Turing Machine の原理

浅川伸一

2009 年 11 月 25 日

Alan Turing (1912.6.23-1954.6.7): イギリスの数学者。数学と数理論理学のほか、計算機の基礎理論と応用の広範な分野で数々の独創的な研究を行った。いわゆる '情報科学' の最初の研究者の一人。大学在学中の 1935 年に数理論理学の研究を開始した。1936/37 年に Turing machine の概念を導入した論文 "On computable numbers, with an application to the Entscheidungsproblem" を発表。数値計算、数理論理学、人工知能の可能性 ('Turing test', チェスのプログラムなど)、形態発生に関する研究を行った。1954 年毒物の服用により死亡。アメリカ最大のソフトウエア学会 ACM は、1966 年から毎年 1〜2 名のすぐれた業績を挙げた研究者に対して、最高の栄誉として Turing 賞を贈っている。(岩波情報科学辞典より抜粋)

Turing machine とは、計算可能性を証明するために Turing が考えた仮想的な機械である。ここでは、Turing machine の動作を通してコンピュータの動作を学ぶ。ワープロにしても、お絵書きソフトにしても内部で行っていることは計算である。現在使われているコンピュータは原理的には、ここで示す Turing machine と同じものであることを念頭において以下の文章を読んで欲しい。

1 Puzzle

次の図に示すような 1 と 0 とが書かれたテープがある。

	1	1	0	1	1	1	
	△						

さらに、ポケットには一枚のメモと、5 枚の命令カードがある。ゲーム開始時、メモには '1' と書かれている。5 枚の命令カードの内容は以下のとおり。

命令 1

q	a	a'	q'
1	1	B	1

命令 2

q	a	a'	q'
1	B	R	2

命令 3

q	a	a'	q'
2	1	R	2

命令 4

q	a	a'	q'
2	B	R	3

命令 5

q	a	a'	q'
3	1	B	3

以下に示すルールに従ってこのパズルを解いてみよう。

1. メモの内容を読む
2. 図中の△の位置に書かれているテープの内容を読み出す。ただし何も書かれていない場合は 'B' と読み変える。
3. 命令カードの中から、メモの内容が q の数字と一致し、かつ、読み出したテープの内容が a と一致する命令カードを探す。
4. 探し出した命令カードの a' に

   L と書かれていたら、△の位置を一つ左に移動する。

   R と書かれていたら、△の位置を一つ右に移動する。

   その他の記号なら、テープの△の位置にその記号を書き込む。すでに何か書かれていたら、その記号を消してから、書き込む。
5. メモの内容を q' に書き換える
6. 以上の操作を命令カードに書かれている操作ができる限り繰り返す。

以上の操作を終えたときにテープに書かれている内容が puzzle の答である。
Tape head の動作は homunculus analogy で考えてもよいかも知れない。

1.1 puzzle を解く

では実際にやってみよう。step 1 を参考に step 2 から step 6 までのテープの内容を埋めよ。

Step 1: メモの状態 [1](命令カード 1 に合致)

	1	1	0	1	1	1	
	△						

Step 2: メモの状態 [ ]( )

	△						

Step 3:メモの状態 [ ]( )

		△					

Step 4:メモの状態 [ ]( )

			△				

Step 5:メモの状態 [ ]( )

				△			

Step 6:メモの状態 [ ]( )

				△			

1.2 tape の説明

テープに書かれている n+1 個の '1' 連続は数字を表している。例えば '3' を表現したい場合には、| | 0 | 1 | 1 | 1 | 1 | 0 | | となる。数字の 1 と 2 とを表現する場合には、次のように間に空白 (または 0) を入れて | | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | | となる。結果の方は、テープに並んだ '1' の個数で表現される。例えば次のテープは数字の 5 を表現してる | | 0 | 1 | ♡ | 1 | ↑ | 1 | 1 | × | 1 | W | | 最初にテープに書かれている数字 (入力) は $n+1$ 個の 1 で数字を表現しているにも関わらず。結果は '1' の 数そのものが意味を持つ ことに注意。

1.3 puzzle のカラクリ

実は上記の puzzule を $1+2=3$ という計算を実行していた。すなわち、

1. 命令カード 1 は 2 つ並んだ数字のうち、最初の数字をあらわす '1' の連続から先頭の '1' を削る働きをする
2. 命令カード 2,3,4 は△を移動させて 2 つめの数字の先頭に△を移動させる
3. 命令カード 5 は、 2 つめの数字を表す '1' の連続から先頭の '1' を削る

結果としてテープには 2 つの数の和だけ '1' が残っていることになる。なぜなら、もともとテープには 2 つの数 + 2 個の '1' が書かれているのだから。

2 Turing machine simulator

第 1 節の命令カード (動作表) は足し算用のものである。このことを確かめるために、Turing machine の動作を示すシミュレーターを用意した。使用方法は、まずエディタで Turing machine を動かすための動作表を作成する。以下のように命令カードに書かれていた内容を 1 カード 1 行に q,a,a',q' の順番で入力する。

```
1 1 B 1
1 B R 2
2 1 R 2
2 B R 3
3 1 B 3
```

入力し終わったら、例えば add.turing のような名前で保存しておく。シミュレーターを実行するには、turing　< 動作表ファイル >　< 入力テープ表現 > という形で起動する。このカード使って $1+2$ を実行した結果を以下に示す。

```
(0001)..........11.111.....................................................
                ^ Rule 0(1 1 B 1) matched.
(0002)...........1.111.....................................................
                ^ Rule 1(1 B R 2) matched.
(0003)...........1.111.....................................................
                 ^ Rule 2(2 1 R 2) matched.
(0004)...........1.111.....................................................
                  ^ Rule 3(2 B R 3) matched.
(0005)...........1.111.....................................................
                   ^ Rule 4(3 1 B 3) matched.
(0006)...........1..11.....................................................
```

```
                    ^ HALT: No rule matched.
 Result = 3
```

入力されたテープの ‘0’ の部分は ‘.’ に置き換えられていることに注意。Turing machine は左右に無限に長いテープを仮定しているので、入力されたテープはテープの中ほどから始まっている。また、△は ‘^’ と表現されている。

問題 2.1 $1+2$ 以外の計算を実行させて正しく計算されることを確認しなさい。例えば $3+4$ を実行させるためには、`turing add.turing 1111011111` という形で起動する。テープが十分長ければ、すべての自然数の和が計算できることになる。すなわち、Turing machine の動作は、テープによって決まるのではなく、与えられたルールよって決まるということを実感せよ。

問題 2.2 用いたルールは、1行目に書いてある動作を除いて順番は関係ない。動作表ファイル add.turing の行を適当に入れ替えて実行せよ。

問題 2.3 第 1 節の命令カードは、2つの数字のあいだにある ‘0’ が一つしか存在しないことを仮定している。`turing add.turing 1100111` という入力を与えると途中で止まってしまう。このことを確かめよ。

問題 2.4 2 つの数字の間に複数個の ‘0’ が入っても正しく計算されるように、add.turing を書き換えよ。

問題 2.5 $x \geq y$ として $x-y$ を計算する Turing machine の動作表の例を挙げる。

```
1 1 R 1 # 2 つの数字の間の 0 を探す
1 B L 2
2 1 B 3 # 最初の数字から ‘1’ を消す
3 B R 3 # 2 番目の数字を探す
3 1 B 4 # 2 番目の数字から ‘1’ を消す
4 B R 5 # となりが 0 なら終り
5 1 L 2 # となりが 1 なら左へ行って ‘1’ を探す
2 B L 2
```

この動作表を sub.turing という名前で file に保存し動作を確認せよ。

問題 2.6 x, y が与えられたとき、絶対値 $|x-y|$ を計算する Turing machine を設計する。2数の差を求めるときには、2つの数字を表す ‘1’ から交互に ‘1’ を消していき、どちらか一方が無くなるまで繰り返せばよい。この動作をする Turing machine が正しく動作するように次の動作表の空欄を埋めよ。

```
1 1 R 1
1 B L 2
2 1 B 3
2 B L 2
3 B L 4
4 B R [ ]
4 1 R 5
5 1 B 6
5 B[ ] 5
6 B R 7
7 1 L 2
8 B R 8
8 1 B 9
```

問題 2.7 x, y が与えられたとき、$x \times y$ を計算する Turing machine を設計する。具体的には、$x \times y$ は x を y 回足せばよい。y を表す '1' を消すたびに x 個の '1' をコピーすることにする。以下の動作表を入力して実行せよ。3×4 の計算結果は計何ステップになるか？

```
1 1 A 1
1 A R 2
2 1 R 2
2 B R 3
3 1 B 3
3 B R 4
4 B L 14
4 1 C 6
5 1 C 6
5 B L 7
6 C R 5
7 C L 7
7 B L 7
7 1 B 8
8 B R 8
8 C R 9
9 C R 9
9 1 L 13
9 B L 13
10 1 R 10
10 A R 10
10 B 1 11
10 C R 10
11 1 L 11
11 A C 12
11 C L 11
12 C L 13
13 B L 7
13 C A 10
14 B L 14
14 1 B 15
15 B L 14
```

3 万能 Turing machine と計算可能性

ある関数が計算可能であるか否かは、その関数の返す値を計算する Turing machine が作れるかどうかによってきまる。このことを "Turing machine に基づく計算可能性" という。

任意の Turing machine と同じ働きをする能力を持つ特定の Turing machine が存在する。この機械 M_U を万能 Turing machine[1] という。すなわち万能 Turing machine とは、任意の Turing machine を M、i を M のゲーデル数とし、M に入力 x を与え M_U に i と x の組 (i, x) を与えて動かすと、M が出力 y を出し

[1]万能 Turing machine が構成可能なことを証明するためには、Gödel 数の議論が必要であるが、ここでは省略する。

て停止するなら M_U も同じ出力 y を出して停止し、M が停止しないなら M_U も停止しない、というマシンである。

万能 Turing machine の概念はいろいろな問題が決定可能であることの証明の基礎になった。

実習に用いた Turing machine simulator は、あらかじめルールを記述しておかなければならなかった。万能チューリングマシンではルールを表す記号列と入力データの組を与えれば、それぞれを勝手に読み込んで計算してくれる。これは計算の内容に応じたプログラムを書くことによって、その計算を行う現在のコンピュータそのものである。

コンピュータで行うことのできる計算はすべて Turing machine でも計算できる。逆に Turing machine で計算不可能な問題は、どんなに高性能なコンピュータを用いても計算できない。この意味で Turing machine は、現在のコンピュータ科学の最も重要な基礎理論の一つである。

4 Turing machine の停止問題

Turing machine の動作表の作り方によっては、実行を停止せずに無限に計算し続けることが起こりうる。ある入力データに対しては計算が停止するけれど、他のデータに対しては無限に計算を続けて停止しない動作表も作ることができる。

Turing machine のふるまいは、与える動作表によって完全に決まるので、動作表と入力データを与えれば、停止か否かを判断できるはずである。そこで以下のような前提を考えてみよう。

1. Turing machine M_A に入力データ x を与えたとき、M_A が停止するか否かを判定するために、別の Turing machine M_C を作ったと仮定する。
2. M_C は停止判定用の Turing machine で、M_A の動作表とその入力 x を入力して、M_A が停止するならば 0, 停止しない場合には 1 を出力して停止するものとする。
3. M_C と同じような動作をする M_D を作る。M_D は M_A の動作表とその入力 x を入力して、M_A が停止するときには停止せず、M_A が停止しないなら停止するようにする。

さて、この M_D に M_D 自身のデータを入力したらどうなるだろうか？

- M_D が停止しないと仮定すると、停止しないというデータを受け取った M_D は停止する。
- M_D が停止すると仮定すると、停止するというデータを受け取った M_D は停止しない。

これは矛盾である。この例は “Turing machine の停止問題” という有名な問題であり、次のように結論することができる。『任意の Turing machine と任意のデータを与えたとき、そのマシンが停止するかしないかを必ず判断できるような判定 Turing machine は存在しない。』これは、背理法あるいは帰謬法と呼ばれている証明方法の一つである。Turing machine の停止問題は決定不能問題のうちで最も基本的なものである[2]。

[2]その他のほとんどの決定不能問題を証明する場合、“もしその問題を解くアルゴリズムが存在すれば Turing machine の停止問題を解くアルゴリズムも存在する” ということを直接的あるいは間接的に示すという手段が用いられる

数学には Gödel の定理というのがあって、『整数論が無矛盾であることを、その体系の中では証明できない』というものである。論理学では Russel の集合論のパラドックスというのがあって、『すべての集合をその要素に持つ集合 (すなわち集合の集合) を考えたとき、その集合は、自分自身をその要素として含むだろうか？』という問題である。コンピュータでも計算できず、数学でも、数理論理学でも証明できない問題が存在する。それらはすべて自己参照した場合に矛盾が生ずるというところが特徴的である。

5 Turing test

最後に人工知能に関する Turing test を紹介しておく。Turing の定義は以下の通り、“1 つの部屋に計算機、別の部屋に人をおき、どちらの部屋に計算機が入っているか知らない人が、外から通信線を通してそれぞれの部屋に種々の質問をしたとする。それらの質問に対する応答がいずれも適切であって、計算機によるものなのか、人間によるものか判断しにくいときに、計算機は人間のように考えている。”

あらゆるコンピュータは Turing machine と原理的に同じであると述べた。もし、近い将来人工知能が完成して、コンピュータが我々の心を理解する日が来たとする。このとき、人間の心は計算可能であることが証明されたことになるのだが、はたして人間 (自分) が人工知能 (自分自身) を作ろうとする試みは、成功するのであろうか？

ちなみに、2 つの Turing machine[3]が、すべての入力に対して同じ答を出すか否かという問題は決定不能であることを、先に挙げた背理法を使って証明できるのだが…。

[3]Turing test における人間の応答と機械の応答？あるいは 2 人の人間の個性？