

チューリングマシンにも、非決定性チューリングマシンと決定性チューリングマシンがある。非決定性チューリングマシンは、次の動作にいくつかの選択肢があるチューリングマシンである。任意の非決定性チューリングマシンに対して、等価な決定性チューリングマシンを構成できることが知られている。以下では決定性チューリングマシンのみを考える。

例: 足し算をするチューリングマシン（初期状態は q_0 ）

	B	1
q_0	$(B, \rightarrow, q_0)$	$(1, \rightarrow, q_1)$
q_1	$(1, \rightarrow, q_2)$	$(1, \rightarrow, q_1)$
q_2	$(B, \leftarrow, q_3)$	$(1, \rightarrow, q_2)$
q_3	$(B, \leftarrow, q_3)$	$(B, \leftarrow, q_4)$
q_4		$(1, \rightarrow, \text{stop})$

読み方の例: 例えば、行 q_2 と列 B の交点には、 $(B, \leftarrow, q_3)$ がある、これは「テープのヘッドの位置に B を書き込んで、ヘッドを左に移動し、新しい状態を q_3 とする。」という動作を表す。

問 5.1.1 次のような初期値のテープ（ヘッドの位置は下線部）で、上の足し算をするチューリングマシンを実行せよ。

... $B B \underline{B} 1 1 B 1 1 1 1 B B B \dots$

例: 掛け算をするチューリングマシン（初期状態は q_0 ）

	B	1
q_0	$(B, \rightarrow, q_0)$	$(B, \rightarrow, q_1)$
q_1	$(B, \rightarrow, q_2)$	$(1, \rightarrow, q_1)$
q_2		$(B, \rightarrow, q_3)$
q_3	$(B, \rightarrow, q_4)$	$(1, \rightarrow, q_3)$
q_4	$(1, \leftarrow, q_5)$	$(1, \rightarrow, q_4)$
q_5	$(B, \leftarrow, q_6)$	$(1, \leftarrow, q_5)$
q_6	$(1, \leftarrow, q_{10})$	$(1, \leftarrow, q_7)$
q_7	$(1, \rightarrow, q_2)$	$(1, \leftarrow, q_7)$
q_8	$(B, \rightarrow, q_0)$	$(1, \leftarrow, q_8)$
q_9	$(B, \leftarrow, q_{11})$	$(1, \leftarrow, q_8)$
q_{10}	$(B, \leftarrow, q_9)$	$(1, \leftarrow, q_{10})$
q_{11}	$(B, \leftarrow, q_{12})$	$(B, \leftarrow, q_{11})$
q_{12}	$(B, \rightarrow, q_{12})$	$(B, \rightarrow, q_{13})$
q_{13}	$(B, \rightarrow, q_{14})$	$(B, \rightarrow, q_{13})$
q_{14}	$(B, \rightarrow, q_{15})$	$(1, \rightarrow, q_{14})$
q_{15}	$(B, \leftarrow, \text{stop})$	

問 5.1.2 （結構大変） 次のような初期値のテープ（ヘッドの位置は下線部）で、上の掛け算をするチューリングマシンを実行せよ。

... $B B \underline{B} 1 1 1 B 1 1 B B B \dots$

5.2 チャーチの提唱

次の事実が知られている。

- 「チューリングマシンで表現できる計算」と「 (μ 再帰関数) で表現できる計算」と「 (λ -calculus) で表現できる計算」はすべて等価である。(帰納的関数、ラムダ計算はいずれも、チューリングマシンと同じところに考えられた「計算」の数学的なモデルである。)
- チューリングマシン (または帰納的関数、またはラムダ計算) で表現できない「計算」を表現できる形式的な手法は、これまでに誰も見つけていない。

 (Church's thesis) あるいは チャーチ=チューリングの提唱 (Church-Turing thesis) とは、次のような主張である。

- 「計算できる」とは、すなわち「チューリングマシン (または帰納的関数、またはラムダ計算) で表現できる」ということである
- 形式的な手法でどのように「計算」を定義しても、チューリングマシン (または...) の能力を超えられない

形式的とは「勘」とか「霊のお告げ」など他人に説明できない方法ではなく、説明可能で誰がやっても同じになる、というような意味である。この主張は証明できるようなことではないので「定理」とは呼ばず「提唱」あるいは「テーゼ」と呼ばれる。

チューリングマシン (または...) と等価な能力をもつ計算体系を (Turing complete) である、という。ほとんどのプログラミング言語は、チューリングマシン (または...) をエミュレートできるので、チューリング完全である。

5.3 チューリングマシンの停止性問題

チューリングマシンは、ヘッドが当たっているテープ上の記号と有限状態部の状態から次の動作を決める、たかが有限個の規則の集まりである。だから、チューリングマシンを記号列としてエンコードすることができる。エンコードしたチューリングマシンと入力記号列の組を受け取って、その動作をエミュレートする (universal Turing machine) を構成することもできる。(コンピューターのソフトウェアで仮想コンピューターを実現するのと同じようなものである。)

すると、次のような問題を考えることができる。

停止性問題 (halting problem) チューリングマシン (をエンコードした記号列) M と入力記号列 α の組を受け取って、チューリングマシン M が入力記号列 α に対して、停止するならば、 $H(M, \alpha)$ が "Yes" を、停止しないならば $H(M, \alpha)$ が "No" を返すようなチューリングマシン H を定義することができるか？

これがもしできるならば、（コンピューターのプログラムはチューリングマシンと等価なので）プログラムが入力に対して停止するかどうか、を判定するプログラムが作成できることになる。（かなりの割合のバグが自動的に発見できる!!）

残念ながら、そのようなチューリングマシンを定義することは無理であることが証明されている。（チューリングマシンの停止性の決定不能性）

証明: もしも停止性を判定できるチューリングマシン H が存在したと仮定する。 H を使って H' を次のように定義することができる。

$$\begin{cases} H(M, M) = \text{"Yes"} \text{ ならば } H'(M) \text{ は停止しない。} \\ H(M, M) = \text{"No"} \text{ ならば } H'(M) \text{ は停止する。} \end{cases}$$

$H'(H')$ が停止するかどうかを考える。停止するとすると、それは、 $H(H', H')$ が "No" であることになる。しかし、これは $H'(H')$ が停止しないことを意味するので矛盾である。同様に $H'(H')$ が停止しないとすると、 $H(H', H')$ が "Yes" であることになる。 $H'(H')$ が停止することになり、やはり矛盾である。よって、停止性を判定できるチューリングマシン H が存在するという仮定が間違っている。（証明終わり）

チューリングマシンの停止性は、代表的な _____（Yes/No を決定する計算方法が存在しないことが証明されている問題）である。多くの問題の決定不能性は、チューリングマシンの停止性の決定不能性に帰着して（...が計算可能ならば、チューリングマシンの停止性も計算できる、という背理法で）証明される。