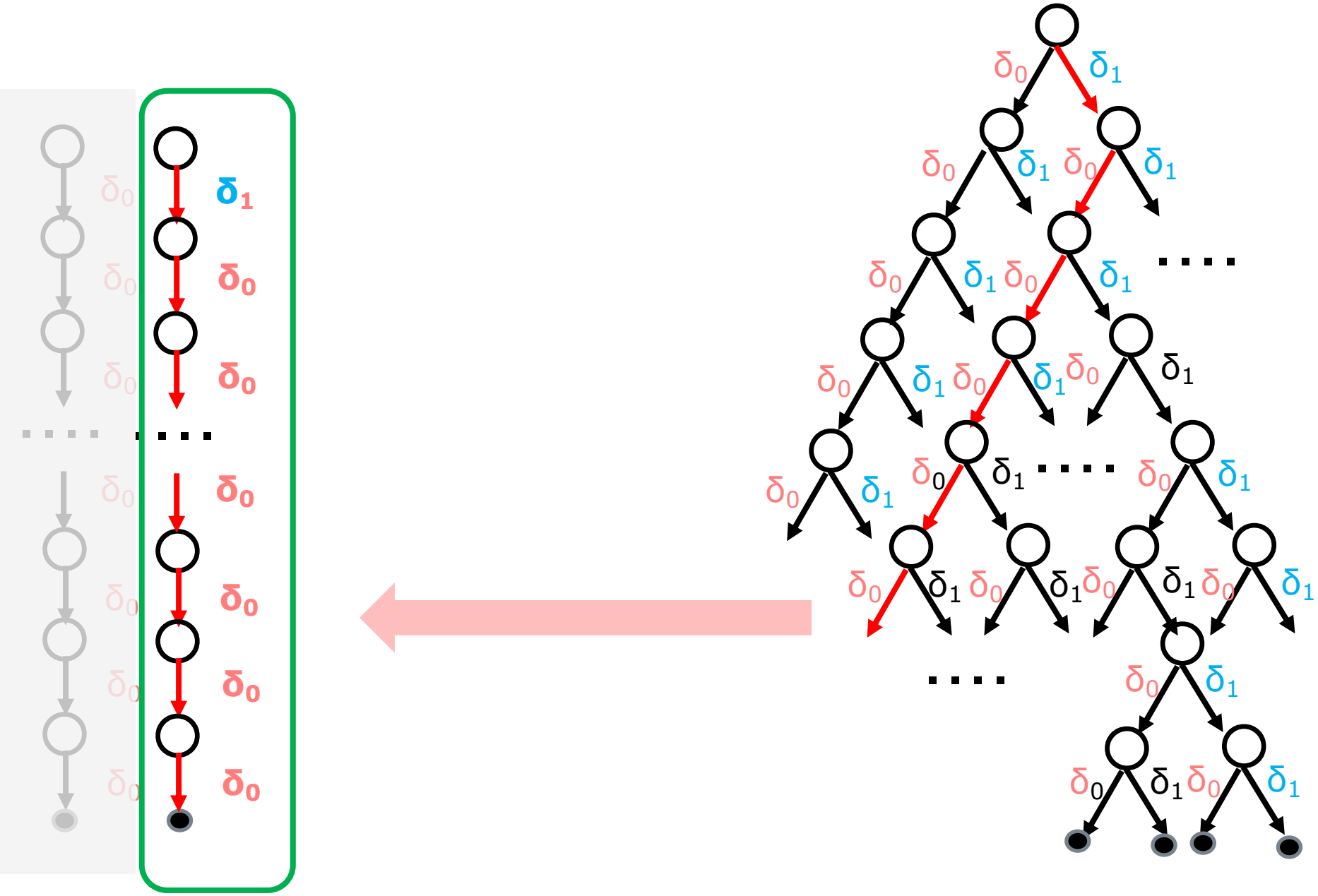


非決定性チューリングマシン

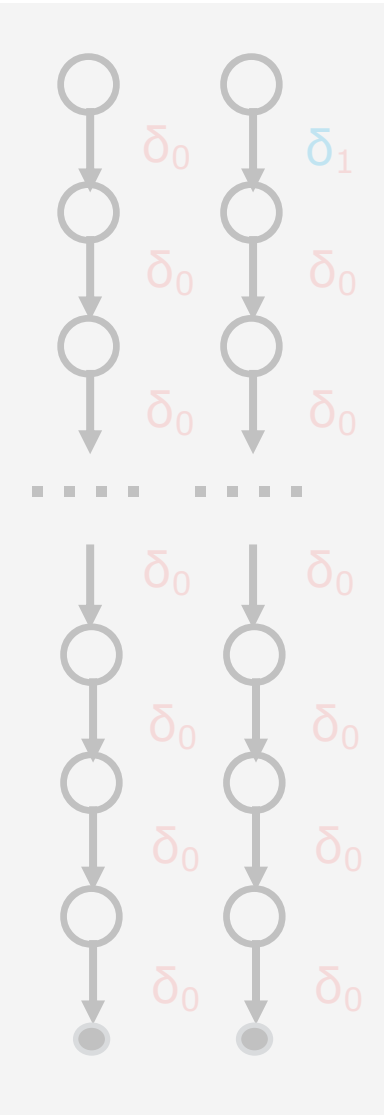


決定性チューリングマシン

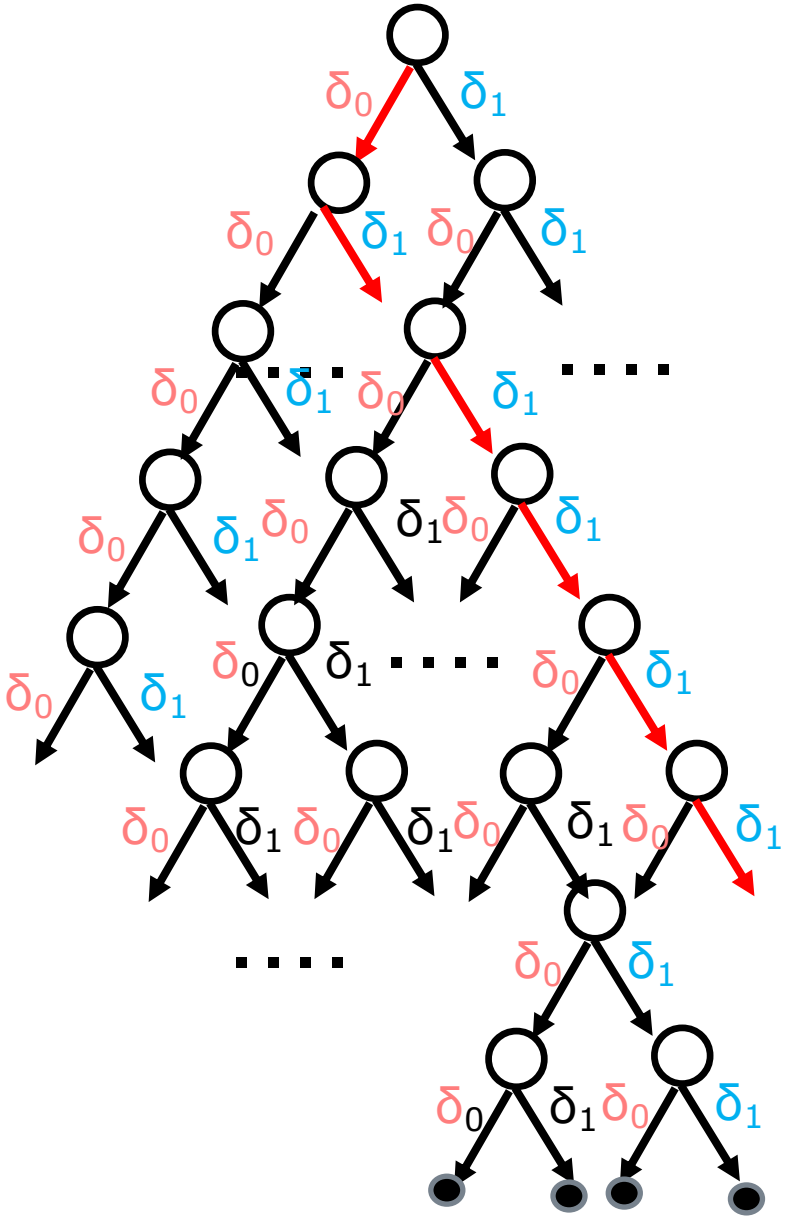
非決定性チューリングマシン



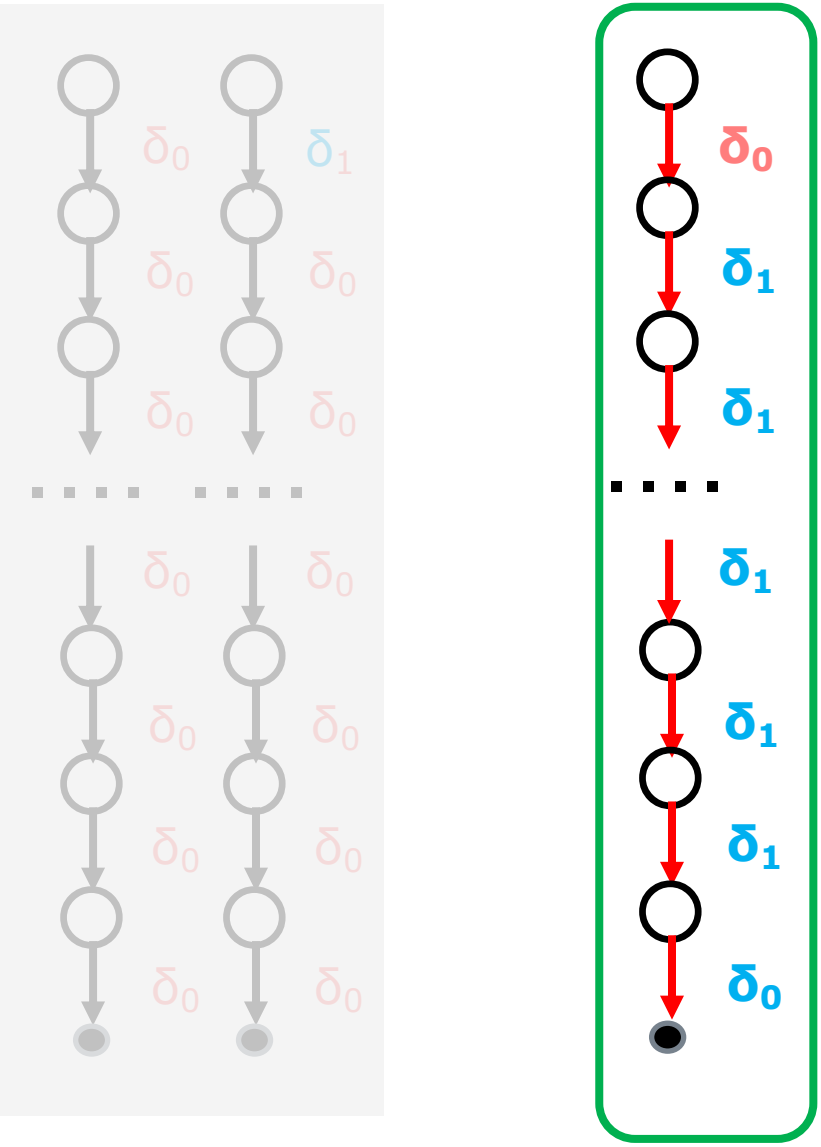
決定性チューリングマシン



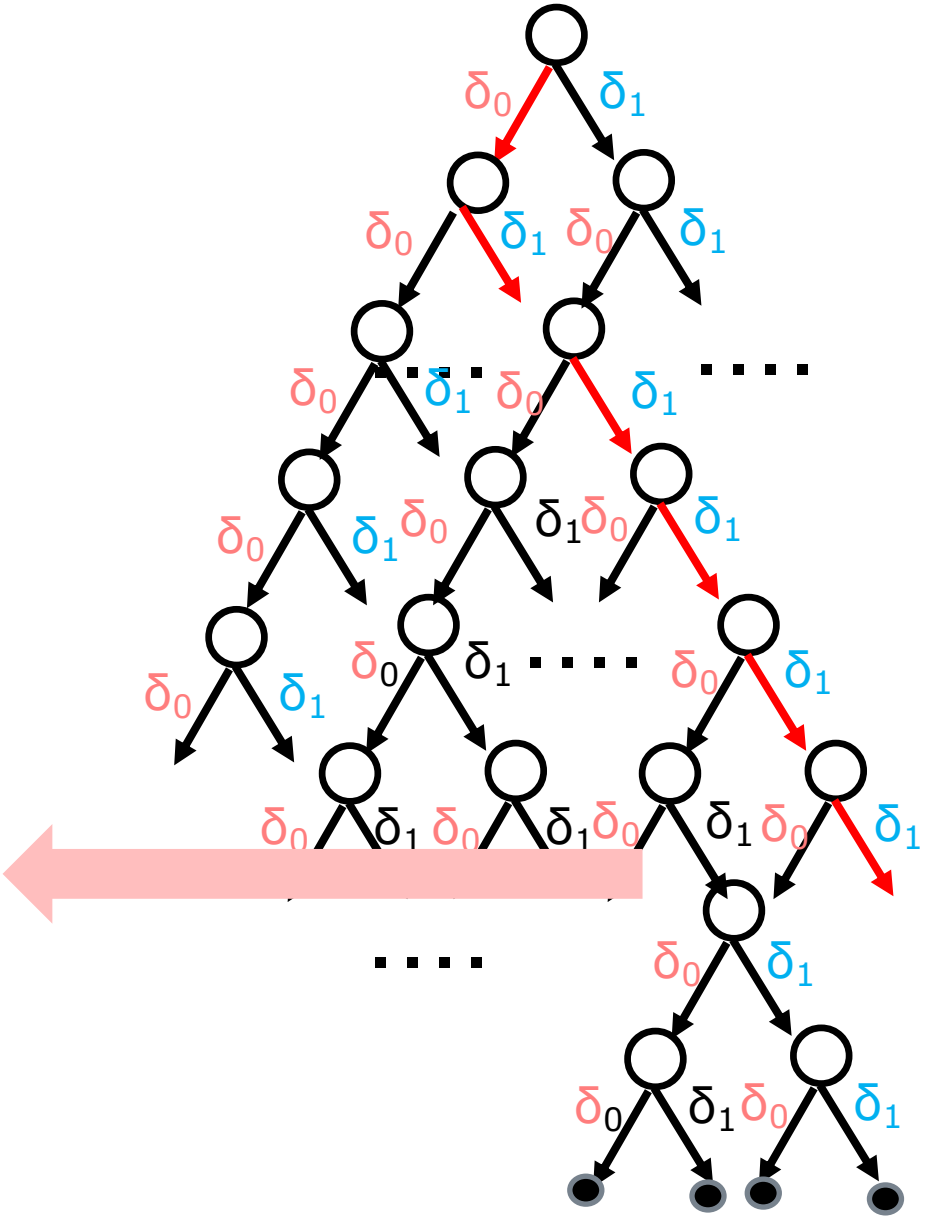
非決定性チューリングマシン



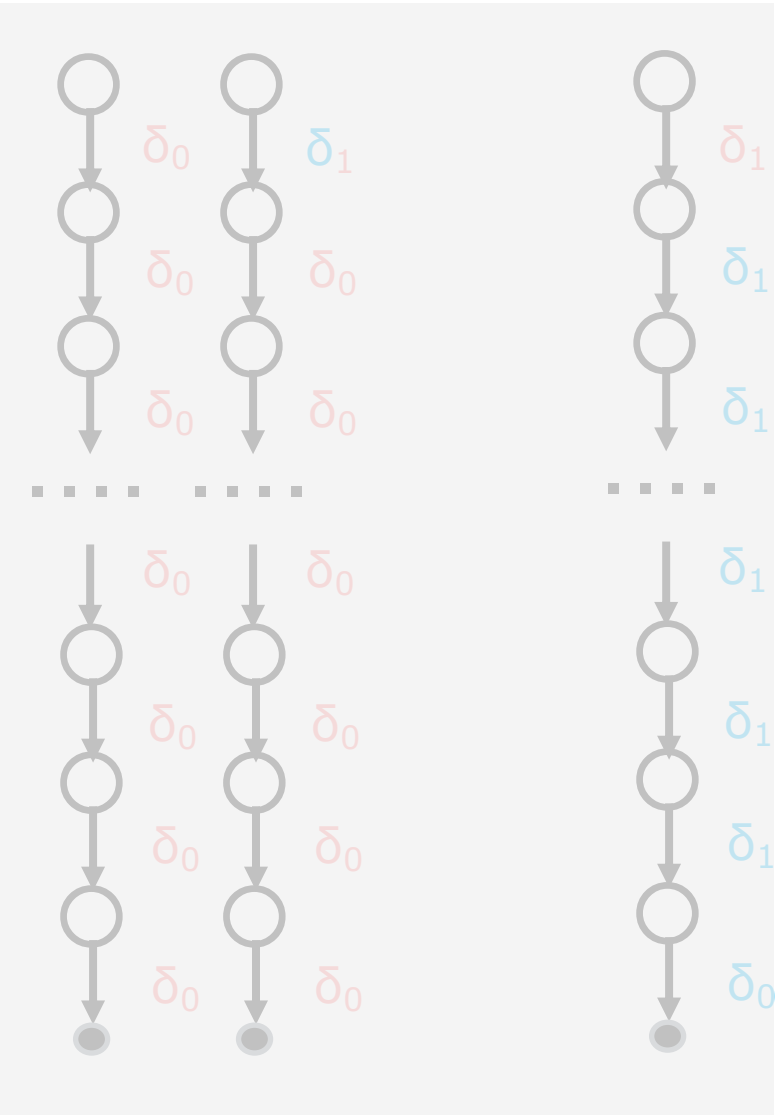
決定性チューリングマシン



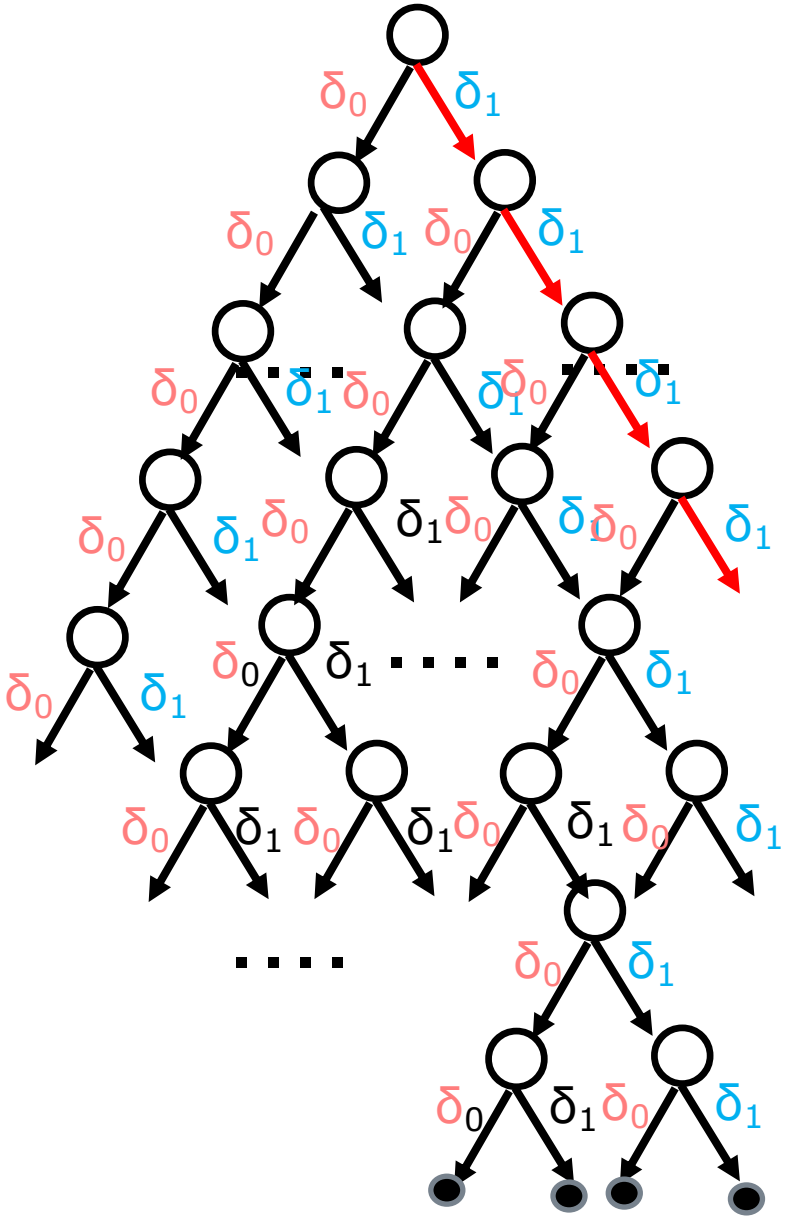
非決定性チューリングマシン



決定性チューリングマシン

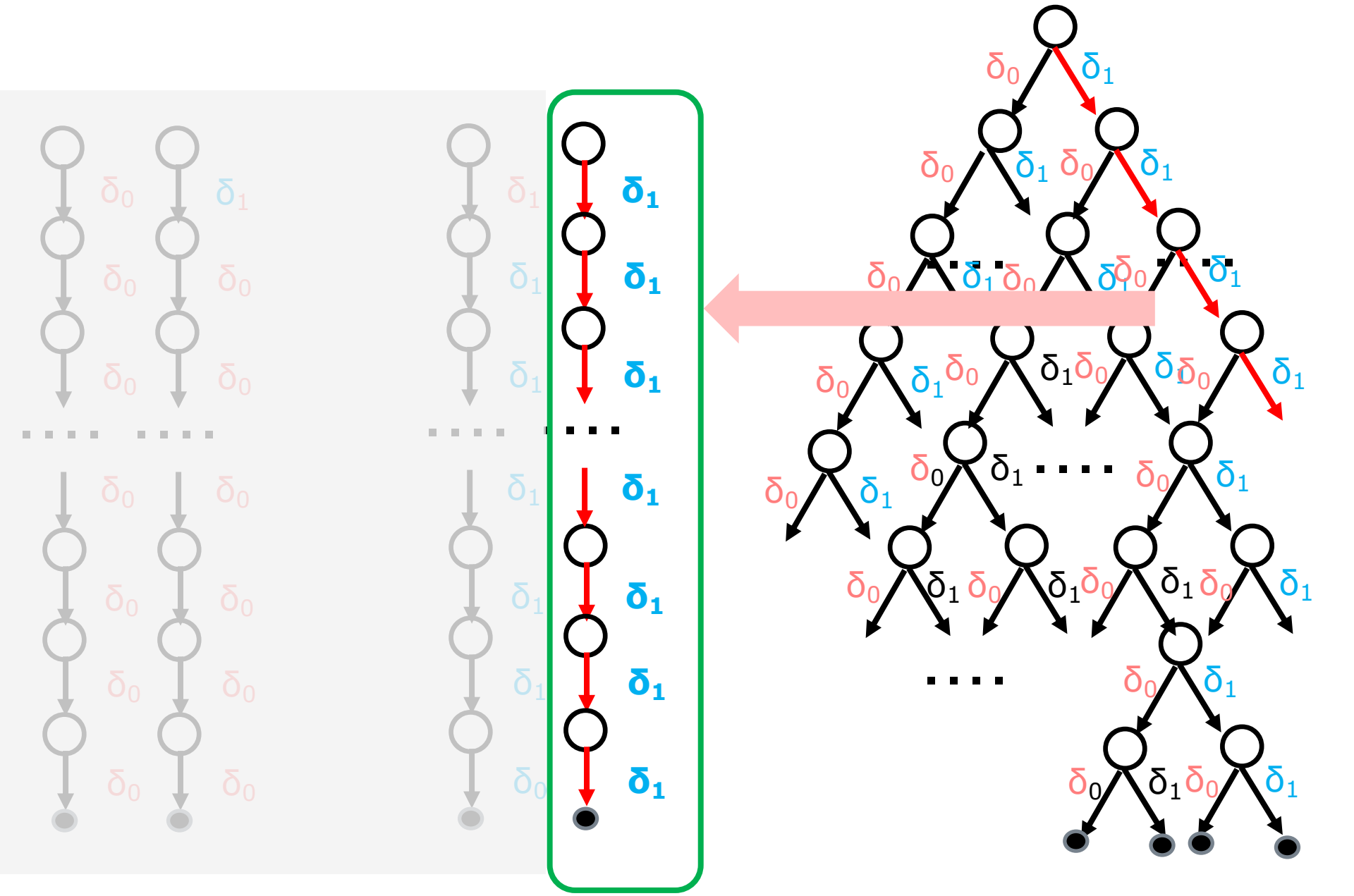


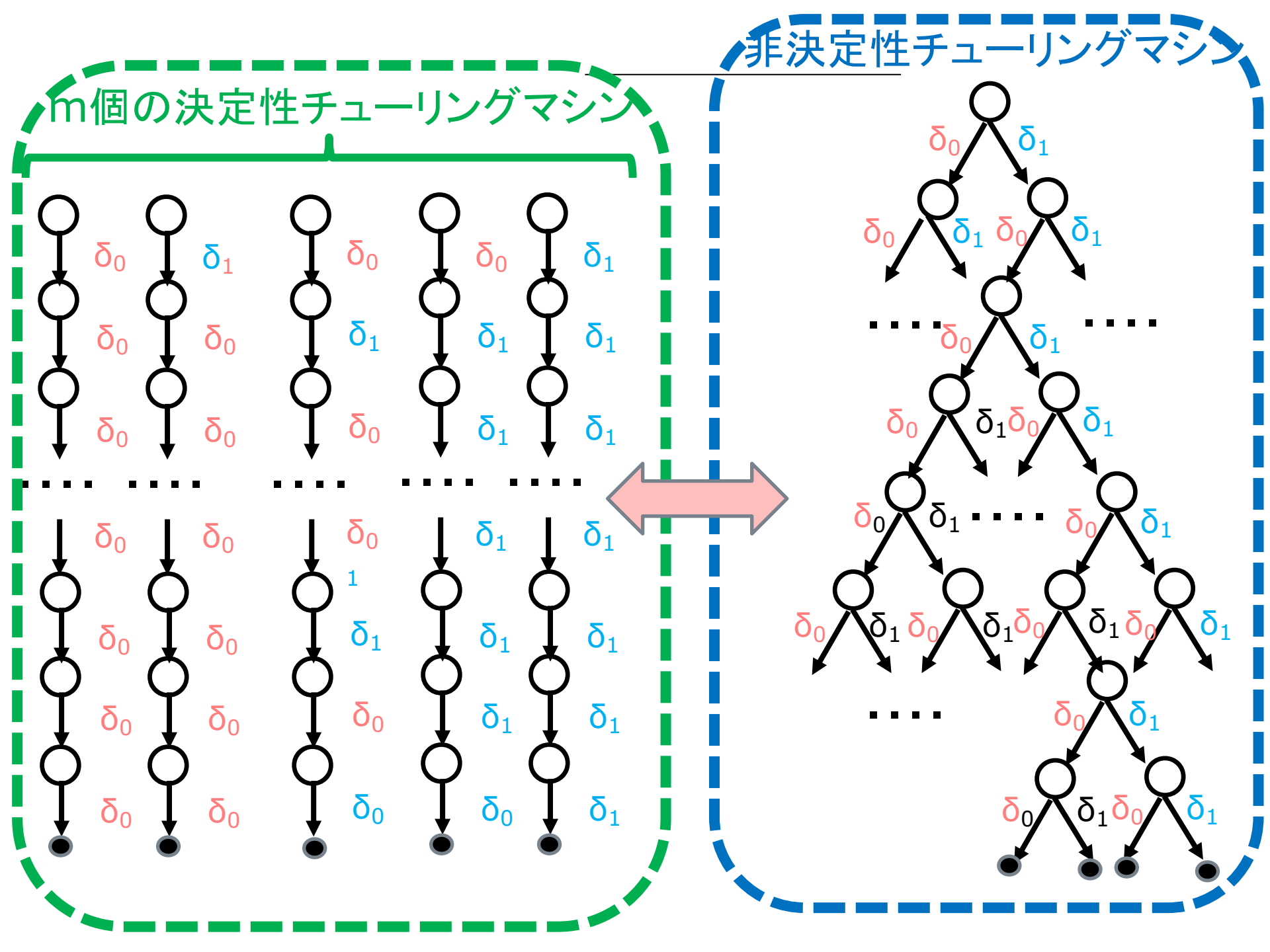
非決定性チューリングマシン



決定性チューリングマシン

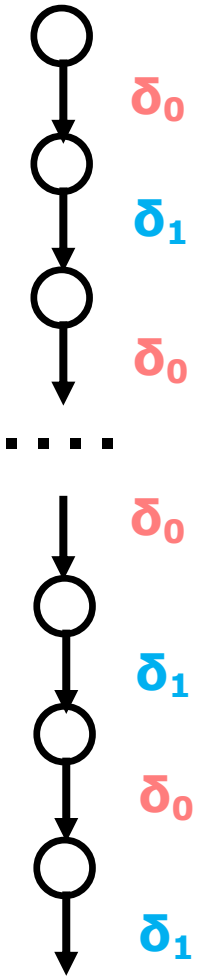
非決定性チューリングマシン




m個の決定性チューリングマシン
非決定性チューリングマシン

三つのテープを持つ
決定性チューリングマシンでシミュレートする

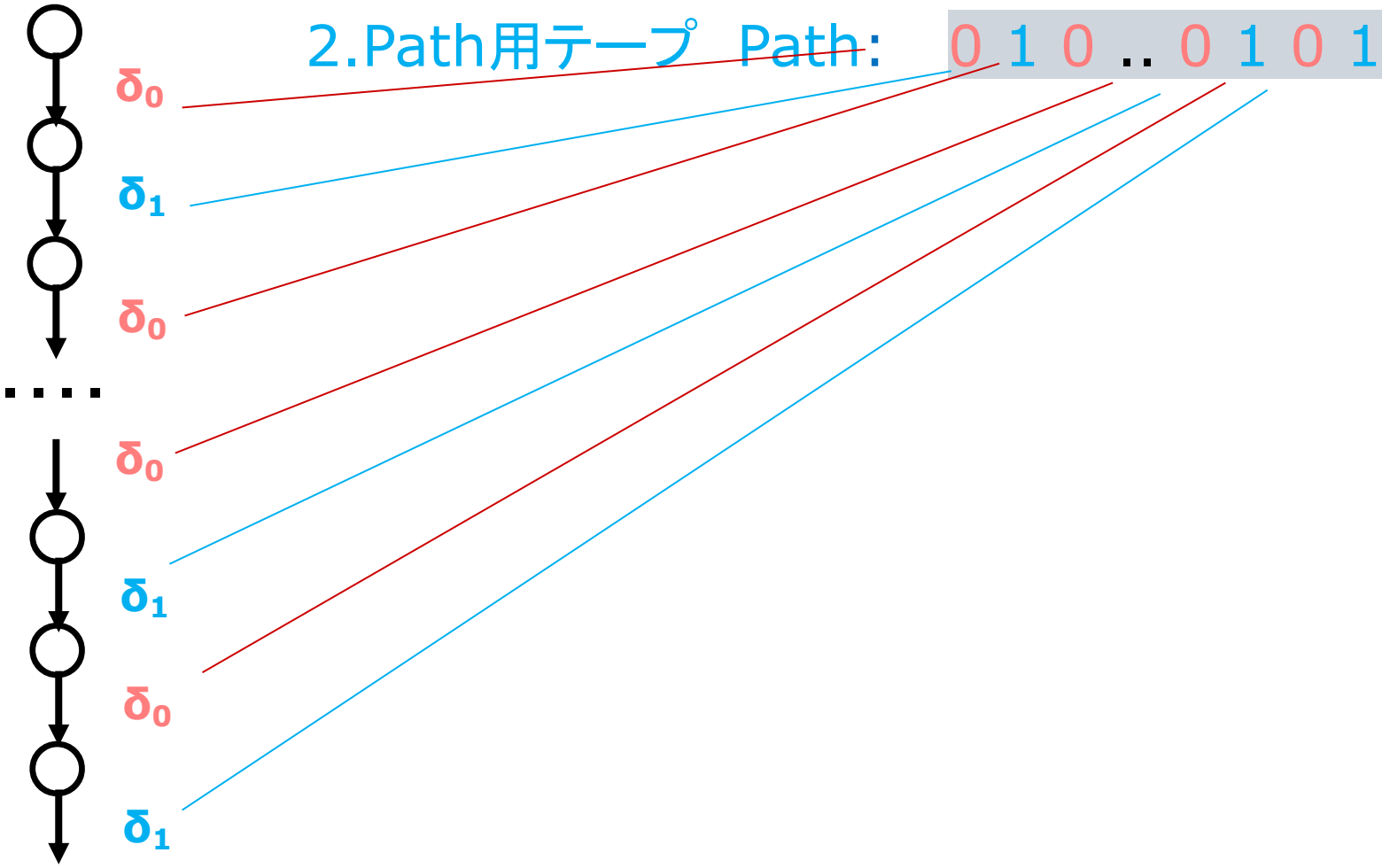
1.入力用テープ　Input:　1 0 1 1 0 ..



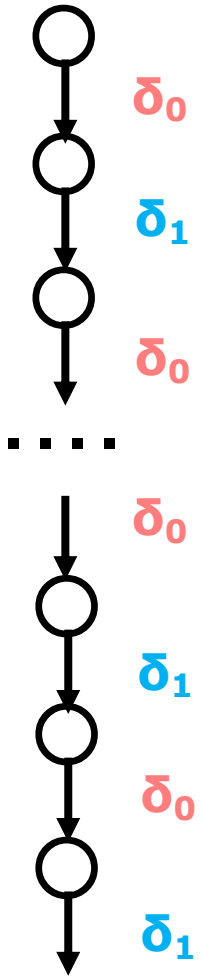
三つのテープを持つ
決定性チューリングマシンでシミュレートする

1.入力用テープ　Input:　1 0 1 1 0 ..

2.Path用テープ　Path:　0 1 0 .. 0 1 0 1



三つのテープを持つ
決定性チューリングマシンでシミュレートする



1.入力用テープ　Input:　1 0 1 1 0 ..

2.Path用テープ　Path:　0 1 0 .. 0 1 0 1

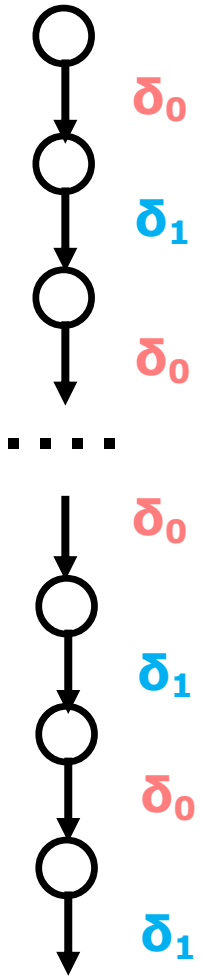
3.出力用テープ　Output:

....

<Path n>Path nの出力
<Path n’>Path n’の出力
<Path n’’>Path n’’の出力

....

三つのテープを持つ
決定性チューリングマシンでシミュレートする



1.入力用テープ　Input:　1 0 1 1 0 ..

2.Path用テープ　Path:　0 1 0 .. 0 1 0 1

3.出力用テープ　Output:

....

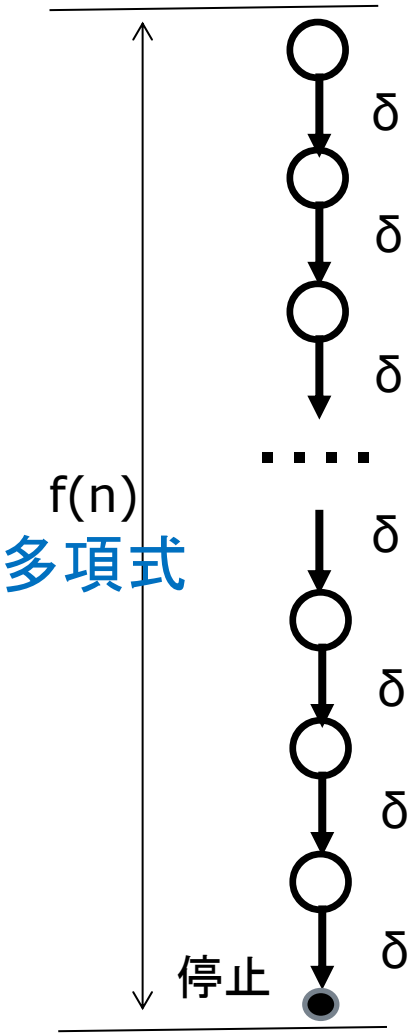
<Path n>Path nの出力
<Path n'>Path n'の出力
<Path n''>Path n''の出力

....

Pathテープが可能なパスを全て枚挙するようにプログラムし(幅優先or深さ優先)、Inputが受理された時点で停止する。
全てのパスで不受理なら不受理とする。

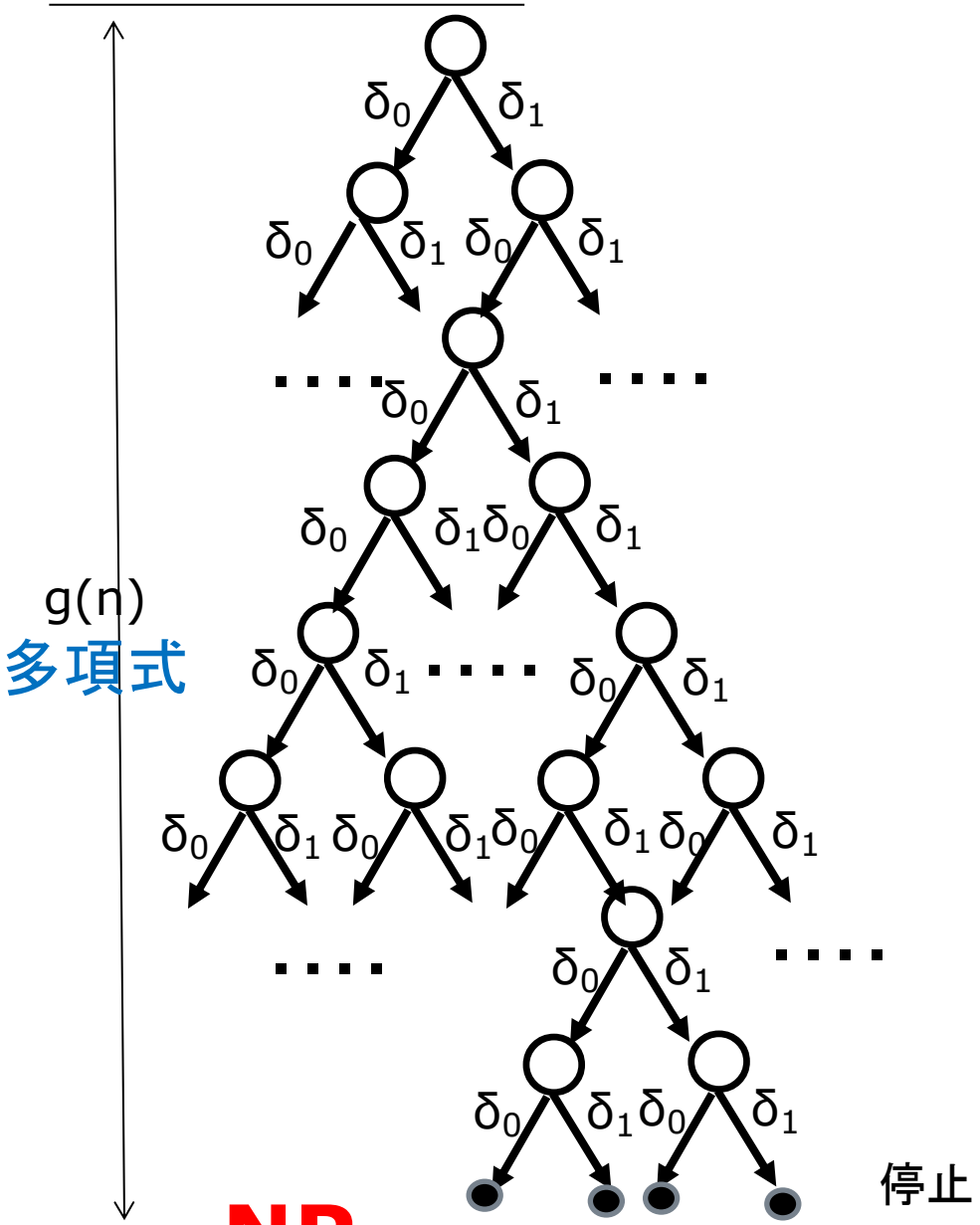
$$P \subseteq NP$$

決定性チューリングマシン



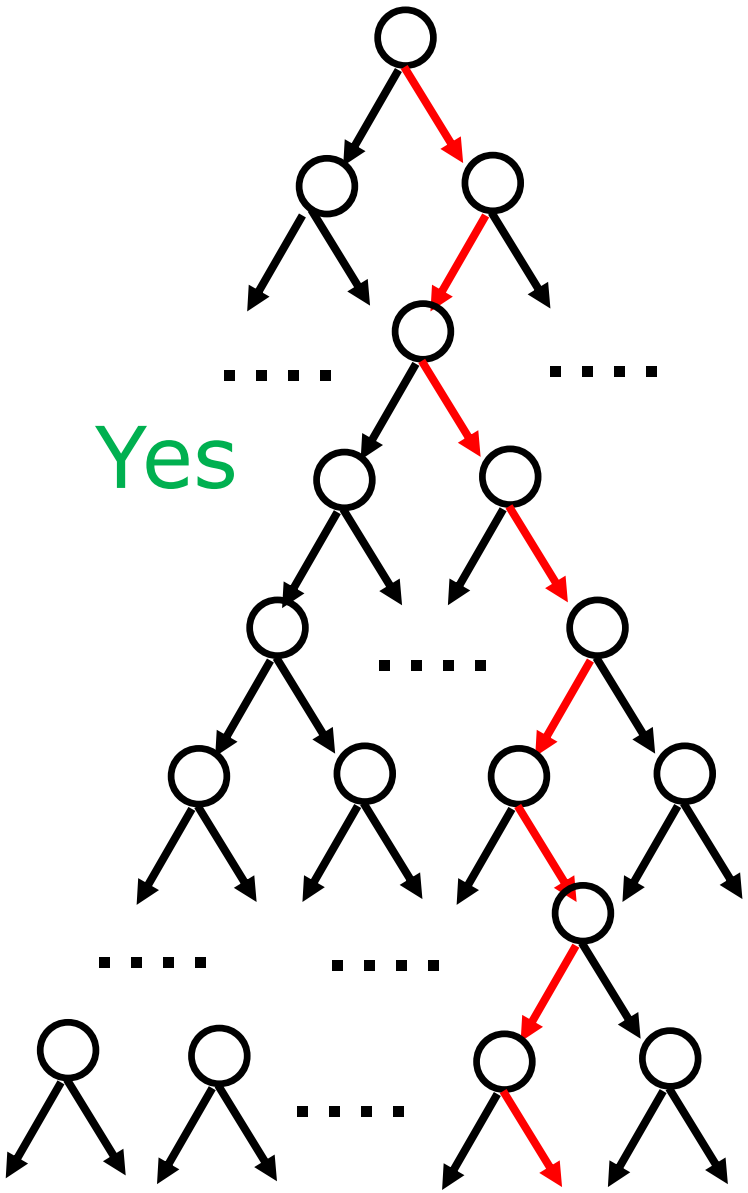
P

非決定性チューリングマシン

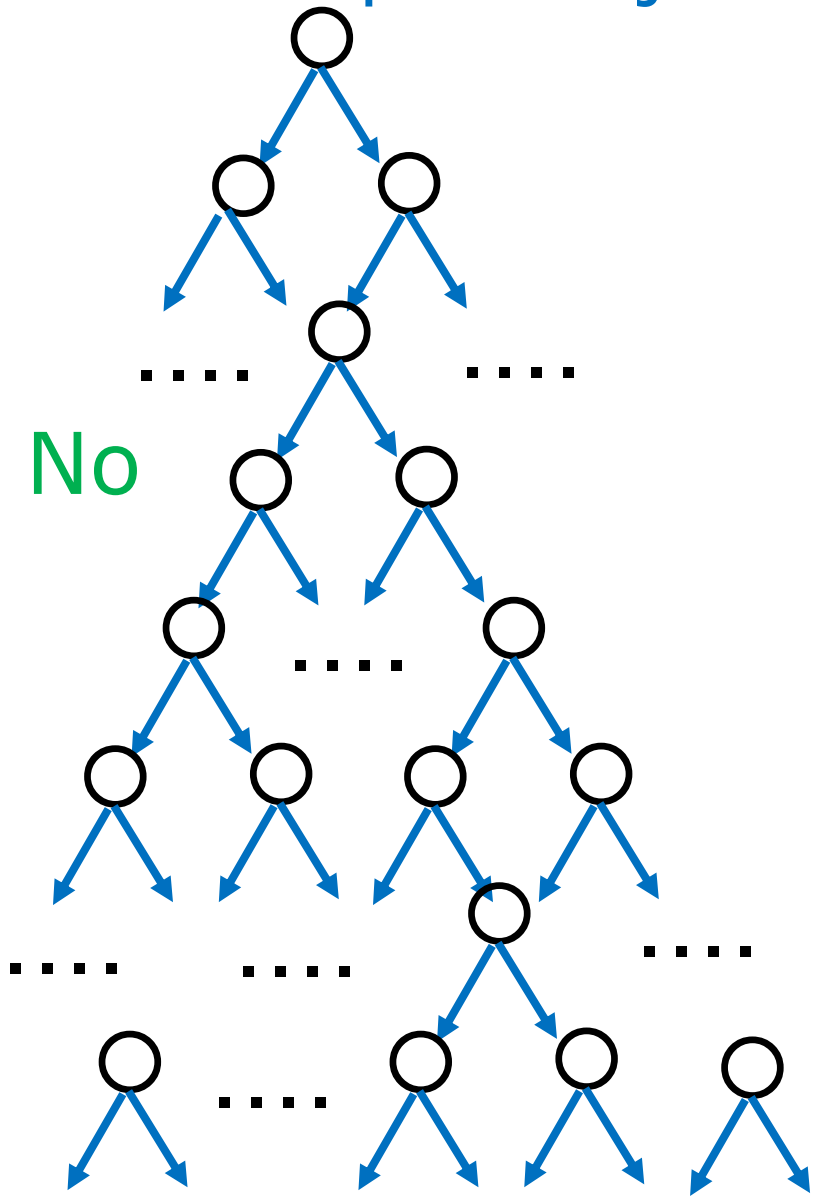


NP

非決定性チューリングマシンのAccept と Reject

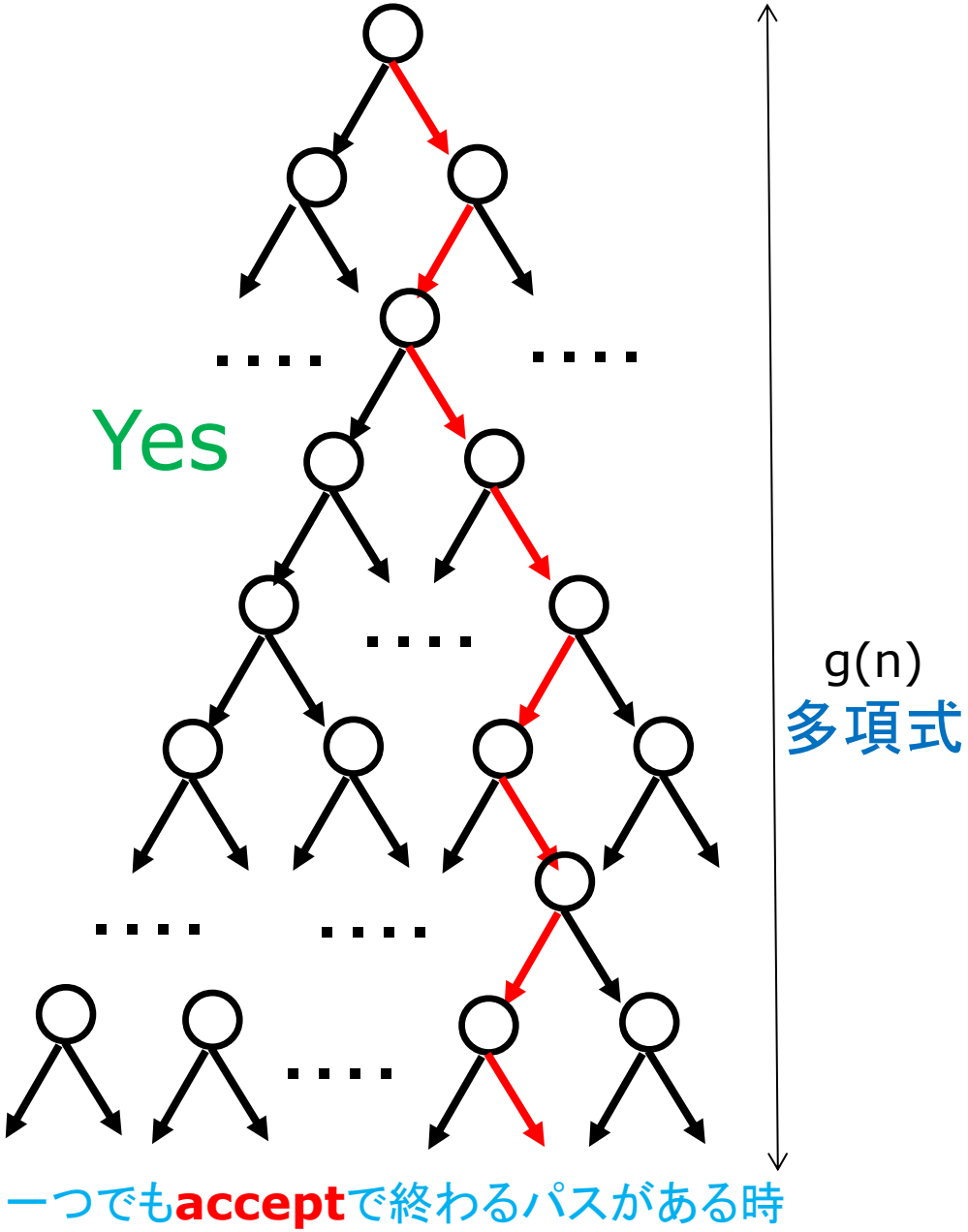


一つでも**accept**で終わるパスがある時



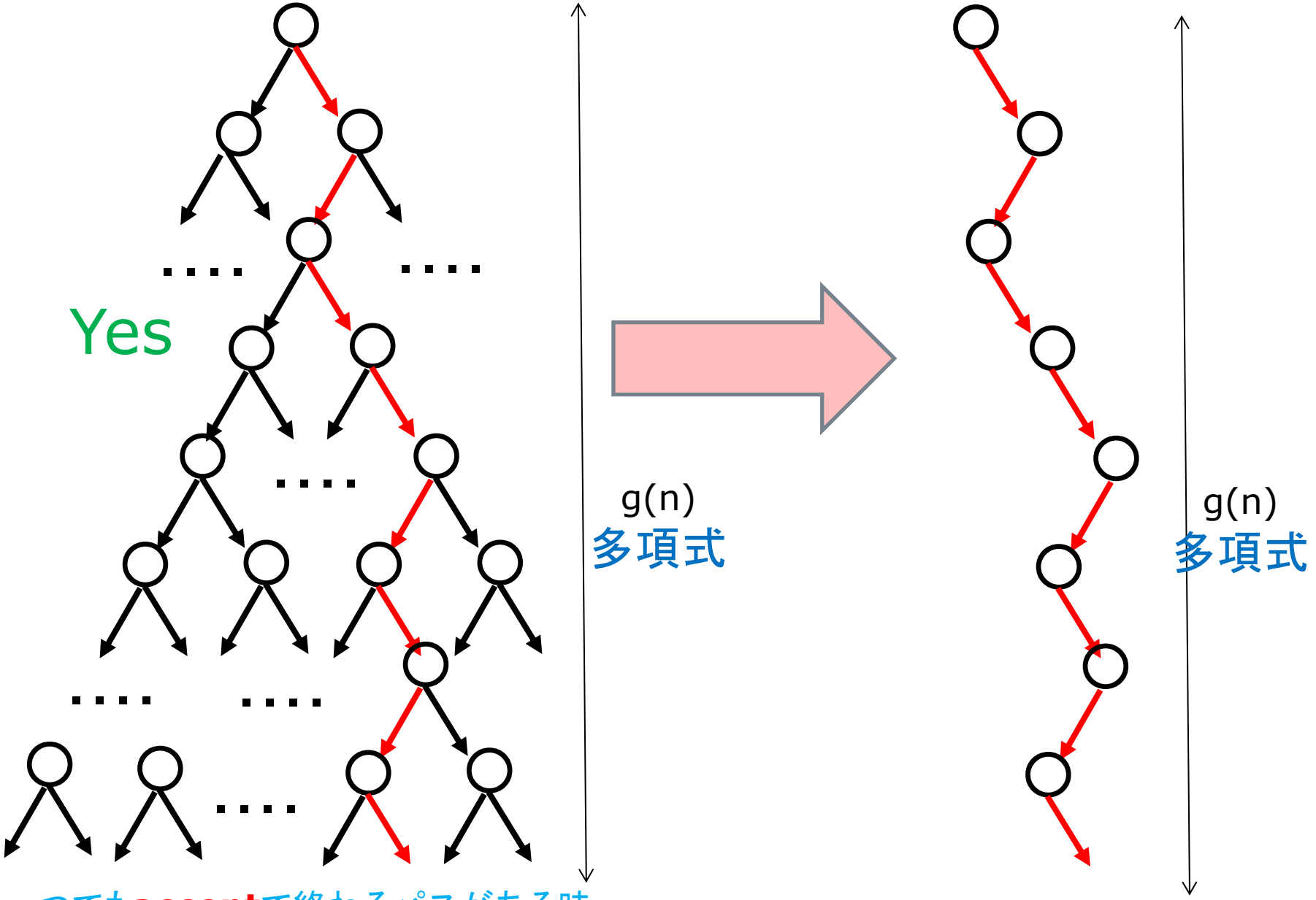
全てのパスが**reject** で終わる場合

NP：非決定性チューリングマシンで受理される場合

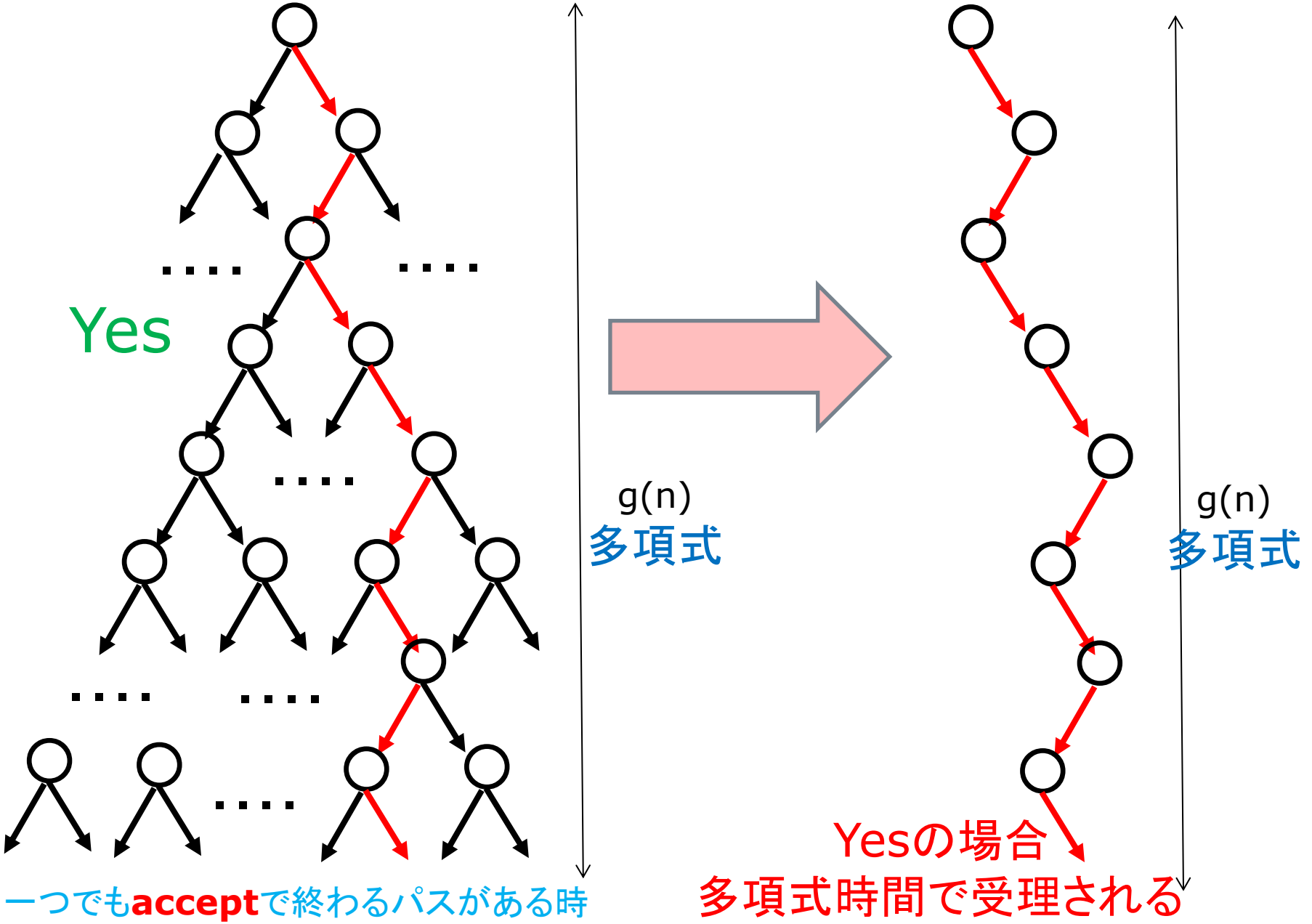


一つでも**accept**で終わるパスがある時

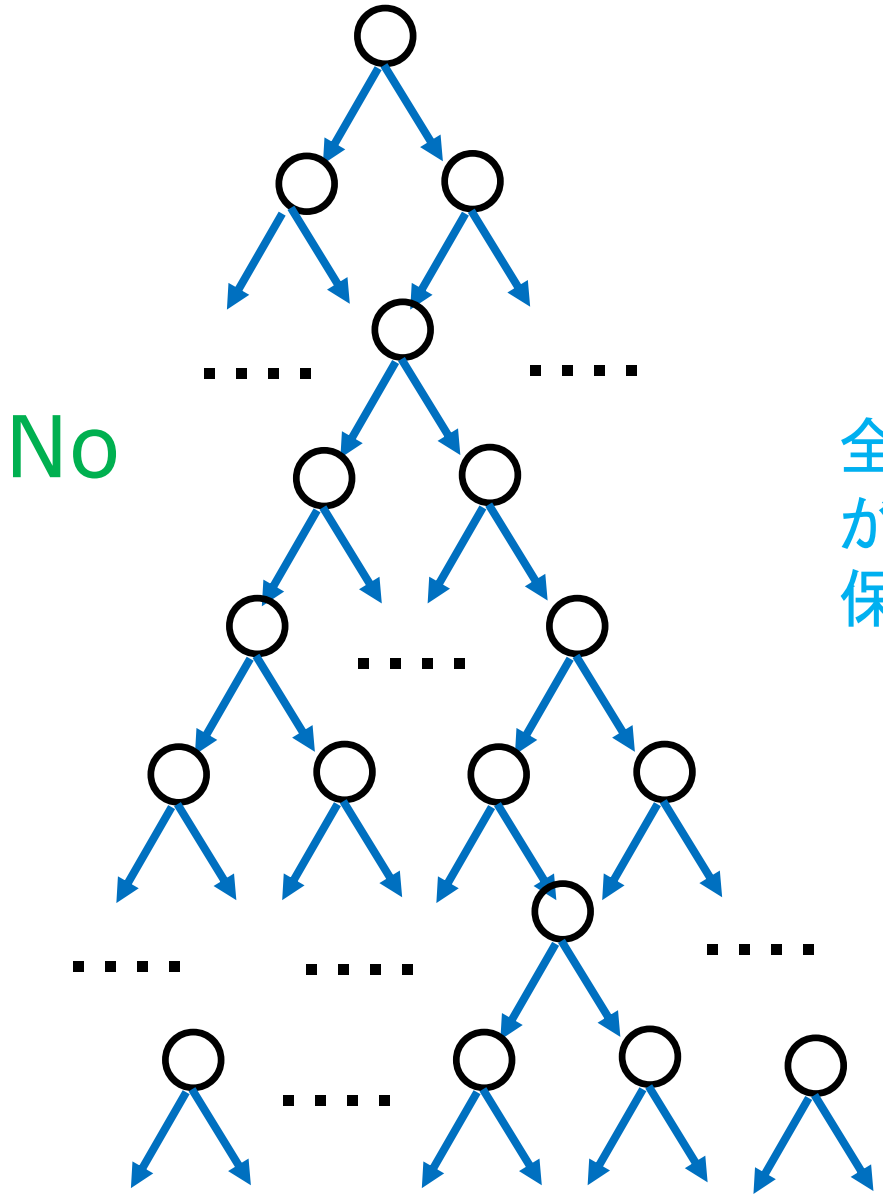
NP：非決定性チューリングマシンで受理される場合



NP：非決定性チューリングマシンで受理される場合



非決定性チューリングマシンで受理されない場合



Noの場合
全てのパスのチェック
が多項式時間で終わる
保証はない

No: 全てのパスが**reject** で終わる場合

決定性チューリングマシンは、
非決定性チューリングマシンの
特別な場合と考えることができる
ので、$P \subseteq NP$である。


確率的チューリングマシンとBPPクラス

確率的チューリングマシンとBPPクラス

今回のセッションでは、「確率的チューリングマシン」と呼ばれるチューリングマシンの拡大の話をします。

「確率的チューリングマシン」で定義される複雑性のクラスをBPPといいます。

「確率的チューリングマシン」の構成の仕方は、前回見た「非決定性チューリングマシン」の構成とよく似ています。

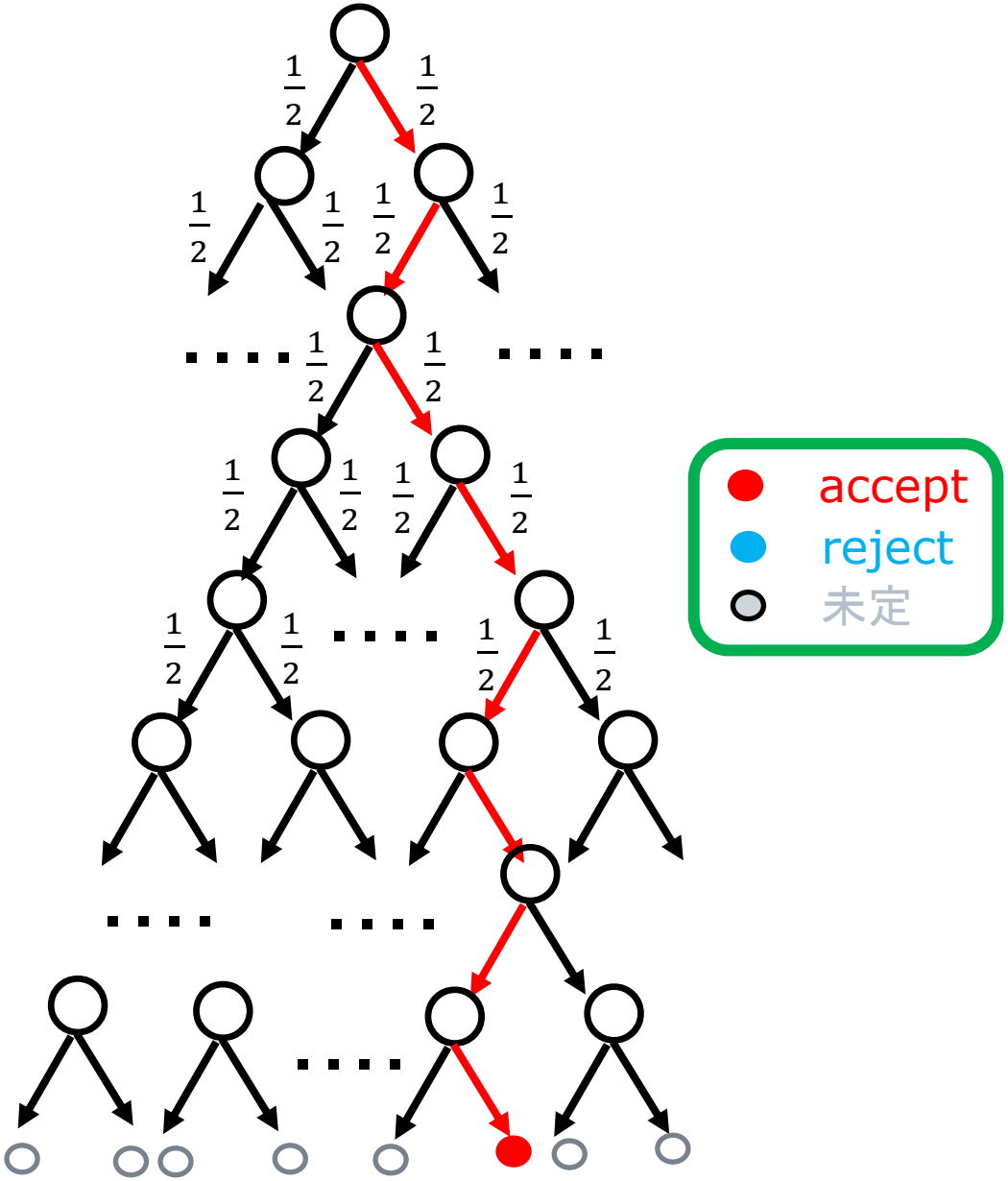
ただ、accept, rejectの条件が大きく変わっています。その定義には、「確率」が導入されています。

「確率的チューリングマシン」PTM では、あるパス上のDTMが入力x をacceptすることを見つけたとしても、PTM自身が xをacceptするわけではないのです。

PTMがxをacceptするためには、PTMのツリー全体を見渡して（それは、非決定性チューリングマシンと同じように巨大なものです）、そのなかでacceptするDTMの比率が、rejectするDTMの比率よりも高いことを言わなければなりません。

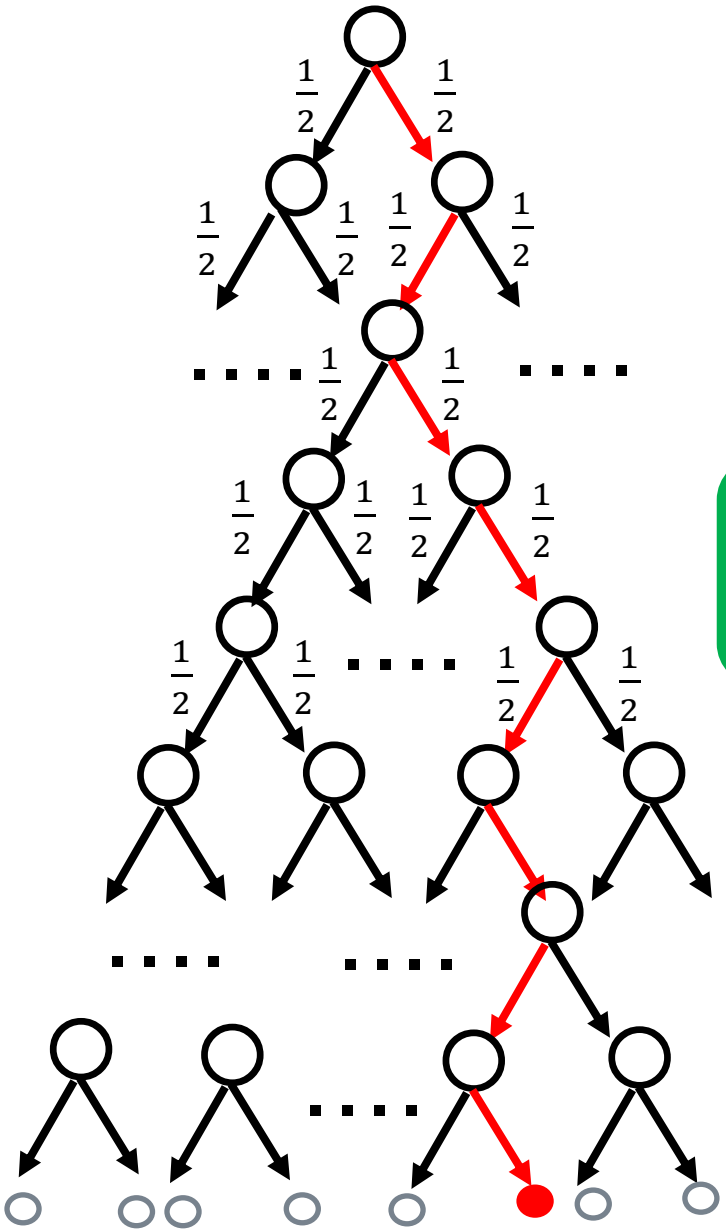
非決定性チューリングマシンと確率的チューリングマシン

非決定性チューリングマシン



一つでもacceptされれば、accept

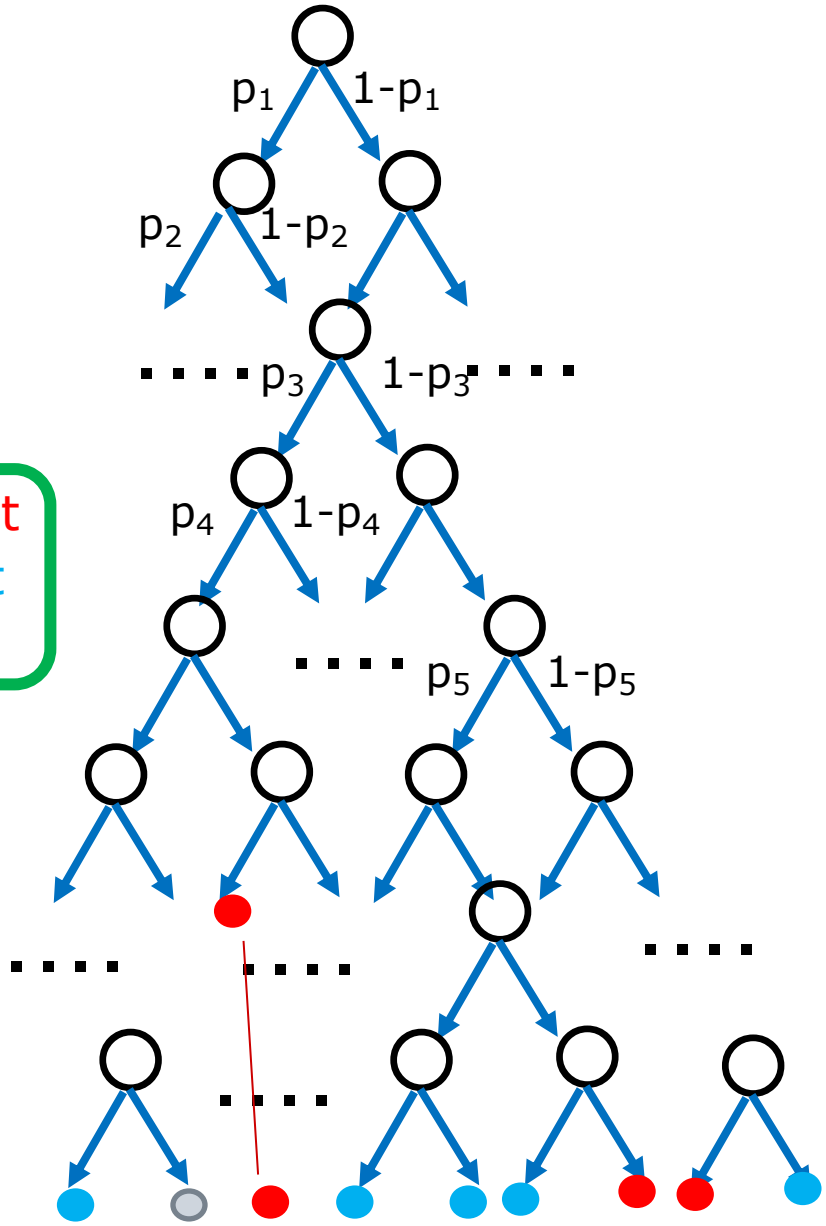
非決定性チューリングマシン



一つでもacceptされれば、accept

accept
reject
未定

確率的チューリングマシン

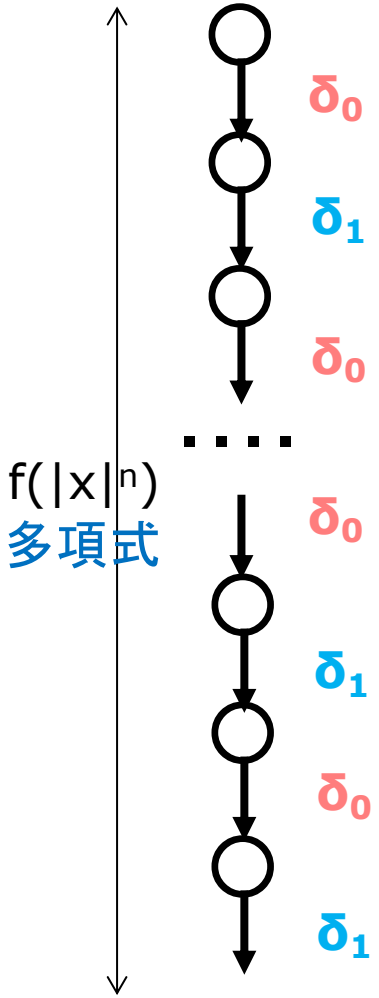


acceptとrejectの確率を考える

NPクラス

入力用テープ Input:
x = 1 0 1 1 0 ..

Path用テープ Path:
p = 0 1 0 .. 0 1 0 1

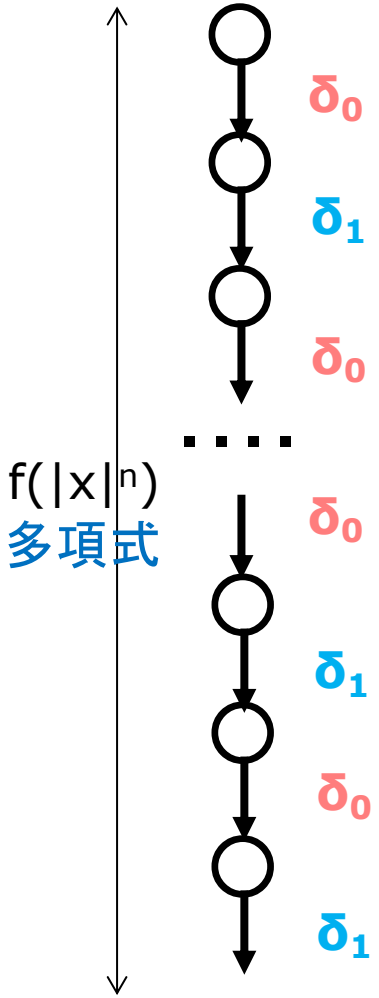


出力用テープ Output:
1:accept, 0:reject

NPクラス

入力用テープ Input: x = 1 0 1 1 0 ..

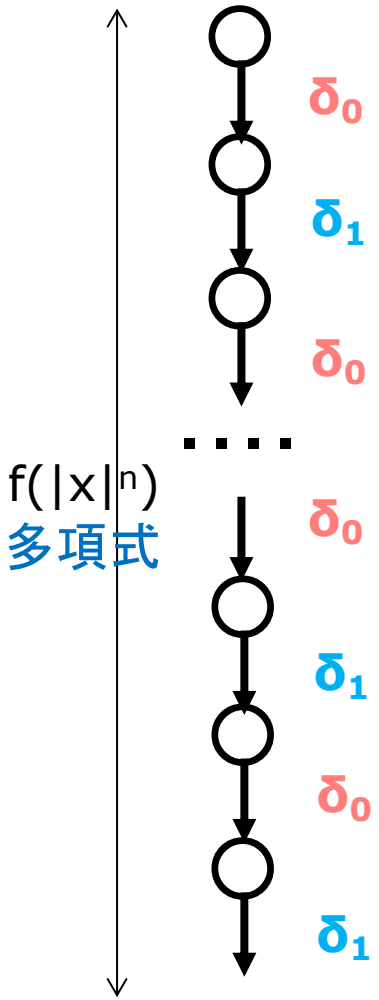
Path用テープ Path: p = 0 1 0 .. 0 1 0 1



xとpを入力とする決定性チューリングマシンMがあって、次の条件を満たすとき、LはNPクラスに属する。

出力用テープ Output:
1:accept, 0:reject

NPクラス



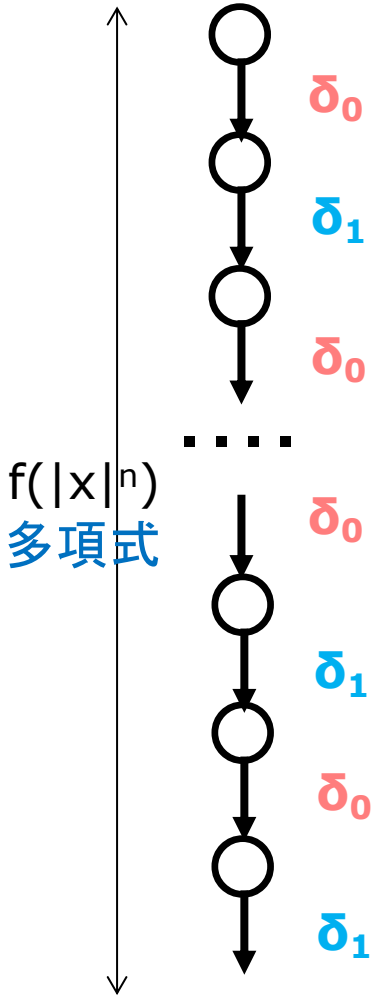
xとpを入力とする決定性チューリングマシンMがあって、次の条件を満たすとき、LはNPクラスに属する。

- 全ての入力(x, p)について、Mは多項式時間$f(|x|^n)$で走る。
- $x \in L$ なら、あるpが存在して、$M(x, p)=1$
- $x \notin L$ なら、全てのpに対して、$M(x, p)=0$

BPPクラス

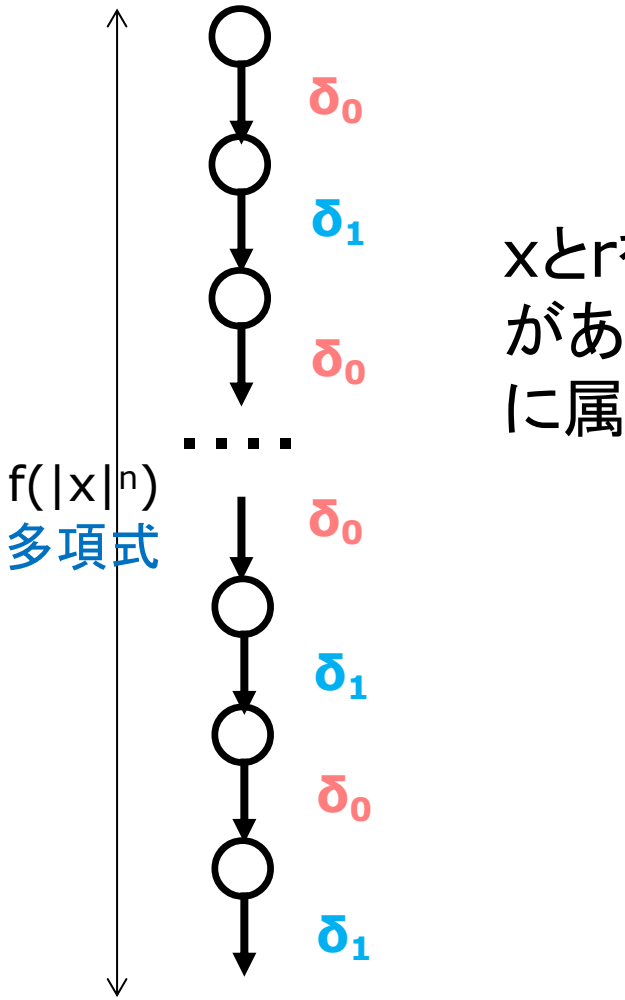
入力用テープ Input:
x = 1 0 1 1 0 ..

乱数用テープ Rand:
r = 0 1 0 .. 0 1 0 1



出力用テープ Output:
1:accept, 0:reject

BPPクラス

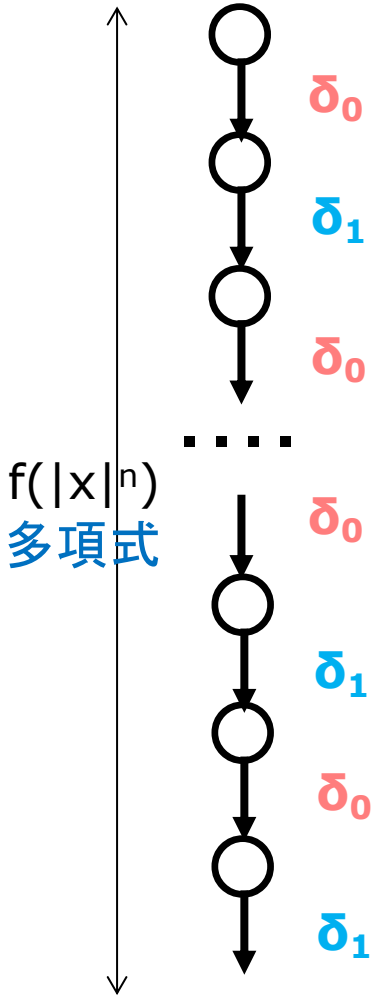


xとrを入力とする確率的チューリングマシンMがあって、次の条件を満たすとき、LはBPPクラスに属する。

BPPクラス

入力用テープ Input: x = 1 0 1 1 0 ..

乱数用テープ Rand: r = 0 1 0 .. 0 1 0 1

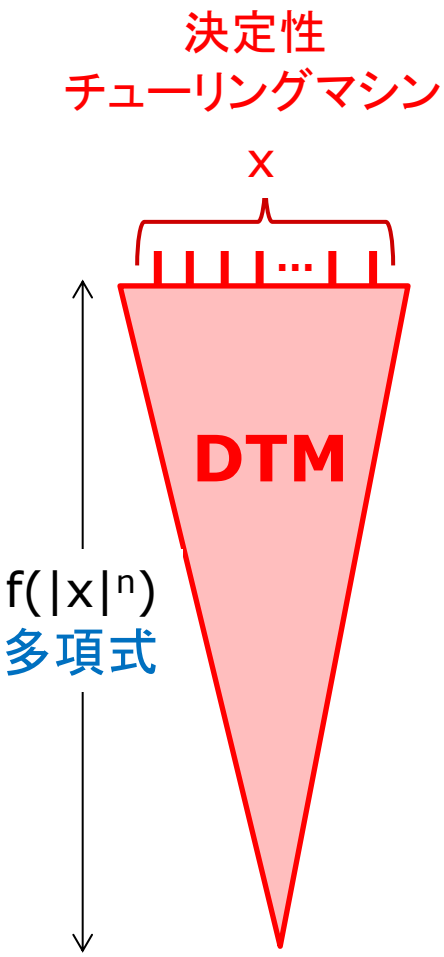


出力用テープ Output:
1:accept, 0:reject

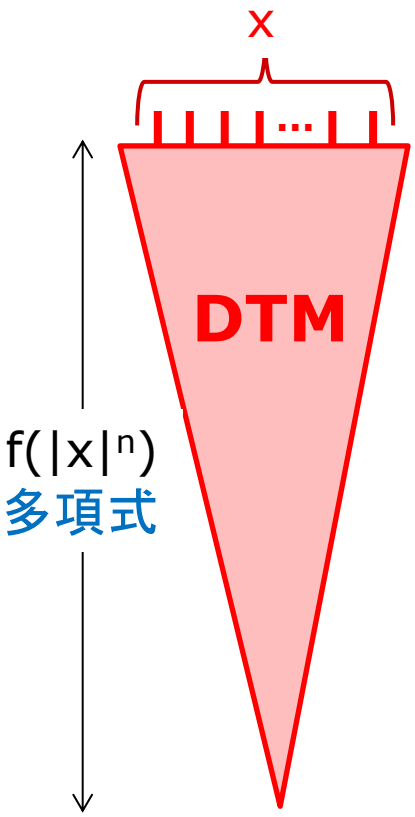
xとrを入力とする確率的チューリングマシンMがあって、次の条件を満たすとき、LはBPPクラスに属する。

- 全ての入力(x, p)について、Mは多項式時間$f(|x|^n)$で走る。
- $x \in L$ なら、$M(x, r)=1$である確率は2/3以上である。
- $x \notin L$ なら、$M(x, r)=0$である確率は1/3以下である。

これまで見たチューリングマシンと
それが受理する複雑性のクラス

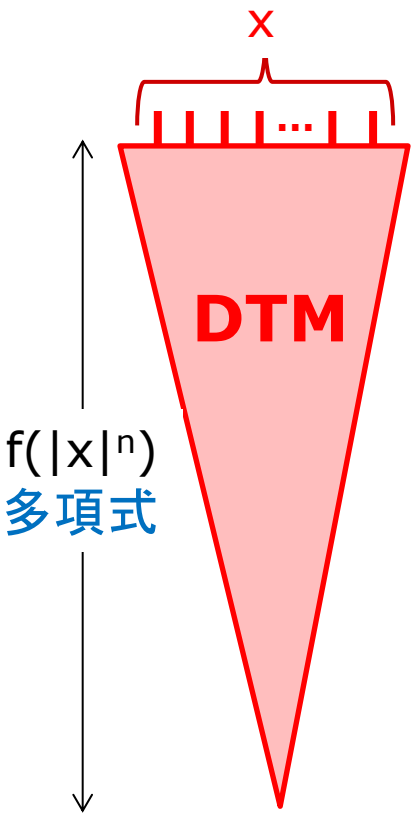

決定性
チューリングマシン
x
DTM
f(|x|ⁿ)
多項式

決定性
チューリングマシン



$x \in L$
$\rightarrow \mathrm{DTM}(x)=1$
$x \notin L$
$\rightarrow \mathrm{DTM}(x)=0$

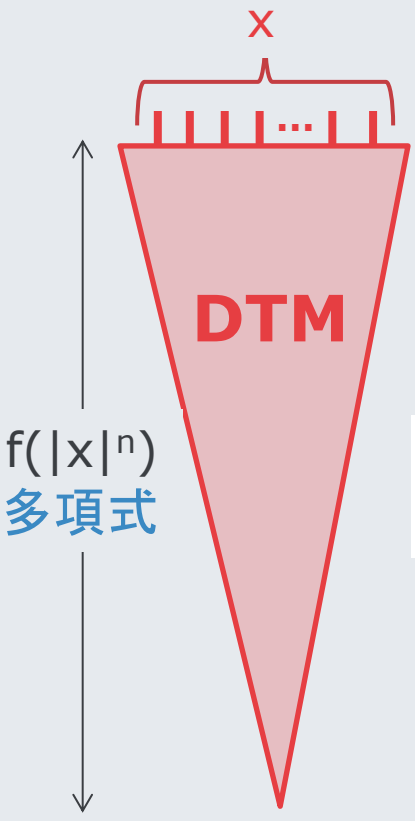
決定性
チューリングマシン



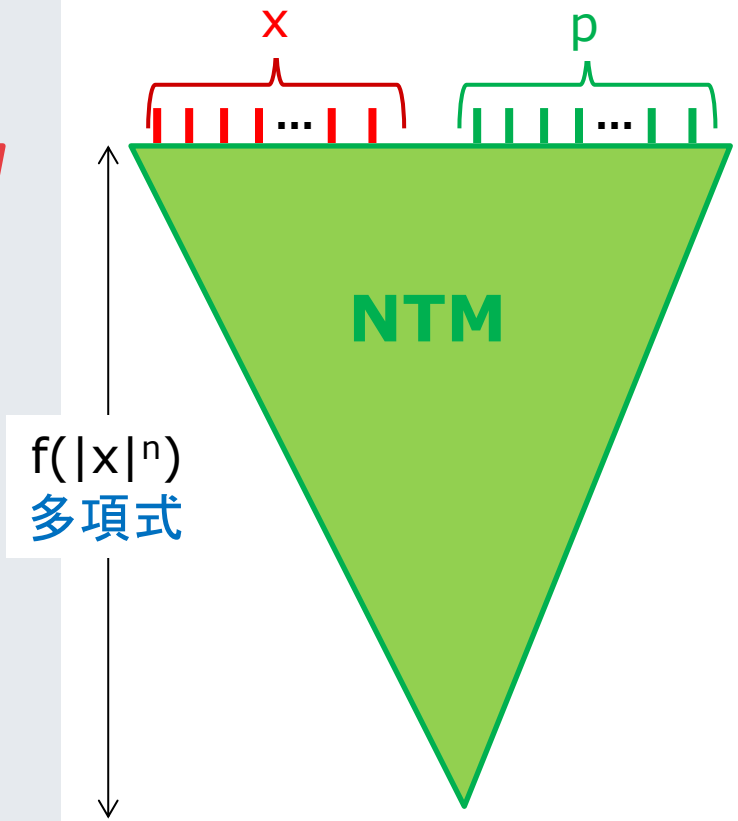
$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

決定性
チューリングマシン



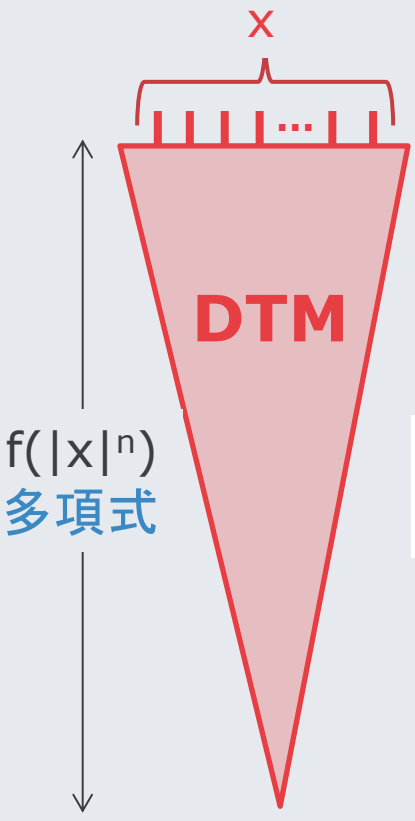
非決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

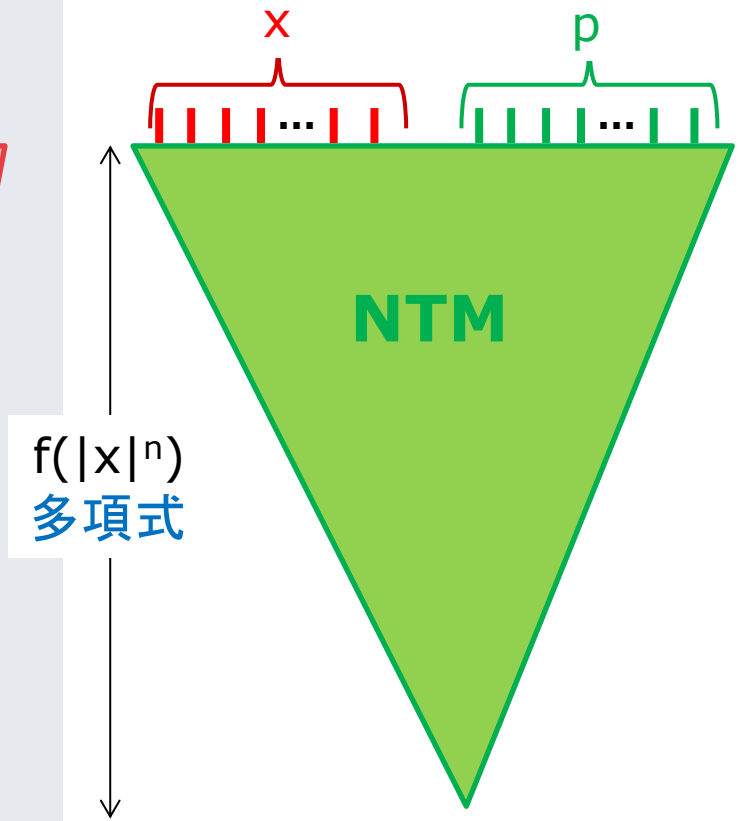
決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

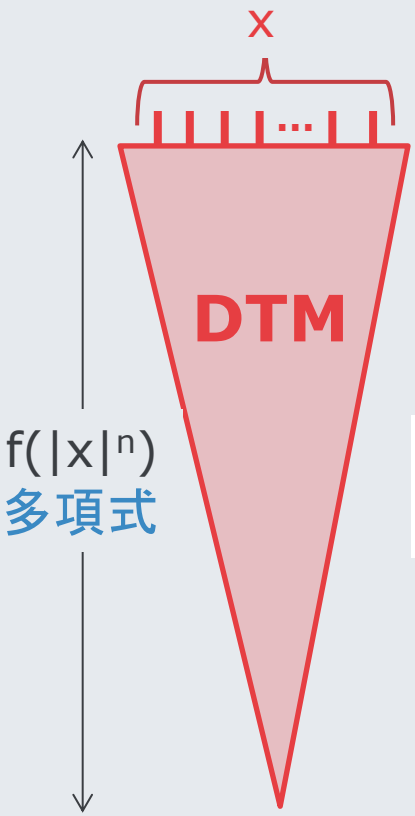
$L \in P$

非決定性
チューリングマシン



$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

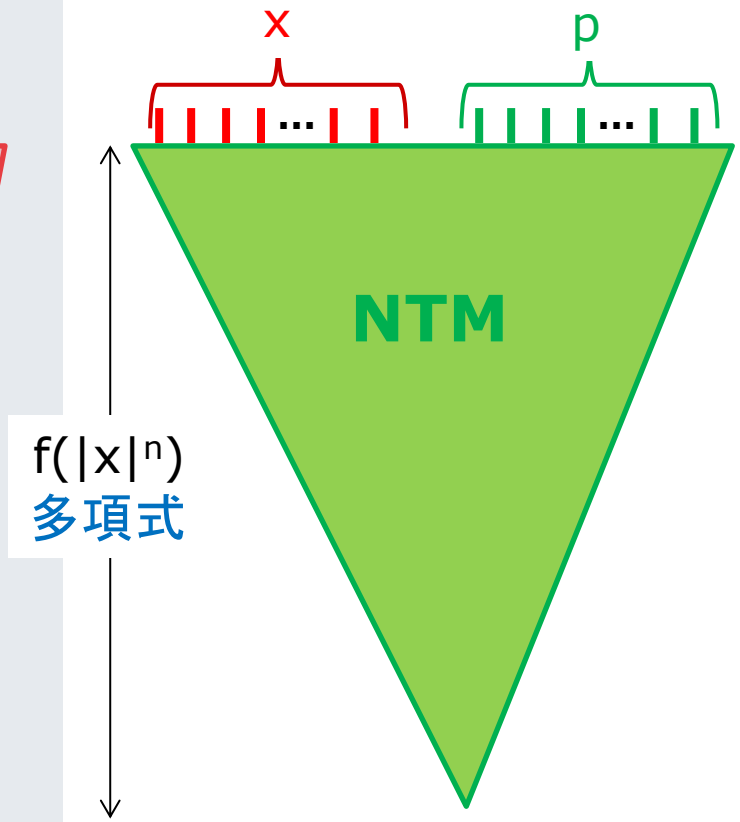
決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

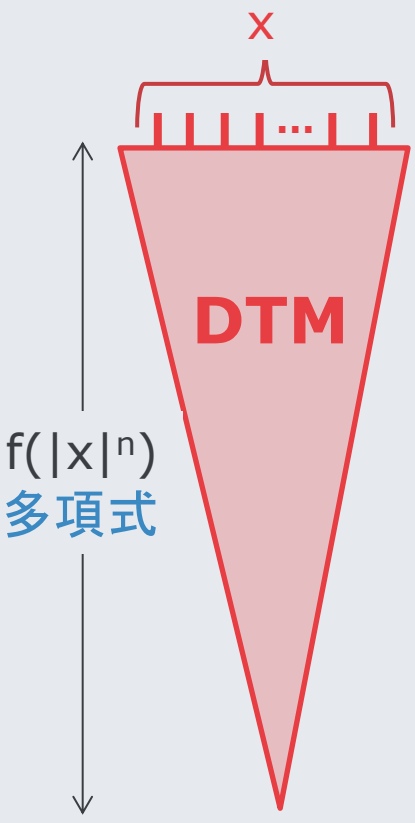
非決定性
チューリングマシン



$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

$L \in NP$

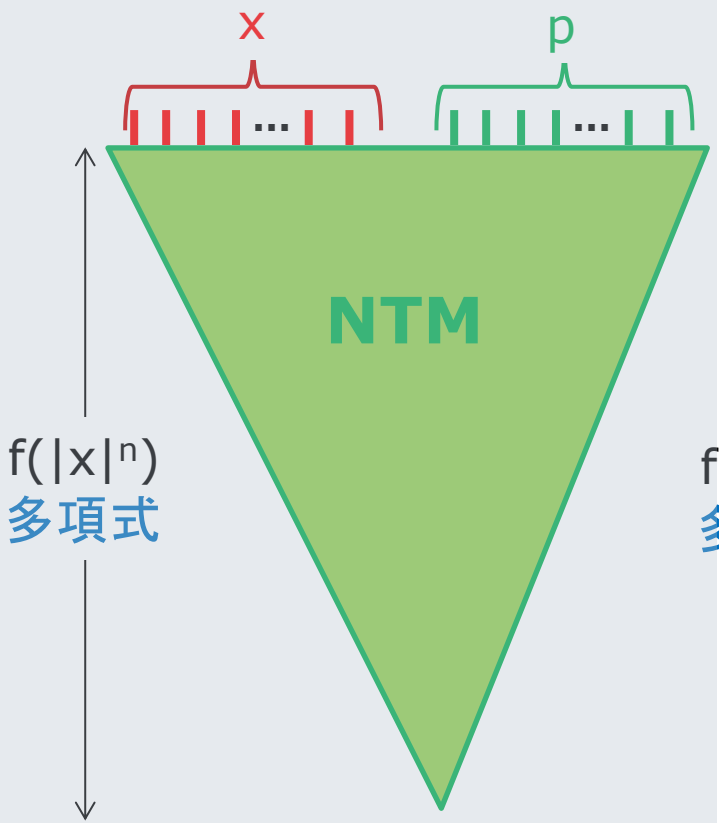
決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

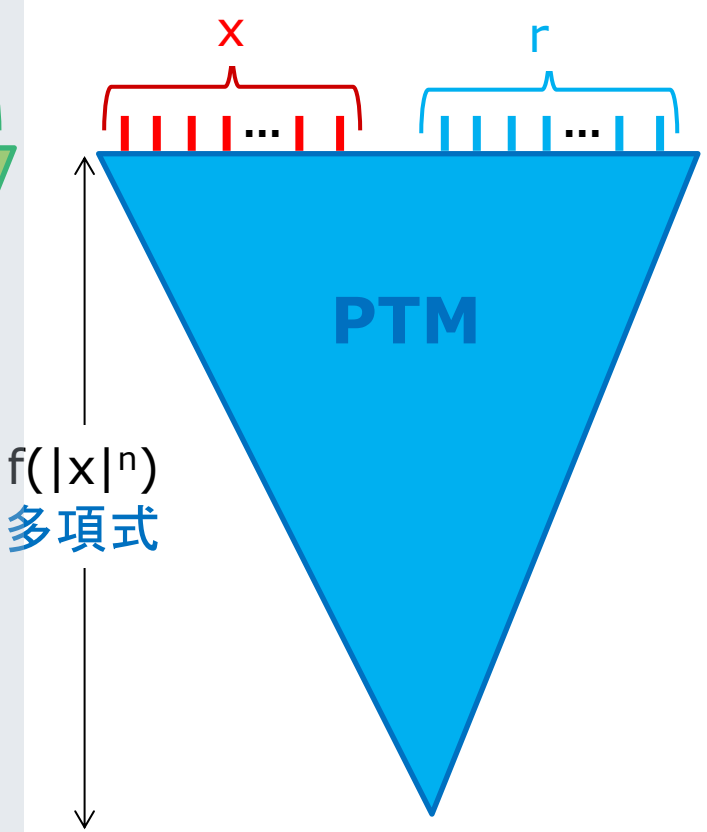
非決定性
チューリングマシン



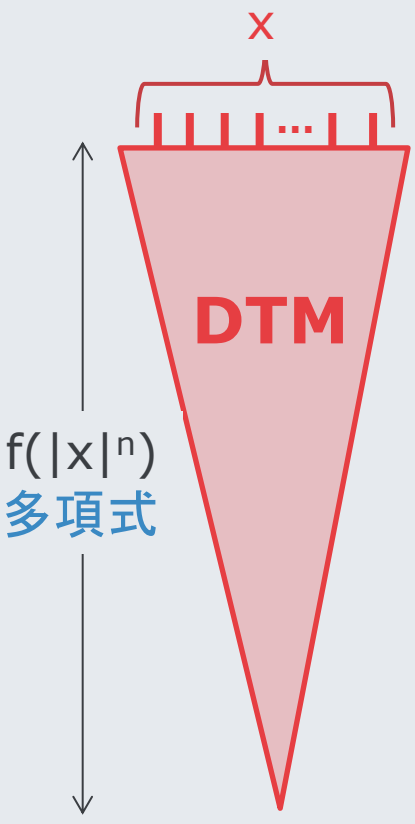
$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

$L \in NP$

確率的
チューリングマシン



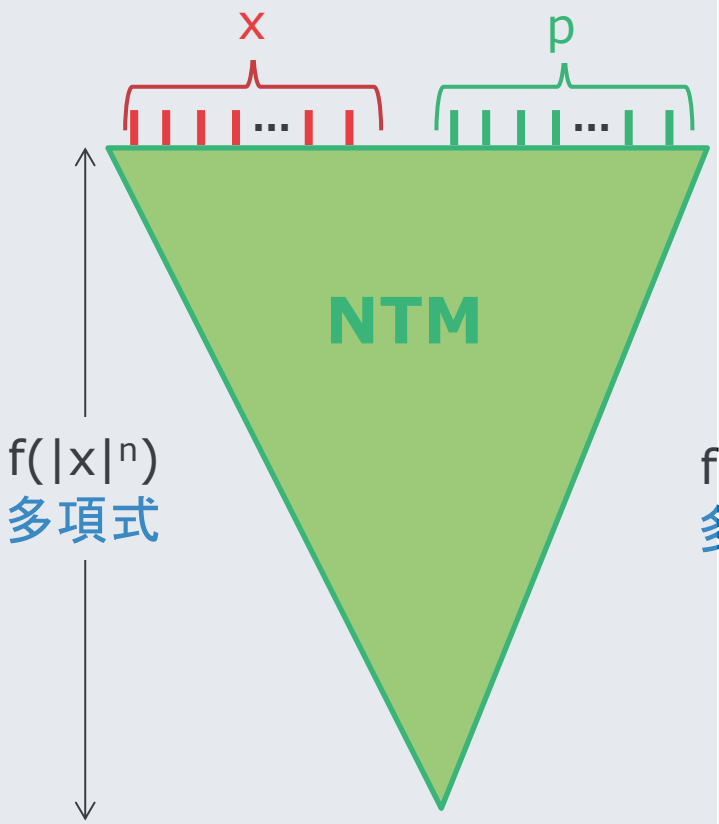
決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

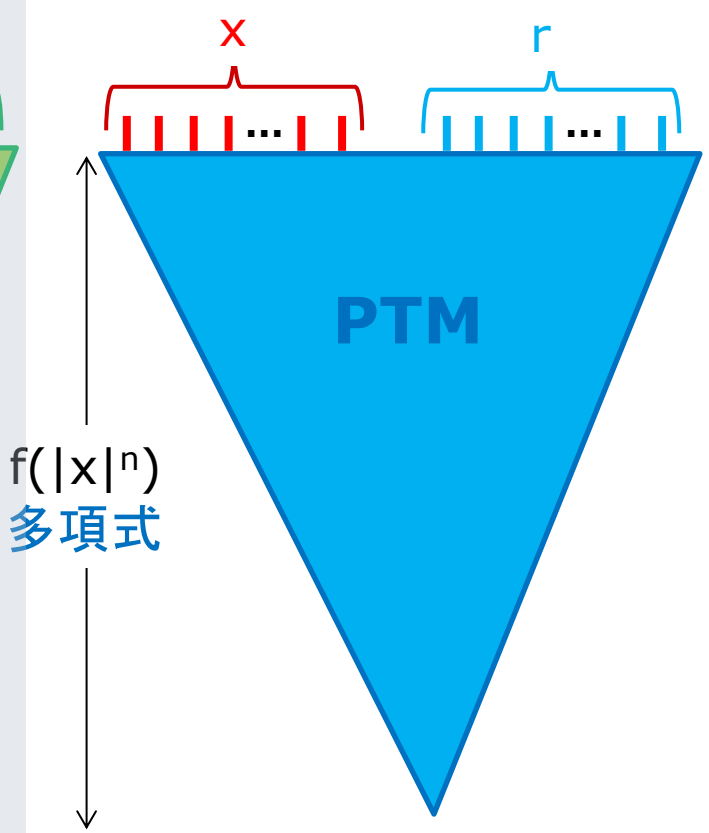
非決定性
チューリングマシン



$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

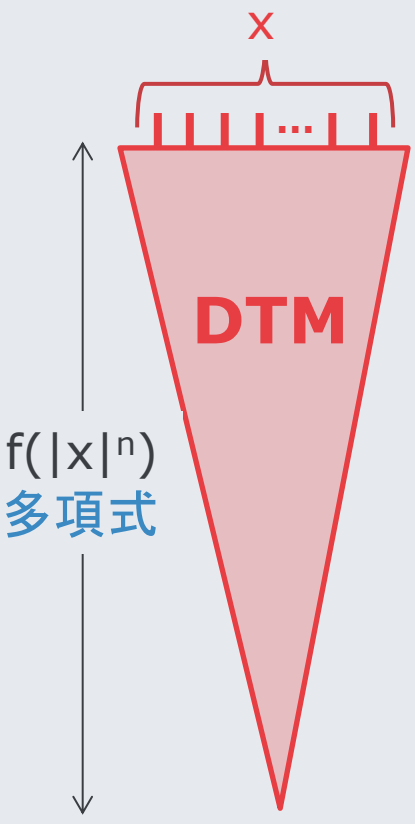
$L \in NP$

確率的
チューリングマシン



$x \in L$
$\rightarrow \mathbf{Prob}(PTM(x, r)=1) \geq 2/3$
$x \notin L$
$\rightarrow \mathbf{Prob}(PTM(x, r)=0) \leq 1/3$

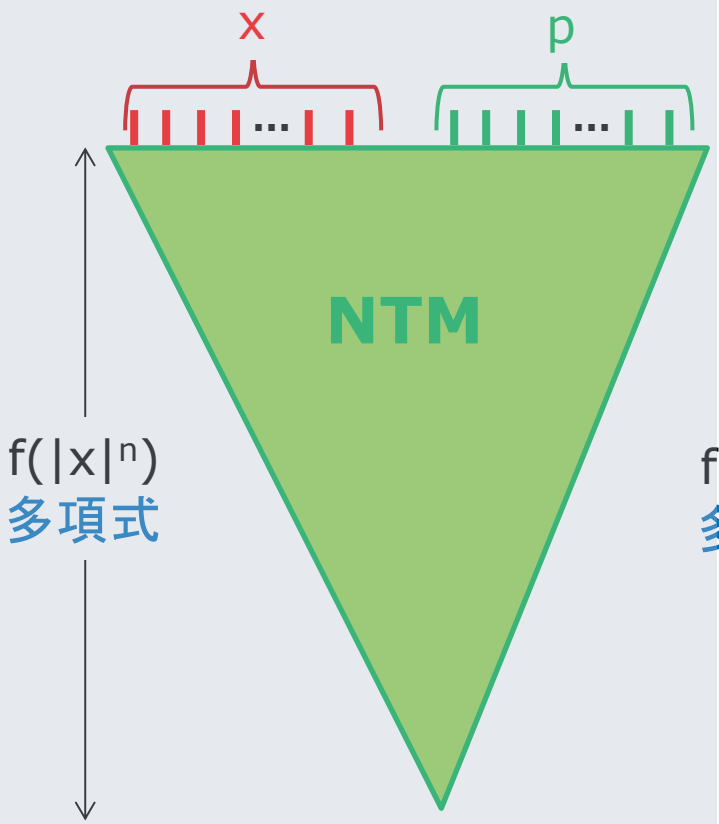
決定性
チューリングマシン



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

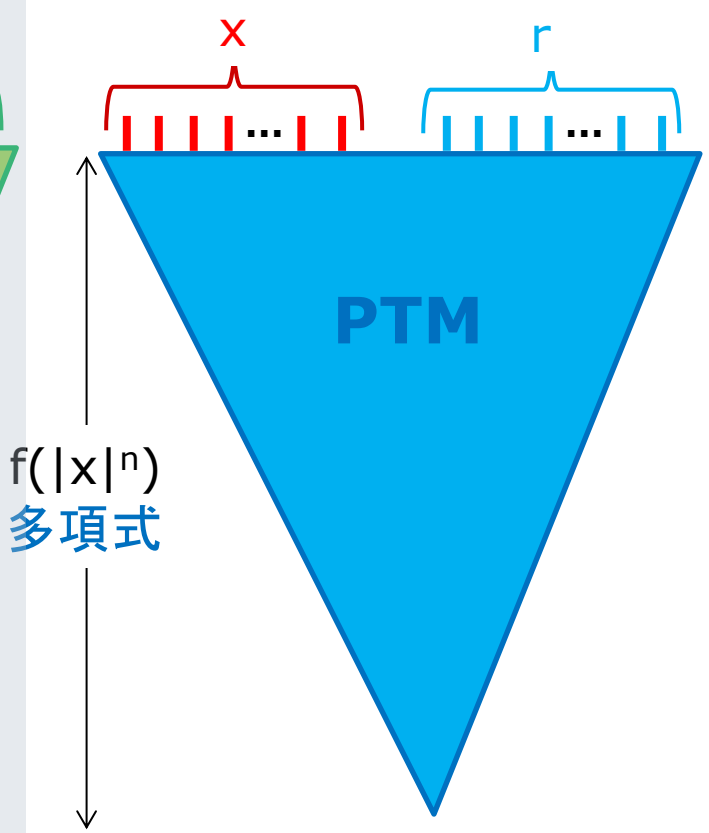
非決定性
チューリングマシン



$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

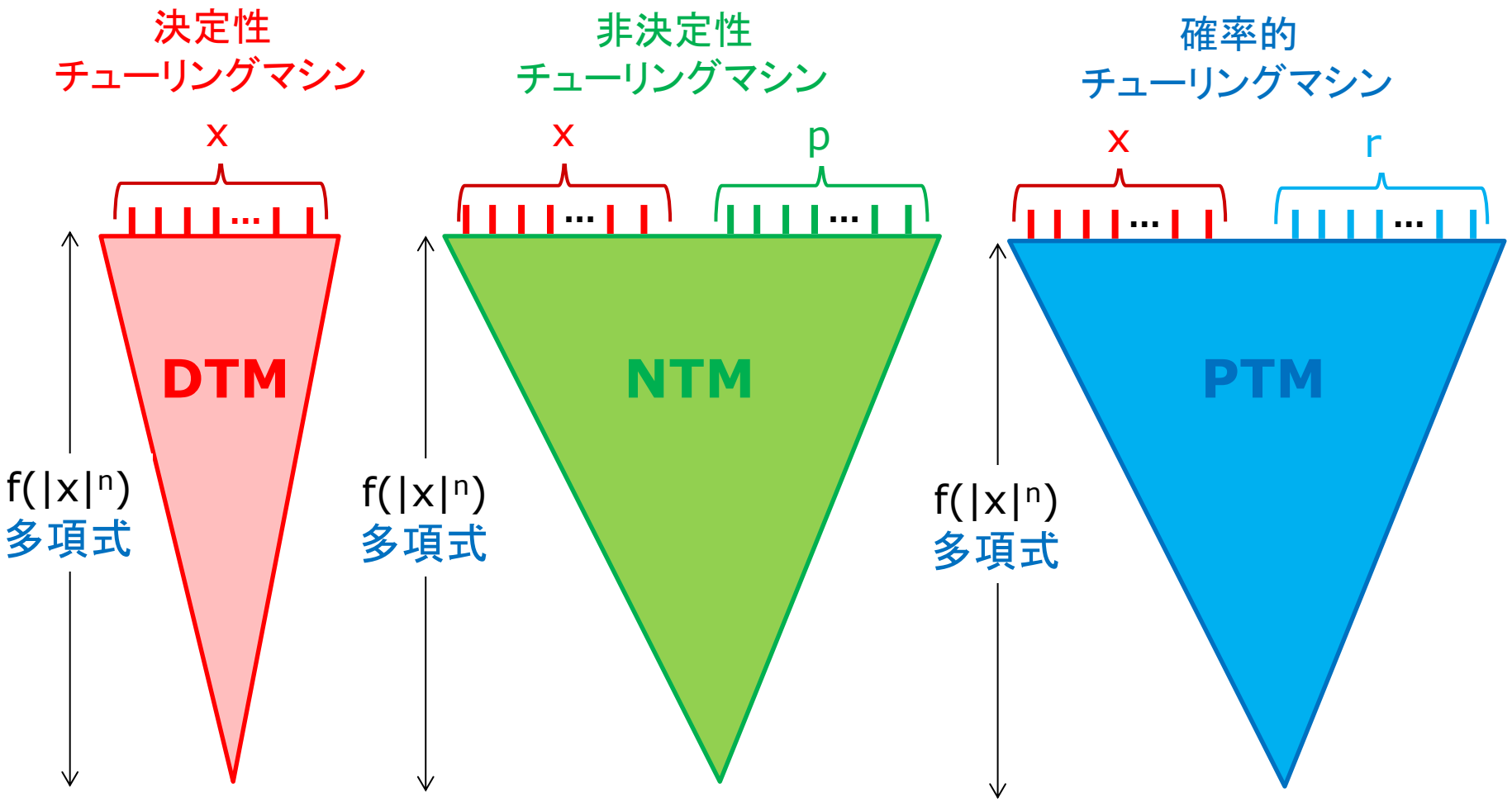
$L \in NP$

確率的
チューリングマシン



$x \in L$
$\rightarrow Prob(PTM(x, r)=1) \geq 2/3$
$x \notin L$
$\rightarrow Prob(PTM(x, r)=0) \leq 1/3$

$L \in BPP$



$x \in L$
$\rightarrow DTM(x)=1$
$x \notin L$
$\rightarrow DTM(x)=0$

$L \in P$

$x \in L$
$\rightarrow \exists p\{NTM(x, p)=1\}$
$x \notin L$
$\rightarrow \forall p\{NTM(x, p)=0\}$

$L \in NP$

$x \in L$
$\rightarrow Prob(PTM(x, r)=1) \geq 2/3$
$x \notin L$
$\rightarrow Prob(PTM(x, r)=0) \leq 1/3$

$L \in BPP$

「確率的チューリングマシン」では「非決定性チューリングマシン」とくらべても、何か強力で新しい計算の定義が可能になったように見えるかもしれません。

ただし、多くの数学者は、P ≠ NP だが P = BPP だろうと考えています。


量子チューリングマシンとBQPクラス

量子複雑性理論

1993年に、Bernstein と Vazirani は、これまでのTuringマシンの拡大である「量子Turingマシン」を新しく定義して、その上で複雑性理論を展開した。

ここから始まったこの複雑性理論の新しい分野を「量子複雑性理論」と呼ぶ。量子複雑性理論は、現在の複雑性理論の中心分野である。

BQP

量子複雑性理論で最も基本的なクラスは、BQPである。それは、従来の複雑性理論での多項式時間で決定可能な複雑性のクラス P に相当するものである。

ただし、量子Turingマシンの特性として、その出力は古典的なTuringマシンのように常に確定した値を返すのではなく、確率分布として与えられる。

BQPは、“Bounded error, Quantum, Polynomial time” の略である。

この Bounded error は、このマシンの出力の「誤り」が一定の確率（一般には 1/3 を使う）以下であることを表している。

その点では、BQPは古典的な複雑性理論でのBPP (bounded-error probabilistic polynomial time)によく似ている。

BPP

BPPは、入力 $x \in \{0,1\}^n$に対して、Mが最大q(n)ステップで終了するような、チューリングマシンMと多項式qが存在する言語$L \subset \{0,1\}^*$のクラスである。

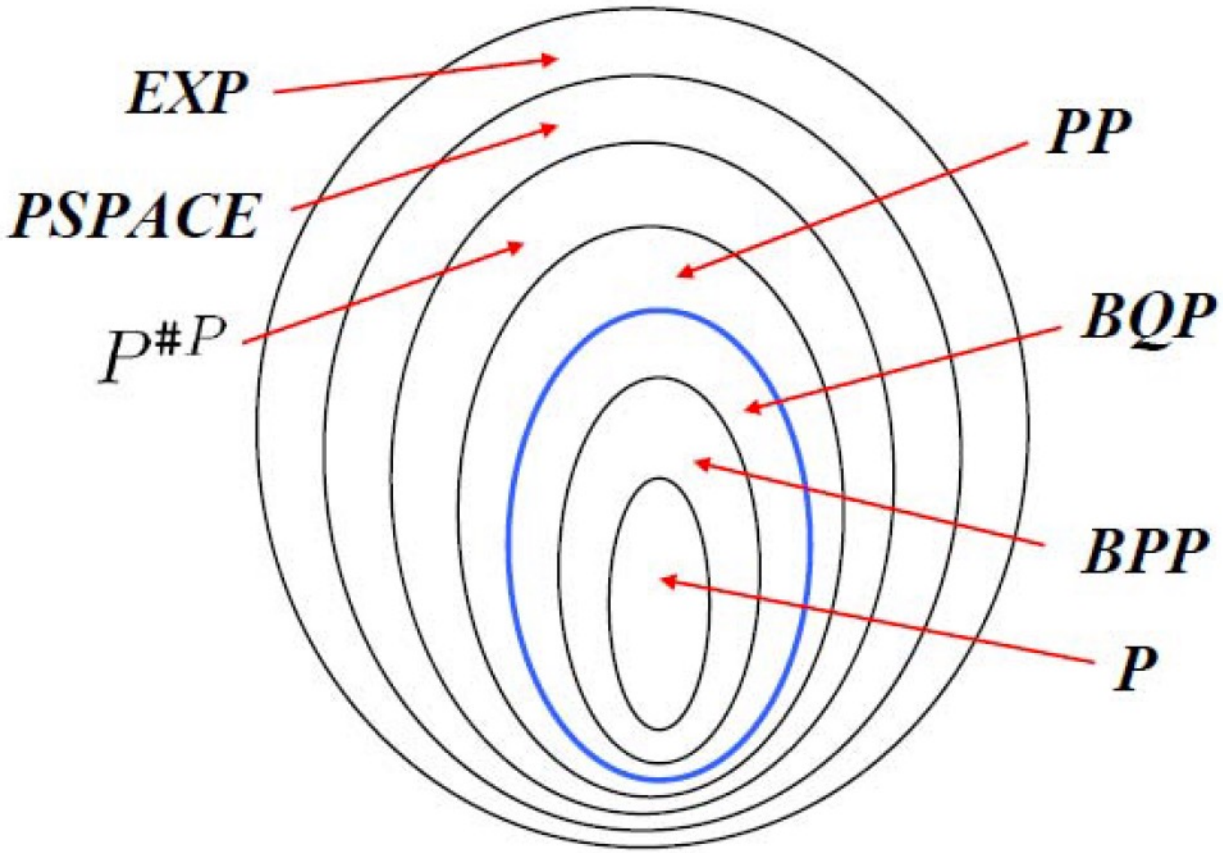
- もし x∈L なら、Mは
  確率 > 2/3 でacceptする
- もし x∉L なら、Mは
  確率 < 1/3 acceptする

BQP

BQPは言語$L \subset \{0,\ 1\}^*$のクラスであり、このクラスにはユニバーサルゲート上の基底と多項式qを持つ多項式サイズの量子回路$\{C_n\}$の一様な族が存在し、全てのnと入力$x \in \{0,\ 1\}^n$に対して次のようになる。

- もし x ∈ L なら、
  $C_n(|x> |0>^{\otimes q(n)})$は
  確率 > 2/3 でacceptする
- もし x ∉ L なら、
  $C_n(|x> |0>^{\otimes q(n)})$は
  確率< 1/3 acceptする

BQPの基本的性質



$$P \subseteq BPP \subseteq BQP \subseteq PP \subseteq P^{\#P} \subseteq PSPACE \subseteq EXP$$

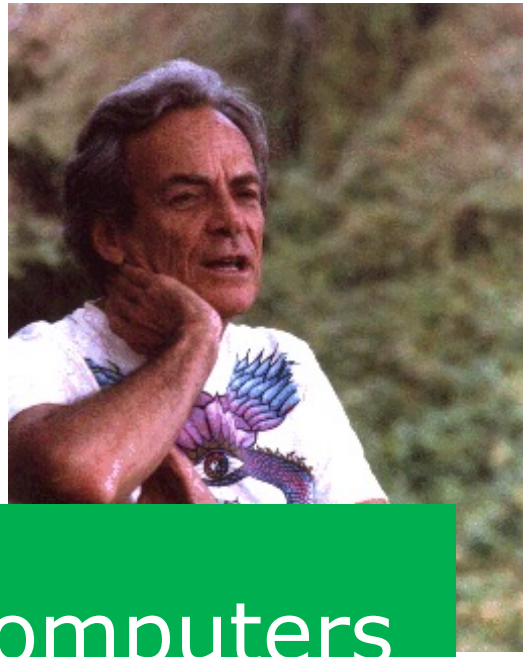
量子コンピュータの理論的可能性をめぐって先行した議論

「量子コンピュータ」は、今から約40年前のFeynmanの洞察を発端とする。DeutschがChurch=Turing Thesis を書き換え、Vaziraniが量子チューリングマシンを構成し、量子複雑性の基本的クラスBQPを提案する

ただ、「量子コンピュータ」が広く注目され研究者が爆発的に増大したのは、Shorが、量子コンピュータによる素因数分解のアルゴリズムを発見してからであった。

1982年

Feynmanの洞察



Simulating Physics with Computers

1982年　Richard P. Feynman

https://catonmat.net/ftp/simulating-physics-with-computers-richard-feynman.pdf

自然をシミュレートするコンピュータ

コンピューターが、正確に自然と同じように振る舞う、正確なシミュレーションが存在する可能性について話そうと思う。

それが証明されて、そのコンピュータのタイプが先に説明したようなものであるなら、必然的に、有限の大きさの時空の中で起きる全てのものは、有限な数の論理的な操作で正確に分析可能でなければならないことになるだろう。

量子論的システムは、古典的なコンピューターでシミュレートされるか？

量子論的なシステムは、古典的な万能計算機で、確率論的にシミュレートされるだろうか？　別の言い方をすれば、コンピューターは、量子論的なシステムが行うのと、同じ確率を与えるだろうか？

コンピューターを今まで述べてきたような古典的なものだとすれば（前節で述べたような量子論的なものではないとすれば）、また法則はすべて変更されないままで、ごまかしもないとすれば、答えは明らかにノーである。

量子コンピュータ -- 万能量子シミュレーター

それは、新しいタイプのコンピューター、量子コンピューター？で可能になるだろう。

私が理解する限りでは、それは量子論的なシステムによって、量子コンピューターの要素によって、シミュレート出来るようになることは、いまや、明らかになった。

それはチューリング・マシンではない。別のタイプのマシンである。

1985年
Deutsch

Church=Turing
Thesis の拡大

Quantum theory, the Church-Turing principle and the universal quantum computer

1985年　David Deutsch

https://www.cs.princeton.edu/courses/archive/fall04/cos576/papers/deutsch85.pdf

Church-Turing-Deutche Thesis

Church-Turingの仮説の基礎には、暗黙の物理学的主張があることが考察される。

この物理学的主張は、次のような物理学的原理として、明確に表現することが出来る。

「有限な方法で実現可能な物理システムは、有限な手段によって操作される万能計算機械のモデルで完全にシミュレート可能である。」

古典物理学と万能チューリング・マシンは、前者は連続的で後者は離散的であるので、この原理には従っていない。

万能量子コンピューター

チューリング・マシンのクラスの量子論的一般化である計算機械のクラスが記述され、量子論と、この「万能量子コンピューター」が、先の原理と両立可能であることが示される。

この万能量子コンピューターをまねた計算機械は、原理的には、構築可能である。

そしてそれは、いかなるチューリング・マシンによっても再生不可能な、多くの目覚ましい特徴を持つであろう。

Quantum Parallelism

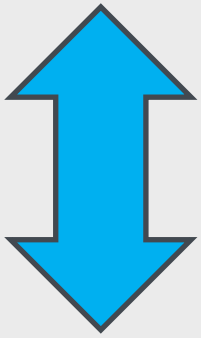
これらの中には、非帰納的な計算は含まれていない。しかし、「量子並行計算」が含まれる。

その方法によれば、古典的な制限の下でのコンピューターよりはるかに高速に、ある種の確率論的問題を万能量子コンピューターが実行出来る。

こうした特質の直観的な説明は、エヴェレット以外の量子論の解釈ではすべて、堪え難い緊張をもたらす。

計算理論は従来、純粋数学のトピックとして、ほとんど抽象的に研究されてきた。しかし、これは本質を見誤っている。コンピュータは物理的な物体であり、計算は物理的なプロセスである。コンピュータが何を計算できるか、あるいは何ができないかは、物理法則だけによって決まるのであって、純粋数学によって決まるのではない。

-- David Deutsch



数学と同様に、コンピュータサイエンスは、そのなかでは.確実なことは決してわからない自然法則ではなく、証明可能な人工的な法則を扱うという点で、他の科学とは多少異なるだろう。

-- Donald Knuth

1993年

Vazirani

量子チューリングマシンと
BQPクラスの導入

Quantum Complexity Theory

1993年　Ethan Bernstein, Umesh Vazirani

https://dl.acm.org/doi/pdf/10.1145/167088.167097

論文の「はじめに」から

計算可能性の理論がチャーチ・チューリング理論を基礎としていたように、計算複雑性理論はこの理論を現代的に強化したものであり、計算のどのような「合理的な」モデルも確率的チューリングマシン上で効率的にシミュレーションできると主張している（効率的なシミュレーションとは、その実行時間がシミュレーションされる機械の実行時間の多項式に束縛されるものである）。

例えば、単位時間内に任意の長さの単語を操作できるコンピュータや、無限精度の実数を正確に計算できるコンピュータは、不合理なモデルである。

有限精度の計算プリミティブしか実装できないと仮定すれば、チューリング・マシン・モデル（または多項式時間と等価なセル・オートマトン・モデル）が必然的な選択であると主張されてきた。

しかし、チューリング・マシンは、物理的に実現可能なすべてのコンピューティング・デバイスのモデルとしては不十分であるという根本的な理由がある。

チューリングマシンは、宇宙の古典物理学モデルに基づいているのに対し、現在の物理理論では、宇宙は量子物理学的であると主張されている。

シミュレーションの難しさ

量子物理学に基づいて、本質的に新しい種類の（離散的な）コンピューティング・デバイスを得ることはできるのだろうか？

そのようなデバイスが確率チューリング・マシンよりも強力になる可能性を最初に示唆したのは、10年ほど前のファインマンの論文であった。

その論文の中で、ファインマンは非常に不思議な問題を指摘している。それは、指数関数的な速度低下なしに、一般的な量子物理システムを確率的チューリング・マシン上でシミュレーションすることは不可能であるように見えるということである。

シミュレーションの難しさは、連続系を離散系でシミュレートする問題とは無関係である。シミュレートされる量子物理系は離散系であり、ある種の量子セル・オートマトンであると仮定することができるからである。

量子チューリングマシン

ファインマンの観察に鑑み、我々は計算複雑性理論の基礎と、チャーチ・チューリング論文の複雑性理論的な形式を再検討し、量子物理学に基づく計算装置の計算能力を研究しなければならない。

量子物理コンピュータの正確なモデル（以下、量子チューリング・マシンと呼ぶ）は、ドイチュによって定式化された。このモデルは、確率的チューリング・マシンの量子物理的類似物と考えることができ、無限テープと有限状態制御を持ち、マシンの動作はこの有限状態制御によって局所的かつ完全に規定される。量子TMは、与えられた入力に対して、確率分布からランダムなサンプルを生成する。

一般的な量子チューリングマシンは物理的に実現可能か？

有限状態制御の仕様ごとに新しい物理的実装を考案する必要はない。従って、この問題を解決するための第一歩は、普遍的な量子チューリング機械が存在するかどうかに答えることである。

Deutschは、量子チューリングマシンの普遍的なシミュレータについて述べている。このシミュレータは、シミュレーションのオーバーヘッドが、最悪の場合、シミュレートされたチューリングマシンの実行時間に対して指数関数的であるため、複雑性理論の観点からは満足できるものではない。

本論文では、シミュレーションのオーバヘッドが多項式に束縛される普遍的な量子チューリング機械の存在を証明する。

量子チューリング機械は、与えられた入力に対して、確率分布からランダムなサンプルを生成する。量子TM T'がTをϵの精度でシミュレートするのは、すべての入力zに対して、T'がTに対応する分布の全変動距離ϵ以内にある分布からサンプルを出力するときである。

時間tと入力zを入力とし、時間tのT(x)を精度ϵでシミュレートするユニバーサル量子TMが存在することを証明する。速度の低下は、tの多項式で $1/\epsilon$ である。

本論文では、量子TMが古典的確率的TMよりも強力である可能性を示す最初の証拠を提示する。

我々は、量子TMが多項式時間で受理できるが、有界エラー古典確率的TMが$n^{O(\log n)}$時間で受理できない言語の相対的なオラクルが存在することを証明する。

より注意深く構成すると、量子多項式時間が 検証者が$n^{O(\log n)}$時間をもつArthur-Merlinクラスにも含まれないようなオラクルが示される。このクラスの定義については[BMW]を参照）。

多項式時間量子TMが最大1/3の誤り確率で受理する言語のクラスをBQP(bounded-error quantum polynomial time)とする。

BQP ⊆ PSPAC'Eを示すのは難しくない。

したがって、BPP ≠ BSPACE ？ という長年の未解決問題を解決することなく、BPP ⊂ BQPを証明することは望めない。

実際、最近Valiant氏から指摘されたことだが、上記の結果は次のように強化することができる。BQP ⊆ #P である。

量子計算の問題は、計算機科学だけでなく物理学にも関連していると感じている。量子TMの計算能力の研究は、量子物理学が提案するモデルと古典的モデルとの本質的な違いを定量的な方法で実証する方法を与える。

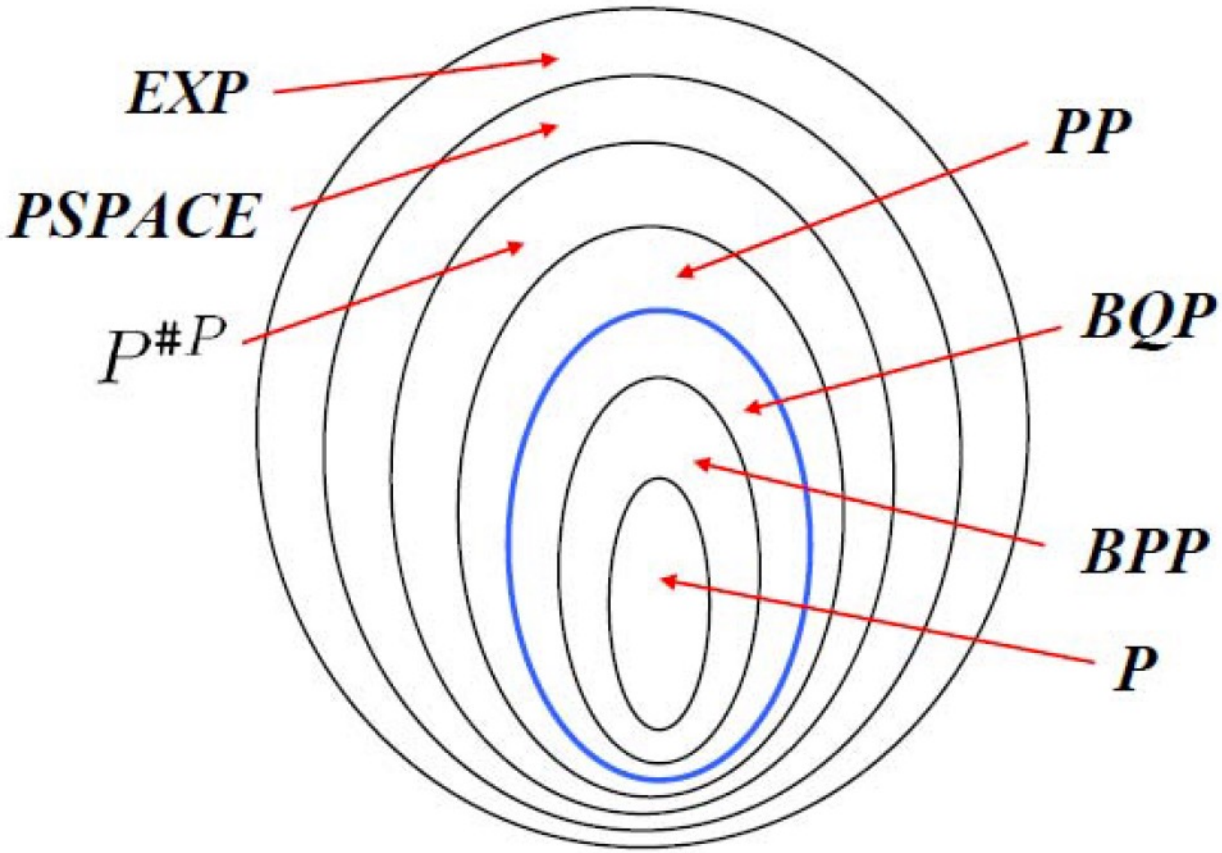
これは、有名なアインシュタイン-ポドルスリ-ローゼンのパラドックスやベルの不等式で議論されている）の精神で捉えることができ、量子モデルと局所的な隠れ変数モデルの統計的性質の違いを示すことができる。

また、BQPがBPPと等価であることを示せば（あるいは、その前段階としてBQPが準指数時間に含まれることを示せば）、量子物理系のコンピュータシミュレーションを効率的に行うことができる。

論文の構成

量子TMのモデルを紹介するために、新しい用語が必要である。セクション2では、古典的な確率的TMの文脈でこの用語を紹介する。そしてセクション3では、このモデルの自然な拡張として量子TMを定式化する。セクション4では、量子TMの2つの有用な特徴を示す。セクション5では、これらの特徴を用いて、ユニバーサルな量子TMを構築する。最後に、セクション6では、量子TMが古典的な確率的TMよりも強力であるという証拠を示す。本論文の最終版には、この暫定版では抑制された多くの議論が含まれる予定である。

Basic properties of BQP



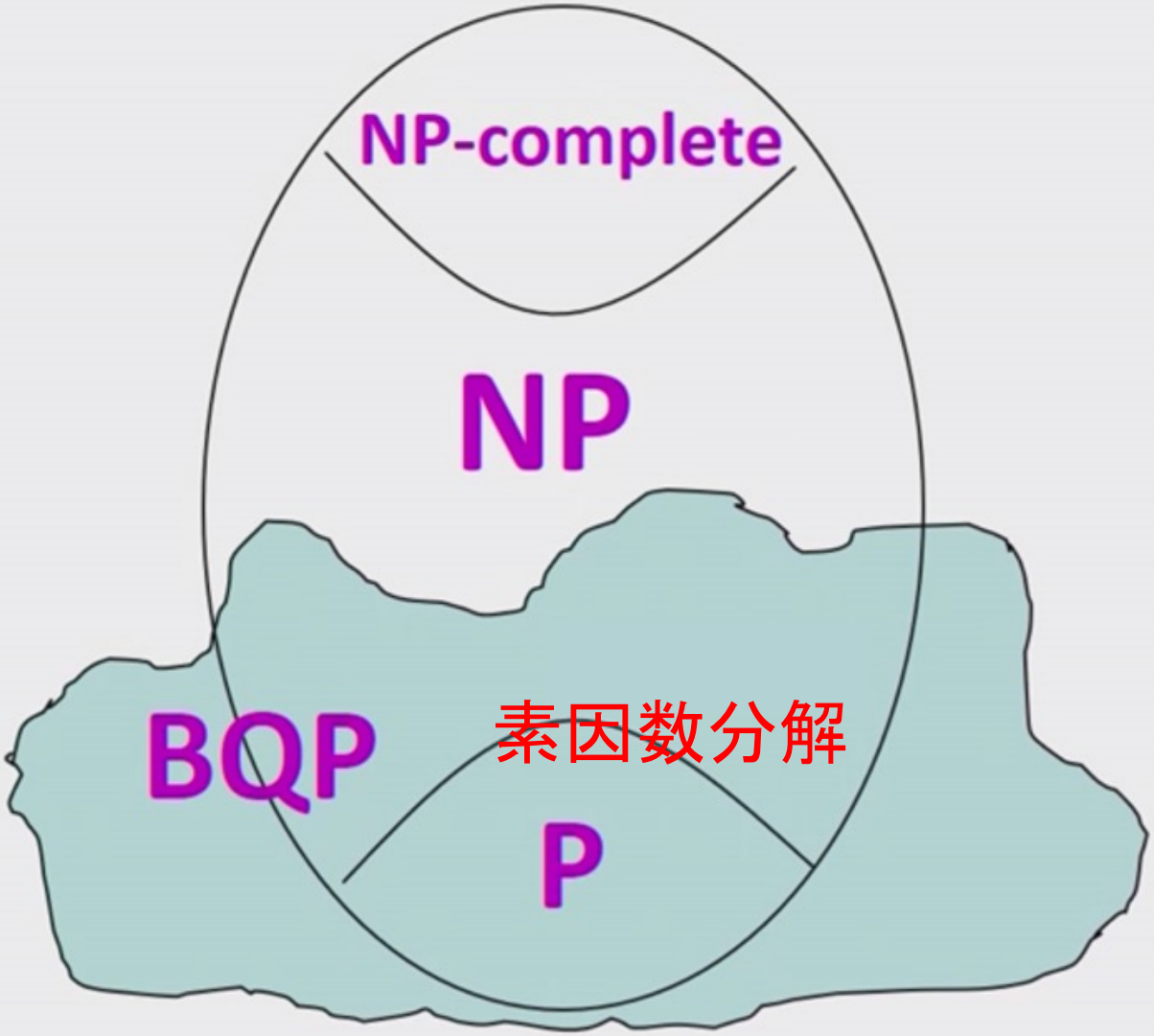
1994年
Shor

量子コンピュータによる
素因数分解アルゴリズの発見

Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer

1995年　Peter W. Shor

http://arxiv.org/pdf/quant-ph/9508027v2.pdf


NP-complete
NP
BQP
素因数分解
P

2019年
Martinisら

量子優越性の実証実験

Quantum supremacy using a programmable superconducting processor

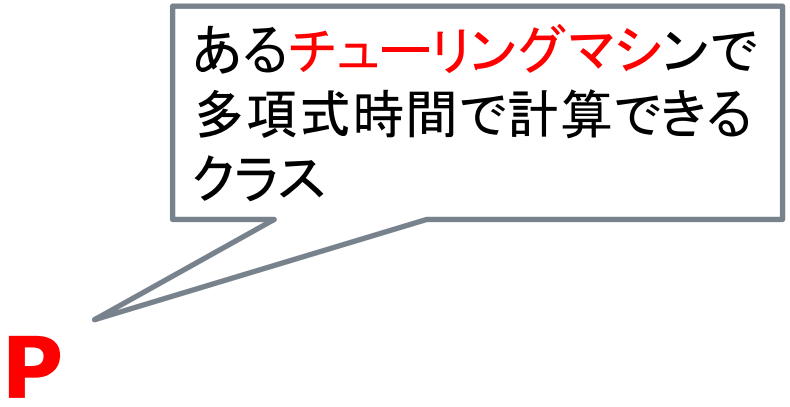
2019/10/23 Google

https://www.nature.com/articles/s41586-019-1666-5

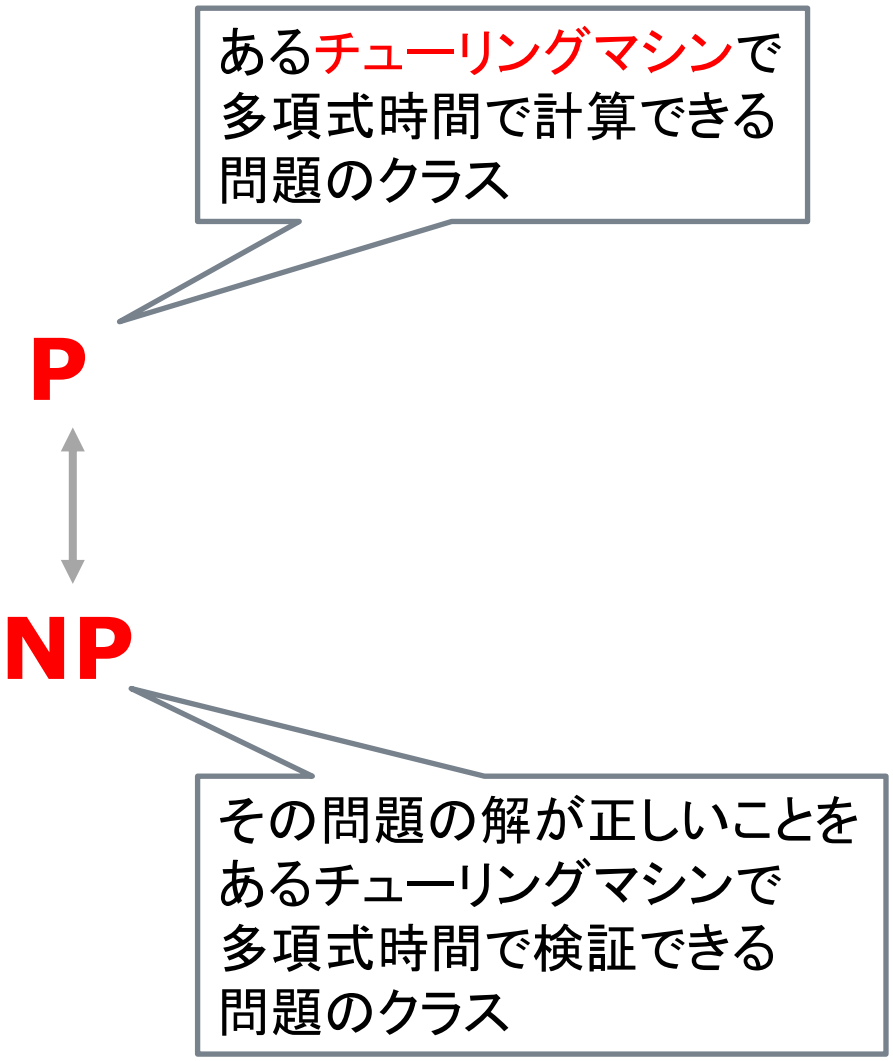
https://www.marulabo.net/docs/q-supremacy/


主要な計算複雑性クラスと
対応するチューリングマシン

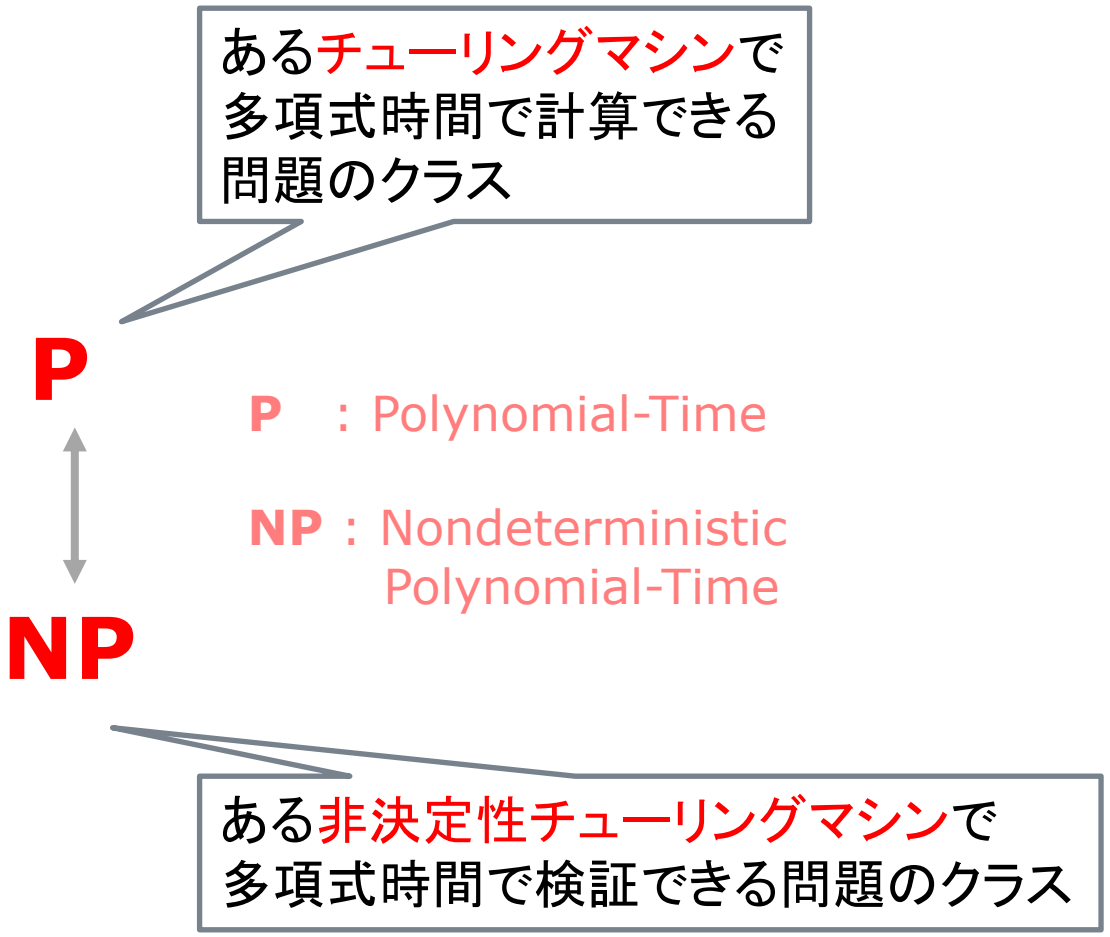
Pクラス



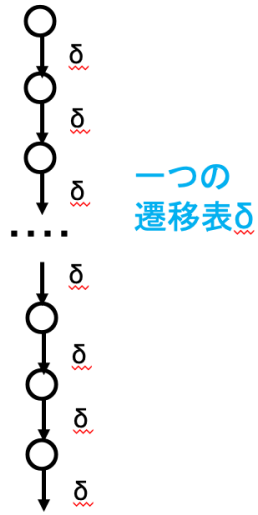
PクラスとNPクラス


あるチューリングマシンで
多項式時間で計算できる
問題のクラス
P
NP
その問題の解が正しいことを
あるチューリングマシンで
多項式時間で検証できる
問題のクラス

NPクラスの別の定義


あるチューリングマシンで
多項式時間で計算できる
問題のクラス
P
P ：Polynomial-Time
NP：Nondeterministic
Polynomial-Time
NP
ある非決定性チューリングマシンで
多項式時間で検証できる問題のクラス

NPクラスの別の定義



一つの
遷移表δ

決定性チューリングマシン

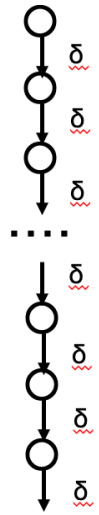
ある決定性チューリングマシンで
多項式時間で計算できる問題のクラス

P

NP

ある非決定性チューリングマシンで
多項式時間で検証できる問題のクラス

NPクラスの別の定義



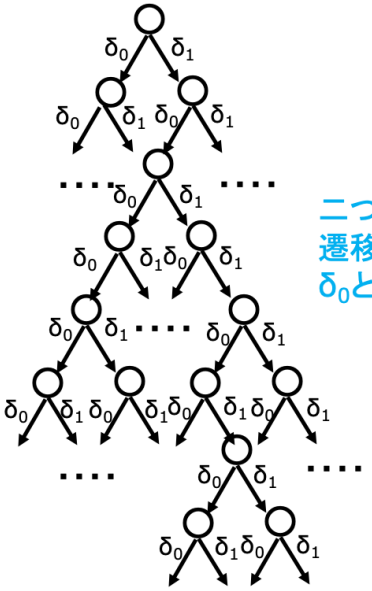
一つの
遷移表δ

決定性チューリングマシン

あるチューリングマシンで
多項式時間で計算できる
問題のクラス

P

NP



二つの
遷移表
δ_0とδ_1

非決定性チューリングマシン

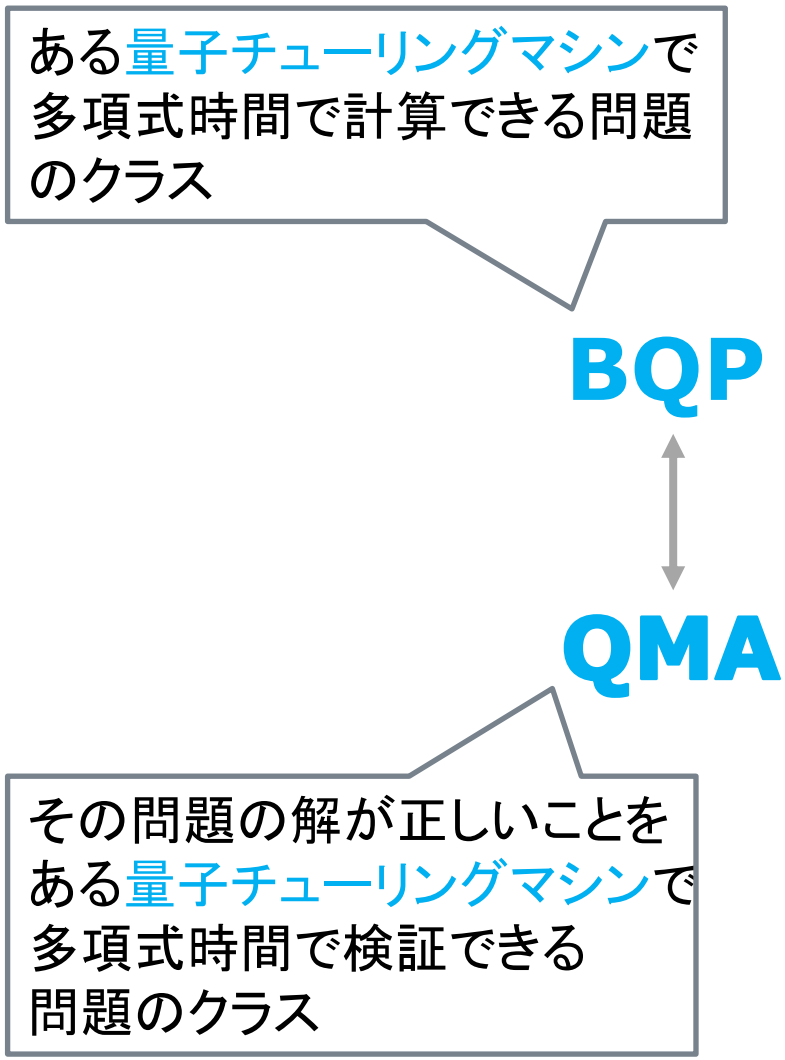
ある非決定性チューリングマシンで
多項式時間で検証できる問題のクラス

BQPクラス

ある量子チューリングマシンで多項式時間で計算できる問題のクラス

BQP

BQPクラスとQMAクラス



BQPクラスとQMAクラス

ある量子チューリングマシンで
多項式時間で計算できる問題
のクラス

BQP

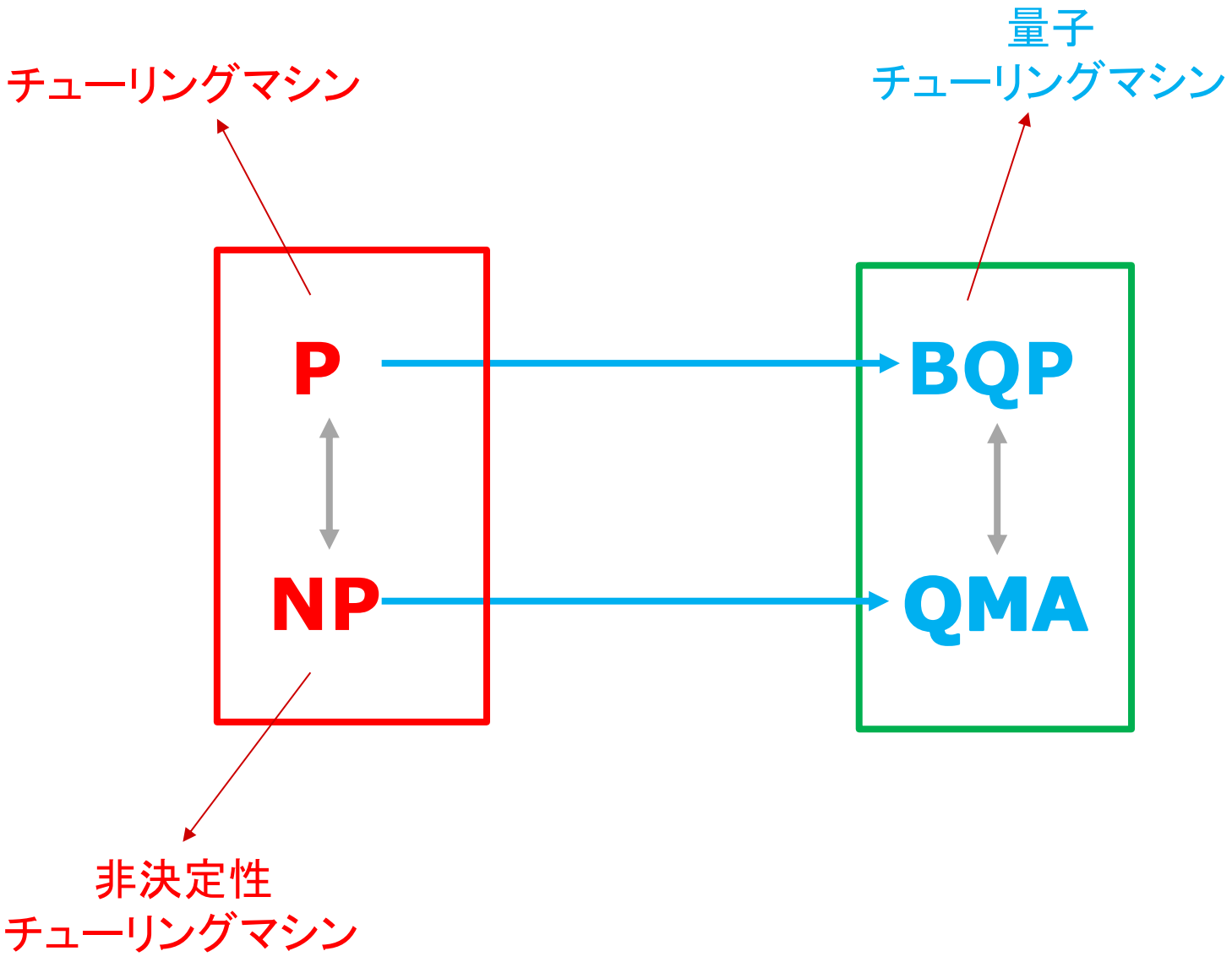
BQP：Bounded-Error Quantum
Polynomial-Time

QMA：Quantum MA

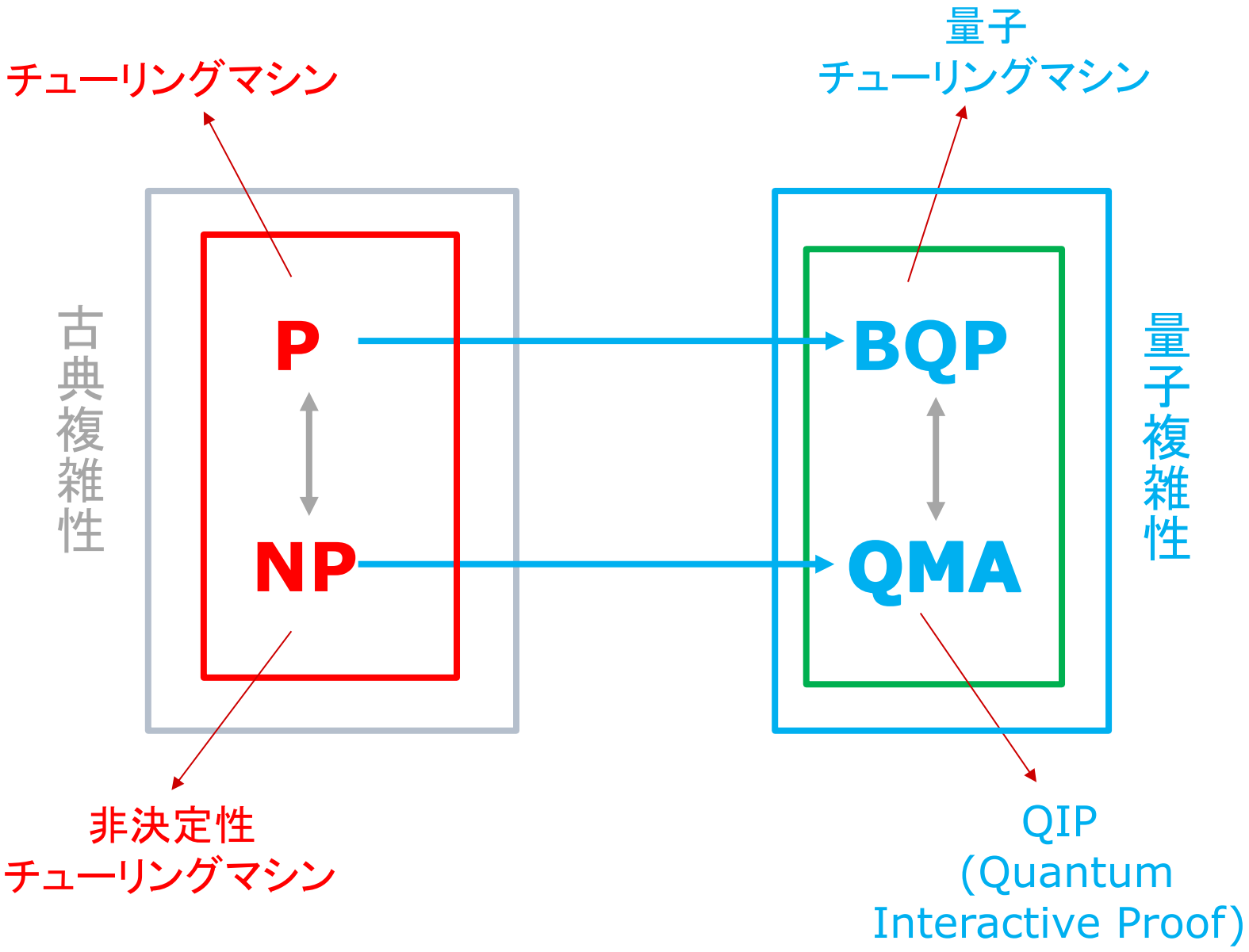
QMA

その問題の解が正しいことを
ある量子チューリングマシンで
多項式時間で検証できる
問題のクラス

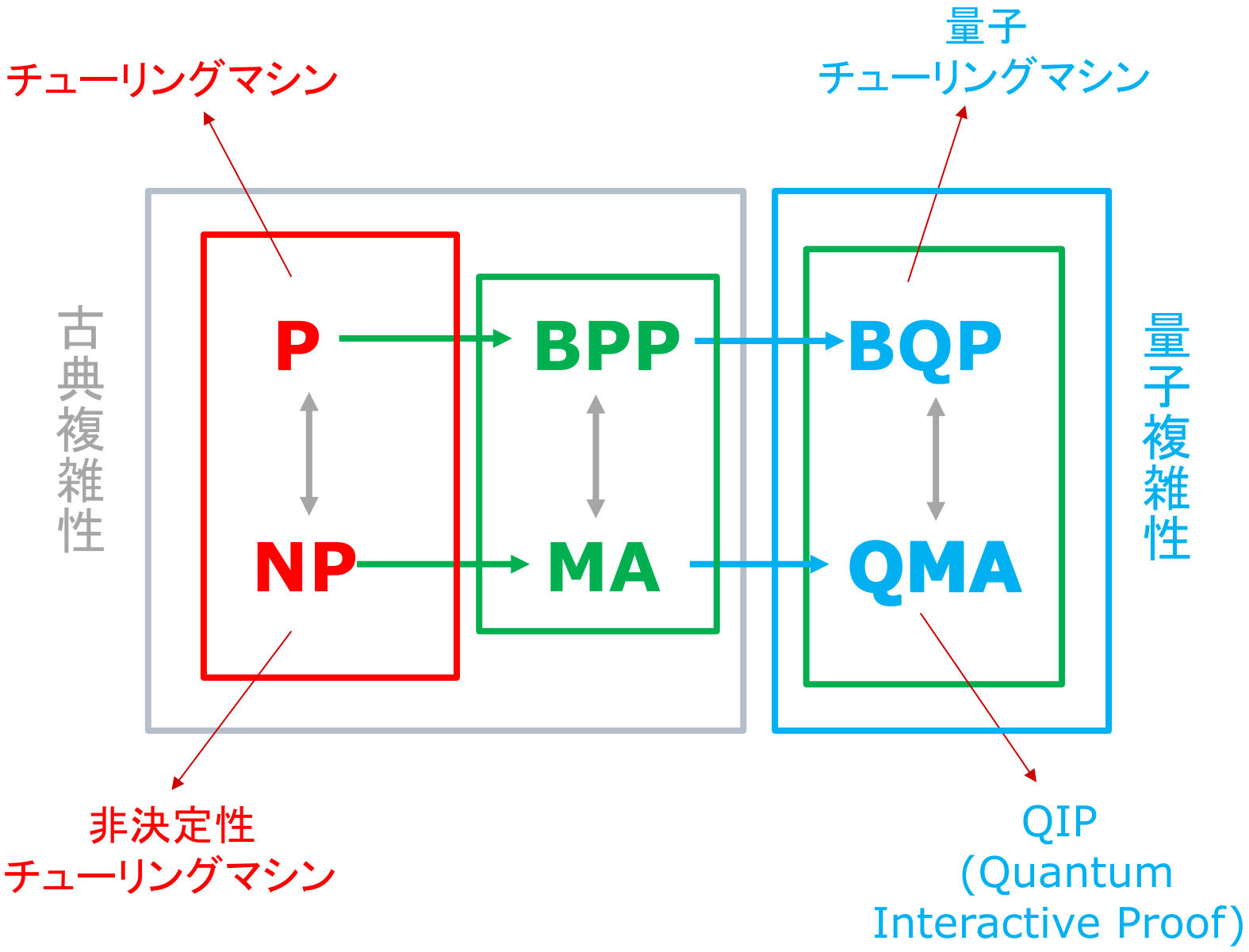
チューリングマシンと量子チューリングマシン



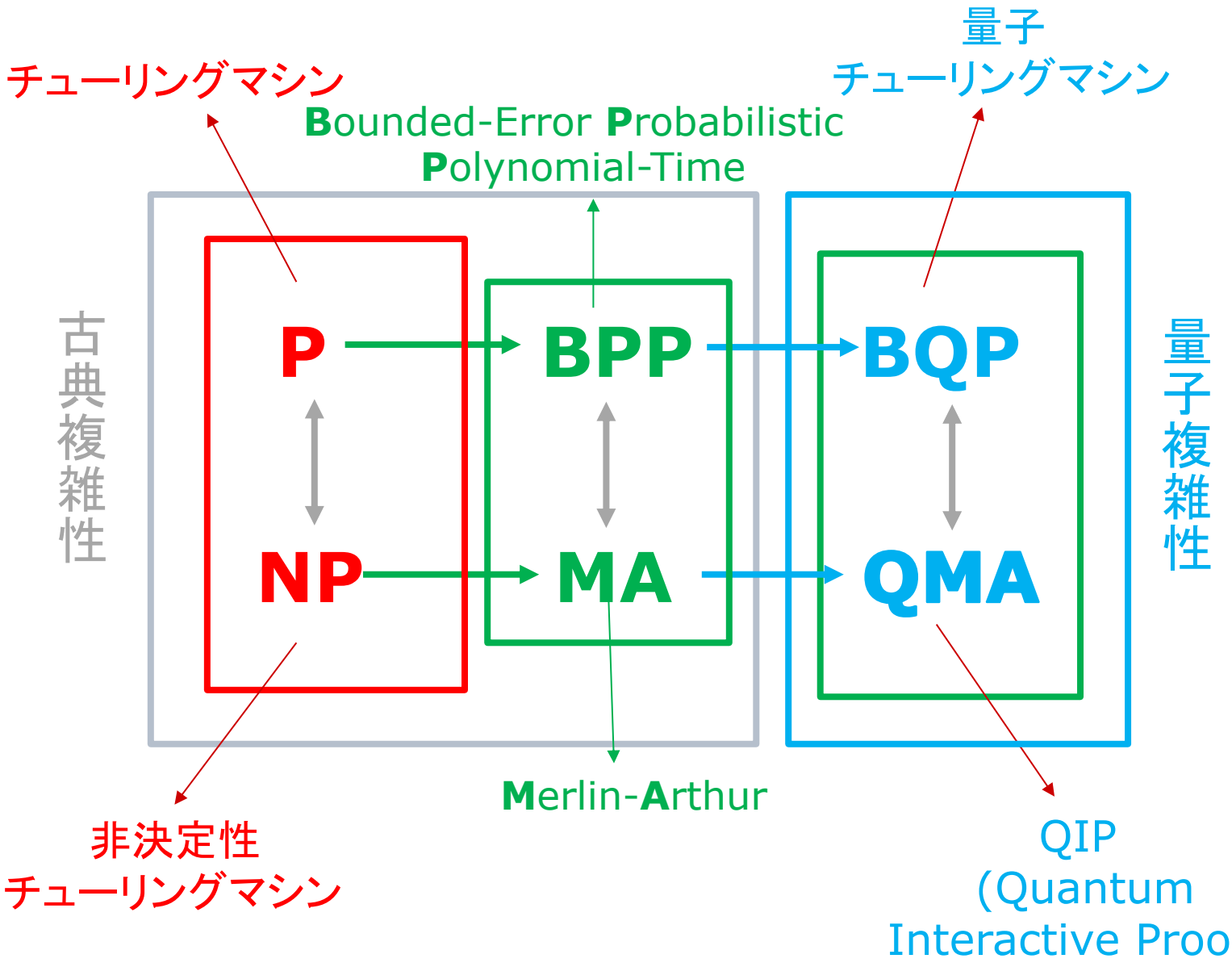
古典複雑性と量子複雑性



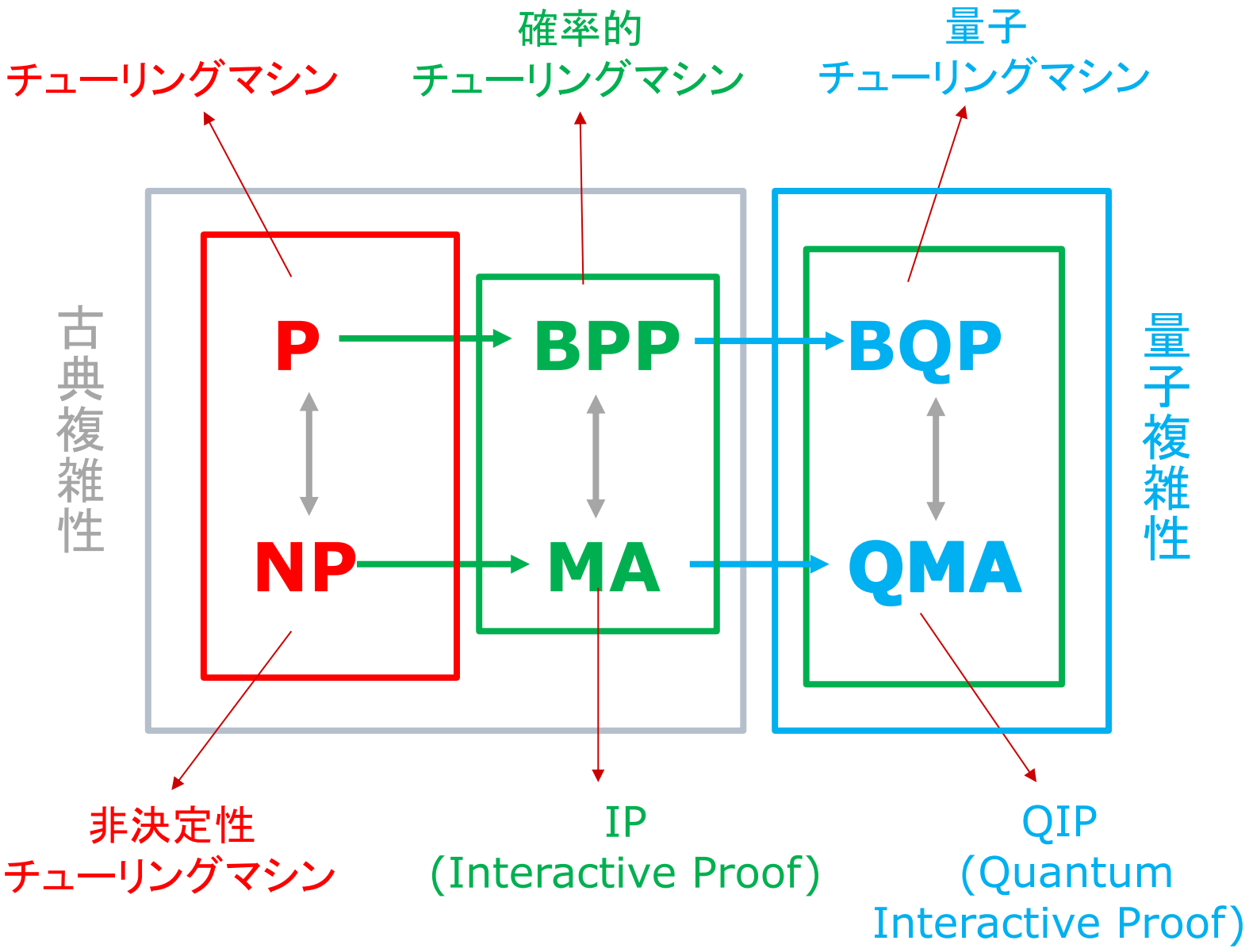
BPPクラスとMAクラス



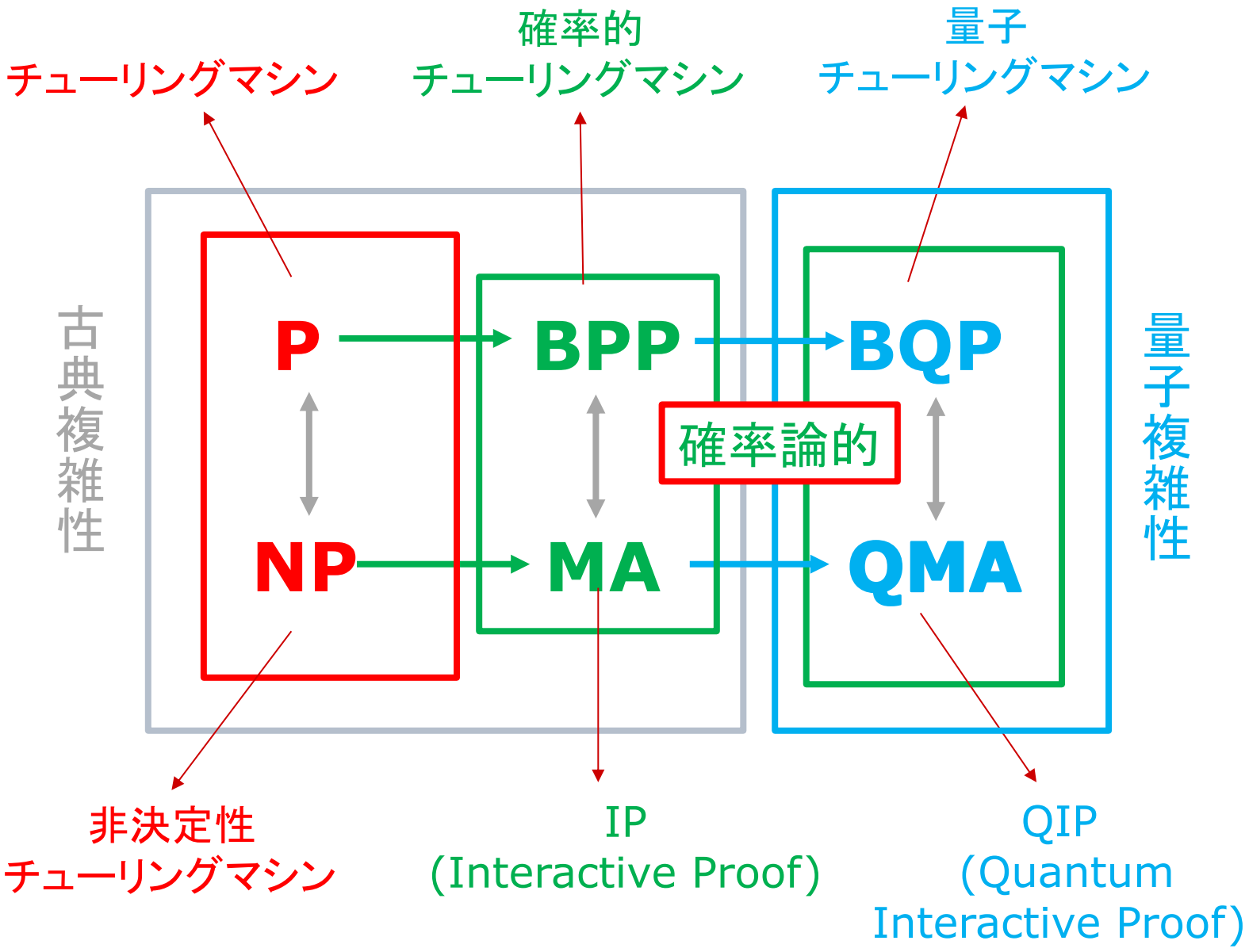
BPPクラスとMAクラス

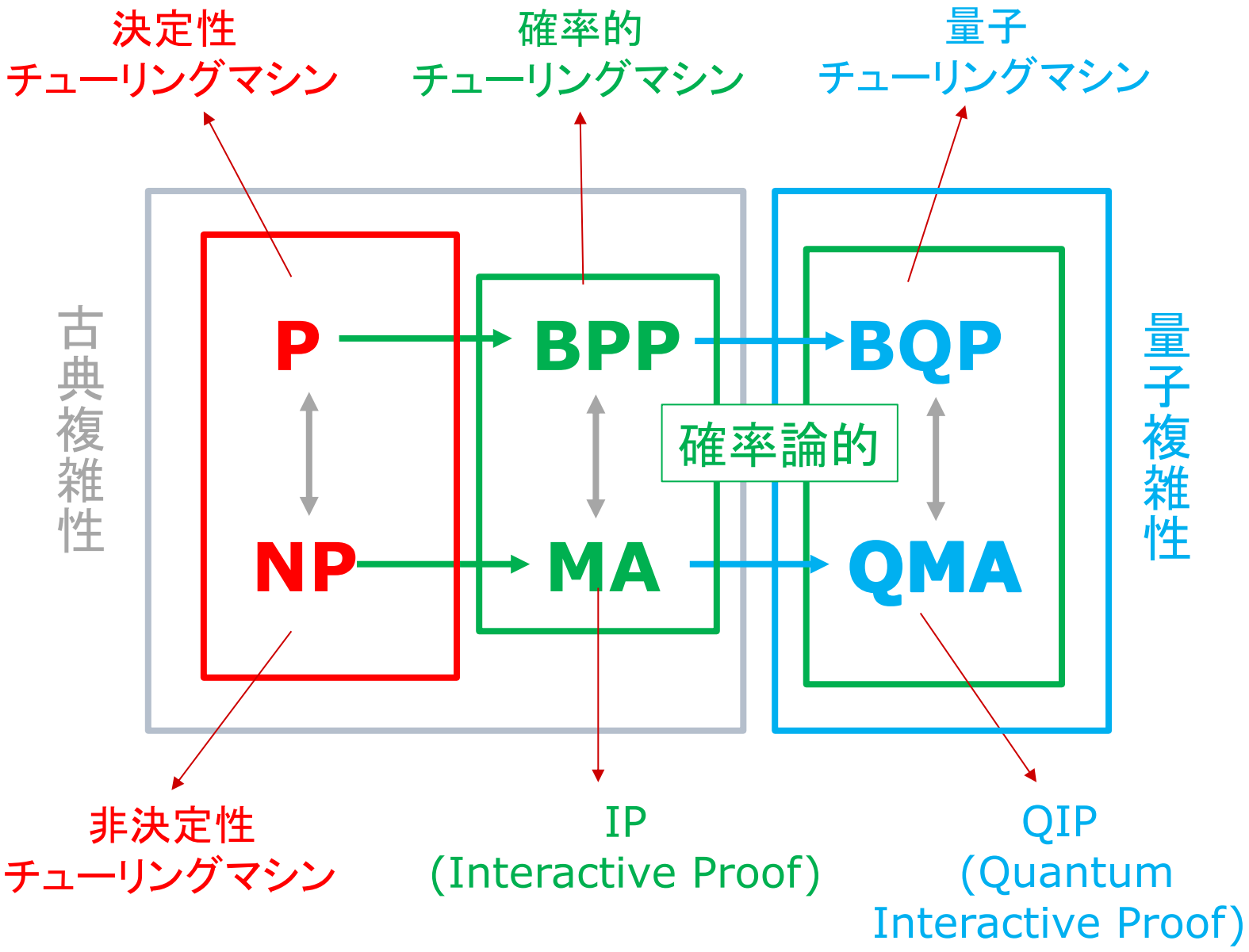


確率的チューリングマシンとInteractive Proof



確率と証明の結びつき


チューリングマシン
確率的
チューリングマシン
量子
チューリングマシン
古典複雑性
P
BPP
BQP
確率論的
NP
MA
QMA
量子複雑性
非決定性
チューリングマシン
IP
(Interactive Proof)
QIP
(Quantum
Interactive Proof)


決定性
チューリングマシン
確率的
チューリングマシン
量子
チューリングマシン
古典複雑性
量子複雑性
P
BPP
BQP
確率論的
NP
MA
QMA
非決定性
チューリングマシン
IP
(Interactive Proof)
QIP
(Quantum
Interactive Proof)


Church-Turing Thesisの変化と
量子情報理論

Church-Turing Thesis

Church-Turing Thesis：
すべての計算可能な計算は、あるTuringマシンによって実行される。

ここでの「計算可能性」は、数学的に構成されたTuringマシンと対応づけられた数学的に定義されたものである。

Church-Turing-Deutsch Thesis

Church-Turing-Deutsch Thesis:
物理システムによって実行されるすべての計算可能な計算は、あるTuringマシンによって実行される。

ここでの計算は、抽象化された数学的な計算ではなく、物理的システムによって実行される物理的なものである。

Church-Turing-Deutsch Thesis

Church-Turing-Deutsch Thesis:
物理システムによって実行されるすべての計算可能な計算は、あるTuringマシンによって実行される。

この命題を言い換えれば、Turingマシンによって実行できない計算は、物理的システムでは実行できないということになるのだが、この命題は物理的な法則の限界として計算可能性について語っている。

Turingマシンによって実行できない計算は、物理法則を破らない限り実行できない

拡張されたChurch-Turing Thesis

The extended CT thesis:
Turingマシンで、多項式時間で効率的に実行できない計算は、いかなる物理システムによっても実行できない。

あるいは同じことだが、

物理システムによって実行されるすべての計算は、あるTuringマシンによって多項式時間で実行される。

という命題「拡張されたChurch-Turing Thesis 」を考えよう。

拡張されたChurch-Turing Thesis と量子コンピュータ

しかし、Turingマシン（古典コンピュータ）で、多項式時間で効率的に実行できない計算（例えば素因数分解）が、量子コンピュータで多項式時間で解けるなら（Shorのアルゴリズム）、この命題は成り立たない。

「量子超越性」は、拡張されたChurch-Turing Thesisが成り立たないことを示す。

Church-Turing Thesisの量子化

次のような、Church-Turing Thesisの量子版を考える。

The Quantum-extended CT thesis (qECT):
量子Turingマシン（あるいは量子回路）によって実効的に実行できない全ての計算は、物理法則と矛盾しない、いかなる物理システムによっても実効的に実行されない。

すなわち、

物理システムで実効的に計算可能な計算は、量子Turingマシン（あるいは量子回路）によって実効的に実行される

Church-Turing Thesisの量子化

The Quantum-extended CT thesis (qECT):
量子Turingマシン（あるいは量子回路）によって実効的に実行できない全ての計算は、物理法則と矛盾しない、いかなる物理システムによっても実効的に実行されない。
すなわち、
物理システムで実効的に計算可能な計算は、量子Turingマシン（あるいは量子回路）によって実効的に実行される。

これは、正しそうだ。

ただ、Susskindによると、ブラックホールに落ち込んだ時、これも成り立たなくなり、修正が必要となるという。

Church-Turing Thesisの量子化の修正版

The Modified Quantum-extended CT thesis:
量子Turingマシン（あるいは量子回路）によって実効的に実行できない全ての計算は、空間のホログラフィック境界とコミュニケーション可能なままにある、いかなる物理システムによっても実効的に実行されない。

Church-Turing Thesis をめぐる最近の議論

Church-Turing Thesis を巡るこうした議論については、次のSusskindの論文を参照されたい。

ここでは、Church-Turing Thesis を巡る議論が、数学的な計算理論の枠を超えて、物理学の量子重力理論との関係で論じられている。

"Horizons Protect Church-Turing"

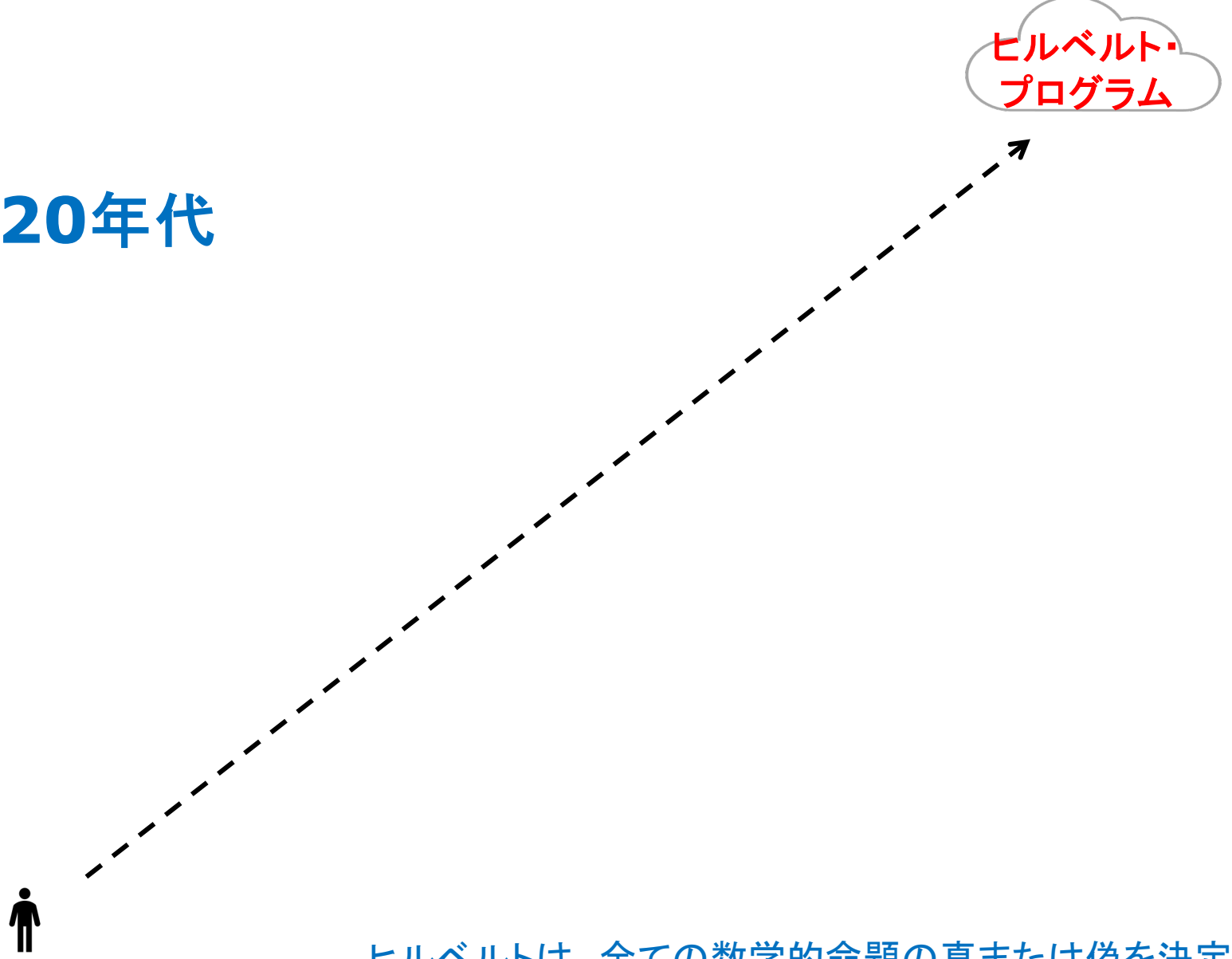
Susskind　2020年5月

https://arxiv.org/pdf/2003.01807.pdf

量子複雑性理論から量子情報理論へ

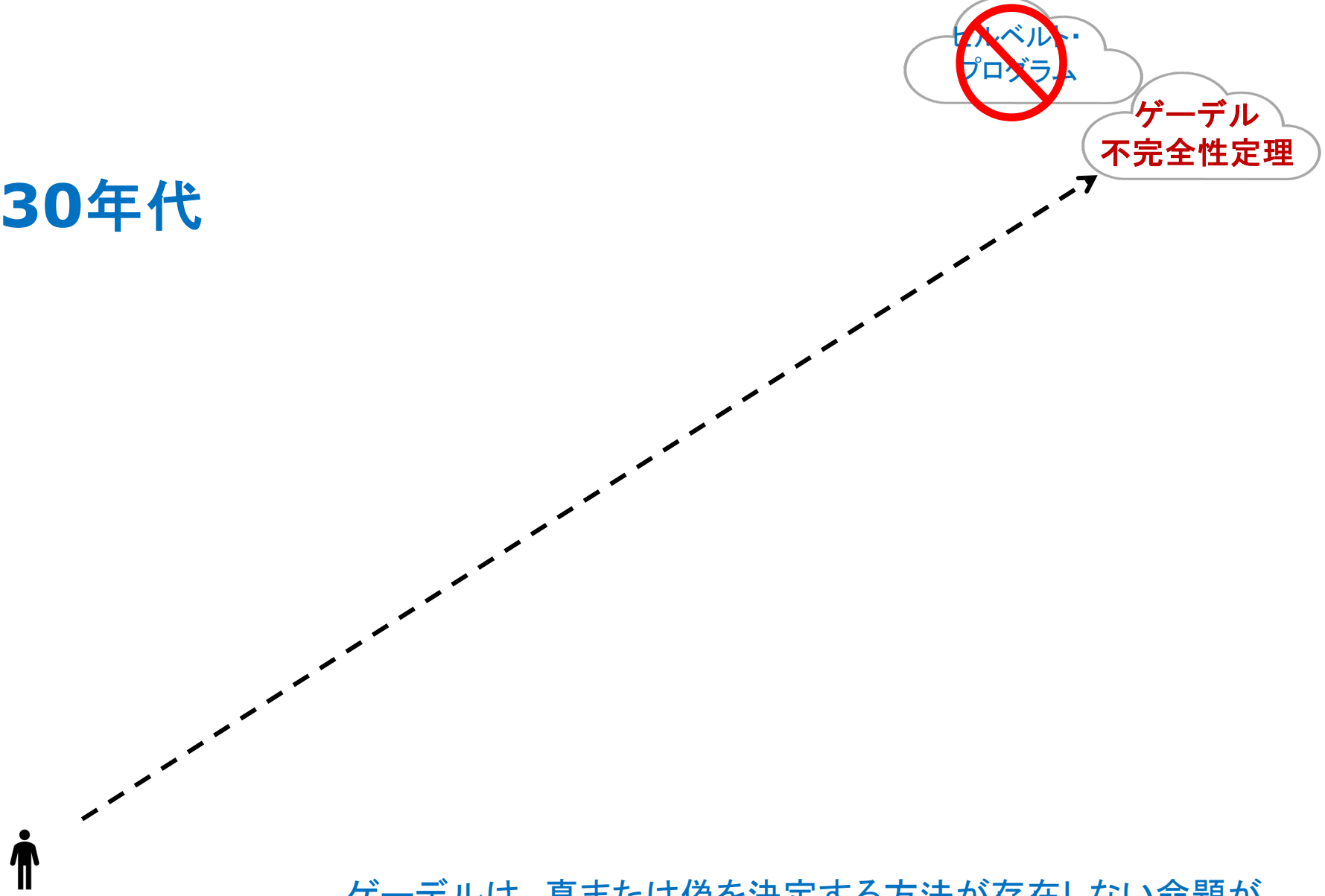
「知能」あるいは「数学的知識」の限界の
「数学的」あるいは「物理学的」な特徴付け

20年代



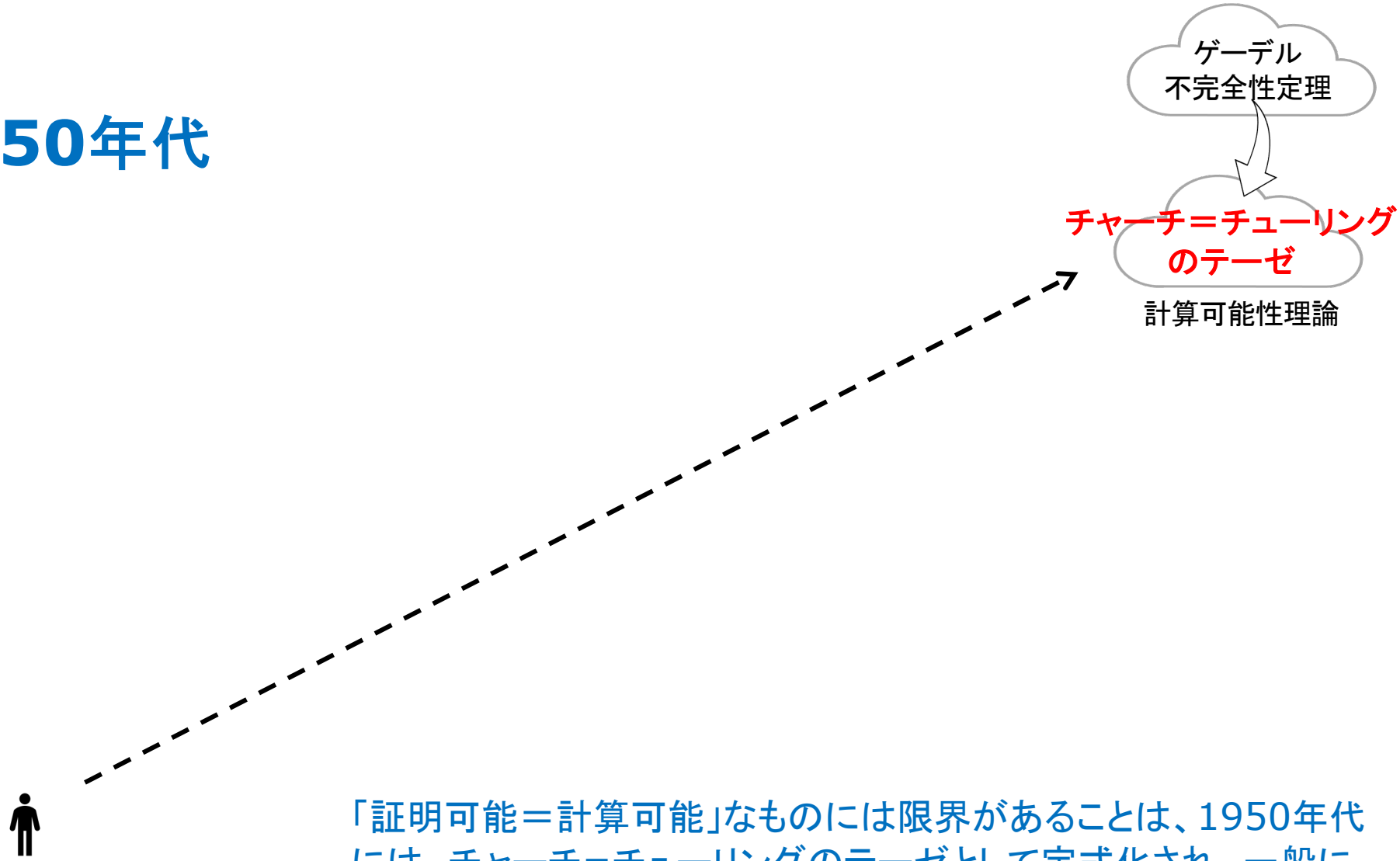
ヒルベルトは、全ての数学的命題の真または偽を決定する方法が存在すること、数学の形式化が矛盾を含まないことが証明しようとした。。

30年代



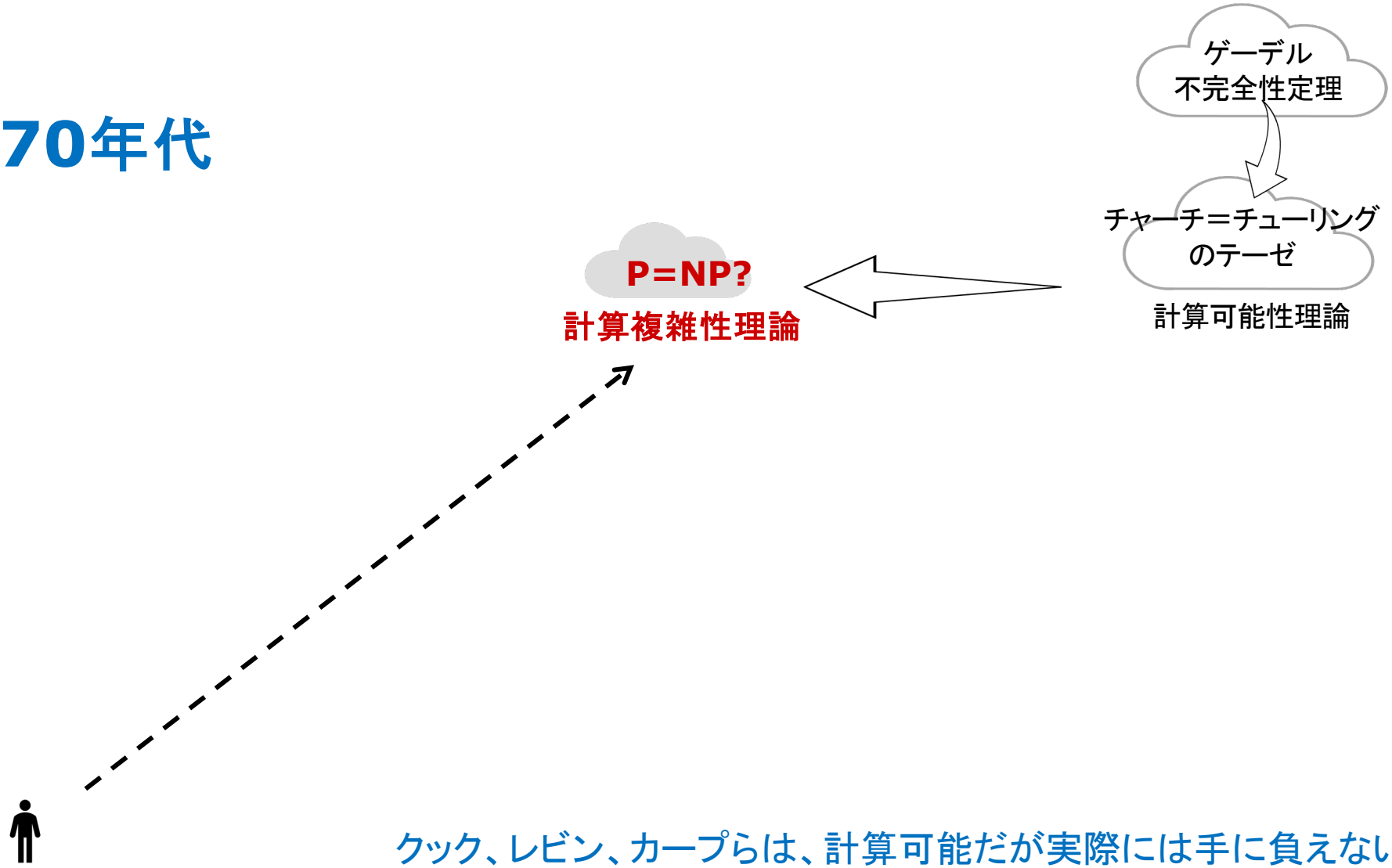
ゲーデルは、真または偽を決定する方法が存在しない命題が存在すること、また、数学の形式化が矛盾を含まないことの証明が不可能であることを示した。それはヒルベルトのプログラムが、遂行できないことを意味した。

50年代



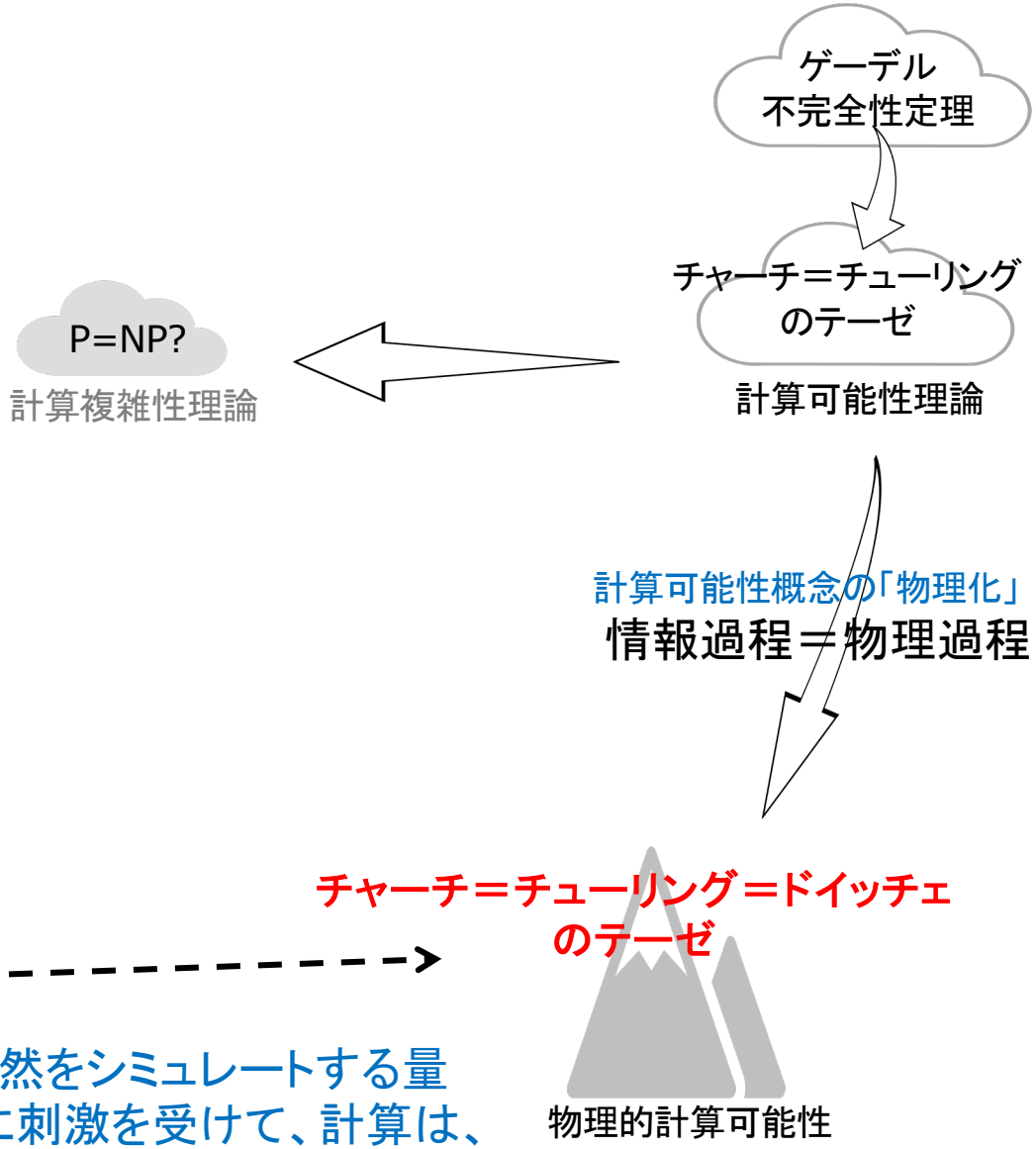
「証明可能＝計算可能」なものには限界があることは、1950年代には、チャーチ＝チューリングのテーゼとして定式化され、一般に広く認められるようになった。ただ、それは、我々の認識の「限界」としては、非常に荒い、かつ、抽象的で原理的な限界を与えるものでしかなかった。

70年代



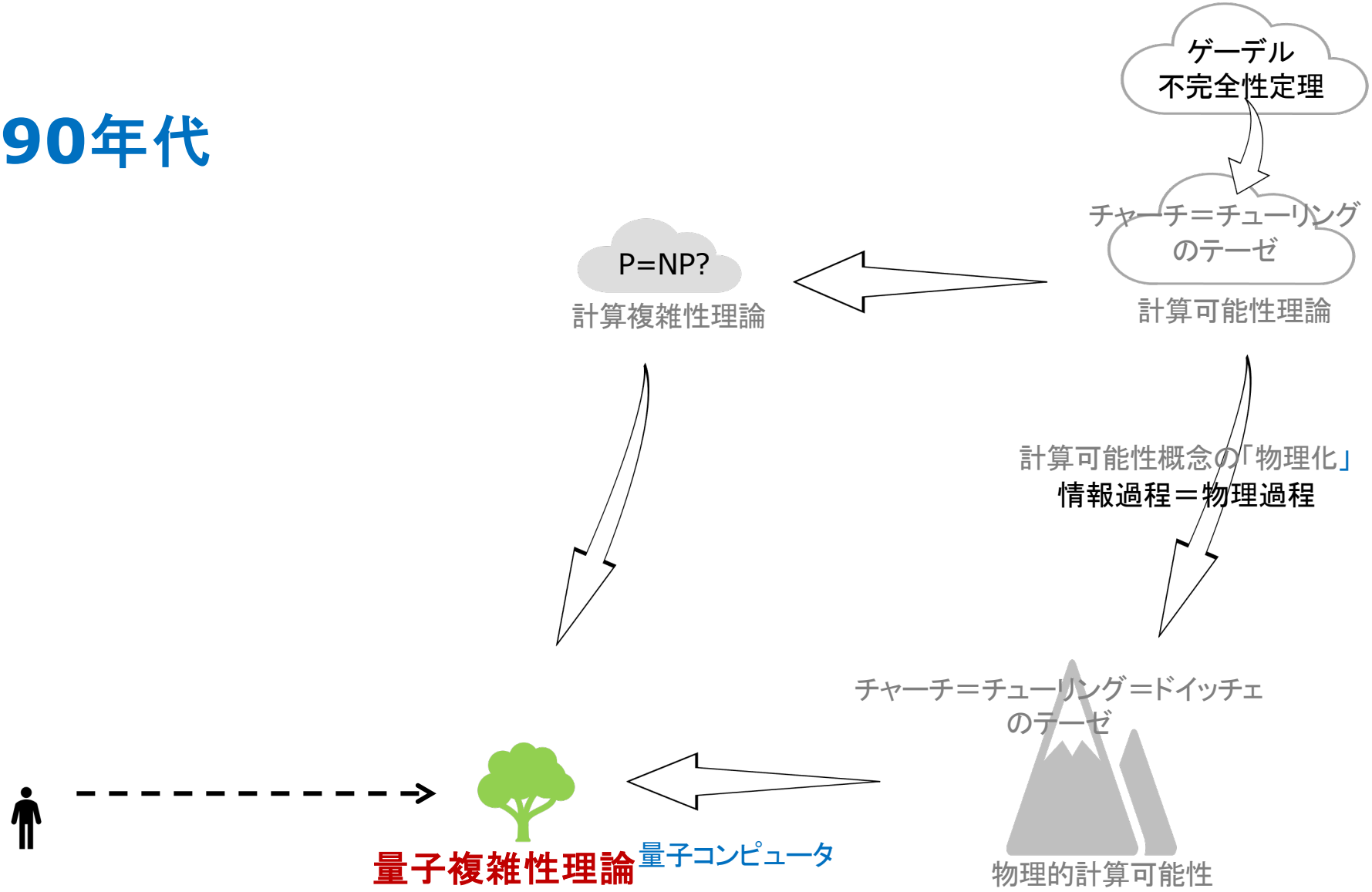
クック、レビン、カープらは、計算可能だが実際には手に負えない計算がある領域を精緻に分類しようとし、計算複雑性理論が生まれる。
「難しい計算」が「易しい計算」には還元できないことを主張する「P ＝ NP ？ 問題」は、現在も未解決である。

80年代



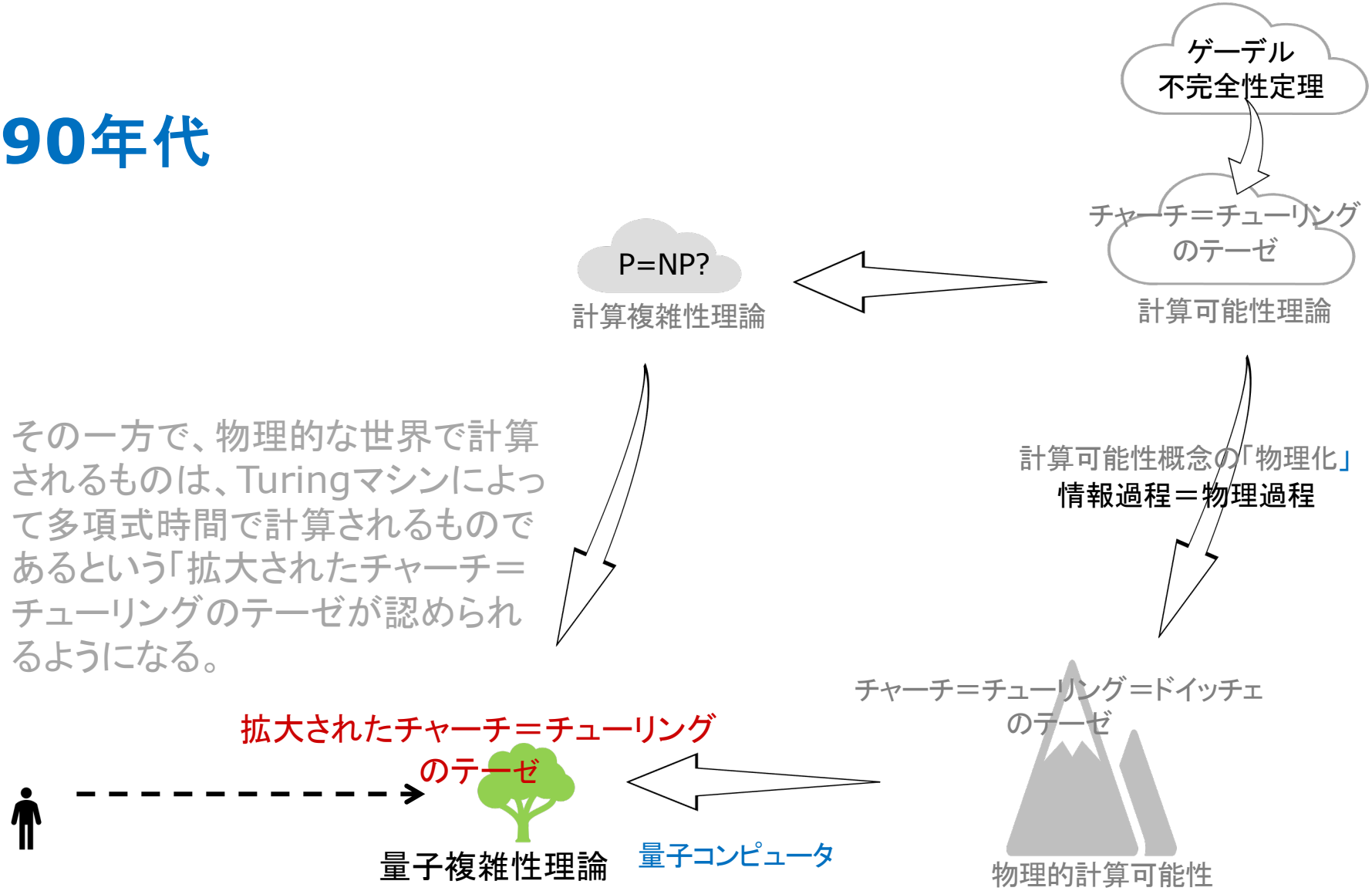
ドイッチェは、ファインマンの自然をシミュレートする量子コンピュータというアイデアに刺激を受けて、計算は、物理的な過程として実現されると主張して、チャーチ＝チューリングのテーゼを拡大した。80年代は、まだ、量子コンピュータは概念としてしか存在していなかった。

90年代



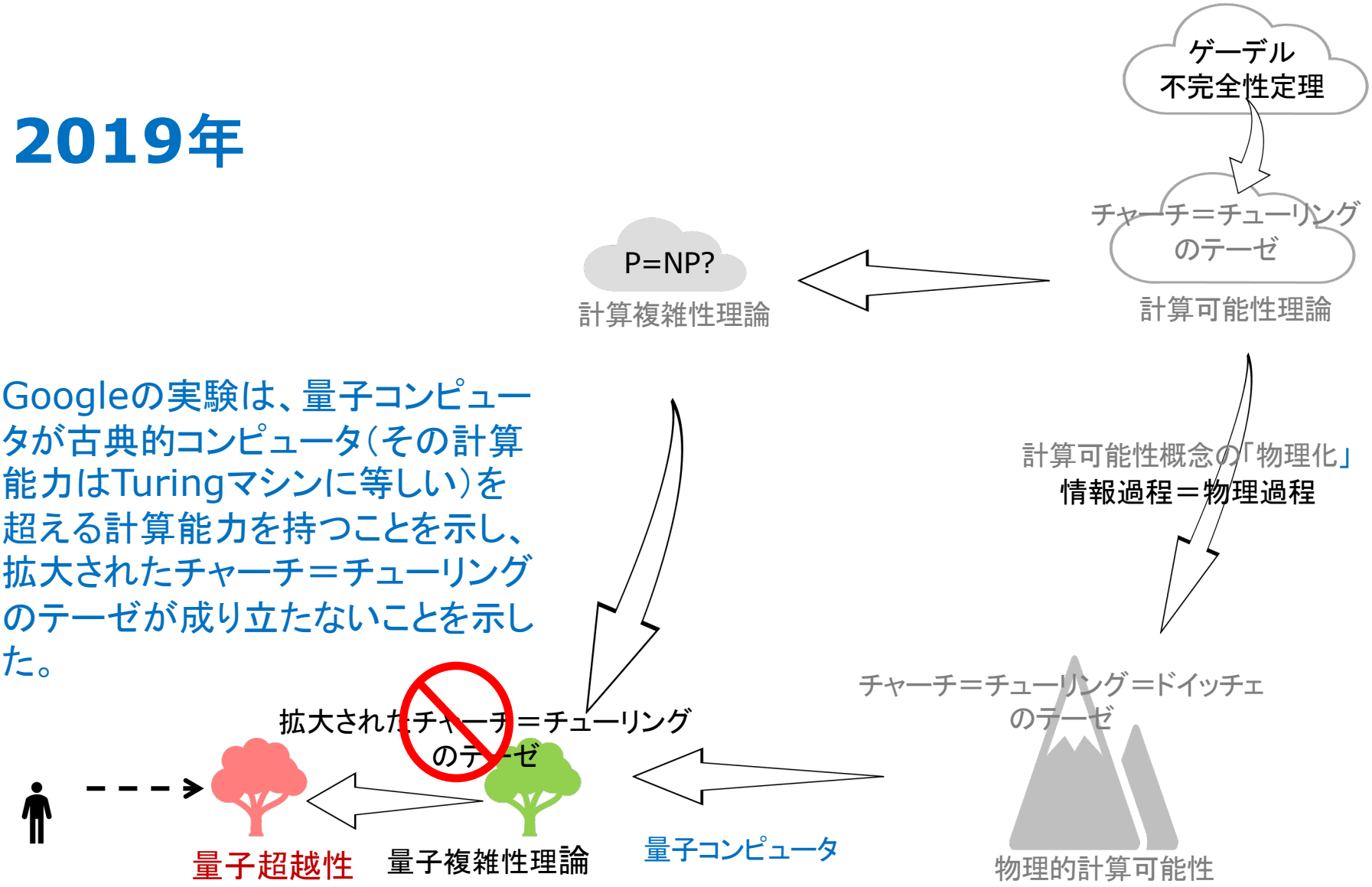
ベルンシュタインとバジラーニによって、量子複雑性の理論が登場し、BQPという新しい複雑性の概念が提案される。広い関心をあつめたショアのアルゴリズムは、このBQPに属するものだった。

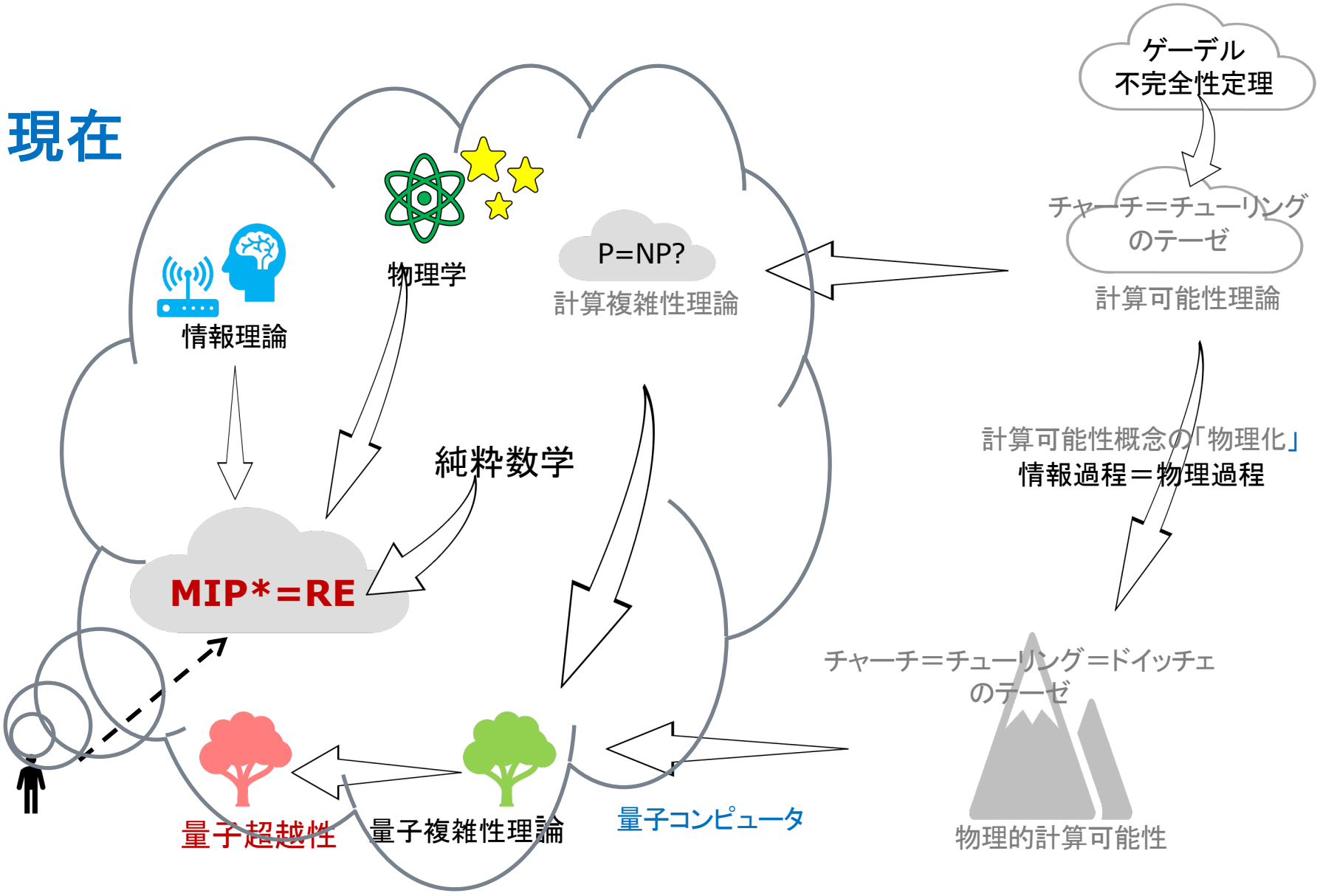
90年代


ゲーデル
不完全性定理
チャーチ＝チューリング
のテーゼ
計算可能性理論
P=NP?
計算複雑性理論
計算可能性概念の「物理化」
情報過程＝物理過程
その一方で、物理的な世界で計算されるものは、Turingマシンによって多項式時間で計算されるものであるという「拡大されたチャーチ＝チューリングのテーゼが認められるようになる。
チャーチ＝チューリング＝ドイッチェ
のテーゼ
拡大されたチャーチ＝チューリング
のテーゼ
量子複雑性理論
量子コンピュータ
物理的計算可能性

2019年

Googleの実験は、量子コンピュータが古典的コンピュータ（その計算能力はTuringマシンに等しい）を超える計算能力を持つことを示し、拡大されたチャーチ＝チューリングのテーゼが成り立たないことを示した。




現在
ゲーデル
不完全性定理
チャーチ＝チューリング
のテーゼ
計算可能性理論
P=NP?
計算複雑性理論
物理学
情報理論
純粋数学
計算可能性概念の「物理化」
情報過程＝物理過程
MIP*=RE
チャーチ＝チューリング＝ドイッチェ
のテーゼ
量子超越性
量子複雑性理論
量子コンピュータ
物理的計算可能性

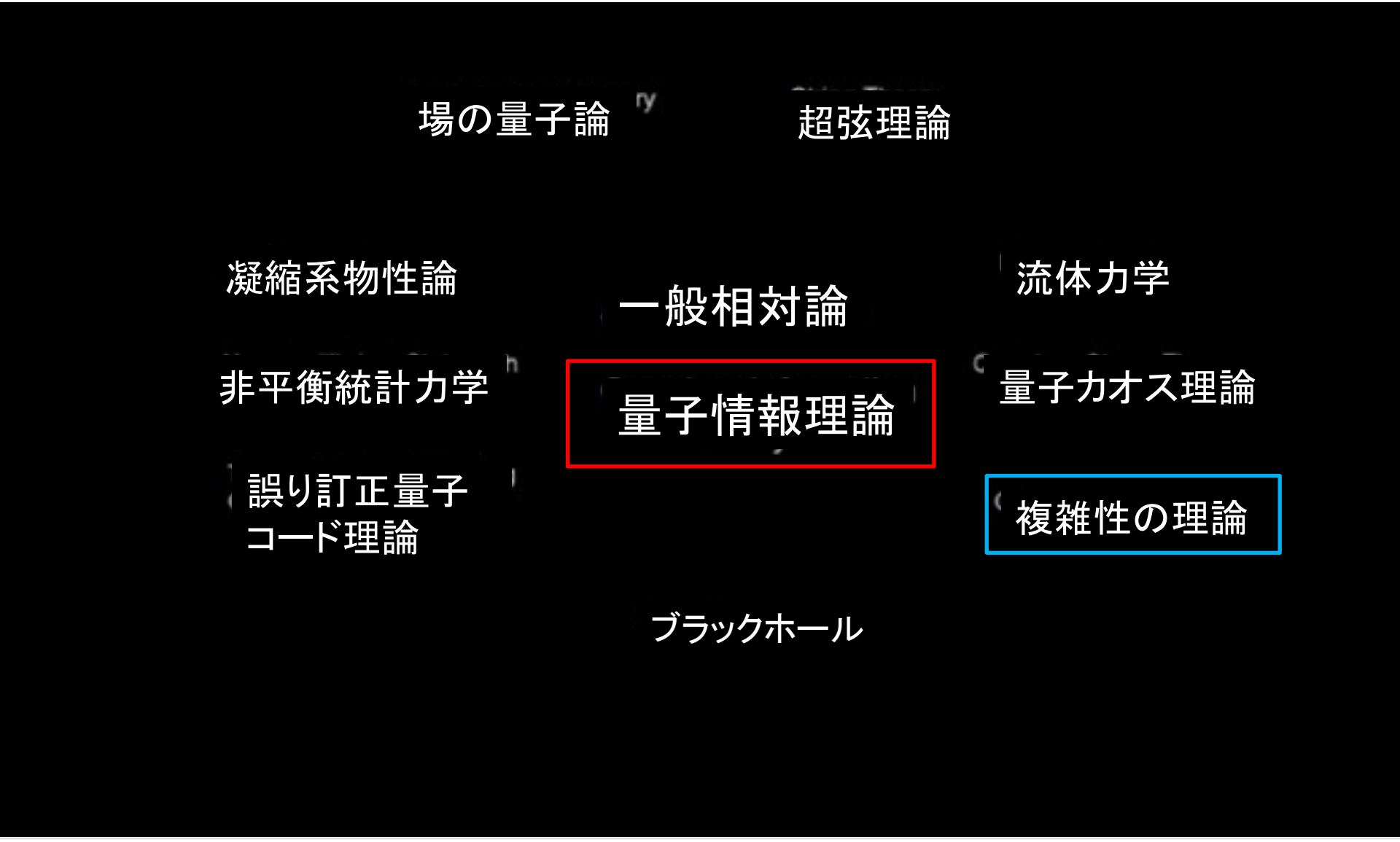
物理学者は、
現代の物理学をどのように捉えているのか？

Church-Turing Thesis の変化は、我々の数学の見方を大きく変えただけではなく、我々の物理学の見方をも大きく変えようとしている。

ここでは、現代の物理学者が、現代物理学の課題を、どのように捉えているのかを見ておこう。

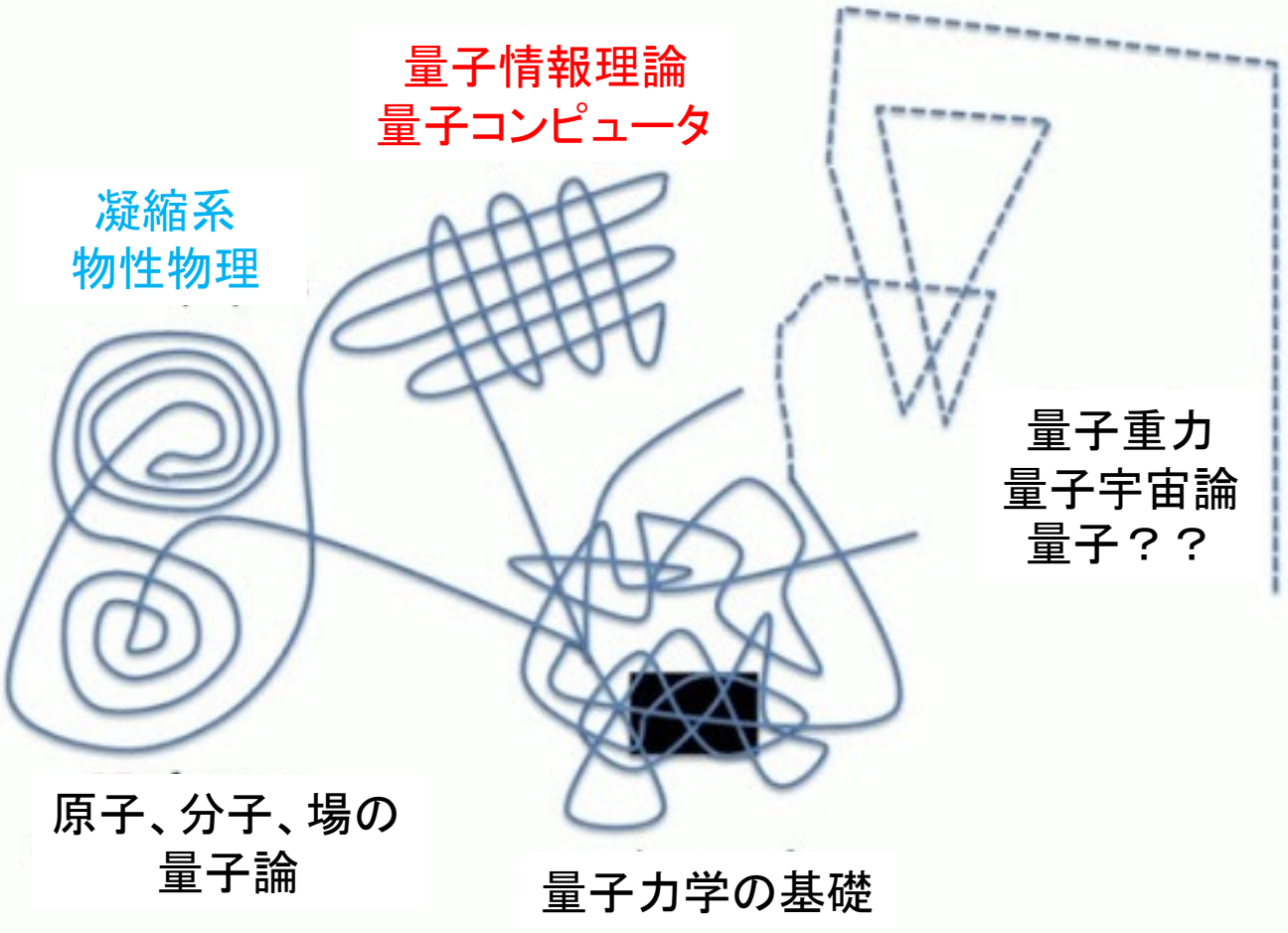
そこでは、複雑性理論の量子複雑性理論への飛躍と量子コンピュータ技術の進歩に伴って発展してきた「量子情報理論」が、物理的世界の認識の基礎理論としての物理学の中心的理論として、大きな関心を集めているのがわかると思う。

現代物理学の俯瞰図 by Susskind

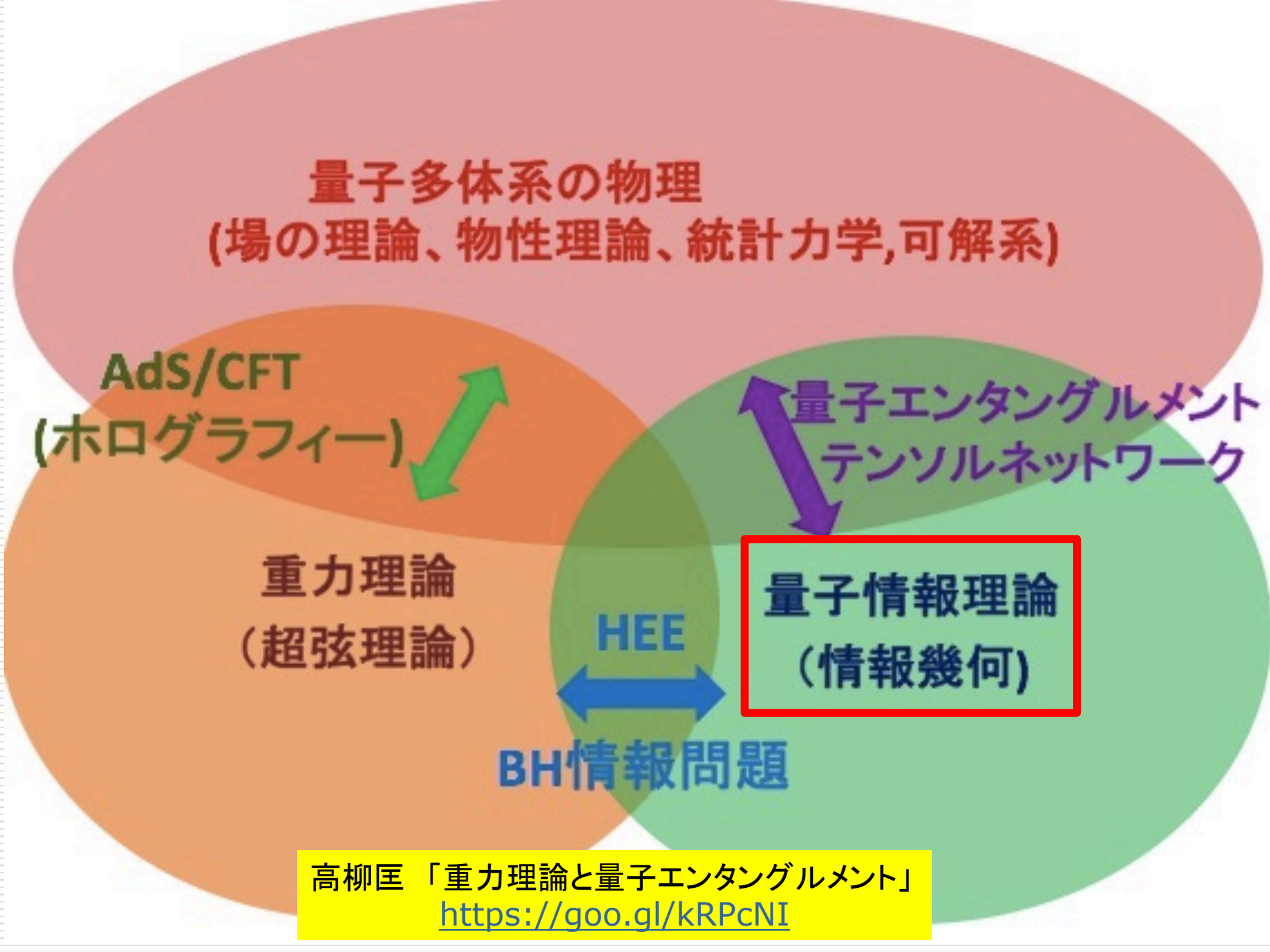


Susskind "Entanglement and Complexity: Gravity and Quantum Mechanics"
https://goo.gl/J0wSkf

エンタングルメントの歴史の絵解き



"Our Quantum Society: Living with Entanglement" https://goo.gl/aWtAzi
Cathryn Carson https://goo.gl/j7bE57



高柳匡 「重力理論と量子エンタングルメント」
https://goo.gl/kRPcNI

Church-Turing Thesis の変化から学ぶべきこと

Church-Turing Thesis の変化から学ぶべきこと

こうした Church-Turing Thesis の変化から、我々は何を学ぶことが出来るのでしょう？

あらためてこのセミナーのはじめに述べたことを確認したいと思います。

私たちは、「機械の知能は、どこまで発展しうるのか？」という問題を考える必要があります。

重要なことは、その問題は、「我々人間の認識は、どこまで発展しうるのか？」という問題と同時に提起されているということです。

楽観論者も悲観論者も入り混じっているのですが、我々がそうした問題の前に立たされていることは、多くの人が、感じていることだと思います。

なぜ今「計算複雑性理論」なのか？

「計算複雑性理論」は、人間の数学的な認識能力の拡大とその限界を「現実的」に境界付けようという理論なのですが、その理論は、基本的に、「チャーチ＝チューリング・テーゼ」に基づいて計算という人間の数学的能力に「チューリング・マシン」という機械的なモデルを与えることで展開します。

それは、人間が可能な計算の現実的な限界の理論であると同時に、そのモデルである機械が可能な計算の限界の理論でもあるのです。

それは、人間の認識の限界の理論であると同時に、機械の認識の限界の理論でもあるのです。

「計算複雑性理論」が示すそうした視点は、我々が今考えるべき、機械と人間の認識能力の発展の可能性について、また、機械と人間の未来での関係について、多くの示唆を与えると僕は考えています。