

再配布禁止

情報科学概論 第4回

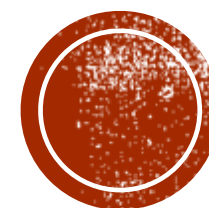
計算の理論

立教大学大学院 人工知能科学研究科

2023年5月8日

アンドラーデ ダニエル

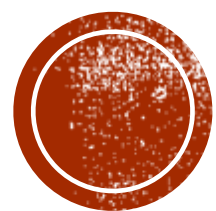
andrade@rikkyo.ac.jp



「問題」と「計算機」の概念を抽象化し、計算機で解ける問題と解けない問題を洗い出す。

1. 計算機に解ける問題の種類
2. PとNPの問題集合





1. 計算機に解ける問 題の種類

「問題」の定義

これから「問題」は「判定問題」と表し、次のように定義する。

$$L(x) = \begin{cases} \text{True}, & \text{入力}x\text{がある条件を満たしているとき} \\ \text{False}, & \text{そうでないとき} \end{cases}$$

$L(x)$ は判定問題 (decision problem)と呼ぶ。ここでは単に「問題」と呼ぶ。入力 x は文字列。

Pythonで考えると、例えば、次の関数はある「問題」を定義する：

```
def L(x):  
    if "aba" in x:  
        return True  
    else:  
        return False
```

つまり

$$L(x) = \begin{cases} \text{True}, & \text{入力}x\text{に"aba"が含まれている。} \\ \text{False}, & \text{そうでないとき。} \end{cases}$$



「計算機」の定義

(Pythonの例の続き) 前頁の問題はパソコンという計算機で解ける。

理論計算機科学では「計算機」を抽象化した概念として次のようなものが利用されている。

- Finite State Automaton (有限状態機械)
- Pushdown Automaton (プッシュダウン・オートマトン)
- Turing Machine (チューリング機械)

解ける問題の種類が機械によって異なる。

今回の授業はFinite State Automaton と Turing Machineのみを紹介する。



「問題を解く」の定義

「機械Aが問題Lを解く」の意味は

任意の入力 $x \in \Sigma^*$ に対して

機械Aが停止 \wedge

$L(x) \Leftrightarrow A$ が停止した時の状態が「受理」(Accept)である。

つまり、任意の入力 $x \in \Sigma^*$ に対して機械Aが止まる（永遠に計算はしない）。また、 $L(x) == \text{True}$ であれば、最後の状態が「受理」。 $L(x) == \text{False}$ であれば、最後の状態が「受理」ではない。

機械が「受理」ではない状態で止まる場合は「拒否」(Reject)という。

x は文字集合 Σ 上の文字列。例えば、 $\Sigma = \{a, b\}$ であれば、 $\Sigma^* = \{“a”, “aa”, “abab”, “bba”, \dots\}$ 。



計算機への入力

- 説明上の都合で、有限サイズの入力 x が無限サイズのテープに書かれているとする。
- 連続する区画に 1 文字ずつが書かれている。
- 入力 x の前後には 空白文字 $_$ が書かれている。
- 例えば、“aba” が入力の場合

...

$_$	$_$	a	b	a	$_$	$_$	$_$
------	------	---	---	---	------	------	------

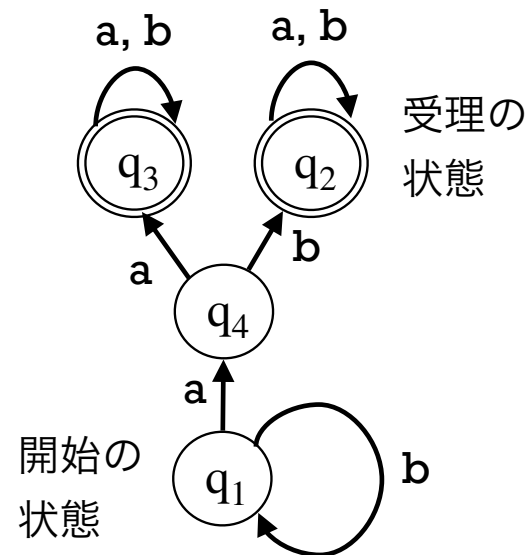
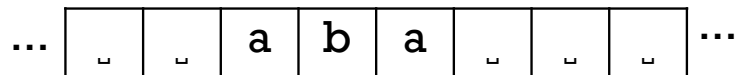
 ...



FINITE STATE AUTOMATON (有限状態機械)

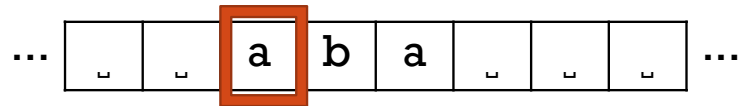
- 有限の状態を持つ。状態の集合 $Q = \{q_1, q_2, \dots, q_r\}$
- 入力 x を 1 文字ずつ左から右に読み、 $_$ に着いたら、機械が停止する。
- 1 字を読んだ後に、現在の状態 q が q' に変わる。 $q, q' \in Q$
- q_1 は開始の状態とする。（1 より大きい添え字は順番など関係がない）
- 「受理」の状態集合は $Q_{\text{accept}} \subseteq Q$. 「受理」の状態は図上で二重丸で表示。

例：

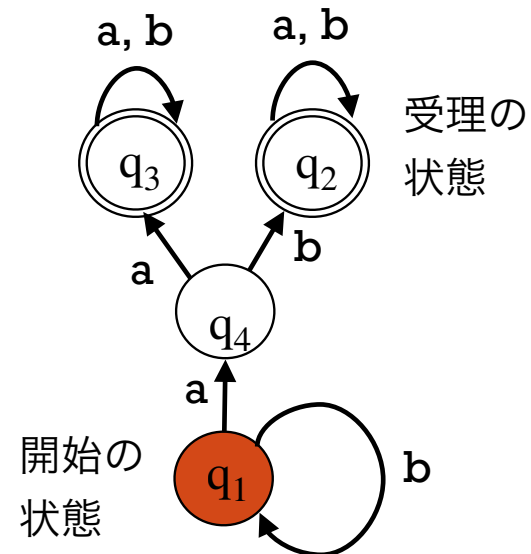


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”aba”

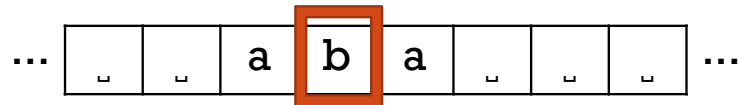


有限状態機械 A

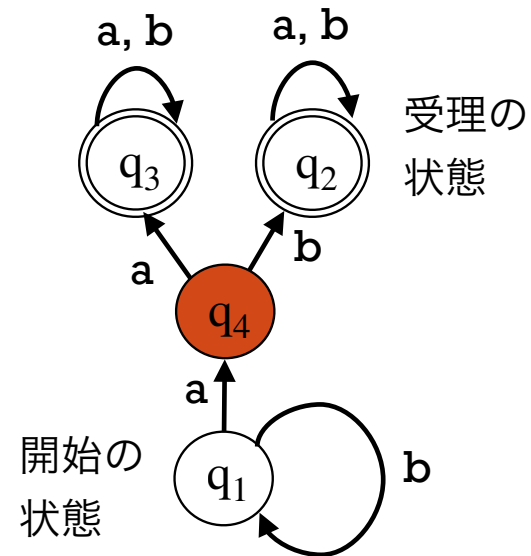


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”aba”

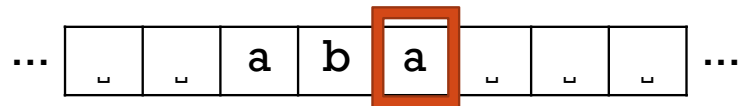


有限状態機械 A

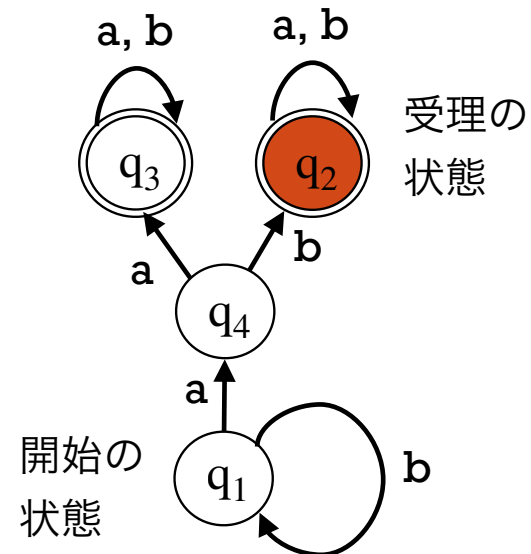


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”aba”

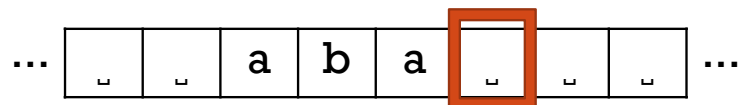


有限状態機械 A



FINITE STATE AUTOMATON (有限状態機械)

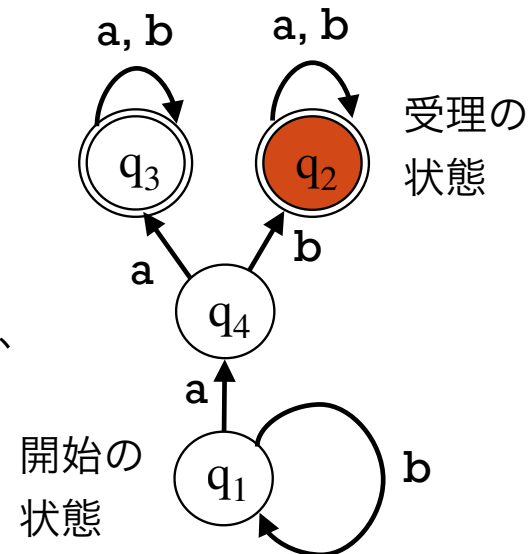
例：入力が"aba"



停止

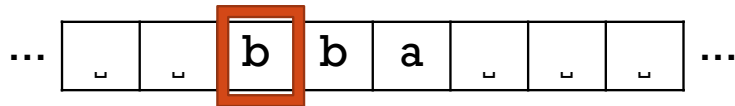
状態が「受理」のため、
入力が受理される

有限状態機械 A

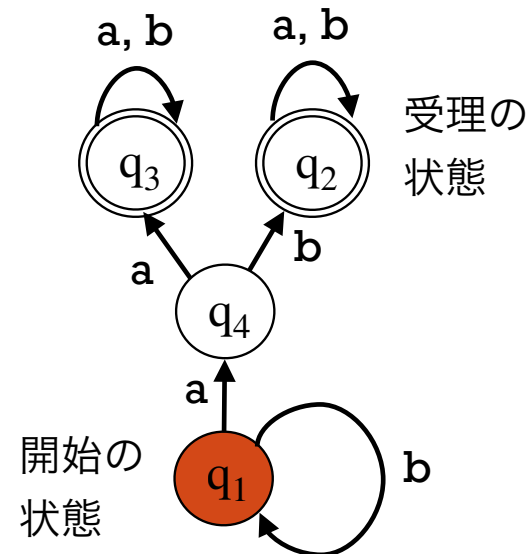


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”bba”

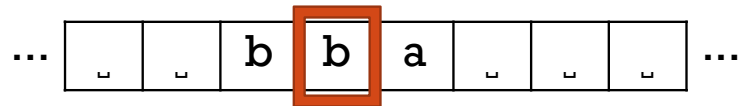


有限状態機械 A

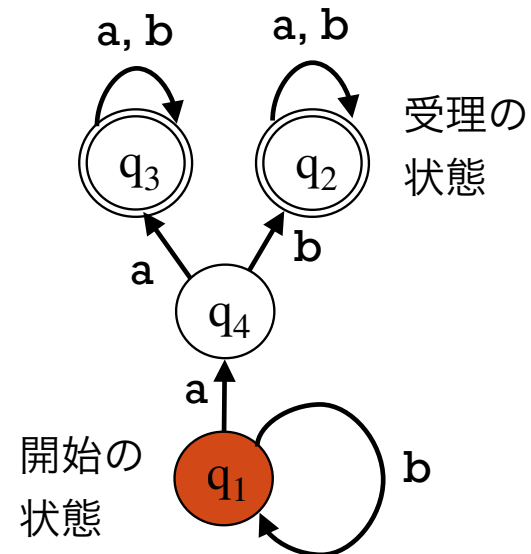


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”bba”

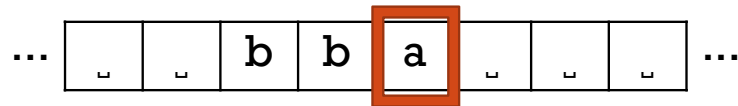


有限状態機械 A

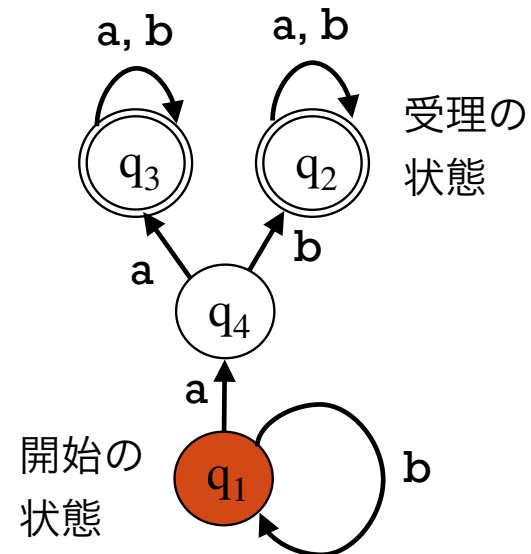


FINITE STATE AUTOMATON (有限状態機械)

例：入力が”bba”

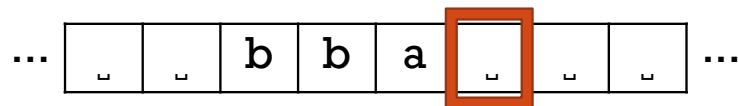


有限状態機械 A



FINITE STATE AUTOMATON (有限状態機械)

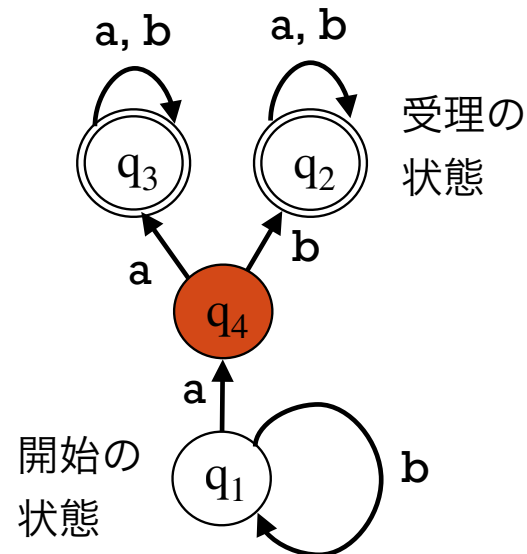
例：入力が”bba”



停止

状態が「受理」
ではないため、
入力が拒否され
る。

有限状態機械 **A**

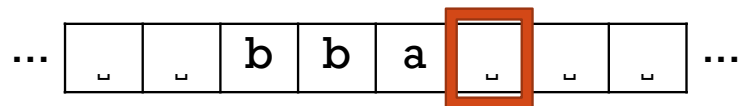


有限状態機械 **A** が解く問題 $L(\mathbf{x})$ は何？



FINITE STATE AUTOMATON (有限状態機械)

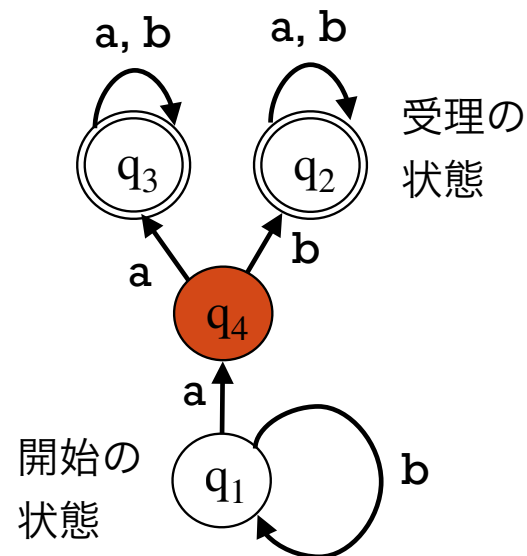
例：入力が”bba”



停止

状態が「受理」
ではないため、
入力が拒否され
る。

有限状態機械 **A**



有限状態機械 **A** が解く問題 $L(\mathbf{x})$ は何？

$$L(x) = \begin{cases} \text{True,} & \text{入力 } x \text{ に "ab" または "aa" が含まれている} \\ \text{False,} & \text{そうでないとき} \end{cases}$$



FINITE STATE AUTOMATONが解けない問題

- 有限状態機械はすべての問題を解けない。

- 例：

$$L(x) = \begin{cases} \text{True}, & \text{入力 } x \text{ は } a^n b^n, n \in \mathbb{N} \\ \text{False}, & \text{そうでないとき} \end{cases}$$

例えば $L(\text{"aba"}) = \text{False}$, $L(\text{"aabb"}) = \text{True}$, $L(\text{"aaabb"}) = \text{False}$ 。

定理：上記の問題 **L** を解ける有限状態機械が存在しない。

証明： 上記の問題 **L** を解ける有限状態機械 **A** があるとする。

A の状態集合が有限のため、 $Q = \{q_1, q_2, \dots, q_r\}$, $r \in \mathbb{N}$ で表現できる。


開始から a^n を読んだ後に **A** の状態が q とする。

開始から a^m を読んだ後に **A** の状態が q' とする。

ただし、 $n > r \Rightarrow \exists m \neq n: q = q'$ 。

入力 $a^n b^n$ が入力された **A** の状態が「受理」の状態になる。

ただし、 $a^m b^n$ の場合でも **A** の状態が同じ「受理」状態になってしまう。


 矛盾 \Rightarrow **L** を解ける有限状態機械が存在しない。

(\mathbb{N} は整数の集合、
つまり $\{0, 1, 2, 3, \dots\}$,
無限自体が \mathbb{N} に含まれてい
ない。つまり、 $\infty \notin \mathbb{N}$)

“aa” \rightarrow q

“aaa” \rightarrow q

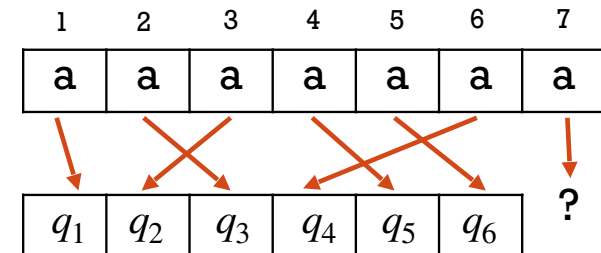
“aabb” \rightarrow 受理

“aaabb” \rightarrow 受理 

鳩の巣原理

例：

$n = 7$,
 $r = 6$

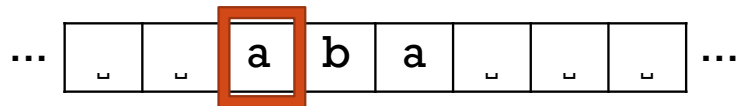


TURING MACHINE（チューリング機械）

下記の点以外には有限状態機と同じ：

- 書き込む機能：テープ上の現在の文字を読んだ後に、その文字を置き換えることが可能。
- 移動方向の変更：テープ上で右か左への移動が可能。
- 停止を自分で決める：読んだ文字と現在の状態によって自分で停止を決める。

関数を使った厳密な定義は定義6、ページ146（「計算の理論」 山口和紀, 情報第2版）に参照。

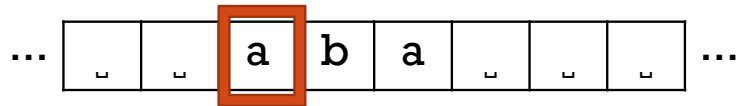


有限状態機械と違って**Turing Machine**が停止しない可能性がある。プログラムの無限ループと同じ。
停止しない場合は、「受理した」とも「拒否した」ともいわない。

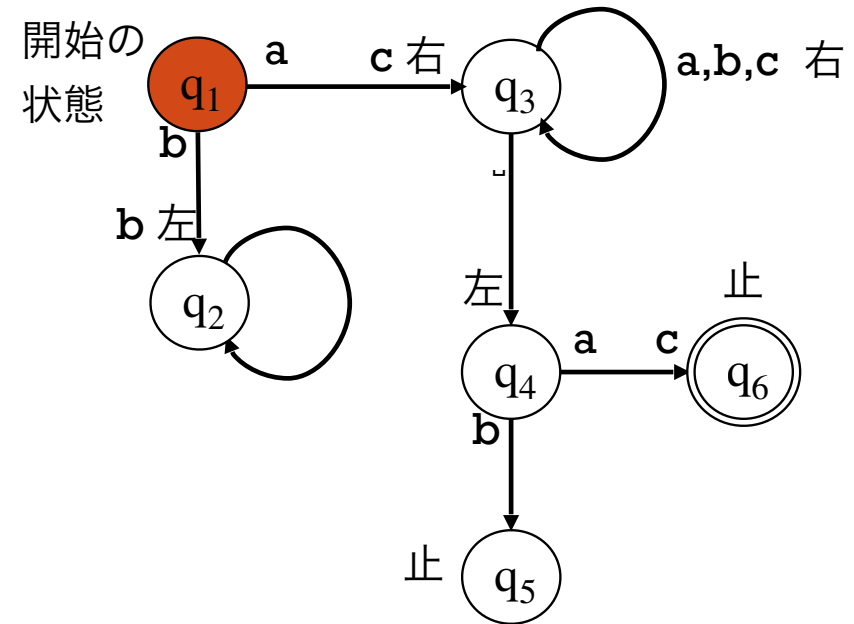


TURING MACHINE

例：

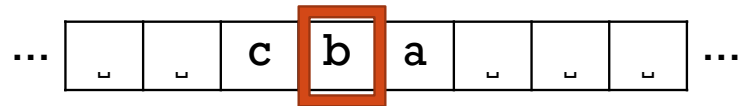


Turing Machine T

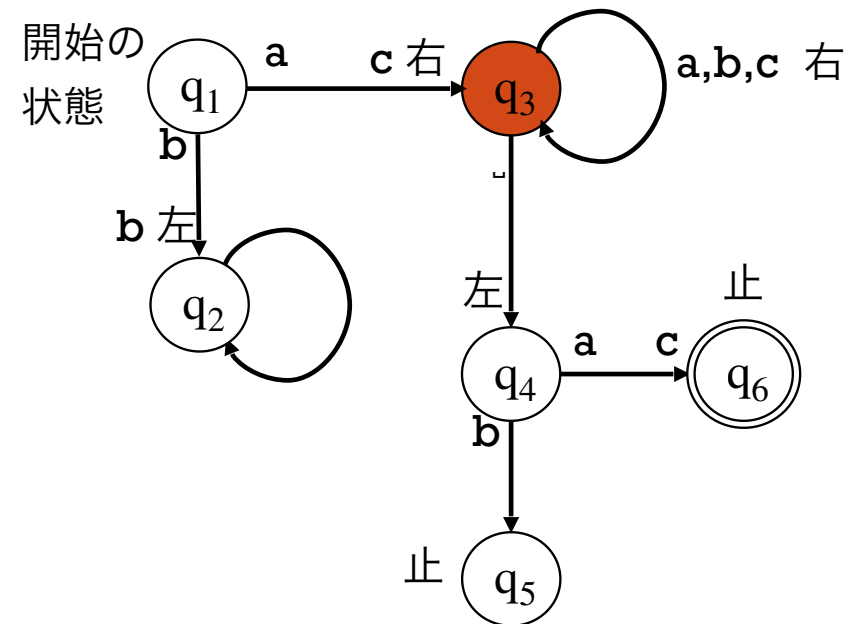


TURING MACHINE

例：

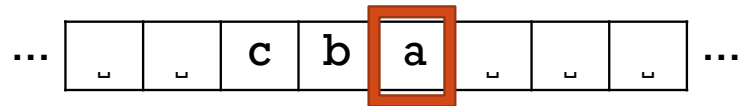


Turing Machine T

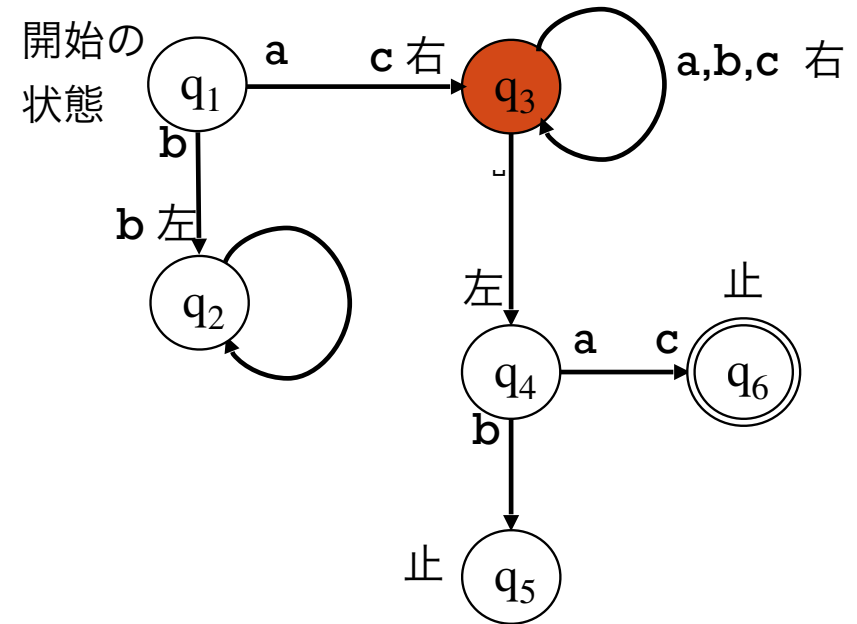


TURING MACHINE

例：

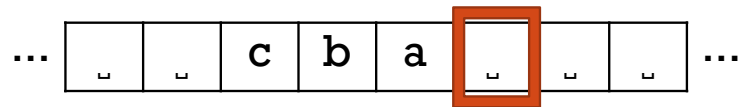


Turing Machine T

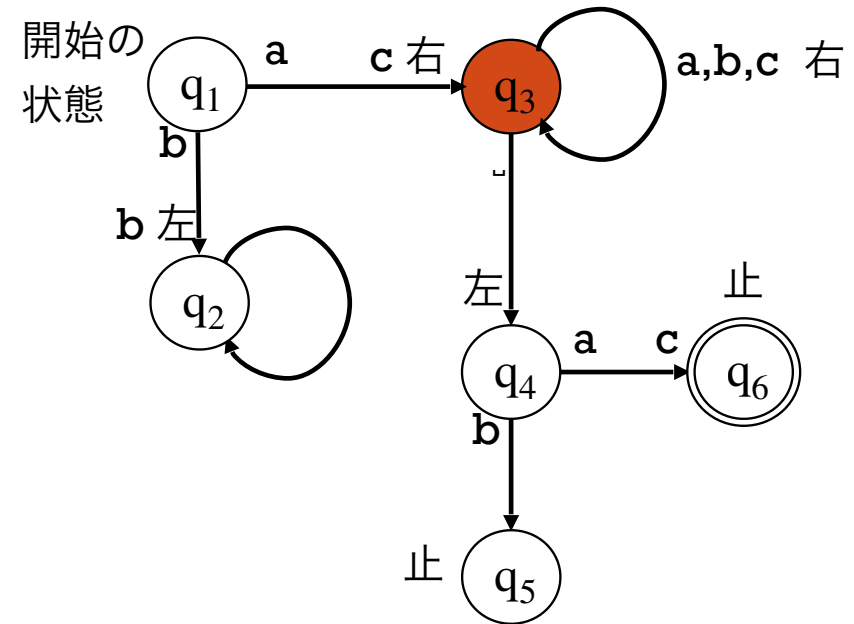


TURING MACHINE

例：

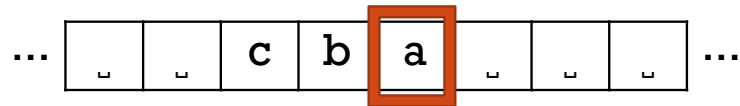


Turing Machine T

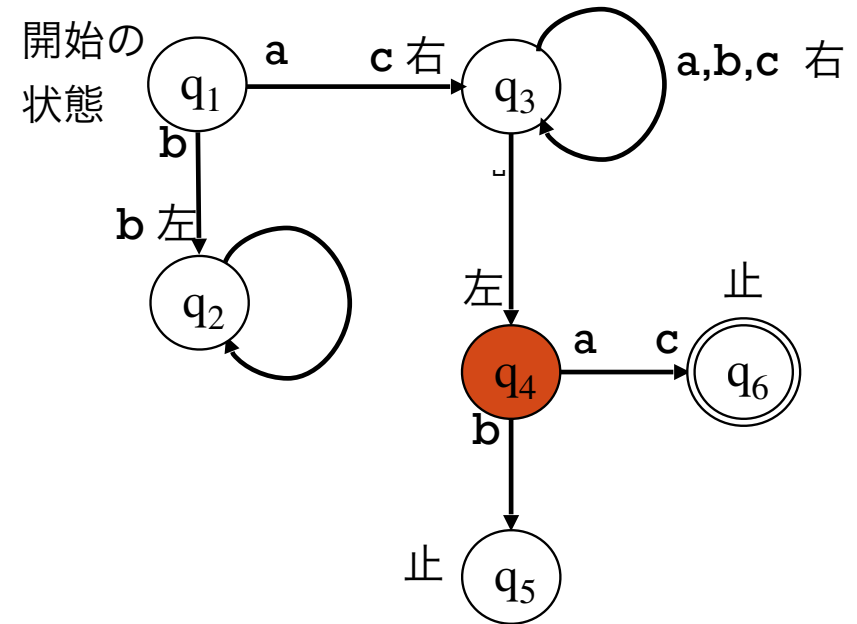


TURING MACHINE

例：

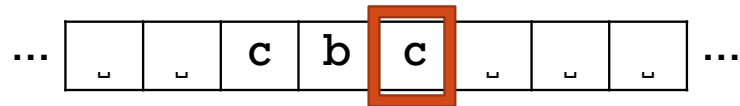


Turing Machine T

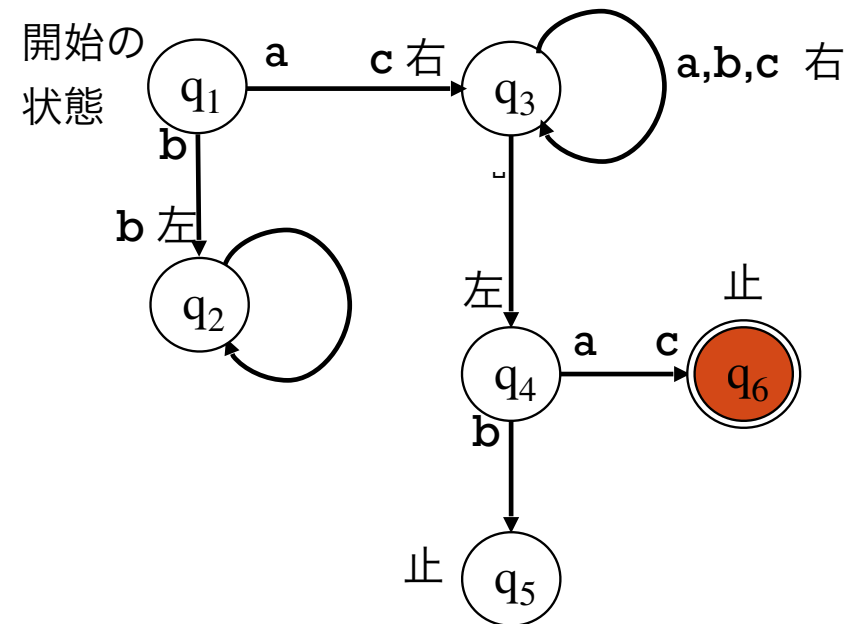


TURING MACHINE

例：



Turing Machine T



TURING MACHINE

有限状態機械と違ってチューリング機械は以下の問題を解ける。

$$L(x) = \begin{cases} \text{True}, & \text{入力 } x \text{ は } a^n b^n, n \in \mathbb{N} \\ \text{False}, & \text{そうでないとき} \end{cases}$$

構築の概要：

$\Sigma = \{a, b\}$ だが、チューリング機械が文字「c」も書き込む。

- 1) テープ上に書かれている文字列の中で一番左の「c」でない文字に移動。そういう文字が存在しなければ「受理」。また「a」でなければ「拒否」。
- 2) 文字「a」を「c」と置き換える。
- 3) テープ上に書かれている文字列の中で一番右の「c」でない文字に移動。そういう文字が存在しなければ「拒否」。また「b」でなければ「拒否」。
- 4) 文字「b」を「c」と置き換えて、1)に戻る。



TURING MACHINEが解ける問題

Turing Machineが有限状態機械より強力：

- \forall 問題 L : 有限状態機械が L を解く \Rightarrow Turing Machineが L を解く
- \exists 問題 L : Turing Machineが L を解く \Rightarrow 有限状態機械が L を解けない

Turing Machineを強化しても、解ける問題が変わらない。

(例えば、テープを増やしたり、各ステップでテープ上の任意の位置に移動できたりしても解ける問題が同じ)

アルゴリズムを実行して計算できる機械はすべてTuring Machineで書けると考えられている。
(Church-Turing Thesis)

Turing Machineは無限のメモリを持つパソコンと同じ能力

(解ける問題の種類が同じだが、計算コストでは別！)

ただしTuring Machineが解けない問題も存在している。



補足：停止問題 (HALTING PROBLEM)

ここでは「機械」 = 「Turing Machine」

おさらい：

機械 M が**停止**したら、入力 x が受理されたか拒否されたか（必ずどちらか）。

機械 M が停止しなかったら、「受理」も「拒否」もどちらとも言わない。（外から見ると、今後「受理」するか、「拒否」するか、永遠に結果が出ないからわからない状況）

「停止問題」は次のように定義されている。

$$L(M\#x) = \begin{cases} True, & x \text{ を機械 } M \text{ に入力すると } M \text{ が停止する} \\ False, & x \text{ を機械 } M \text{ に入力すると } M \text{ が停止しない} \end{cases}$$

「 $M\#x$ 」では機械 M が文字列として符号化された。

停止問題を解く機械が存在しない。



補足：停止問題 (HALTING PROBLEM)

証明

証明：

- 停止問題を解く機械が存在すると仮定しよう。その機械をNと呼ぶ。
- Nを使って、次の機械 D を作る。

$$D(x) = \begin{cases} \text{停止しない,} & x \text{を機械} x \text{に入力すると} x \text{が停止する} \\ \text{停止する,} & x \text{を機械} x \text{に入力すると} x \text{が停止しない} \end{cases}$$

Dの作り方：Nが「 $x\#x$ は停止する」と判断したら、無限ループに入る。

Nが「 $x\#x$ は停止しない」と判断したら、停止する。



補足：停止問題 (HALTING PROBLEM)

証明

- 機械Nの出力テーブル:
- M#xを入力した場合、機械Nが「停止する」と判断。（つまりMが入力xに対して停止する）
 - ✗ M#xを入力した場合、機械Nが「停止しない」と判断。（つまりMが入力xに対して停止しない）

入力 x (機械に限定)

あらゆる機械 M

○	✗	✗	○	✗	○	○
✗	○	✗	○	✗	✗	○
○	○	✗	○	○	○	○
✗	✗	○	○	○	✗	○
○	○	✗	○	✗	○	○
✗	✗	○	✗	○	○	○
○	✗	○	○	○	○	✗

縦軸と横軸で並んでいる機械が同じ順番。

機械 D

✗	✗	○	✗	○	✗	○
---	---	---	---	---	---	---

機械Nの出力テーブルの対角成分の反対。



補足：停止問題 (HALTING PROBLEM)

証明

- 機械Nの出力テーブル:
- M#xを入力した場合、機械Nが「停止する」と判断。（つまりMが入力xに対して停止する）
 - ✗ M#xを入力した場合、機械Nが「停止しない」と判断。（つまりMが入力xに対して停止しない）

		入力 x (機械に限定)						
あらゆる 機械 M	機械 D	○	✗	✗	○	✗	○	○
		✗	○	✗	○	✗	✗	○
		○	○	✗	○	○	○	○
		✗	✗	○	○	○	✗	○
		○	○	✗	○	✗	○	○
		✗	✗	○	✗	○	○	○
		○	✗	○	○	○	○	✗

縦軸と横軸で
並んでいる機械が
同じ順番。

Dが機械であるた
め、Dと対応してい
る行があるはず。

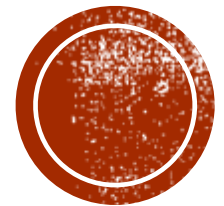
機械 D	✗	✗	○	✗	○	✗	○
------	---	---	---	---	---	---	---

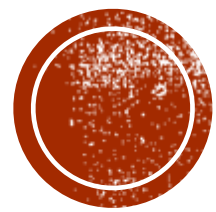
機械Nの出力テーブルの対角成分の反対。

⇒ Nが存在する仮定が間違っているはず。

矛盾

質問タイム





2. 計算量：

PとNPの問題集合

計算量

- 解ける問題の観点では、チューリング機械がパソコン・計算機を代表できる。
- そのため理論情報科学ではチューリング機械の計算量を議論する。
- 計算量は入力の大きさに応じて処理時間がどのように増えるかで測る。
- チューリング機械の場合では
 - 入力の大きさ = 入力されている文字列の長さ n
 - 処理時間 = テープ上の遷移の回数
- 判定問題（アルゴリズム）の 処理時間を測るために、主に二つの見方がある。
 - Average runtime（平均の処理時間）
 - Worst-case runtime（最悪の場合の処理時間）
- ここではWorst-case処理時間が n に対して多項式で表現できるかないかに焦点を置く。



Pの定義

問題集合 **P** (**P**olynomial time solvable) は以下のように定義されている。

問題 $L \in P \iff \exists$ チューリング機械 M : M が L を解く \wedge M の処理時間が $p(n)$

$p(n)$ は n に対する任意の多項式 (例えば $p(n) = n^6 + 4n^2$)

- 例 :
 - 有限状態機械で解ける問題 (なぜ? 考えてみてください)
 - $a^n b^n$ かどうかの問題
 - 無向グラフ上で最短距離の経路かどうかの問題



NPの定義

問題集合 NP (**N**on-deterministic **P**olynomial time solvable) は以下のように定義されている。

問題 $L \in NP \Leftrightarrow \exists$ 問題 $R \in P$: L が以下のように表現できる。

$$L(x) = \begin{cases} \text{True} , & R(x, y) \text{ が True になるような入力 } y \text{ が存在し、} y \text{ の長さが } p(n) \text{ 以下} \\ \text{False,} & \text{そうでないとき} \end{cases}$$

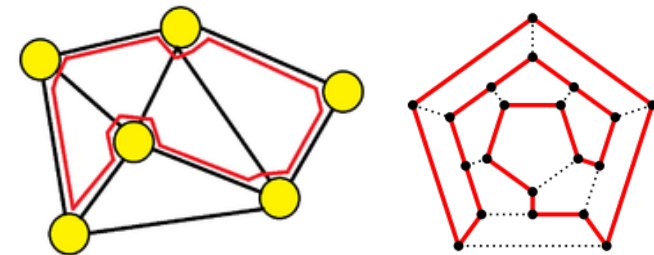
つまり、 $L(x)$ が **True** になるような解 y を多項式の処理時間で検証できると、 L は **NP**に入っている。

例：

入力グラフ x にハミルトン路があるかないか問題が**NP**に入っている。 (*)

y = ハミルトン路

$R = y$ はグラフ x 上のハミルトン路かどうか問題、 $R \in P$



(*) x が文字列として適宜に符号化したもの。

ハミルトン路 = 各ノードを一回だけ通る経路

図は Wikipediaより https://en.wikipedia.org/wiki/Hamiltonian_path



NPの定義

問題集合 NP (**N**on-deterministic **P**olynomial time solvable) は以下のように定義されている。

問題 $L \in NP \Leftrightarrow \exists$ 問題 $R \in P$: L が以下のように表現できる。

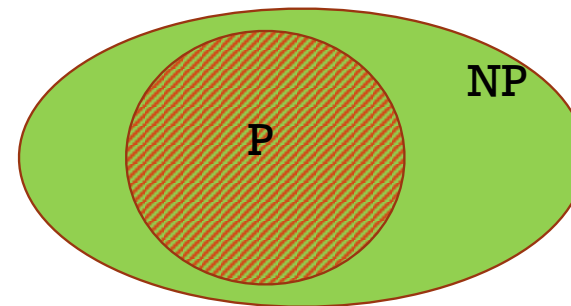
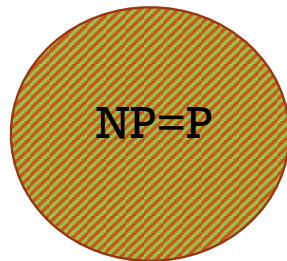
$$L(x) = \begin{cases} \text{True}, & R(x, y) \text{ が True になるような入力 } y \text{ が存在し、} y \text{ の長さが } p(n) \text{ 以下} \\ \text{False}, & \text{そうでないとき} \end{cases}$$

NPの定義によって $L \in P \Rightarrow L \in NP$ が成り立つ。

(証明: R として L を利用し、 y は任意、例えば空欄。)

$L \in NP \Rightarrow L \in P$ は誰もまだ知らない。

皆がこちらが正しいと思っているが、証明はまだない。



NP-COMplete (NP完全)

皆がこちらが正しいと思っているが、
証明はまだない。

- NP-complete \subseteq NPという問題集合には様々な難しい問題が含まれている。

- ハミルトン路
- Traveling Salesman Problem
- SAT問題
-

- どれも半世紀以上、多項式で解けるアルゴリズムは見つかっていない。

- NP-complete集合の重要な特徴：

もし 一つの問題 $L \in \text{NP-complete}$ が P に属していると証明されたら、
 $\text{NP-complete} = P = \text{NP}$

⇒ NP-completeに入っている問題はすべて実は簡単（ P に含まれている）だと信じがたいから、

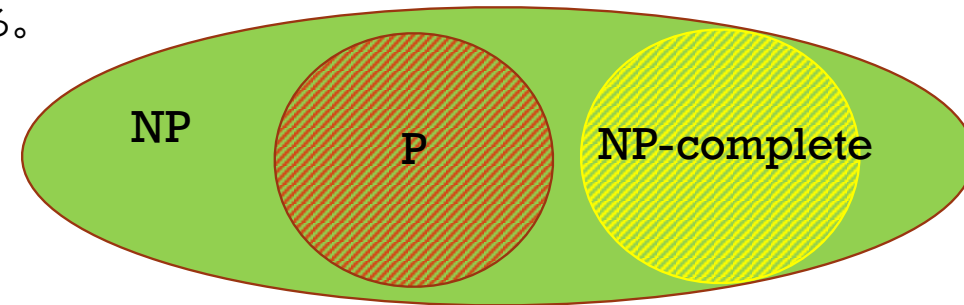
NP-completeに属している問題を効率よく（＝多項式の時間）で解くのが絶望的。

注意点：

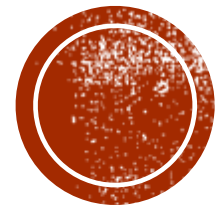
NP と NP-Completeで議論した処理時間はWorst-caseでの時間だった。

つまり、もしあなたの問題が $L \in \text{NP-Complete}$ だと分かったら、すべての入力 x に対して効率よい

アルゴリズムを見つけるのが絶望的だが、うまい具合で特殊のケースに絞って、効率的なアルゴリズムを見つけるチャンスがある。（例えば、機械学習分野で流行っている劣モジュラ関数）



質問タイム



「計算の理論」のまとめ

- 理論上では、「問題」が「判定問題」を表す。
判定問題は文字列の入力を受け、True（真）か False（偽）を返す関数として定義されているものである。
- 「問題」を解くためには様々な理論的な計算機がある。その中で特に重要なのは「有限状態機械」(Finite State Automaton)と「チューリング機械」(Turing Machine)である。
- 有限状態機械は有限の状態を持ち、各入力文字に対して、状態を変え（または状態を維持し）、次の右の文字を読み込む。したがって、入力文字のサイズが n だとすると、遷移の回数が n である。
- 有限状態機械の能力は限られており、 $a^n b^n$ といった問題は解けない。
- チューリング機械は有限状態機械と違って、書き込めるテープが搭載されている。
 $a^n b^n$ 問題も解けるし、チューリング機械より能力のある計算機がないと考えられている。(Church-Turing Thesis)
- チューリング機械の問題を解く能力が本質的にパソコンと同じで、処理時間を大まかに見積もることもチューリング機械を元に議論できる。
- 問題は P か NP に属するかわからないか、つまり、処理時間は多項式時間で解けるか解けないかが重要な分類。(ランダウ記法より大まかな分類)

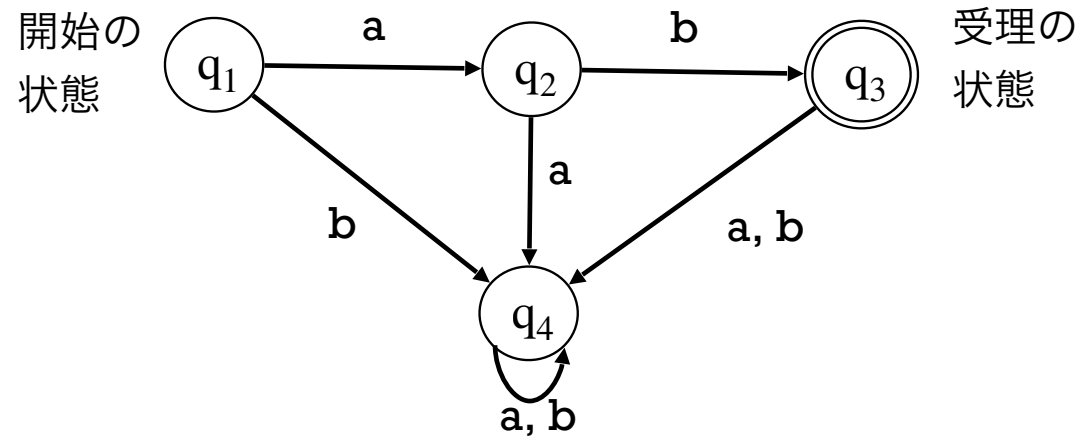


レポートの課題 計算の理論

下記の有限状態機が解く問題を記載してください。

($L(x) = \dots$ の場合分けはBlackboard上で書きやすいように適宜に書いてください。例えば If ... Then ... Else ... で)

$$\Sigma = \{a, b\}, Q = \{q_1, q_2, q_3, q_4\}, Q_{\text{accept}} = \{q_3\}$$



レポートの課題

- 今夜BLACKBOARDに登録します。

「情報の伝達と通信PART2」と「計算の理論」の課題の

締め切りは5月14日の23時59分にします。

締切までは再提出可で、最後に提出されたものを採点します。

- 解答の方法：

最初の一行や一段落で、解答を簡潔に書いてください。

その後に、解答に至るまでの考え方・補足・詳細を簡潔に書いてください。

- レポートに関する質問は5月13日までにお願いします。

(締め切りの当日に私はメール・Slackのメッセージに返事ができないため。)

- Docxファイルとして提出しないでください。pdfに変換したもののご提出をお願いします。

