# 計算と論理

**Jacques Garrigue**

名古屋大学多元数理科学研究科

http://www.math.nagoya-u.ac.jp/~garrigue/

# 歴史的背景

1900年頃〜 ダフィット・ヒルベルトのヒルベルト・プログラムが数学を確実な基礎に据えることを目的とする。そのために曖昧性のない言語として論理を使う。

1928年 ヒルベルトがEntscheidungsproblem (決った手続きによる論理式の決定可能性) を提案する。

1931年 ゲーデルの不完全性定理によって閉じた論理体系で証明できない問題の存在を示す。

第１不完全性定理 算術を含めた体系では成り立つのに証明できない定理が存在する。

第２不完全性定理 さらに量化子を加えると無矛盾性を自分自身の中で証明することが不可能。

1936年 TuringとChurchが計算可能性を形式化することで論理式を決定する手続きが存在しないことを証明する。

# 計算とは何か（1930年頃)

人間が規則に基づいて進める作業

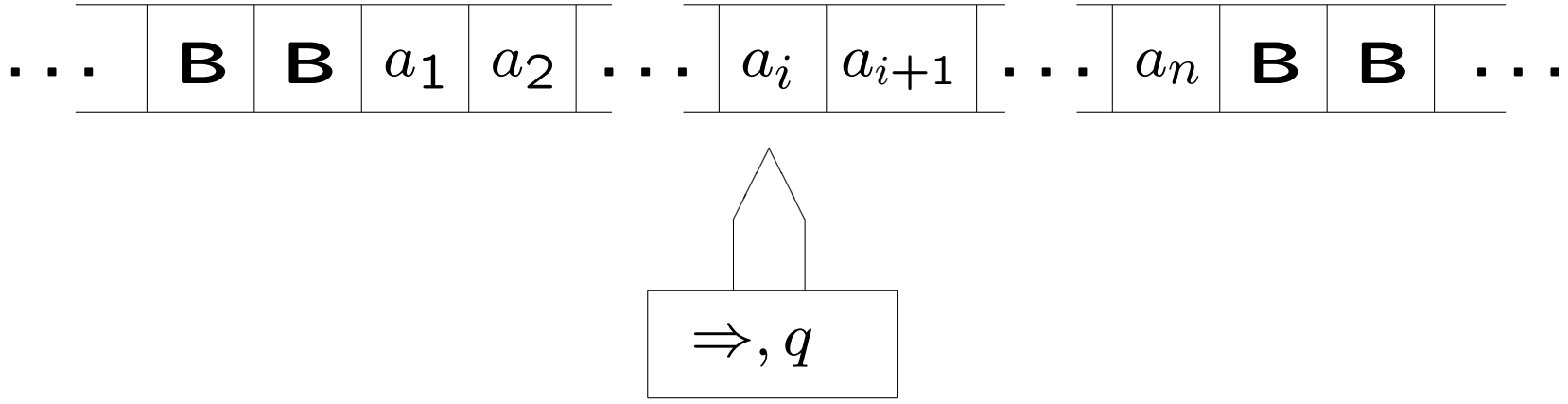
- 規則の適用は頭の中で行う
- 一度に確実に頭に収まるデータの量は有限
- 補助記憶としてノートと鉛筆を使う
- １ページに書く分量は有限
- ノートの枚数を無限としてもよい

# Alan Turingが考えた機械

少し単純化すると、以下のような機械で人間のできる計算が全てできるはず

- ノートの代わりに両端が無限なテープを使う
  - テープは記号が一つだけ書いてある四角の列からできている
  - テープに書ける記号の種類は有限である
- 「頭」はテープのある四角を見て、以下の動作を行う
  - 自分の状態と目の前の記号を元に動作を決定的に決める
  - 同じ位置に新しい記号を書く
  - 右か左に１四角分移動する

# チューリング機械



# チューリング機械の形式的な定義

**定義 1 チューリング機械は次の5つ組によって定義される。**

$$M = (K, \Sigma, \delta, q_0, H)$$

$K$: 空でない有限集合。$K$ の要素を状態という。

$\Sigma$: 空でない有限集合 (アルファベット)。$\Sigma$ の元を記号という。
$\Sigma$ は空白記号Bを含む。

$q_0$: $K$ の要素で、初期状態という。

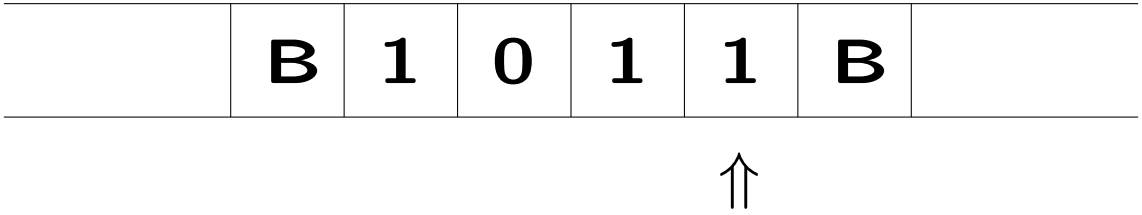
$H$: $K$ の部分集合で、その要素を停止状態という。

$\delta$: $(K \setminus H) \times \Sigma \to \Sigma \times \{\text{左}, \text{右}\} \times K$ なる遷移関数。

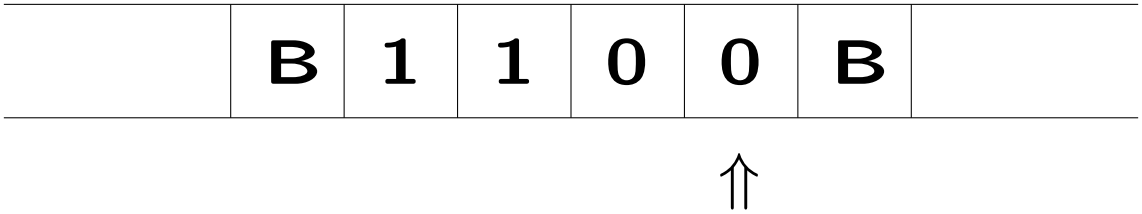
# １を足すチューリング機械

## ２進数で表現された数字に１を足す機械

初期状態は下記の通り $\overline{1011}^2 = 2^3 + 2^1 + 2^0 = 11$




終了時はこうなる $\overline{1100}^2 = 12$



# 1を足すチューリング機械の定義

$\Sigma = \{B, 0, 1\}$

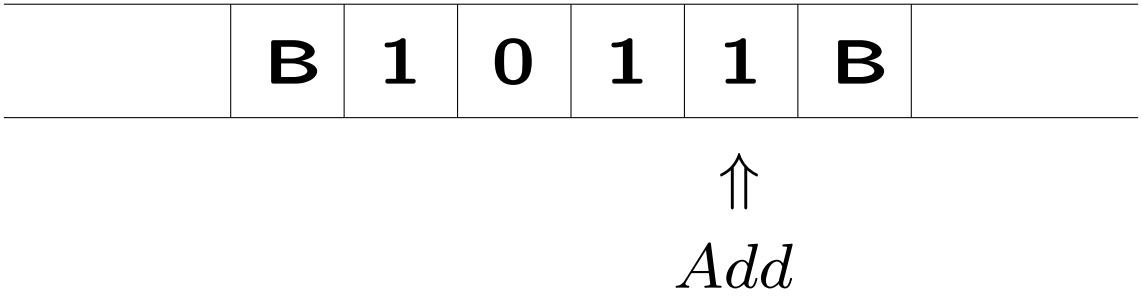
$K = \{Add, Right, Halt\}$, $q_0 = Add$, $H = \{Halt\}$

$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

- 1のときはそれを0に変えて繰り上がり
- 0が見つかるまで左に移動しながら繰り返す
- 見つかったら、1に変えて右に戻る

# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 1を足すチューリング機械の動き

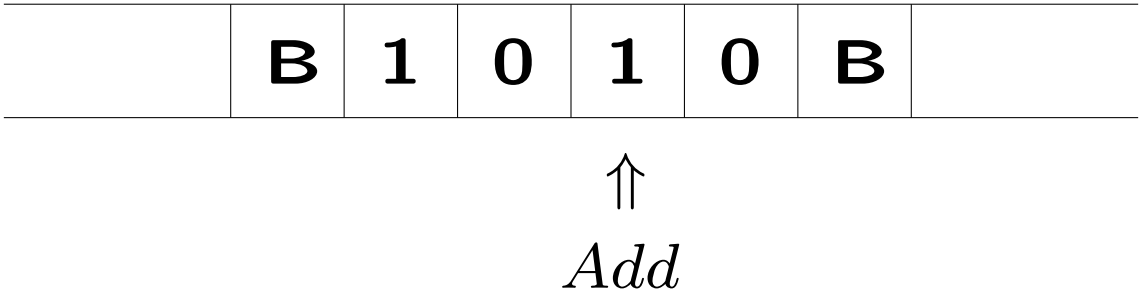
| | B | 1 | 0 | 1 | **0** | B | |
|---|---|---|---|---|---|---|---|

$\Leftarrow$

$Add$

$$\delta = \begin{array}{c|c|c|c} q\backslash a & 0 & 1 & B \\ \hline Add & (1, \text{右}, Right) & (0, \text{左}, Add) & (1, \text{右}, Right) \\ \hline Right & (0, \text{右}, Right) & (1, \text{右}, Right) & (B, \text{左}, Halt) \end{array}$$

# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 1を足すチューリング機械の動き

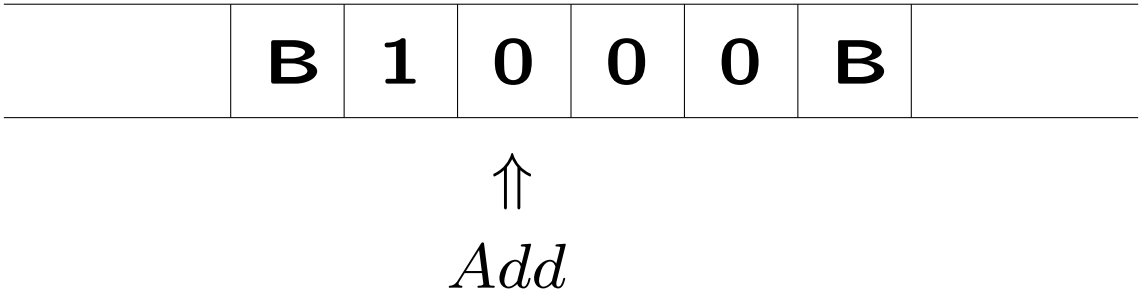
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | B | 1 | 0 | **0** | 0 | B | |

$\Leftarrow$

$Add$

$$\delta = \begin{array}{c|c|c|c} q\backslash a & 0 & 1 & \mathsf{B} \\ \hline Add & (1, \text{右}, Right) & (0, \text{左}, Add) & (1, \text{右}, Right) \\ \hline Right & (0, \text{右}, Right) & (1, \text{右}, Right) & (B, \text{左}, Halt) \end{array}$$

# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 1を足すチューリング機械の動き

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | B | 1 | **1** | 0 | 0 | B | |

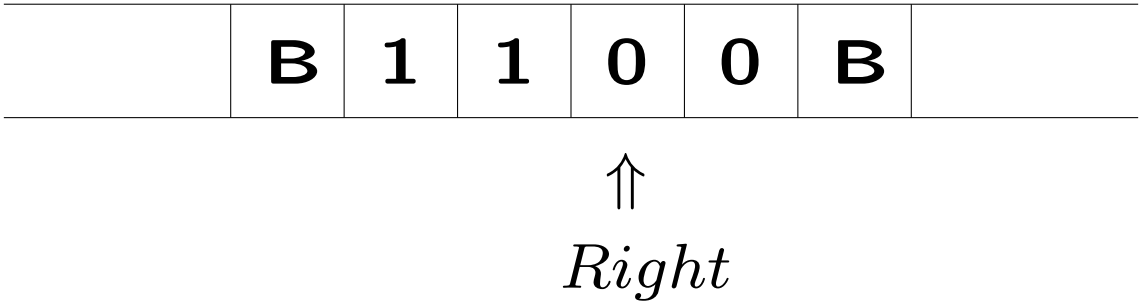
$\Rightarrow$

$Right$

$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

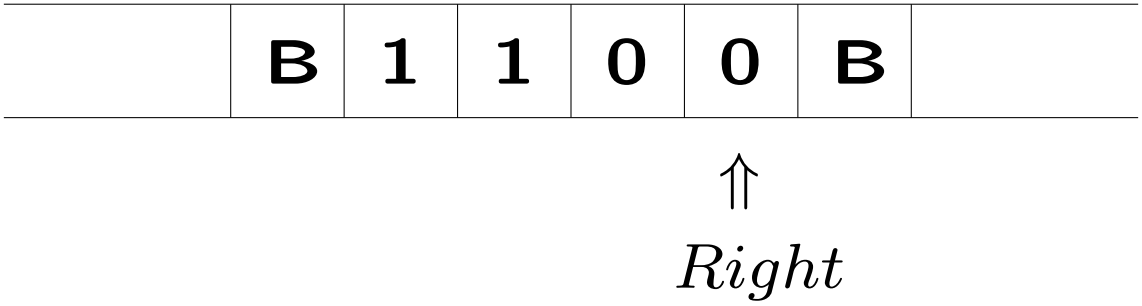
# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | (1, 右, $Right$) | (0, 左, $Add$) | (1, 右, $Right$) |
| $Right$ | (0, 右, $Right$) | (1, 右, $Right$) | ($B$, 左, $Halt$) |

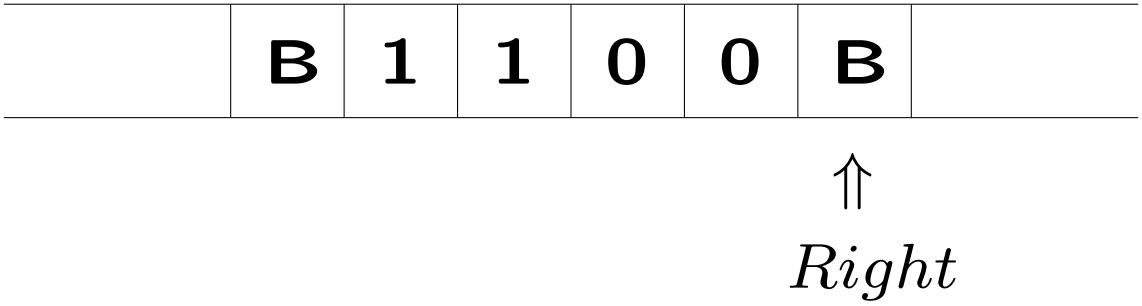
# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 1を足すチューリング機械の動き




$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 1を足すチューリング機械の動き


$\delta =$

| $q \backslash a$ | 0 | 1 | B |
|---|---|---|---|
| $Add$ | $(1, 右, Right)$ | $(0, 左, Add)$ | $(1, 右, Right)$ |
| $Right$ | $(0, 右, Right)$ | $(1, 右, Right)$ | $(B, 左, Halt)$ |

# 足し算を行う機械

２進数で表現された二つの数字を足す

初期状態は下記の通り $\overline{110}^2 = 6,\ \overline{10}^2 = 2$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| B | 1 | 1 | 0 | M | 1 | 0 | B |

⇑

終了時はこうなる $\overline{1000}^2 = 8$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | M | B | B | B |

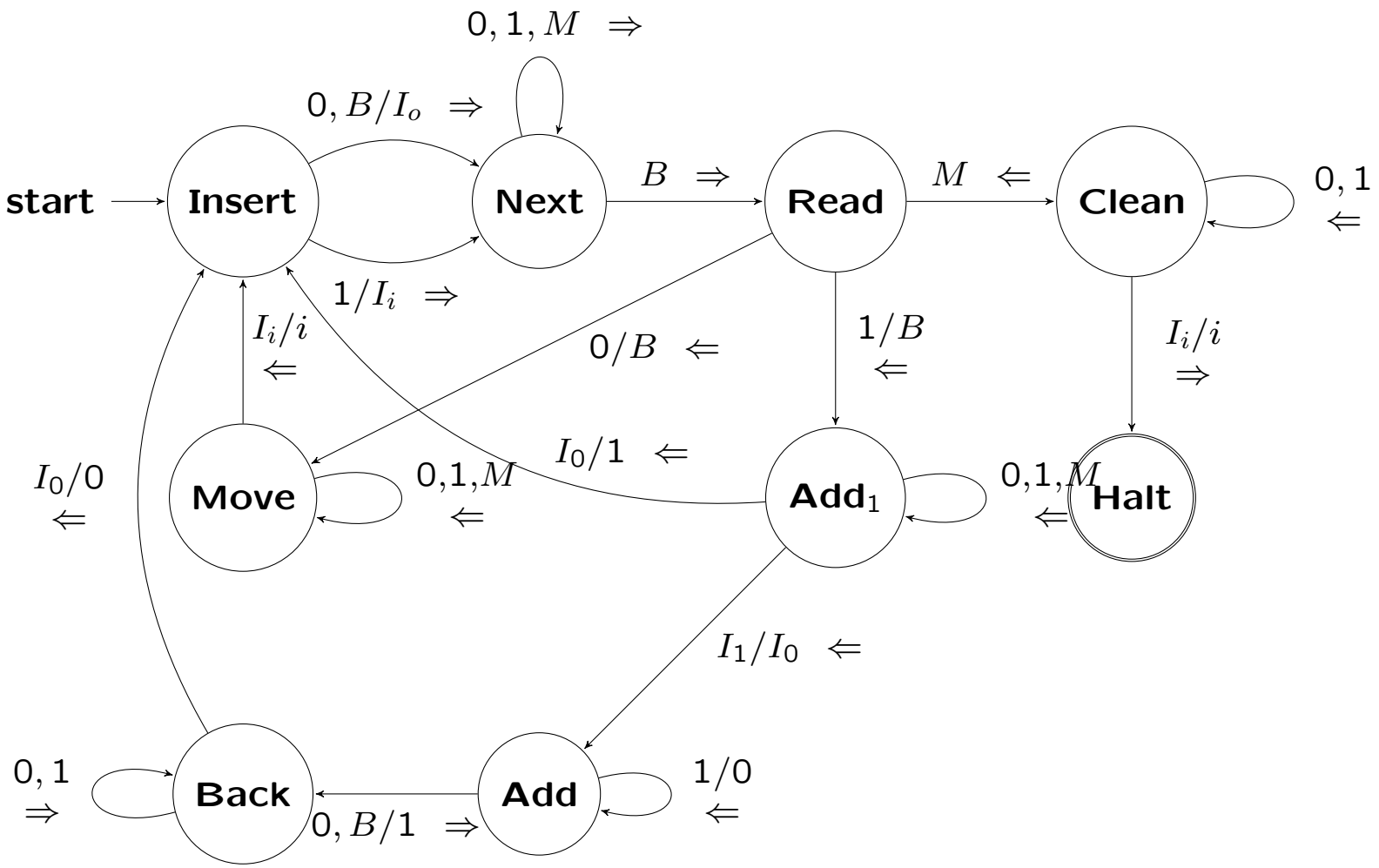
⇑

# 足し算を行う機械の定義

$\Sigma = \{B, 0, 1, M, I_0, I_1\}$
$K = \{Insert, Next, Read, Add_1, Add, Back, Move, Clean, Halt\}$
$q_0 = Insert, \quad H = \{Halt\}$

| $q\backslash a$ | B | 0 | 1 | M | $I_0$ | $I_1$ |
|---|---|---|---|---|---|---|
| $I$ | $(I_0, 右, N)$ | $(I_0, 右, N)$ | $(I_1, 右, N)$ | | | |
| $N$ | $(a, 左, R)$ | $(a, 右, N)$ | $(a, 右, N)$ | $(a, 右, N)$ | | |
| $R$ | | $(B, 左, M)$ | $(B, 左, A_1)$ | $(a, 左, C)$ | | |
| $A_1$ | | $(a, 左, A_1)$ | $(a, 左, A_1)$ | $(a, 左, A_1)$ | $(1, 左, I)$ | $(I_0, 左, A)$ |
| $A$ | $(1, 右, B)$ | $(1, 右, B)$ | $(0, 左, A)$ | | | |
| $B$ | | $(a, 右, B)$ | $(a, 右, B)$ | | $(0, 左, I)$ | |
| $M$ | | $(a, 左, M)$ | $(a, 左, M)$ | $(a, 左, M)$ | $(0, 右, I)$ | $(1, 右, I)$ |
| $C$ | $(a, 右, H)$ | $(a, 左, C)$ | $(a, 左, C)$ | | $(0, 右, H)$ | $(1, 右, H)$ |

# 足し算の状態遷移図


0, 1, M ⇒
0, B/I_o ⇒
start
Insert
Next
B ⇒
Read
M ⇐
Clean
0, 1 ⇐
1/I_i ⇒
I_i/i ⇐
0/B ⇐
1/B ⇐
I_i/i ⇒
I_0/1 ⇐
I_0/0 ⇐
Move
0,1,M ⇐
Add_1
0,1,M ⇐
Halt
I_1/I_0 ⇐
0, 1 ⇒
Back
0, B/1 ⇒
Add
1/0 ⇐

# 万能チューリング機械

今までの例で分かるように、人間が行うあらゆる計算はチューリング機械を使ってできる。

チューリング機械を動かすのも計算の一種だ。

では、チューリング機械の定義と入力テープを与えられて、それを実行するチューリング機械はあるはずだ。

それを万能チューリング機械という。

$\Rightarrow$ 汎用計算機の原型でもある

# ２文字チューリング機械

万能チューリング機械は自身の状態数もアルファベットも固定である

実行する機械の状態はテープに書くので、複数の記号で書けばいい

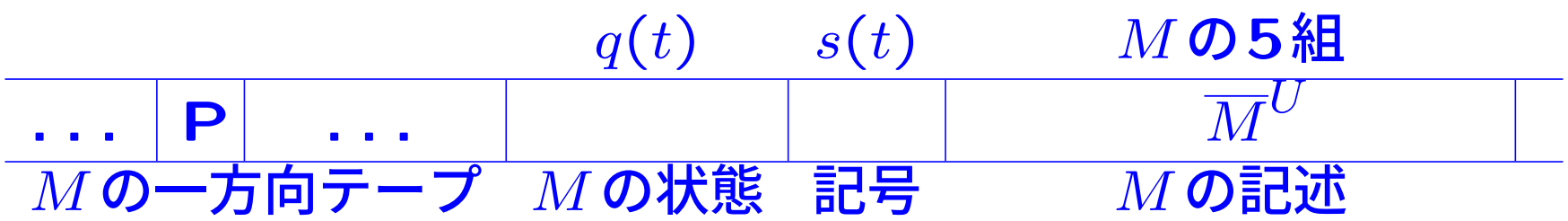
以下の定理によって、実行する機械の記号を２つに制限できる
テープも一方向に制限できる

**定理 1** 任意の機械 $M$ に対し、アルファベット $\Sigma_2 = \{0, 1, \mathsf{B}\}$ だけで動作し、$M$ と同じ計算をする機械 $M'$ が構築できる。さらに、$M'$ のテープが一方向にしか伸びないようにもできる。

# 万能チューリング機械の構築

簡単な演算によって、万能チューリング機械$U$を定義する

動作中にテープは以下のような形をしている




- $q(t)$ は2進表記された$M$の現在の状態を保持する
- $s(t)$ はPを挿入した位置にあった記号を保持する

# 万能チューリング機械の動き

$U$ の動作は以下の通り

1. $(q(t), s(t))$ に合う規則を５組の中に探す。なければ終了
2. その５組に応じて $q(t+1)$ を書き込む
3. 方向を覚えながら、Ｐを新しい記号に書き変える
4. 右か左に一個移動し、そこの記号を読み、Ｐを書き込む
5. 記号を $s(t+1)$ として書き込み、１から繰り返す

# Church-Turing 提唱

チューリング機械によって計算できる関数を計算可能関数という。

厳密には「チューリング計算可能」というべきだろうが、以下の提唱によってその区別が要らない。

提唱 2 実効的な手続きによって計算できる全ての関数はチューリング機械によって計算できる。

この提唱によれば、万能チューリング機械が作れる全ての実効的な枠組みはチューリング機械と全く同じ表現力を持つ。
その性質をチューリング完全性という。

# その他の計算の枠組み

チューリング完全な枠組みが多く存在するが、以下の二つはチューリング機械と同じ時期(1935年頃)に開発され、
Church-Turing提唱の根拠にされた

- Haskell Curry と Alonzo Church のラムダ計算
- Stephen Kleene の帰納的関数

どちらも1936年頃にチューリング機械との等価性が証明された

# ラムダ計算

構文は３種類だけ

$$\begin{array}{rcll} M & ::= & x & \text{変数} \\ & | & \lambda x.M_1 & \lambda \text{抽象} \\ & | & M_1(M_2) & \text{関数適用} \end{array}$$

計算規則は一つだけ

$$(\lambda x.M_1)(M_2) = M_1[x \leftarrow M_2]$$

$M_1$ の中の $x$ の自由な出現を $M_2$ に置き換える

# 帰納的関数

零関数、後者関数、合成と原始帰納法から原始的関数を定義する

定義 2 $g : N^n \to N$ と $h : N^{n+2} \to n$ が原始的関数ならば、

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, y+1) &= h(\vec{x}, y, f(\vec{x}, y)) \end{aligned}$$

で定義される関数 $f : N^{n+1} \to N$ も原始的関数である

原始的関数は必ず停止するが、最小解関数を加えることでチューリング機械と同じ表現力が得られる

$$\mu_p(\vec{x}) = \begin{cases} p(\vec{x}, y) = 0 \text{となるような} y \text{が存在すれば、その最小値} \\ \text{解がなければ、結果は未定義} \end{cases}$$

# 停止問題

任意のチューリング機械の定義と入力テープについて、その機械がその入力に対して停止状態に到達できるかどうか

この問題を判定するするチューリング機械は存在するのか

もしも存在すれば、止まらないプログラムが未然に防げるので、とても役に立つ

# 停止問題は判定不能

**定理 3 任意の Turing 機械の停止問題を判定する実効的な手続きは存在しない**

証明　もしも、停止を判定する Turing 機械 $H$ が存在すれば、それを次のように改造する

まず、引数として $\overline{M}$ だけをもらい、$M$ が $\overline{M}$ に適用されたとき止まるかどうかを判定する機械 $H'$ を作る。$H'(\overline{M}) = H(\overline{M}, \overline{M})$

次に、$H'$ が NO を返したときには止まるが、YES を返したときには無限ループに入る機械 $H^*$ を作る

最後に、$H^*(\overline{H^*})$ が止まるかどうかを判定する、すなわち $H(\overline{H^*}, \overline{H^*}) = H'(\overline{H^*})$ を実行する。もしも YES を返せば、$H^*(\overline{H^*})$ が止まらなくなるので、矛盾になる。NO を返せば、今度は $H^*(\overline{H^*})$ が止まってしまうので、それも矛盾 □

# 印字問題

**定理 4 任意の機械が実行中に特定の記号 $S_0$ をテープに書くことがあるかどうを判定する実効的な手続きは存在しない**

**証明** $S_0$ をアルファベットに含まない機械 $M$ を次のように変更する。各停止状態を変更し、停止する前に $S_0$ を書くようにする。

そうすると、新しい機械の印字問題は $M$ の停止問題に当るので、判定が不可能になる。 □

# 判定が不可能な問題

停止問題に還元することで、様々な問題が判定不可能であることを証明できる

**定理 5** 任意の機械が空テープに適用されたときに停止するかどうかを判定する実効的な手続きは存在しない

**定理 6** 任意の機械がテープの中身に依存せずに停止するかどうかを判定する実効的な手続きは存在しない

# 論理学への影響

チューリング機械は元々計算機としてではなく、論理学の道具として作られた

論理学の大きな目的の一つは、論法を計算として見なすことである

計算によって解けない問題の存在を示すことで、論理学の限界が示されたのである

**Turing**の論文は**Hilbert**の数学を機械化する計画に終止符を打ったことで有名だ

# 計算機科学への影響

Church-Turing提唱と停止問題の判定不可能性が様々なところに影響する

- 一定以上の表現力を持ったプログラミンッグ言語はチューリング完全なので、プログラミング言語の表現力の比較は不毛である
- プログラムに必ず停止して欲しい場面はあるが、それをプログラミング言語が自動的に行おうとすると表現力が弱くなる
- プログラムのほとんどの性質を停止判定に還元できるので、プログラムの正しさの完全な自動証明は不可能である

# 判定不能な問題を超えて

しかし、それぞれの問題について部分的な解はある

- 停止性を自動的に調べるツールは多くある
- それをうまく使うと、他の性質を証明するのにも使える
- 人間のひらめきで証明できるときもある

# 参考文献

**A. M. Turing, On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Ser. 2, Vol. 42, 1937年**

**高橋正子「計算論」（近代科学社）1991年**

**Neil D. Jones, Computability and complexity from a programming perspective, MIT Press, 1995年**

**J. Garrigue 「計算と論理」（講義ノート）2023年**
http://www.math.nagoya-u.ac.jp/~garrigue/lecture/2023_tenbo/index.html