2018 年の決定不能問題ギャラリーを振り返る

y. (@waidotto)

2019 年 3 月 30 日 @ 第 3 回関東すうがく徒のつどい 2 日目

http://iso.2022.jp/math/tsudoi/kanto3/slide.pdf

自己紹介

- y. (@waidotto)
- 決定不能問題と呼ばれるものが好き
- 特に，数学の色々な分野に自然に現れる具体的な決定問題の決定可能性・不能性に興味がある
- 2018 年の元旦から「決定不能問題ギャラリー」というサイトを作って記事を公開している

決定不能問題の何が面白いか: 不可能性・非存在の証明の一例として

単純に，ある事柄が

- 何を
- どうやっても
- 未来永劫
- 絶対に
- できない！

ことが，

- 数学的に証明できる

という現象が面白い．

(cf. 角の三等分，Gödel の不完全性定理，etc.)

決定不能問題の何が面白いか: 不可能性・非存在の証明の一例として

単純に，ある事柄が

- 何を
- どうやっても
- 未来永劫
- 絶対に
- できない！

ことが，

- 数学的に証明できる

という現象が面白い．

(cf. 角の三等分，Gödel の不完全性定理，etc.)

決定不能問題の何が面白いか: 不可能性・非存在の証明の一例として

単純に，ある事柄が

- 何を
- どうやっても
- 未来永劫
- 絶対に
- できない！

ことが，

- 数学的に証明できる

という現象が面白い.

(cf. 角の三等分，Gödel の不完全性定理，etc.)

目次

決定問題
第 1 弾 Turing 機械の定義と停止問題
第 2 弾 Post の対応問題
第 3 弾 Wang のタイル貼り問題
第 4 弾 Polyomino Problem
第 5 弾 Turing 機械の変種
第 6 弾 再帰的関数 & カウンター機械
第 7 弾 Fraction Game と一般化 Collatz 問題
第 8 弾 半 Thue 系
第 9 弾 文脈自由言語の普遍性判定問題
第 10 弾 Hilbert の第 10 問題

決定問題

決定問題とは (1/2)

決定問題 (decision problem) とは，与えられた入力に対して YES/NO で答えるタイプの問題をいう．例えば次のようなもの:

問題 (素数判定問題)

Input: 自然数 n
Question: n は素数か？

決定問題が決定可能 (decidable) であるということを「全ての入力に対して正しい答えを返すアルゴリズムが存在すること」と定義する．そうでないとき，決定不能 (undecidable) であるという．

注意

ここでは例えば「20190330 は素数か？」という問は「問題」ではなく「入力」の一つである．つまり，素数判定問題の例で言えば，

$$\{0 \mapsto \text{NO}, 1 \mapsto \text{NO}, 2 \mapsto \text{YES}, 3 \mapsto \text{YES}, 4 \mapsto \text{NO}, 5 \mapsto \text{YES}, 6 \mapsto \text{NO}, \dots\}$$

という「入力と正答の対応表」の全体を決定問題というのである．

決定問題とは (1/2)

決定問題 (decision problem) とは，与えられた入力に対して YES/NO で答えるタイプの問題をいう．例えば次のようなもの:

問題 (素数判定問題)

Input: 自然数 n
Question: n は素数か？

決定問題が決定可能 (decidable) であるということを「全ての入力に対して正しい答えを返すアルゴリズムが存在すること」と定義する．そうでないとき，決定不能 (undecidable) であるという．

注意

ここでは例えば「20190330 は素数か？」という問は「問題」ではなく「入力」の一つである．つまり，素数判定問題の例で言えば，

$$\{0 \mapsto \text{NO}, 1 \mapsto \text{NO}, 2 \mapsto \text{YES}, 3 \mapsto \text{YES}, 4 \mapsto \text{NO}, 5 \mapsto \text{YES}, 6 \mapsto \text{NO}, \dots\}$$

という「入力と正答の対応表」の全体を決定問題というのである．

決定問題とは (2/2)

素数判定問題は明らかに決定可能である.

命題

素数判定問題は決定可能である.

証明.

以下のようなアルゴリズムを考えればよい.

入力 n に対して，$n \leq 1$ なら No を返し，$n = 2$ なら Yes を返す．$n \geq 3$ なら，n を $2, 3, \ldots, n-1$ で順番に割ってみて，どれかひとつでも割り切れるものがあれば No を返し，そうでなければ Yes を返す. □

決定不能性を証明するには

いま見たように，決定問題が決定可能であることを証明するには，それを解くアルゴリズムを作ってみせればよい．では反対に，ある決定問題が決定不能であることを証明するにはどうすればよいだろうか．

決定可能性の定義より，決定問題 A が決定不能であるということは，A を解くアルゴリズムが存在しないこと，言い換えれば，いかなるアルゴリズムも A を解かない (つまり，ある入力について答えを間違えるか，または無限ループに陥ってしまう) ことに他ならない．

このことを厳密に証明するためには，アルゴリズムの数学的定義が必要になる．アルゴリズムの数学的な定義がなければ，「これで全てのアルゴリズムを調べ尽くしたか？」ということにいつまでも確信が持てないからである．

決定不能性を証明するには

いま見たように，決定問題が決定可能であることを証明するには，それを解くアルゴリズムを作ってみせればよい．では反対に，ある決定問題が決定不能であることを証明するにはどうすればよいだろうか．

決定可能性の定義より，決定問題 A が決定不能であるということは，A を解くアルゴリズムが存在しないこと，言い換えれば，いかなるアルゴリズムも A を解かない (つまり，ある入力について答えを間違えるか，または無限ループに陥ってしまう) ことに他ならない．

このことを厳密に証明するためには，アルゴリズムの数学的定義が必要になる．アルゴリズムの数学的な定義がなければ，「これで全てのアルゴリズムを調べ尽くしたか？」ということにいつまでも確信が持てないからである．

決定不能性を証明するには

いま見たように，決定問題が決定可能であることを証明するには，それを解くアルゴリズムを作ってみせればよい．では反対に，ある決定問題が決定不能であることを証明するにはどうすればよいだろうか．

決定可能性の定義より，決定問題 A が決定不能であるということは，A を解くアルゴリズムが存在しないこと，言い換えれば，いかなるアルゴリズムも A を解かない (つまり，ある入力について答えを間違えるか，または無限ループに陥ってしまう) ことに他ならない．

このことを厳密に証明するためには，アルゴリズムの数学的定義が必要になる．アルゴリズムの数学的な定義がなければ，「これで全てのアルゴリズムを調べ尽くしたか？」ということにいつまでも確信が持てないからである．

第 1 弾 **Turing** 機械の定義と停止問題

Turing 機械の定義 (の概要)

アルゴリズムの数学的な定義 (のひとつ) として Turing 機械と呼ばれるものがある．Turing 機械は

- セルが無限に連なったテープ,
- 各時点でテープ上のひとつのセルを見ているヘッド

からなる (図 1).

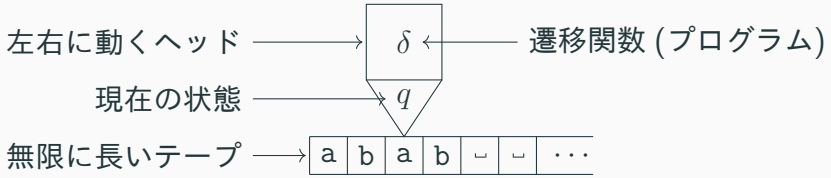


Figure 1: Turing 機械

Turing 機械による計算

Turing 機械の入力文字列 w に対する計算は次のように進む．

1. テープの左端から順番に w を一文字ずつ書き込み，残りのセルは全て空白記号 $\sqcup$ で埋める．
2. 各時点において，現在の内部状態 $q \in Q$ と，ヘッドが見ている文字 $a \in \Gamma$ から，遷移関数 $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ に従って計算を進める．例えば $\delta(q, a) = (r, b, \rightarrow)$ だったら，内部状態を q から r に変更し，ヘッドが見ている文字を a から b に変更し，ヘッドを右にひとつだけ動かす．$\leftarrow$ だったら左に動かす．
3. 状態が q_{accept} に到達したらその時点で計算を停止して $\textsc{Yes}$ を返す．
4. 状態が q_{reject} に到達したらその時点で計算を停止して $\textsc{No}$ を返す．
5. 状態が $q_{\mathsf{accept}}, q_{\mathsf{reject}}$ のどちらにも到達しなければ計算は永遠に停止しない．

Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

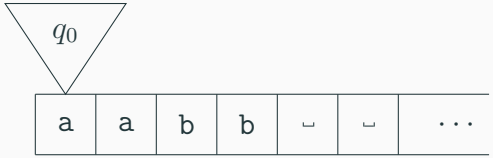
この Turing 機械を入力 aabb に対して動かすと次のようになる.

Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

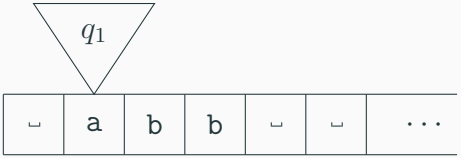


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

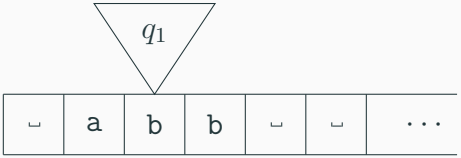


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

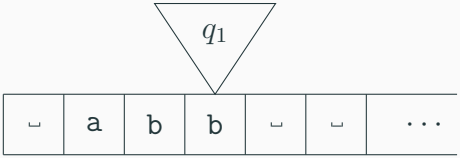


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

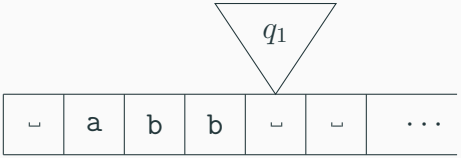


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

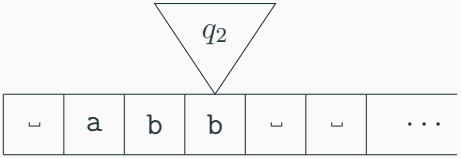


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

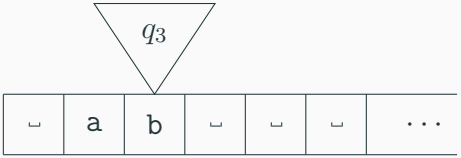


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.



Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

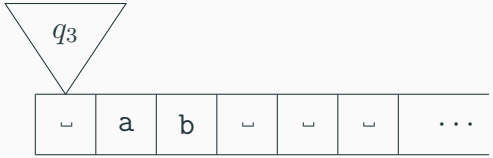
この Turing 機械を入力 aabb に対して動かすと次のようになる.

Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.



Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

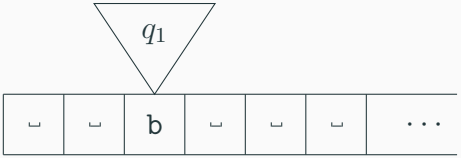
この Turing 機械を入力 aabb に対して動かすと次のようになる.

Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

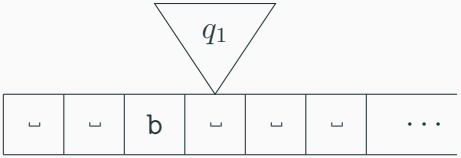


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

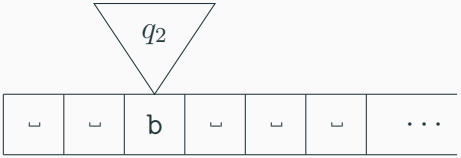


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.



Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

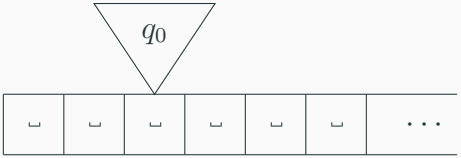
この Turing 機械を入力 aabb に対して動かすと次のようになる.

Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

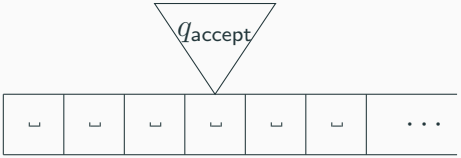


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える.

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる.

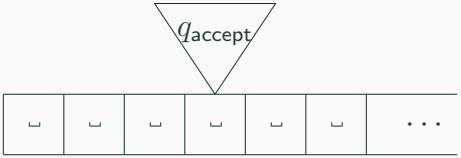


Turing 機械の例

遷移関数 δ が次の表で与えられる Turing 機械 M を考える．

	a			b			␣			
q_0	q_1	␣	$\rightarrow$	q_{reject}	b	$\rightarrow$	q_{accept}	␣	$\rightarrow$	(左端の a を消す)
q_1	q_1	a	$\rightarrow$	q_1	b	$\rightarrow$	q_2	␣	$\leftarrow$	(右端まで行く)
q_2	q_{reject}	a	$\rightarrow$	q_3	␣	$\leftarrow$	q_{reject}	␣	$\rightarrow$	(右端の b を消す)
q_3	q_3	a	$\leftarrow$	q_3	b	$\leftarrow$	q_0	␣	$\rightarrow$	(左端まで行く)

この Turing 機械を入力 aabb に対して動かすと次のようになる．



よって，M は入力 aabb に対して $M(\text{aabb}) = \text{YES}$ という答えを返す．

(入力が $\underbrace{\text{aa}\cdots\text{a}}_{n}\underbrace{\text{bb}\cdots\text{b}}_{n}$ の形になっているかを判定している．)

Church-Turing の提唱

Turing 機械は文字列に対する計算が定義されていないが，計算の対象となるものはたいてい文字列で表すことができるので問題ない．

定義 (文字列)

有限集合 Σ に対し，Σ の元からなる文字列全体の集合を Σ^* で表す．例えば，$\Sigma = \{\mathsf{a}, \mathsf{b}\}$ のとき，

$$\Sigma^* = \{\varepsilon, \mathsf{a}, \mathsf{b}, \mathsf{aa}, \mathsf{ab}, \mathsf{ba}, \mathsf{bb}, \mathsf{aaa}, \dots\}.$$

Turing 機械は入力文字列 $w \in \Sigma^*$ に対して，計算が停止するときは $\textsc{Yes}$ または $\textsc{No}$ を返す．すなわち，Σ^* の部分集合から $\{\textsc{Yes}, \textsc{No}\}$ への関数を定める (このようなものを Σ^* から $\{\textsc{Yes}, \textsc{No}\}$ への部分関数という)．Turing 機械は実はあらゆる計算を行うことができる (と信じられている)．

Church-Turing の提唱

$f\colon \Sigma^* \rightharpoonup \{\textsc{Yes}, \textsc{No}\}$ が計算可能 $\iff$ f を計算する Turing 機械が存在する

Church-Turing の提唱

Turing 機械は文字列に対する計算が定義されていないが，計算の対象となるものはたいてい文字列で表すことができるので問題ない．

定義 (文字列)

有限集合 Σ に対し，Σ の元からなる文字列全体の集合を Σ^* で表す．例えば，$\Sigma = \{\mathsf{a}, \mathsf{b}\}$ のとき，

$$\Sigma^* = \{\varepsilon, \mathsf{a}, \mathsf{b}, \mathsf{aa}, \mathsf{ab}, \mathsf{ba}, \mathsf{bb}, \mathsf{aaa}, \dots\}.$$

Turing 機械は入力文字列 $w \in \Sigma^*$ に対して，計算が停止するときは $\textsc{Yes}$ または $\textsc{No}$ を返す．すなわち，Σ^* の部分集合から $\{\textsc{Yes}, \textsc{No}\}$ への関数を定める (このようなものを Σ^* から $\{\textsc{Yes}, \textsc{No}\}$ への部分関数という)．Turing 機械は実はあらゆる計算を行うことができる (と信じられている)．

Church-Turing の提唱

$f\colon \Sigma^* \rightharpoonup \{\textsc{Yes}, \textsc{No}\}$ が計算可能 $\iff$ f を計算する Turing 機械が存在する

停止問題

与えられたプログラムが無限ループに陥るかどうかを自動的に判定するようなプログラムを書くことはできない．これが計算機科学で最も重要な決定問題である停止問題である．

問題 (停止問題; HALT)

Input: Turing 機械 M と入力文字列 w
Question: M の入力 w に対する計算 $M(w)$ は停止して $\textsc{Yes}$ を返すか？

定理

停止問題 HALT は決定不能である．

停止問題は決定不能である

証明.

背理法で示す．仮に HALT が決定可能であったとすると，HALT を解く Turing 機械 H が存在するはずである．すなわち，任意の M と w に対して

$$H(M,w) = \textsc{Yes} \iff M(w) = \textsc{Yes}$$
$$H(M,w) = \textsc{No} \iff M(w) = \textsc{No} \text{ または停止しない}$$

が成り立つ．ここで Turing 機械 D を

$$D(M) = \begin{cases} \textsc{No} & (H(M,M) = \textsc{Yes}) \\ \textsc{Yes} & (H(M,M) = \textsc{No}) \end{cases}$$

と定義すると，

$$\begin{aligned} D(D) = \textsc{Yes} &\iff H(D,D) = \textsc{Yes} \quad (H \text{ の定義より}) \\ &\iff D(D) = \textsc{No} \quad (D \text{ の定義より}) \end{aligned}$$

となり矛盾. □

第 2 弾 **Post** の対応問題

Post の対応問題

二つの文字列を上下に並べたものをドミノと呼ぶことにする．例えば

$$\left[\frac{\mathtt{ab}}{\mathtt{cde}}\right]$$

はドミノである．ドミノの列

$$\left[\frac{u_1}{v_1}\right]\left[\frac{u_2}{v_2}\right]\cdots\left[\frac{u_n}{v_n}\right]$$

がマッチであるとは，上段を連結した文字列 $u_1u_2\cdots u_n$ と下段を連結した文字列 $v_1v_2\cdots v_n$ とが等しいことをいう．

問題 (Post の対応問題; PCP)

Input: ドミノの有限集合 P

Question: P はマッチを持つか？ (ただし，同じドミノを何回使ってもよい)

Post の対応問題

二つの文字列を上下に並べたものをドミノと呼ぶことにする．例えば

$$\left[\frac{\mathtt{ab}}{\mathtt{cde}}\right]$$

はドミノである．ドミノの列

$$\left[\frac{u_1}{v_1}\right]\left[\frac{u_2}{v_2}\right]\cdots\left[\frac{u_n}{v_n}\right]$$

がマッチであるとは，上段を連結した文字列 $u_1u_2\cdots u_n$ と下段を連結した文字列 $v_1v_2\cdots v_n$ とが等しいことをいう．

問題 (Post の対応問題; PCP)

Input: ドミノの有限集合 P
Question: P はマッチを持つか？ (ただし，同じドミノを何回使ってもよい)

例

例 1

$$P_1 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}, \left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}, \left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}, \left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} \right\}$$ とおく．このとき，

$$\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} = \left[\frac{\mathtt{abababababbaaaa}}{\mathtt{abababababbaaaa}}\right]$$

となるので P_1 はマッチを持つ．

例 2

$P_2 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{b}}{\mathtt{bab}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$，$P_3 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{aba}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{bab}}{\mathtt{a}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$ とおくと，P_2 は下段の方が上段より常に長いのでマッチを持たず，P_3 は右端に置けるドミノがないのでマッチを持たない．

例

例 1

$P_1 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}, \left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}, \left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}, \left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} \right\}$ とおく．このとき，

$$\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} = \left[\frac{\mathtt{abababbbaaaa}}{\mathtt{abababbbaaaa}}\right].$$

となるので P_1 はマッチを持つ．

例 2

$P_2 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{b}}{\mathtt{bab}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$，$P_3 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{aba}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{bab}}{\mathtt{a}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$ とおくと，P_2 は下段の方が上段より常に長いのでマッチを持たず，P_3 は右端に置けるドミノがないのでマッチを持たない．

例

例 1

$P_1 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}, \left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}, \left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}, \left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} \right\}$ とおく．このとき，

$$\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{ab}}{\mathtt{abab}}\right]_{①}\left[\frac{\mathtt{aba}}{\mathtt{b}}\right]_{③}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{b}}{\mathtt{a}}\right]_{②}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④}\left[\frac{\mathtt{aa}}{\mathtt{a}}\right]_{④} = \left[\frac{\mathtt{abababbbaaaa}}{\mathtt{abababbbaaaa}}\right].$$

となるので P_1 はマッチを持つ．

例 2

$P_2 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{abab}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{b}}{\mathtt{bab}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$，$P_3 = \left\{ \left[\frac{\mathtt{ab}}{\mathtt{aba}}\right], \left[\frac{\mathtt{b}}{\mathtt{a}}\right], \left[\frac{\mathtt{bab}}{\mathtt{a}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right] \right\}$ とおくと，P_2 は下段の方が上段より常に長いのでマッチを持たず，P_3 は右端に置けるドミノがないのでマッチを持たない．

決定不能性の証明: アイデア

定理

Post の対応問題 PCP は決定不能である.

証明のアイデア

仮に PCP が決定可能だったとして，HALT も決定可能になってしまうことを示す．HALT は決定不能だったので，このことから PCP も決定不能であることが導かれる.

PCP が決定可能であると仮定したので，PCP を判定するアルゴリズムが存在する．このアルゴリズムを利用して，HALT を判定するアルゴリズムを作ればよい．そのためには，任意の M, w に対して，あるドミノの有限集合 $P_{M,w}$ が作れて

$$M(w) = \textsc{Yes} \iff P_{M,w} \text{ はマッチを持つ}$$

となることを言えば十分である．いま仮定から右辺は決定可能なので，左辺も決定可能となる.

PCP の決定不能性の証明 (1/2)

HALT への入力を M と $w = w_1 \cdots w_n$ とする．#を M で用いられていない新しい文字とする．以下の手順で $P = P_{M,w}$ を構成する．

1. 計算の開始状況を表すドミノ $d = \left[\frac{\#}{\#q_0 w_1 \cdots w_n \#}\right]$ を P に追加する．
2. 各 $a, b \in \Gamma, q, r \in Q\,(r \neq q_{\mathsf{reject}})$ に対し $\delta(q, a) = (r, b, \rightarrow)$ のとき $\left[\frac{qa}{br}\right]$ を P に追加する．
3. 各 $a, b, c \in \Gamma, q, r \in Q\,(r \neq q_{\mathsf{reject}})$ に対し $\delta(q, a) = (r, b, \leftarrow)$ のとき $\left[\frac{cqa}{rcb}\right]$ を P に追加する．
4. 各 $a \in \Gamma$ に対して $\left[\frac{a}{a}\right]$ を追加する．
5. $\left[\frac{\#}{\#}\right], \left[\frac{\#}{\sqcup\#}\right]$ を P に追加する．
6. 各 $a \in \Gamma$ に対して $\left[\frac{aq_{\mathsf{accept}}}{q_{\mathsf{accept}}}\right], \left[\frac{q_{\mathsf{accept}}a}{q_{\mathsf{accept}}}\right]$ を P に追加する．
7. $\left[\frac{q_{\mathsf{accept}}\#\#}{\#}\right]$ を P に追加する．

このとき「$M(w) = \textsc{Yes} \iff P$ が d を左端とするマッチを持つ」が成り立つ．

PCP の決定不能性の証明 (2/2)

$*, \lozenge$ を P に現れない新しい文字とする．一般に文字列 $u = u_1 \cdots u_l$ に対し

$$\begin{aligned} \star u &:= *u_1 * u_2 * \cdots * u_l, \\ u\star &:= u_1 * u_2 * \cdots * u_l *, \\ \star u\star &:= *u_1 * u_2 * \cdots * u_l * \end{aligned}$$

と定義する．

$$P = \left\{ d = \left[\frac{u_1}{v_1}\right], \left[\frac{u_2}{v_2}\right], \ldots, \left[\frac{u_n}{v_n}\right] \right\}$$

に対し，PCP の入力を

$$P' := \left\{ \left[\frac{\star u_1}{\star v_1 \star}\right], \left[\frac{\star u_2}{v_2 \star}\right], \ldots, \left[\frac{\star u_n}{v_n \star}\right], \left[\frac{\star \lozenge}{\lozenge}\right] \right\}$$

と定める．このとき

P が左端が d であるマッチを持つ $\iff$ P' がマッチを持つ

となる．(証終)

例

HALT への入力を $(M, \mathtt{ab})$ とすると，

$$P = \left\{ d = \left[\frac{\#}{\#q_0\mathtt{ab}\#}\right], \left[\frac{q_0\mathtt{a}}{\sqcup q_1}\right], \left[\frac{q_0\sqcup}{\sqcup q_{\mathtt{acc}}}\right], \left[\frac{q_1\mathtt{a}}{\mathtt{a}q_1}\right], \left[\frac{q_1\mathtt{b}}{\mathtt{b}q_1}\right], \left[\frac{q_3\sqcup}{\sqcup q_0}\right], \left[\frac{\mathtt{a}q_1\sqcup}{q_2\mathtt{a}\sqcup}\right], \left[\frac{\mathtt{b}q_1\sqcup}{q_2\mathtt{b}\sqcup}\right], \left[\frac{\sqcup q_1\sqcup}{q_2\sqcup\sqcup}\right], \right.$$
$$\left[\frac{\mathtt{a}q_2\mathtt{b}}{q_3\mathtt{a}\sqcup}\right], \left[\frac{\mathtt{b}q_2\mathtt{b}}{q_3\mathtt{b}\sqcup}\right], \left[\frac{\sqcup q_2\mathtt{b}}{q_3\sqcup\sqcup}\right], \left[\frac{\mathtt{a}q_3\mathtt{a}}{q_3\mathtt{a}\mathtt{a}}\right], \left[\frac{\mathtt{b}q_3\mathtt{a}}{q_3\mathtt{b}\mathtt{a}}\right], \left[\frac{\sqcup q_3\mathtt{a}}{q_3\sqcup\mathtt{a}}\right], \left[\frac{\mathtt{a}q_3\mathtt{b}}{q_3\mathtt{a}\mathtt{b}}\right], \left[\frac{\mathtt{b}q_3\mathtt{b}}{q_3\mathtt{b}\mathtt{b}}\right], \left[\frac{\sqcup q_3\mathtt{b}}{q_3\sqcup\mathtt{b}}\right], \left[\frac{\mathtt{a}}{\mathtt{a}}\right], \left[\frac{\mathtt{b}}{\mathtt{b}}\right],$$
$$\left.\left[\frac{\sqcup}{\sqcup}\right], \left[\frac{\#}{\#}\right], \left[\frac{\#}{\sqcup\#}\right], \left[\frac{\mathtt{a}q_{\mathtt{acc}}}{q_{\mathtt{acc}}}\right], \left[\frac{\mathtt{b}q_{\mathtt{acc}}}{q_{\mathtt{acc}}}\right], \left[\frac{\sqcup q_{\mathtt{acc}}}{q_{\mathtt{acc}}}\right], \left[\frac{q_{\mathtt{acc}}\mathtt{a}}{q_{\mathtt{acc}}}\right], \left[\frac{q_{\mathtt{acc}}\mathtt{b}}{q_{\mathtt{acc}}}\right], \left[\frac{q_{\mathtt{acc}}\sqcup}{q_{\mathtt{acc}}}\right], \left[\frac{q_{\mathtt{acc}}\#\#}{\#}\right] \right\}$$

となり，P は例えば

$$\left[\frac{\#}{\#q_0\mathtt{ab}\#}\right]\left[\frac{q_0\mathtt{a}}{\sqcup q_1}\right]\left[\frac{\mathtt{b}}{\mathtt{b}}\right]\left[\frac{\#}{\#}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{q_1\mathtt{b}}{\mathtt{b}q_1}\right]\left[\frac{\#}{\sqcup\#}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\mathtt{b}q_1\sqcup}{q_2\mathtt{b}\sqcup}\right]\left[\frac{\#}{\#}\right]\left[\frac{\sqcup q_2\mathtt{b}}{q_3\sqcup\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\#}{\#}\right]\left[\frac{q_3\sqcup}{\sqcup q_0}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]$$
$$\left[\frac{\#}{\#}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{q_0\sqcup}{\sqcup q_{\mathtt{acc}}}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\#}{\#}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{q_{\mathtt{acc}}\sqcup}{q_{\mathtt{acc}}}\right]\left[\frac{\#}{\#}\right]\left[\frac{\sqcup}{\sqcup}\right]\left[\frac{\sqcup q_{\mathtt{acc}}}{q_{\mathtt{acc}}}\right]\left[\frac{\#}{\#}\right]\left[\frac{\sqcup q_{\mathtt{acc}}}{q_{\mathtt{acc}}}\right]\left[\frac{\#}{\#}\right]\left[\frac{q_{\mathtt{acc}}\#\#}{\#}\right]$$

というマッチを持つ．

コメント & 余談

もしマッチが存在すれば，全ての組み合わせを試していけばいつかは必ず見つかる．一方で，マッチがないときには全ての組み合わせを順番に試していっても (当然) マッチは見つからない．よって，PCP が決定不能であるということは「どれだけの組み合わせを試したとしても，マッチがないことを確信することはできない」ということを意味する．

はじめに示したいくつかの例でマッチがないことを証明することができたが，これは個々の入力に対してたまたまよい性質が発見できたために証明できたことである．一般には，そのようなマッチの非存在の証明を自動的に行うことはできない．実際，「入力のマッチの探索」と「入力がマッチを持たないことの ZFC からの証明の探索」を同時並行で行う Turing 機械 M を作ることができるが，PCP の決定不能性からこの M は停止しない，すなわち「マッチが存在しないにも関わらず，そのことを ZFC から証明できないようなドミノの有限集合」が存在する．

コメント & 余談

もしマッチが存在すれば，全ての組み合わせを試していけばいつかは必ず見つかる．一方で，マッチがないときには全ての組み合わせを順番に試していっても (当然) マッチは見つからない．よって，PCP が決定不能であるということは「どれだけの組み合わせを試したとしても，マッチがないことを確信することはできない」ということを意味する．

はじめに示したいくつかの例でマッチがないことを証明することができたが，これは個々の入力に対してたまたまよい性質が発見できたために証明できたことである．一般には，そのようなマッチの非存在の証明を自動的に行うことはできない．実際，「入力のマッチの探索」と「入力がマッチを持たないことの ZFC からの証明の探索」を同時並行で行う Turing 機械 M を作ることができるが，PCP の決定不能性からこの M は停止しない，すなわち「マッチが存在しないにも関わらず，そのことを ZFC から証明できないようなドミノの有限集合」が存在する．

第 3 弾 Wang のタイル貼り問題

タイル貼り問題の定義

Wang のタイル貼り問題とは，次のような決定問題である．

問題 (Wang のタイル貼り問題)

Input: 各辺に色が塗られたタイル (単位正方形) の有限集合 T
Question: T の元を使って以下のルールに沿って平面全体を充填できるか？

- 隣接する辺の色は同じでなければならない．
- 回転や反転は禁止．
- 同じタイルを何度使ってもよい．

注意

ここでいう「色」は区別が付くものならなんでもよく，自然数や文字列を代わりに使ってもよい．

タイル貼り問題の定義

Wang のタイル貼り問題とは，次のような決定問題である．

問題 (Wang のタイル貼り問題)

Input: 各辺に色が塗られたタイル (単位正方形) の有限集合 T
Question: T の元を使って以下のルールに沿って平面全体を充填できるか？

- 隣接する辺の色は同じでなければならない．
- 回転や反転は禁止．
- 同じタイルを何度使ってもよい．

注意

ここでいう「色」は区別が付くものならなんでもよく，自然数や文字列を代わりに使ってもよい．

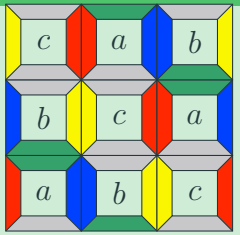
例

例 1

タイル集合 T を

$$T = \left\{ \boxed{a}, \boxed{b}, \boxed{c}, \boxed{d} \right\}$$

と定義すると，T は右のパターンを繰り返した周期的なタイル貼りを持つ．



例 2

一方で $T' = \left\{ \boxed{a}, \boxed{b} \right\}$ とおくと (a の上，b の下に置けるタイルがないので) T' はタイル貼りを持たない．

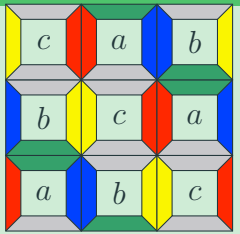
例

例 1

タイル集合 T を

$$T = \left\{ \boxed{a}, \boxed{b}, \boxed{c}, \boxed{d} \right\}$$

と定義すると，T は右のパターンを繰り返した周期的なタイル貼りを持つ．



例 2

一方で $T' = \left\{ \boxed{a}, \boxed{b} \right\}$ とおくと (a の上，b の下に置けるタイルがないので) T' はタイル貼りを持たない．

Wang のタイル貼り問題の決定不能性

Wang のタイル貼り問題の決定不能性を直接証明するのは大変なので，以下のような変種を考える．

問題 (制約付きのタイル貼り問題)

Input: タイルの有限集合 T とその元 $t \in T$
Question: T は t を原点に置くタイル貼りを持つか？

定理

制約付きのタイル貼り問題は決定不能である．

証明のアイデア

再び Turing 還元の技法を用いて，制約付きのタイル貼り問題が決定可能ならば HALT (の変種) も決定可能になることを示す．任意の Turing 機械 M に対して，あるタイル集合 T_M と $t \in T_M$ が作れて

$M(\varepsilon)$ の計算が停止しない $\iff$ T_M が t を原点とするタイル貼りを持つ

が成り立つことを示せばよい．

Wang のタイル貼り問題の決定不能性

Wang のタイル貼り問題の決定不能性を直接証明するのは大変なので，以下のような変種を考える．

問題 (制約付きのタイル貼り問題)

Input: タイルの有限集合 T とその元 $t \in T$
Question: T は t を原点に置くタイル貼りを持つか？

定理

制約付きのタイル貼り問題は決定不能である．

証明のアイデア

再び Turing 還元の技法を用いて，制約付きのタイル貼り問題が決定可能ならば HALT (の変種) も決定可能になることを示す．任意の Turing 機械 M に対して，あるタイル集合 T_M と $t \in T_M$ が作れて

$M(\varepsilon)$ の計算が停止しない $\iff$ T_M が t を原点とするタイル貼りを持つ

が成り立つことを示せばよい．

Wang のタイル貼り問題の決定不能性

Wang のタイル貼り問題の決定不能性を直接証明するのは大変なので，以下のような変種を考える．

問題 (制約付きのタイル貼り問題)

Input: タイルの有限集合 T とその元 $t \in T$
Question: T は t を原点に置くタイル貼りを持つか？

定理

制約付きのタイル貼り問題は決定不能である．

証明のアイデア

再び Turing 還元の技法を用いて，制約付きのタイル貼り問題が決定可能ならば HALT (の変種) も決定可能になることを示す．任意の Turing 機械 M に対して，あるタイル集合 T_M と $t \in T_M$ が作れて

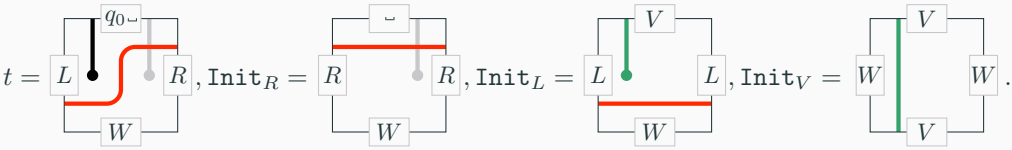
$M(\varepsilon)$ の計算が停止しない $\iff$ T_M が t を原点とするタイル貼りを持つ

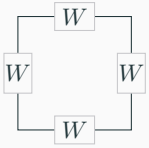
が成り立つことを示せばよい．

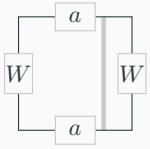
証明 (1/2)

$T = T_M$ を次のように構成していく．

1. まず，原点に置くタイル t を含めた初期タイル $t, \mathtt{Init}_R, \mathtt{Init}_L, \mathtt{Init}_V$ を T_M に加える:



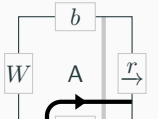
2. 次に空白タイル $\mathtt{Blank} =$  を T_M に加える．

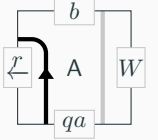
3. 各文字 $a \in \Gamma$ ごとに記号タイル $\mathtt{Alph}_a =$  $(a \in \Gamma)$ を T_M に加える．

証明 (2/2)

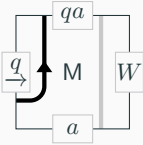
4. 各状態 $q \in Q\,(q \notin \{q_{\text{accept}}, q_{\text{reject}}\})$ と各 $a \in \Gamma$ ごとに遷移タイル $\texttt{Act}_{(q,a)}$ を T_M に加える．

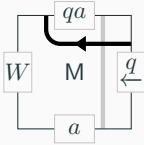
$$\texttt{Act}_{(q,a)} = \begin{cases} \texttt{Act}^{\rightarrow}_{(q,a)} = & \text{if } \delta(q,a) = (r, b, \rightarrow), \\ \texttt{Act}^{\leftarrow}_{(q,a)} = & \text{if } \delta(q,a) = (r, b, \leftarrow), \end{cases} \quad (q \in Q, q \notin \{q_{\text{accept}}, q_{\text{reject}}\}, a \in \Gamma).$$





5. 最後に，各状態 $q \in Q\,(q \notin \{q_{\text{accept}}, q_{\text{reject}}\})$ と各文字 $a \in \Gamma$ ごとに合流タイル $\texttt{Merg}_{q \to a}, \texttt{Merg}_{a \leftarrow q}$ を T_M に加える．

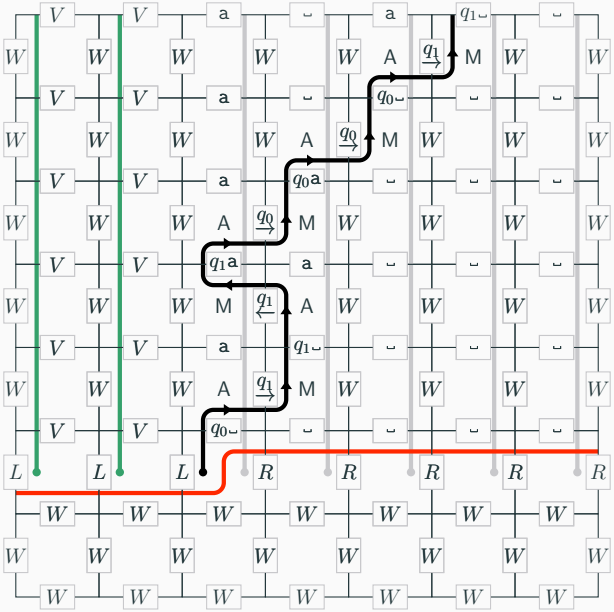
$\texttt{Merg}_{q \to a} =$ 

, $\texttt{Merg}_{a \leftarrow q} =$ 

$(q \in Q, q \notin \{q_{\text{accept}}, q_{\text{reject}}\}, a \in \Gamma).$

(証終)

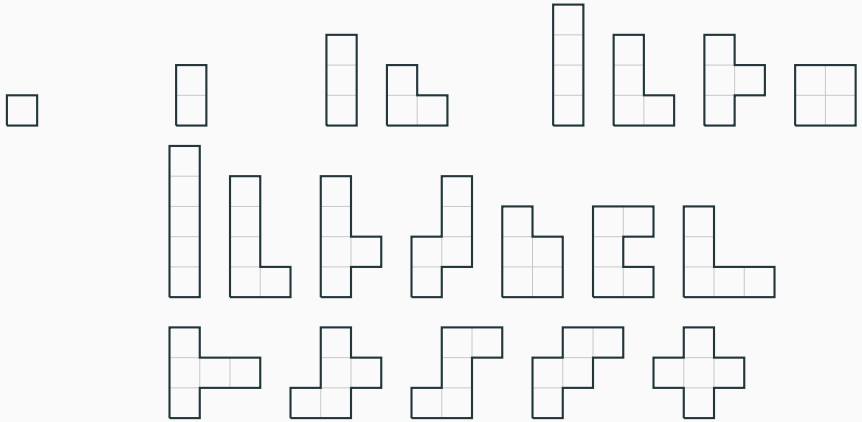
$M(\varepsilon)$ の計算が停止しないような Turing 機械 M に対して T_M を作ると，例えば次のようなタイル貼りを持つ．



第 4 弾 Polyomino Problem

ポリオミノ

ポリオミノとは，いくつかの単位正方形を辺で貼り合わせてできる図形のことである．



Polyomino Problem

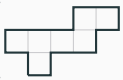
問題 (Polyomino Problem)

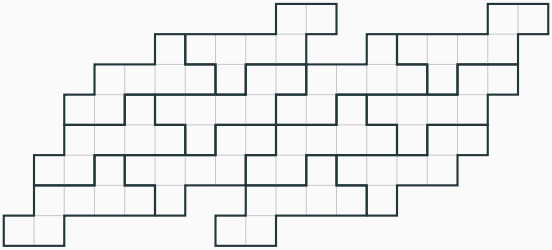
Input: ポリオミノの有限集合 S
Question: S の元を用いて平面全体を充填できるか？ (ただし，同じ元は何回使ってもよく，また回転・反転をしてもよいとする)

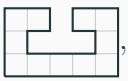
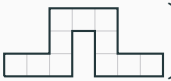
問題 (Polyomino Translation Problem)

Input: ポリオミノの有限集合 S
Question: S の元を用いて平面全体を充填できるか？ (ただし，同じ元は何回使ってもよいが，回転・反転は禁止とする)

例

1 つのポリオミノからなる集合 $S = \left\{ \right.$  $\left. \right\}$ を考えると，S は (回転を許せば) 以下のように平面全体を充填することができる．



一方，$S' = \left\{ \right.$  ,  $\left. \right\}$ とおくと，S' の元をどのように並べても平面全体を充填することはできない．

Polyomino Translation Problem の決定不能性

定理

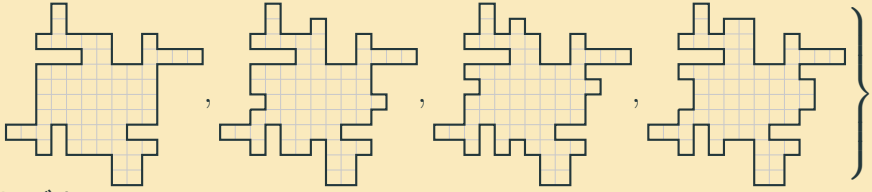
Polyomino Translation Problem は決定不能である．

証明の概略

仮に Polyomino Translation Problem が決定可能だったとすると，Wang のタイル貼り問題も決定可能になってしまうことを示す．タイルの色をポリオミノの凹凸で表現することで

$$T = \left\{ \begin{smallmatrix} & 0 & \\ 0 & & 0 \\ & 0 & \end{smallmatrix}, \begin{smallmatrix} & 1 & \\ 1 & & 1 \\ & 1 & \end{smallmatrix}, \begin{smallmatrix} & 2 & \\ 2 & & 2 \\ & 2 & \end{smallmatrix}, \begin{smallmatrix} & 3 & \\ 3 & & 3 \\ & 3 & \end{smallmatrix} \right\}$$

$$\mapsto S_T = \left\{ \quad , \quad , \quad , \quad \right\}$$



のようにすればよい．

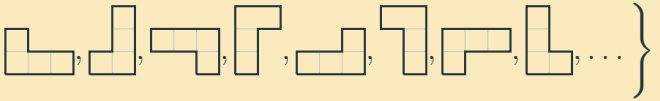
2 つの問題の等価性 (1/2)

定理

Polyomino Translation Problem が決定不能 $\iff$ Polyomino Problem が決定不能.

証明.

($\Longrightarrow$) polyomino problem の入力が例えば $S = \left\{ \ldots, \right.$  $\left. , \ldots \right\}$ だったとすると，S の各ポリオミノに対し，その回転・反転によって作られる全てのポリオミノ (高々 8 通り) を加えた集合を

$S' =$ 

とおけば，

S が回転・反転ありで全平面を充填できる

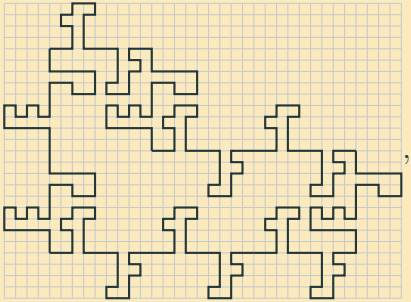
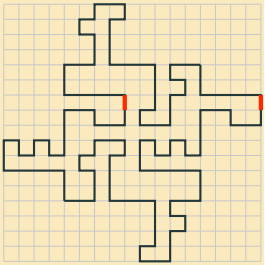
$\iff S'$ が回転・反転なしで全平面を充填できる

となる. □

2 つの問題の等価性 (2/2)

証明.

($\Longleftarrow$) polyomino translation problem の入力が例えば $S = \left\{\ldots, \right.$  $\left.,\ldots\right\}$ だったとする．このとき

$$S_{\circlearrowleft} = \left\{\ldots, \right.$$  $$\left.,\ldots\right\}, \square \mapsto$$ 

という変換を施せば，

S が回転・反転なしで全平面を充填できる

$\Longleftrightarrow S_{\circlearrowleft}$ が回転・反転ありで全平面を充填できる． $\square$

第 5 弾 Turing 機械の変種

Turing 機械の定義いろいろ

Turing 機械の定義を次のように変更しても計算能力は変化しない.

- テープを両方向に無限に伸ばす.
- ヘッドを左右に動かすだけでなく，その場に留まることも許す.
- ヘッドの動作を「右へ 1 ステップ移動」と「左端に移動」のみにする.
- ヘッドやテープをひとつではなく複数個にする.
- 非決定性 Turing 機械 (詳細は略).

(証明は略)

Turing 機械で自然数値関数を計算する

はじめの Turing 機械の定義においては $f\colon \Sigma^* \rightharpoonup \{\textsc{Yes}, \textsc{No}\}$ という関数を扱ったが，Turing 機械の定義を少し変更することで自然数値関数 $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ を計算するようにできる．

- $q_{\mathrm{accept}}, q_{\mathrm{reject}}$ の代わりにただひとつの停止状態 q_{halt} を用意する．
- 入力アルファベットを $\Sigma = \{1\}$ とする．
- 関数の引数が $(x_1, \ldots, x_k) \in \mathbb{N}^k$ であるときは，テープの左端から連続する $x_i + 1$ 個の 1 を空白記号␣で区切って並べ，残りを空白記号␣で埋めた状態 $1^{x_1+1}\text{␣}1^{x_2+1}\text{␣}\cdots\text{␣}1^{x_k+1}\text{␣␣}\cdots$ から計算を始める．
- q_{halt} に到達した時点でテープ上にある空白記号以外の文字の個数 (有限個) を出力値とする．
- q_{halt} に到達しなければ計算結果は未定義とする．

例えば，$q_0 = q_{\mathrm{halt}}$ なる機械 (テープの内容を全く変更せず最初のステップで停止する) を 1 変数関数 $f\colon \mathbb{N} \rightharpoonup \mathbb{N}$ を計算する機械と考えると $f(x) = x + 1$ である．

Turing 機械で自然数値関数を計算する

はじめの Turing 機械の定義においては $f\colon \Sigma^* \rightharpoonup \{\mathrm{YES}, \mathrm{NO}\}$ という関数を扱ったが，Turing 機械の定義を少し変更することで自然数値関数 $f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ を計算するようにできる．

- $q_{\mathsf{accept}}, q_{\mathsf{reject}}$ の代わりにただひとつの停止状態 q_{halt} を用意する．
- 入力アルファベットを $\Sigma = \{1\}$ とする．
- 関数の引数が $(x_1, \ldots, x_k) \in \mathbb{N}^k$ であるときは，テープの左端から連続する $x_i + 1$ 個の 1 を空白記号␣で区切って並べ，残りを空白記号␣で埋めた状態 $1^{x_1+1}␣1^{x_2+1}␣\cdots␣1^{x_k+1}␣␣\cdots$ から計算を始める．
- q_{halt} に到達した時点でテープ上にある空白記号以外の文字の個数 (有限個) を出力値とする．
- q_{halt} に到達しなければ計算結果は未定義とする．

例えば，$q_0 = q_{\mathsf{halt}}$ なる機械 (テープの内容を全く変更せず最初のステップで停止する) を 1 変数関数 $f\colon \mathbb{N} \rightharpoonup \mathbb{N}$ を計算する機械と考えると $f(x) = x + 1$ である．

第 6 弾 再帰的関数 & カウンター機械

再帰的関数 (1/2)

再帰的関数は次のように帰納的に定義される自然数上の関数のクラスである.

1. 以下の 3 つの初期関数は全て再帰的関数である.
   - 0 変数零関数 (定数) $\mathtt{zero}\colon \mathbb{N}^0 \to \mathbb{N};\quad \mathtt{zero}() = 0$
   - 射影 $\mathtt{proj}_i^n\colon \mathbb{N}^n \to \mathbb{N};\quad \mathtt{proj}_i^n(x_1, \ldots, x_n) = x_i\ (1 \leq i \leq n)$
   - 後者関数 $\mathtt{succ}\colon \mathbb{N} \to \mathbb{N};\quad \mathtt{succ}(x) = x + 1$
2. $g\colon \mathbb{N}^n \rightharpoonup \mathbb{N}\,(n > 0), h_1, \ldots, h_n\colon \mathbb{N}^m \rightharpoonup \mathbb{N}\,(m \geq 0)$ が再帰的関数ならば，それらの合成 $f\colon \mathbb{N}^m \rightharpoonup \mathbb{N}; f(\vec{x}) :\simeq g(h_1(\vec{x}), \ldots, h_n(\vec{x}))$ も再帰的関数である. ここで $h_1, \ldots, h_n, g$ のうちでひとつでも値が未定義なものがあれば f も未定義となる.

(続く)

再帰的関数 (2/2)

3. $g\colon \mathbb{N}^n \rightharpoonup \mathbb{N}, h\colon \mathbb{N}^{n+2} \rightharpoonup \mathbb{N}\,(n \geq 0)$ が再帰的関数ならば，原始再帰法により
$$f(0, \vec{y}) :\simeq g(\vec{y}),$$
$$f(x+1, \vec{y}) :\simeq h(x, \vec{y}, f(x, \vec{y}))$$
で定義される $f\colon \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}$ も再帰的関数である．
4. $g\colon \mathbb{N}^{n+1} \rightharpoonup \mathbb{N}\,(n \geq 0)$ が再帰的関数ならば，最小化演算子 μ によって
$$f(\vec{x}) :\simeq \mu y[g(\vec{x}, y)] := \min\{\, y \in \mathbb{N} \mid g(\vec{x}, y)\downarrow = 0 \,\}$$
で定義される $f\colon \mathbb{N}^n \to \mathbb{N}$ も再帰的関数である．ここで $f(\vec{x})$ は $g(\vec{x}, y)\downarrow = 0$ となるような $y \in \mathbb{N}$ が存在しないときは定義されない．
5. 以上によって定義されるもののみが再帰的関数である．

再帰的関数の例

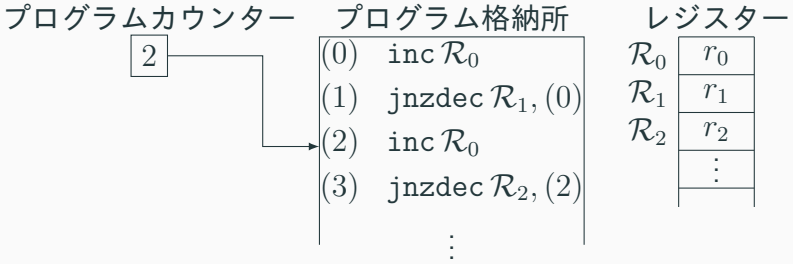
例

1 変数零関数，加算，乗算，指数，階乗は再帰的関数である．

$$
\begin{aligned}
\mathtt{zero}_1(0) &= \mathtt{zero}(),\\
\mathtt{zero}_1(x+1) &= \mathtt{proj}^2_2(x, \mathtt{zero}_1(x)),\\
\mathtt{add}(0, y) &= \mathtt{proj}^1_1(y),\\
\mathtt{add}(x+1, y) &= \mathtt{succ} \circ \mathtt{proj}^3_3(x, y, \mathtt{add}(x, y))),\\
\mathtt{mult}(0, y) &= \mathtt{zero}_1(y),\\
\mathtt{mult}(x+1, y) &= \mathtt{add} \circ (\mathtt{proj}^3_3, \mathtt{proj}^3_2)(x, y, \mathtt{mult}(x, y)),\\
x^0 &= 1,\\
x^{y+1} &= x^y \cdot x,\\
0! &= 1,\\
(x+1)! &= x! \cdot (x+1)
\end{aligned}
$$

カウンター機械 (1/2)

カウンター機械は以下のようなプログラムカウンターとプログラム格納所，可算個のレジスター $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2, \ldots$ からなる．



カウンター機械 (2/2)

カウンター機械のプログラムに利用可能な命令は次の 2 つのみである．

- $\texttt{inc}\,\mathcal{R}_i$: r_i の値を 1 増加させ，プログラムカウンターの値を 1 増加させる．
- $\texttt{jnzdec}\,\mathcal{R}_i, (n)$: $r_i = 0$ ならプログラムカウンターの値を 1 増加させる．$r_i \neq 0$ なら r_i の値を 1 減少させ，プログラムカウンターの値を n にする．

N 行からなるカウンター機械のプログラム P に対し，次のようにして計算される n 変数関数 $F_n^P(x_1, \ldots, x_n)$ が定まる．

1. レジスター $\mathcal{R}_1, \ldots, \mathcal{R}_n$ にそれぞれ $x_1, \ldots, x_n$ を格納する．
2. プログラムカウンターの値を 0 に設定する．
3. プログラムカウンターに保持されている行番号の命令を実行する．これをプログラムカウンターの値がプログラムの行数 N 以上になるまで繰り返す．プログラムカウンターの値が N 以上になった時点で計算を終了する．
4. 計算終了時点での r_0 を出力値 $F_n^P(x_1, \ldots, x_n)$ とする．

カウンター機械 (2/2)

カウンター機械のプログラムに利用可能な命令は次の 2 つのみである．

- $\texttt{inc}\,\mathcal{R}_i$: r_i の値を 1 増加させ，プログラムカウンターの値を 1 増加させる．
- $\texttt{jnzdec}\,\mathcal{R}_i, (n)$: $r_i = 0$ ならプログラムカウンターの値を 1 増加させる．$r_i \neq 0$ なら r_i の値を 1 減少させ，プログラムカウンターの値を n にする．

N 行からなるカウンター機械のプログラム P に対し，次のようにして計算される n 変数関数 $F_n^P(x_1, \ldots, x_n)$ が定まる．

1. レジスター $\mathcal{R}_1, \ldots, \mathcal{R}_n$ にそれぞれ $x_1, \ldots, x_n$ を格納する．
2. プログラムカウンターの値を 0 に設定する．
3. プログラムカウンターに保持されている行番号の命令を実行する．これをプログラムカウンターの値がプログラムの行数 N 以上になるまで繰り返す．プログラムカウンターの値が N 以上になった時点で計算を終了する．
4. 計算終了時点での r_0 を出力値 $F_n^P(x_1, \ldots, x_n)$ とする．

例

次のプログラム P を考える．

$$
\begin{array}{ll}
(0) & \texttt{inc}\,\mathcal{R}_0 \\
(1) & \texttt{inc}\,\mathcal{R}_0 \\
(2) & \texttt{jnzdec}\,\mathcal{R}_1, (0) \\
(3) & \texttt{inc}\,\mathcal{R}_0 \\
(4) & \texttt{jnzdec}\,\mathcal{R}_2, (3)
\end{array}
$$

P は関数 $F_2^P(x_1, x_2) = 2(x_1 + 1) + (x_2 + 1) = 2x_1 + x_2 + 3$ を計算するカウンター機械のプログラムである．実際，P を $r_1 = 2, r_2 = 1$ として実行すると次のように動作する．

行番号	0	1	2	0	1	2	0	1	2	3	4	3	4	5
$\mathcal{R}_0$	0	1	2	2	3	4	4	5	6	6	7	7	8	8
$\mathcal{R}_1$	2	2	2	1	1	1	0	0	0	0	0	0	0	0
$\mathcal{R}_2$	1	1	1	1	1	1	1	1	1	1	1	0	0	0

例

次のプログラム P を考える．

$$
\begin{aligned}
&(0) \quad \texttt{inc}\,\mathcal{R}_0\\
&(1) \quad \texttt{inc}\,\mathcal{R}_0\\
&(2) \quad \texttt{jnzdec}\,\mathcal{R}_1, (0)\\
&(3) \quad \texttt{inc}\,\mathcal{R}_0\\
&(4) \quad \texttt{jnzdec}\,\mathcal{R}_2, (3)
\end{aligned}
$$

P は関数 $F_2^P(x_1, x_2) = 2(x_1 + 1) + (x_2 + 1) = 2x_1 + x_2 + 3$ を計算するカウンター機械のプログラムである．実際，P を $r_1 = 2, r_2 = 1$ として実行すると次のように動作する．

行番号	0	1	2	0	1	2	0	1	2	3	4	3	4	5
$\mathcal{R}_0$	0	1	2	2	3	4	4	5	6	6	7	7	8	8
$\mathcal{R}_1$	2	2	2	1	1	1	0	0	0	0	0	0	0	0
$\mathcal{R}_2$	1	1	1	1	1	1	1	1	1	1	1	0	0	0

計算能力の等価性

定理

$f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ に対して，以下は同値.

1. f は Turing 機械で計算可能.
2. f は再帰的関数である.
3. f はカウンター機械で計算可能.

証明の概略.

1. 素因数分解を用いた有限列コード化の技法を用いて，Turing 機械の計算履歴を μ 演算子で求めればよい.
2. 再帰的関数の定義に沿って帰納的に計算すればよい.
3. カウンター機械が使用するレジスターと同じ数だけテープを用意した多テープ Turing 機械でシミュレートすればよい. □

計算能力の等価性

定理

$f\colon \mathbb{N}^k \rightharpoonup \mathbb{N}$ に対して，以下は同値.

1. f は Turing 機械で計算可能.
2. f は再帰的関数である.
3. f はカウンター機械で計算可能.

証明の概略.

1. 素因数分解を用いた有限列コード化の技法を用いて，Turing 機械の計算履歴を μ 演算子で求めればよい.
2. 再帰的関数の定義に沿って帰納的に計算すればよい.
3. カウンター機械が使用するレジスターと同じ数だけテープを用意した多テープ Turing 機械でシミュレートすればよい. □

第 7 弾 Fraction Game と一般化 Collatz 問題

Vector Game

問題 (Vector Game)

Input: $d, n \in \mathbb{N}$ と有限個のベクトル $v_1, \ldots, v_k \in \mathbb{Z}^d$

Question: $v = (n, 0, \ldots, 0) \in \mathbb{Z}^d$ からスタートして「$v + v_i \in \mathbb{N}^d$ となる最初の v_i を v に加算する」という操作を無限に続けることはできるか？

例

$d = 2, n = 7, v_1 = (-4, 1), v_2 = (-2, 0), v_3 = (3, -1)$ とおく．

$v = (n, 0) = (7, 0)$ から開始すると，

$$(7,0) \xrightarrow{+v_1} (3,1) \xrightarrow{+v_2} (1,1) \xrightarrow{+v_3} (4,0) \xrightarrow{+v_1} (0,1) \xrightarrow{+v_3} (3,0) \xrightarrow{+v_2} (1,0)$$

となり，これ以上操作を適用することはできないので停止する．

一方，$d = 2, n = 3, v_1 = (1, 2)$ とおけば，

$$(3,0) \xrightarrow{+v_1} (4,2) \xrightarrow{+v_1} (5,4) \xrightarrow{+v_1} \cdots$$

と無限に続く．

Vector Game

問題 (Vector Game)

Input: $d, n \in \mathbb{N}$ と有限個のベクトル $v_1, \ldots, v_k \in \mathbb{Z}^d$

Question: $v = (n, 0, \ldots, 0) \in \mathbb{Z}^d$ からスタートして「$v + v_i \in \mathbb{N}^d$ となる最初の v_i を v に加算する」という操作を無限に続けることはできるか？

例

$d = 2, n = 7, v_1 = (-4, 1), v_2 = (-2, 0), v_3 = (3, -1)$ とおく．

$v = (n, 0) = (7, 0)$ から開始すると，

$$(7,0) \xrightarrow{+v_1} (3,1) \xrightarrow{+v_2} (1,1) \xrightarrow{+v_3} (4,0) \xrightarrow{+v_1} (0,1) \xrightarrow{+v_3} (3,0) \xrightarrow{+v_2} (1,0)$$

となり，これ以上操作を適用することはできないので停止する．

一方，$d = 2, n = 3, v_1 = (1, 2)$ とおけば，

$$(3,0) \xrightarrow{+v_1} (4,2) \xrightarrow{+v_1} (5,4) \xrightarrow{+v_1} \cdots$$

と無限に続く．

Vector Game

問題 (Vector Game)

Input: $d, n \in \mathbb{N}$ と有限個のベクトル $v_1, \ldots, v_k \in \mathbb{Z}^d$

Question: $v = (n, 0, \ldots, 0) \in \mathbb{Z}^d$ からスタートして「$v + v_i \in \mathbb{N}^d$ となる最初の v_i を v に加算する」という操作を無限に続けることはできるか？

例

$d = 2, n = 7, v_1 = (-4, 1), v_2 = (-2, 0), v_3 = (3, -1)$ とおく．

$v = (n, 0) = (7, 0)$ から開始すると，

$$(7,0) \xrightarrow{+v_1} (3,1) \xrightarrow{+v_2} (1,1) \xrightarrow{+v_3} (4,0) \xrightarrow{+v_1} (0,1) \xrightarrow{+v_3} (3,0) \xrightarrow{+v_2} (1,0)$$

となり，これ以上操作を適用することはできないので停止する．

一方，$d = 2, n = 3, v_1 = (1, 2)$ とおけば，

$$(3,0) \xrightarrow{+v_1} (4,2) \xrightarrow{+v_1} (5,4) \xrightarrow{+v_1} \cdots$$

と無限に続く．

Vector Game の決定不能性

定理

Vector Game は決定不能である．

証明の概略.

仮に Vector Game が決定可能であるとすると，カウンター機械の停止問題が決定可能になってしまうことを示す．正確には，カウンター機械の変種である Minsky 機械をシミュレートできることを示す．P を Minsky 機械のプログラムとする．P の行数を N とし，P で使用されるレジスターは $\mathcal{R}_{M-1}$ までであるとする．$d = N + M + 1$ とおいて，$\mathbb{Z}^d$ の元を前半の M 個の成分と後半の $N + 1$ 個の成分に分けて考える．前半をレジスターの値を保持するために用い，後半を現在の行番号を保持するために利用する． □

例

P を次のものとする．

(0) $\texttt{jnzdec}\,\mathcal{R}_0, (1), (3)$
(1) $\texttt{inc}\,\mathcal{R}_1, (2)$
(2) $\texttt{inc}\,\mathcal{R}_1, (0)$

このプログラムは入力を 2 倍する関数
$f(x) = 2x$ を計算するプログラムである．
このプログラムに対応する vector game の
リスト L は右のようになる．

$$
\begin{aligned}
v_1 &= (-1, 0 \mid -1, 1, 0, 0)\\
v_2 &= \ (0, 0 \mid -1, 0, 0, 1)\\
v_3 &= \ (0, 1 \mid 0, -1, 1, 0)\\
v_4 &= \ (0, 1 \mid 1, 0, -1, 0)\\
v_5 &= (-1, 0 \mid 1, 0, 0, 0)\\
v_6 &= \ (0, 1 \mid 0, 0, 0, -1).
\end{aligned}
$$

これを $v = (2 + 1, 0 \mid 0, 0, 0, 0)$ から開始すると

$$
\begin{aligned}
(3, 0 \mid 0, 0, 0, 0) &\xrightarrow{+v_5} (2, 0 \mid 1, 0, 0, 0) \xrightarrow{+v_1} (1, 0 \mid 0, 1, 0, 0) \xrightarrow{+v_3} (1, 1 \mid 0, 0, 1, 0)\\
&\xrightarrow{+v_4} (1, 2 \mid 1, 0, 0, 0) \xrightarrow{+v_1} (0, 2 \mid 0, 1, 0, 0) \xrightarrow{+v_3} (0, 3 \mid 0, 0, 1, 0)\\
&\xrightarrow{+v_4} (0, 4 \mid 1, 0, 0, 0) \xrightarrow{+v_2} (0, 4 \mid 0, 0, 0, 1) \xrightarrow{+v_6} (0, 5 \mid 0, 0, 0, 0)
\end{aligned}
$$

で停止し，入力 2 に対する P の出力値は $5 - 1 = 4$ であるとわかる．

例

P を次のものとする．

(0) $\texttt{jnzdec}\,\mathcal{R}_0, (1), (3)$
(1) $\texttt{inc}\,\mathcal{R}_1, (2)$
(2) $\texttt{inc}\,\mathcal{R}_1, (0)$

$$\begin{aligned}
v_1 &= (-1, 0 \mid -1, 1, 0, 0)\\
v_2 &= (0, 0 \mid -1, 0, 0, 1)\\
v_3 &= (0, 1 \mid 0, -1, 1, 0)\\
v_4 &= (0, 1 \mid 1, 0, -1, 0)\\
v_5 &= (-1, 0 \mid 1, 0, 0, 0)\\
v_6 &= (0, 1 \mid 0, 0, 0, -1).
\end{aligned}$$

このプログラムは入力を 2 倍する関数
$f(x) = 2x$ を計算するプログラムである．
このプログラムに対応する vector game の
リスト L は右のようになる．
これを $v = (2 + 1, 0 \mid 0, 0, 0, 0)$ から開始すると

$$\begin{aligned}
(3, 0 \mid 0, 0, 0, 0) &\xrightarrow{+v_5} (2, 0 \mid 1, 0, 0, 0) \xrightarrow{+v_1} (1, 0 \mid 0, 1, 0, 0) \xrightarrow{+v_3} (1, 1 \mid 0, 0, 1, 0)\\
&\xrightarrow{+v_4} (1, 2 \mid 1, 0, 0, 0) \xrightarrow{+v_1} (0, 2 \mid 0, 1, 0, 0) \xrightarrow{+v_3} (0, 3 \mid 0, 0, 1, 0)\\
&\xrightarrow{+v_4} (0, 4 \mid 1, 0, 0, 0) \xrightarrow{+v_2} (0, 4 \mid 0, 0, 0, 1) \xrightarrow{+v_6} (0, 5 \mid 0, 0, 0, 0)
\end{aligned}$$

で停止し，入力 2 に対する P の出力値は $5 - 1 = 4$ であるとわかる．

Fraction Game

問題 (Fraction Game)

Input: 正整数 $n > 0$ と有限個の正の有理数 $q_1, \ldots, q_n \in \mathbb{Q}$
Question: n に対して「$q_i n \in \mathbb{Z}$ となる最初の q_i を n にかける」という操作を無限に繰り返すことはできるか？

素因数分解を用いて $\mathbb{Z}^d$ と有理数を対応させることで次がわかる.

系

Fraction Game は決定不能である.

Fraction Game

問題 (Fraction Game)

Input: 正整数 $n > 0$ と有限個の正の有理数 $q_1, \ldots, q_n \in \mathbb{Q}$
Question: n に対して「$q_i n \in \mathbb{Z}$ となる最初の q_i を n にかける」という操作を無限に繰り返すことはできるか？

素因数分解を用いて $\mathbb{Z}^d$ と有理数を対応させることで次がわかる.

系

Fraction Game は決定不能である.

VAS & 一般化 Collatz 問題

VAS の決定可能性

Vector Game における操作を少し変更して「$v_i + v \in \mathbb{N}^d$ となるようなあるv_i を v に加える」としたものは Vector Addition System (VAS) と呼ばれている．Vector Game の場合と異なり，VAS の到達可能性問題は決定可能であることが知られている．

問題 (一般化 Collatz 問題)

Input: 正整数 $m \in \mathbb{Z}_{>0}$ と有理数 $a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1} \in \mathbb{Q}$ であって，$n \equiv r \pmod{m}$ ならば $a_r n + b_r \in \mathbb{Z}$ となるようなもの
Question: 全ての正整数 $n \in \mathbb{Z}_{>0}$ について「$n \equiv r \pmod{m}$ ならば $a_r n + b_r$ を改めて n とおく」という操作を繰り返すといずれ 1 に到達するか？

定理

一般化 Collatz 問題は決定不能である．(証明は略)

VAS & 一般化 Collatz 問題

VAS の決定可能性

Vector Game における操作を少し変更して「$v_i + v \in \mathbb{N}^d$ となるようなあるv_i を v に加える」としたものは Vector Addition System (VAS) と呼ばれている．Vector Game の場合と異なり，VAS の到達可能性問題は決定可能であることが知られている．

問題 (一般化 Collatz 問題)

Input: 正整数 $m \in \mathbb{Z}_{>0}$ と有理数 $a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1} \in \mathbb{Q}$ であって，$n \equiv r \pmod{m}$ ならば $a_r n + b_r \in \mathbb{Z}$ となるようなもの
Question: 全ての正整数 $n \in \mathbb{Z}_{>0}$ について「$n \equiv r \pmod{m}$ ならば $a_r n + b_r$ を改めて n とおく」という操作を繰り返すといずれ 1 に到達するか？

定理

一般化 Collatz 問題は決定不能である．(証明は略)

VAS & 一般化 Collatz 問題

VAS の決定可能性

Vector Game における操作を少し変更して「$v_i + v \in \mathbb{N}^d$ となるようなあるv_i を v に加える」としたものは Vector Addition System (VAS) と呼ばれている．Vector Game の場合と異なり，VAS の到達可能性問題は決定可能であることが知られている．

問題 (一般化 Collatz 問題)

Input: 正整数 $m \in \mathbb{Z}_{>0}$ と有理数 $a_0, \ldots, a_{m-1}, b_0, \ldots, b_{m-1} \in \mathbb{Q}$ であって，$n \equiv r \pmod m$ ならば $a_r n + b_r \in \mathbb{Z}$ となるようなもの
Question: 全ての正整数 $n \in \mathbb{Z}_{>0}$ について「$n \equiv r \pmod m$ ならば $a_r n + b_r$ を改めて n とおく」という操作を繰り返すといずれ 1 に到達するか？

定理

一般化 Collatz 問題は決定不能である．(証明は略)

第 8 弾 半 Thue 系

半 Thue 系

半 Thue 系とは，文字列の書き換え規則 $u \longrightarrow v$ の有限集合 S のことをいう．文字列 w が u を部分文字列として含むとき，S のある書き換え規則 $u \longrightarrow v$ によって w 中の u を v に書き換えた文字列 w' が得られることを $w \underset{S}{\to} w'$ で表す．文字列 w に $\underset{S}{\to}$ を 0 回以上適用して w' に到達できることを $w \overset{*}{\underset{S}{\to}} w'$ で表し，そのときの文字列の列 $w = w_0 \underset{S}{\to} w_1 \underset{S}{\to} \cdots \underset{S}{\to} w_l = w'$ を $w \overset{*}{\underset{S}{\to}} w'$ の導出と呼ぶ．

対称的な半 Thue 系 S を特に Thue 系という．

半 Thue 系

半 Thue 系とは，文字列の書き換え規則 $u \longrightarrow v$ の有限集合 S のことをいう．文字列 w が u を部分文字列として含むとき，S のある書き換え規則 $u \longrightarrow v$ によって w 中の u を v に書き換えた文字列 w' が得られることを $w \underset{S}{\to} w'$ で表す．文字列 w に $\underset{S}{\to}$ を 0 回以上適用して w' に到達できることを $w \overset{*}{\underset{S}{\to}} w'$ で表し，そのときの文字列の列 $w = w_0 \underset{S}{\to} w_1 \underset{S}{\to} \cdots \underset{S}{\to} w_l = w'$ を $w \overset{*}{\underset{S}{\to}} w'$ の導出と呼ぶ．

対称的な半 Thue 系 S を特に Thue 系という．

半 Thue 系

半 Thue 系とは，文字列の書き換え規則 $u \longrightarrow v$ の有限集合 S のことをいう．文字列 w が u を部分文字列として含むとき，S のある書き換え規則 $u \longrightarrow v$ によって w 中の u を v に書き換えた文字列 w' が得られることを $w \underset{S}{\rightarrow} w'$ で表す．文字列 w に $\underset{S}{\rightarrow}$ を 0 回以上適用して w' に到達できることを $w \overset{*}{\underset{S}{\rightarrow}} w'$ で表し，そのときの文字列の列 $w = w_0 \underset{S}{\rightarrow} w_1 \underset{S}{\rightarrow} \cdots \underset{S}{\rightarrow} w_l = w'$ を $w \overset{*}{\underset{S}{\rightarrow}} w'$ の導出と呼ぶ．

対称的な半 Thue 系 S を特に Thue 系という．

半 Thue 系の例

例

$S = \{\, \mathtt{aba} \longrightarrow \mathtt{b}, \mathtt{ba} \longrightarrow \mathtt{abba} \,\}$ とおく．このとき，例えば

$$\begin{aligned} \mathtt{a\underline{ba}} &\underset{S}{\to} \mathtt{aab\underline{ba}} && (\mathtt{ba} \longrightarrow \mathtt{abba} \text{ を適用}) \\ &\underset{S}{\to} \mathtt{a\underline{aba}bba} && (\mathtt{ba} \longrightarrow \mathtt{abba} \text{ を適用}) \\ &\underset{S}{\to} \mathtt{abbba} && (\mathtt{aba} \longrightarrow \mathtt{b} \text{ を適用}) \end{aligned}$$

となるので $\mathtt{aba} \overset{*}{\underset{S}{\to}} \mathtt{abbba}$ である．他にも $\mathtt{ab\underline{ba}} \underset{S}{\to} \mathtt{\underline{aba}bba} \underset{S}{\to} \mathtt{bbba}$ から $\mathtt{abba} \overset{*}{\underset{S}{\to}} \mathtt{bbba}$ であることがわかる．

一方で，S の規則を用いて $\mathtt{abababa}$ から $\mathtt{abb}$ を導出することはできない．なぜなら，S には $\mathtt{b}$ の個数を減少させるような書き換え規則がないからである．

半 Thue 系の例

例

$S = \{\, \mathtt{aba} \longrightarrow \mathtt{b}, \mathtt{ba} \longrightarrow \mathtt{abba} \,\}$ とおく．このとき，例えば

$$\begin{aligned} \mathtt{a}\underline{\mathtt{ba}} &\underset{S}{\to} \mathtt{aa}\underline{\mathtt{ba}}\mathtt{bba} && (\mathtt{ba} \longrightarrow \mathtt{abba} \text{ を適用}) \\ &\underset{S}{\to} \mathtt{a}\underline{\mathtt{aba}}\mathtt{bba} && (\mathtt{ba} \longrightarrow \mathtt{abba} \text{ を適用}) \\ &\underset{S}{\to} \mathtt{abbba} && (\mathtt{aba} \longrightarrow \mathtt{b} \text{ を適用}) \end{aligned}$$

となるので $\mathtt{aba} \overset{*}{\underset{S}{\to}} \mathtt{abbba}$ である．他にも $\mathtt{ab}\underline{\mathtt{ba}} \underset{S}{\to} \underline{\mathtt{aba}}\mathtt{bba} \underset{S}{\to} \mathtt{bbba}$ から $\mathtt{abba} \overset{*}{\underset{S}{\to}} \mathtt{bbba}$ であることがわかる．

一方で，S の規則を用いて $\mathtt{abababa}$ から $\mathtt{abb}$ を導出することはできない．なぜなら，S には $\mathtt{b}$ の個数を減少させるような書き換え規則がないからである．

半 Thue 系に関わる決定問題たち

問題 (到達可能性問題)

Input: 半 Thue 系 S と文字列 w_1, w_2
Question: $w_1 \xrightarrow[S]{*} w_2$ か？

問題 (Thue 系の到達可能性問題)

Input: Thue 系 S と文字列 w_1, w_2
Question: $w_1 \xrightarrow[S]{*} w_2$ か？

問題 (共通子孫問題)

半 Thue 系 S を固定する.
Input: 文字列 w_1, w_2
Question: $w_1 \xrightarrow[S]{*} w$ かつ $w_2 \xrightarrow[S]{*} w$ となる文字列 w は存在するか？

到達可能性問題の決定不能性

定理

半 Thue 系の到達可能性問題は決定不能である．

証明.

仮に決定可能だったとすると，Turing 機械の停止問題も決定可能になってしまうことを示す．PCP の場合とほとんど同様に証明することができる．実際，PCP の決定不能性の証明で用いたドミノ

$$\left[\frac{qa}{br}\right], \left[\frac{cqa}{rcb}\right], \left[\frac{\#}{\sqcup\#}\right], \left[\frac{aq_{\mathsf{accept}}}{q_{\mathsf{accept}}}\right], \left[\frac{q_{\mathsf{accept}}a}{q_{\mathsf{accept}}}\right]$$

をそれぞれ

$$qa \to br, cqa \to rcb, q\# \to q\sqcup\#, aq_{\mathsf{accept}}\# \to q_{\mathsf{accept}}\#, q_{\mathsf{accept}}a \to q_{\mathsf{accept}}.$$

に書き換えればよい． □

M をはじめに定義した $\{\, \mathtt{a}^n\mathtt{b}^n \mid n \geq 0 \,\}$ を判定する Turing 機械とする．HALT への入力を $(M, \mathtt{ab})$ とすると，

$$
S^M = \left\{ \begin{array}{ccccc}
q_0\mathtt{@} \to q_0, & & q_0\mathtt{a} \to \mathtt{\_}q_1, & & q_0\mathtt{\_} \to \mathtt{\_}q_{\mathsf{accept}}, \quad q_1\mathtt{a} \to \mathtt{a}q_1, \\
q_1\mathtt{b} \to \mathtt{b}q_1, & & q_3\mathtt{\_} \to \mathtt{\_}q_0, & & q_0\mathtt{\#} \to q_0\mathtt{\_\#}, \quad q_1\mathtt{\#} \to q_1\mathtt{\_\#}, \\
\mathtt{a}q_1\mathtt{\_} \to q_2\mathtt{a\_}, & & \mathtt{b}q_1\mathtt{\_} \to q_2\mathtt{b\_}, & & \mathtt{\_}q_1\mathtt{\_} \to q_2\mathtt{\_\_}, \\
\mathtt{a}q_2\mathtt{b} \to q_3\mathtt{a\_}, & & \mathtt{b}q_2\mathtt{b} \to q_3\mathtt{b\_}, & & \mathtt{\_}q_2\mathtt{b} \to q_3\mathtt{\_\_}, \\
\mathtt{a}q_3\mathtt{a} \to q_3\mathtt{aa}, & & \mathtt{b}q_3\mathtt{a} \to q_3\mathtt{ba}, & & \mathtt{\_}q_3\mathtt{a} \to q_3\mathtt{\_a}, \\
\mathtt{a}q_3\mathtt{b} \to q_3\mathtt{ab}, & & \mathtt{b}q_3\mathtt{b} \to q_3\mathtt{bb}, & & \mathtt{\_}q_3\mathtt{b} \to q_3\mathtt{\_b}, \\
q_{\mathsf{accept}}\mathtt{a} \to q_{\mathsf{accept}}, & & q_{\mathsf{accept}}\mathtt{b} \to q_{\mathsf{accept}}, & & q_{\mathsf{accept}}\mathtt{\_} \to q_{\mathsf{accept}}, \\
\mathtt{a}q_{\mathsf{accept}}\mathtt{\#} \to q_{\mathsf{accept}}\mathtt{\#}, & & \mathtt{b}q_{\mathsf{accept}}\mathtt{\#} \to q_{\mathsf{accept}}\mathtt{\#}, & & \mathtt{\_}q_{\mathsf{accept}}\mathtt{\#} \to q_{\mathsf{accept}}\mathtt{\#}
\end{array} \right\},
$$

$$
\begin{aligned}
q_0\mathtt{@ab\#} &\xrightarrow[S^M]{} q_0\mathtt{ab\#} \xrightarrow[S^M]{} \mathtt{\_}q_1\mathtt{b\#} \xrightarrow[S^M]{} \mathtt{\_b}q_1\mathtt{\#} \xrightarrow[S^M]{} \mathtt{\_b}q_1\mathtt{\_\#} \xrightarrow[S^M]{} \mathtt{\_}q_2\mathtt{b\_\#} \xrightarrow[S^M]{} q_3\mathtt{\_\_\_\#} \\
&\xrightarrow[S^M]{} \mathtt{\_}q_0\mathtt{\_\_\#} \xrightarrow[S^M]{} \mathtt{\_\_}q_{\mathsf{accept}}\mathtt{\_\#} \xrightarrow[S^M]{} \mathtt{\_\_}q_{\mathsf{accept}}\mathtt{\#} \xrightarrow[S^M]{} \mathtt{\_}q_{\mathsf{accept}}\mathtt{\#} \xrightarrow[S^M]{} q_{\mathsf{accept}}\mathtt{\#}
\end{aligned}
$$

だから $q_0\mathtt{@ab\#} \xrightarrow[S^M]{*} q_{\mathsf{accept}}\mathtt{\#}$ となる．

第 9 弾 文脈自由言語の普遍性判定問題

文脈自由言語とプッシュダウンオートマトン

一般に，文字列の集合 $L \subseteq \Sigma^*$ を言語と呼ぶ．

プッシュダウンオートマトンとは，スタック付きの非決定性有限オートマトンのことをいう．プッシュダウンオートマトンによって認識される言語を文脈自由言語という．

問題 (文脈自由言語の普遍性判定問題)

Input: プッシュダウンオートマトン A
Question: A は全ての文字列を受理するか？

文脈自由言語とプッシュダウンオートマトン

一般に，文字列の集合 $L \subseteq \Sigma^*$ を言語と呼ぶ．

プッシュダウンオートマトンとは，スタック付きの非決定性有限オートマトンのことをいう．プッシュダウンオートマトンによって認識される言語を文脈自由言語という．

問題 (文脈自由言語の普遍性判定問題)

Input: プッシュダウンオートマトン A
Question: A は全ての文字列を受理するか？

文脈自由言語とプッシュダウンオートマトン

一般に，文字列の集合 $L \subseteq \Sigma^*$ を言語と呼ぶ．

プッシュダウンオートマトンとは，スタック付きの非決定性有限オートマトンのことをいう．プッシュダウンオートマトンによって認識される言語を文脈自由言語という．

問題 (文脈自由言語の普遍性判定問題)

Input: プッシュダウンオートマトン A
Question: A は全ての文字列を受理するか？

証明の概略

定理

文脈自由言語の普遍性判定問題は決定不能である．

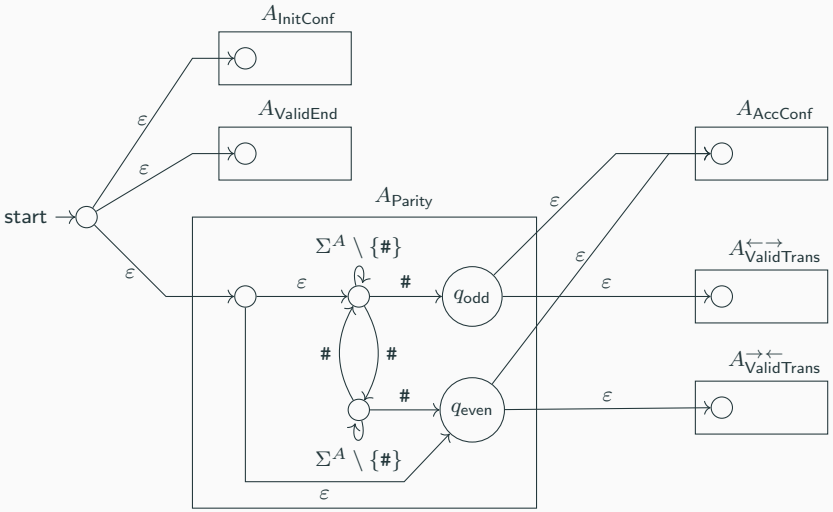
証明のアイデア．

仮に決定可能だったとすると，Turing 機械の停止問題が決定可能になってしまうことを示す．HALT への入力 M, w に対して，プッシュダウンオートマトン A を

$$M(w) \text{ の計算が停止する} \iff A \text{ がある文字列を拒否する}$$

となるように設計する．より具体的には，A は入力された文字列が $M(w)$ の受理計算履歴でないことを検査する機械である． □

構成の概略図


A_{InitConf}
A_{ValidEnd}
A_{AccConf}
A_{Parity}
start
ε
ε
ε
ε
ε
ε
ε
ε
$\Sigma^A \setminus \{\#\}$
$\#$
$\#$
$\#$
q_{odd}
q_{even}
$\Sigma^A \setminus \{\#\}$
$A^{\leftarrow\rightarrow}_{\mathsf{ValidTrans}}$
$A^{\rightarrow\leftarrow}_{\mathsf{ValidTrans}}$

第 10 弾 Hilbert の第 10 問題

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29$

$x^3 + y^3 + z^3 = 30$

$x^3 + y^3 + z^3 = 31$

$x^3 + y^3 + z^3 = 32$

$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29$
$x^3 + y^3 + z^3 = 30$
$x^3 + y^3 + z^3 = 31$
$x^3 + y^3 + z^3 = 32$
$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29 \quad (x, y, z) = (3, 1, 1)$

$x^3 + y^3 + z^3 = 30$

$x^3 + y^3 + z^3 = 31$

$x^3 + y^3 + z^3 = 32$

$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29 \quad (x, y, z) = (3, 1, 1)$

$x^3 + y^3 + z^3 = 30 \quad (x, y, z) = (-283059965, -2218888517, 2220422932)$

$x^3 + y^3 + z^3 = 31$

$x^3 + y^3 + z^3 = 32$

$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29 \quad (x, y, z) = (3, 1, 1)$

$x^3 + y^3 + z^3 = 30 \quad (x, y, z) = (-283059965, -2218888517, 2220422932)$

$x^3 + y^3 + z^3 = 31$ 解なし ($\mathrm{mod}\ 9$ で考えるとわかる)

$x^3 + y^3 + z^3 = 32$

$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29$ $(x, y, z) = (3, 1, 1)$

$x^3 + y^3 + z^3 = 30$ $(x, y, z) = (-283059965, -2218888517, 2220422932)$

$x^3 + y^3 + z^3 = 31$ 解なし ($\mathrm{mod}\ 9$ で考えるとわかる)

$x^3 + y^3 + z^3 = 32$ 解なし ($\mathrm{mod}\ 9$ で考えるとわかる)

$x^3 + y^3 + z^3 = 33$

Diophantus 方程式を解く

Diophantus 方程式を解くことは数論において重要な問題のひとつである.

例えば，Fermat の最終定理は無限個の Diophantus 方程式

$$x^3 + y^3 = z^3, x^4 + y^4 = z^4, x^5 + y^5 = z^5, \dots$$

が解を持たないという主張である.

他の例:

$x^3 + y^3 + z^3 = 29$ $(x, y, z) = (3, 1, 1)$

$x^3 + y^3 + z^3 = 30$ $(x, y, z) = (-283059965, -2218888517, 2220422932)$

$x^3 + y^3 + z^3 = 31$ 解なし ($\mathrm{mod}\ 9$ で考えるとわかる)

$x^3 + y^3 + z^3 = 32$ 解なし ($\mathrm{mod}\ 9$ で考えるとわかる)

$x^3 + y^3 + z^3 = 33$ $(8866128975287528, -8778405442862239, -2736111468807040)$
(2019 年 3 月に発見された)

Hilbert の第 10 問題

問題 (Hilbert の第 10 問題)

Input: (多変数) 多項式 $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, x_2, \ldots]$
Question: $f(x_1, \ldots, x_n) = 0$ は整数解 $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ を持つか？

定理

Hilbert の第 10 問題は決定不能である.

Hilbert の第 10 問題

問題 (Hilbert の第 10 問題)

Input: (多変数) 多項式 $f(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, x_2, \ldots]$
Question: $f(x_1, \ldots, x_n) = 0$ は整数解 $(x_1, \ldots, x_n) \in \mathbb{Z}^n$ を持つか？

定理

Hilbert の第 10 問題は決定不能である.

証明の大まかな流れ

1. 整数解でなく，自然数解の存否の判定が決定不能なことを確かめれば十分であることを示す．
2. Diophantus 方程式から Diophantus 的集合を，Turing 機械から c.e. 集合を定義する．
3. これらの 2 つの集合のクラスが等しいことを示す (Diophantus 的集合が c.e. であることは簡単)．
4. Diophantus 的集合のクラスが $\wedge, \vee, \exists$ で閉じていることを示す．
5. 基本的な集合・関係・関数が Diophantus 的であることを示す．
6. 指数関数 $a = b^c$ のグラフが Diophantus 的集合であることを示す (これが一番重要！)．
7. 指数関数を用いて，有限列のコード化の技法を開発する．
8. Turing 機械の計算を Diophantus 的な関係でシミュレートする．

指数関数 $a = b^c$ を定義する多項式

$b \geq 2$ に対し数列 $\alpha_b(n)$ を

$$\alpha_b(0) = 0, \alpha_b(1) = 1, \alpha_b(n+2) = b\alpha_b(n+1) - \alpha_b(n)$$

で定義する．三つ組 $(a, b, c) \in \mathbb{N}^3$ が $b \geq 4 \wedge a = \alpha_b(c)$ をみたすことと，以下の連立 Diophantus 方程式が $\mathbb{N}$ で解を持つことは同値である．

$$\begin{cases} b \geq 4, \\ u^2 - buu_1 + u_1^2 = 1, \\ r < s, \\ r^2 - brs + s^2 = 1, \\ u^2 \mid s, \\ v = bs - 2r, \\ w > 2, \\ x^2 - wxx_1 + x_1^2 = 1, \\ v \mid (w - b), \\ u \mid (w - 2), \\ a = \mathrm{arem}(x, v), \\ 2a < u, \\ c = \mathrm{arem}(x, u). \end{cases}$$

三つ組 $(a, b, c) \in \mathbb{N}^3$ が $a = b^c$ をみたすことと，以下の連立 Diophantus 方程式が $\mathbb{N}$ で解を持つことは同値である．

$$\begin{cases} x = 16c\alpha_{b+1}(c+1) + 17, \\ a = \alpha_{bx+4}(c+1) \operatorname{div} \alpha_x(c+1) \end{cases}$$

参考文献

- y., Turing 機械の定義と停止問題 (2018).
- y., Post の対応問題 (2018).
- y., Wang のタイル貼り問題 (2018).
- y., Polyomino Problem (2018).
- y., Turing 機械の変種 (2018).
- y., 再帰的関数 (2018).
- y., カウンター機械 (レジスター機械) (2018).
- y., Fraction Game と一般化 Collatz 問題 (2018).
- y., 半 Thue 系 (2018).
- y., 文脈自由言語の普遍性判定問題 (2018).
- y., Hilbert の第 10 問題 (2018).