計算不可能な関数

計算可能な関数とは？

定義

関数 f を計算するC言語のプログラムが存在するとき，
f は**計算可能**であるという．

- 次の関数 f_1, f_2, f_3, f_4 はいずれも計算可能．

入力	f_1	f_2	f_3	f_4
1	2	7	6	100
2	4	8	6	100
3	6	9	6	100
4	8	10	6	4
5	10	11	6	5
6	12	12	6	6
:	:	:	:	:

関数 f_1 を計算するプログラム

- $f_1(x) = 2x$

```
int f1(int x) {
  return 2 * x;
}
```

入力	f_1	f_2	f_3	f_4
1	2	7	6	100
2	4	8	6	100
3	6	9	6	100
4	8	10	6	4
5	10	11	6	5
6	12	12	6	6
:	:	:	:	:

関数 f_2 を計算するプログラム

- $f_2(x) = x + 6$

```
int f2(int x) {
  return x + 6;
}
```

入力	f_1	f_2	f_3	f_4
1	2	7	6	100
2	4	8	6	100
3	6	9	6	100
4	8	10	6	4
5	10	11	6	5
6	12	12	6	6
:	:	:	:	:

関数 f_3 を計算するプログラム

- $f_3(x) = 6$

```
int f3(int x) {
   return 6;
}
```

入力	f_1	f_2	f_3	f_4
1	2	7	6	100
2	4	8	6	100
3	6	9	6	100
4	8	10	6	4
5	10	11	6	5
6	12	12	6	6
:	:	:	:	:

関数 f_4 を計算するプログラム

- $$f_4(x) = \begin{cases} 100 & (x \leq 3) \\ x & (x \geq 4) \end{cases}$$

```
int f4(int x) {
   if (x < 4) {
      return 100;
   } else {
      return x;
   }
}
```

入力	f_1	f_2	f_3	f_4
1	2	7	6	100
2	4	8	6	100
3	6	9	6	100
4	8	10	6	4
5	10	11	6	5
6	12	12	6	6
	:	:	:	:

どんな関数でも
プログラミングできそうな
気がしてきたが．．．？

計算可能な関数

- 他のプログラミング言語で書かれていても，
  C言語に翻訳できる．よって，
  計算可能性は言語とは無関係である．

- 高校まで，あるいは大学の教養課程で出会ったほとんどすべての関数は計算可能である．

- はたして計算不可能な関数など存在するのか?

計算不能な関数

定義

関数 f を計算するC言語のプログラムが存在しないとき，
f は**計算不（可）能**であるという．

- はたしてそんなものが**存在**するか？
- 存在するとして，それを**具体的に示せる**か？

一対一の対応

- もしも，関数が 4 つあるが，C言語のプログラムが 3 つしかなければ，計算不可能な関数が存在することになる．
- もちろん，C言語のプログラムは無限に存在し，
  自然数から自然数への関数も無限に存在する．
- ただし，無限の度合いが違う．
  - 「可算」と「非可算」

可算集合(Countable set)

- 自然数全体の集合
- 偶数全体の集合
- 整数全体の集合
- 2次元平面上の格子点全体の集合
- 有理数全体の集合
- C言語のプログラム全体の集合

非可算集合(Uncountable set)

- 実数全体の集合
- 自然数から自然数への関数の集合

- 「対角線論法」を使って証明する.

対角線論法

- ３種類のアイスクリームの味の好みについて，３人の少年に聞いた.

好き？	太郎	健二	哲也	少年A
バラ	Y	N	Y	N
チョコ	N	N	Y	Y
ストロベリー	N	Y	Y	N

対角線論法

- 少年Aの好みは，次のようになった．

好き？	太郎	健一	哲也	少年A
バニラ	Y	N	Y	N
チョコレート	N	N	Y	Y
ストロベリー	N	Y	Y	N

- この事実から，少年Aは太郎でも健一でも哲也でもないことがわかる．

対角線論法

- *n* 種類のアイスクリームとn人の少年に対して
  - 少年Aは，*i* 番目の少年と *i* 番目のアイスクリーム味を好むかについて意見が異なるとする．
  - 少年Aは，*n*人の少年のうちのどの1人でもない．

好き？	太郎	健一	哲也	真一	秀夫	伸吾	英樹		少年A
バニラ	Y	N	Y	Y	Y	N	N		N
チョコレート	N	N	Y	Y	Y	N	Y		Y
ストロベリー	N	Y	Y	N	N	N	N		N
バナナ	Y	N	Y	N	N	N	N		Y
ブルーベリー	Y	N	Y	N	N	N	Y		Y
抹茶	N	Y	Y	Y	Y	N	N		Y
レモン	Y	N	Y	N	N	N	N		Y

この議論は（可算）無限個の要素に対しても適用できる．

入力

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	…	
好き？	太郎	健一	哲也	真一	秀夫	伸吾	英樹		少年A
バニラ	Y	N	Y	Y	Y	N	N		N
チョコレート	N	N	Y	Y	Y	N	Y		Y
ストロベリー	N	Y	Y	N	N	N	N		N
バナナ	Y	N	Y	N	N	N	N		Y
ブルーベリー	Y	N	Y	N	N	N	Y		Y
抹茶	N	Y	Y	Y	Y	N	N		Y
レモン	Y	N	Y	N	N	N	N		Y

⋮

プログラムの停止性問題

- ある入力に対して，プログラムが有限の時間で停まるかどうかを判定できるだろうか？

停止性が判定できるプログラム（１）

```
int func1(int x) {
  while (x == x) {
    x = x;
  }
  return x;
}
```

停止性が判定できるプログラム（２）

```
int func2(int x) {
  while (x > 10) {
    x = x;
  }
  return x;
}
```

次のプログラムはすべての入力に対して停止するか？（1）

```
int Hatena1(int x) {
  while (x > 1) {
    if (x % 2 == 0) {
      x = x / 2;
    } else {
      x = 3 * x + 1;
    }
  }
  return 1;
}
```

コラッツ-角谷の予想

次のプログラムはすべての入力に対して停止するか？（２）

```
int Hatena2(int x) {
  int n = 4;
  while (1) {
    if (n = a + bとなる素数a,bが存在しない) {
      return 0;
    }
    n = n + 2;
  }
}
```

ゴールドバッハの予想

プログラムの停止性問題

- ある入力に対して，プログラムが有限の時間で停まるかどうかを判定できるだろうか？

- 実際に実行してみれば、停まる場合はいつかは停まる．しかし，いつまで待てば停まるのかどうかが分からないため，
  - まだ停まっていないだけなのか，
  - そもそも停まらないのか

  を区別するのが難しい…気がする…

プログラムの等価性問題

- 二つのプログラムがどのような入力に対しても同じ結果を返すことを判定できるだろうか.

等価性が判定できるプログラム（１）

```
int A1(int x) {
  return 2 * x;
}
```

```
int A2(int x) {
  return x + x;
}
```

等価性が判定できるプログラム（２）

```
int B1(int x) {
  return x;
}
```

```
int B2(int x) {
  while (x < 10) {
    x = x;
  }
  return x;
}
```

プログラムの等価性問題

- 二つのプログラムがどのような入力に対しても同じ結果を返すことを判定できるだろうか.

- 様々な入力に対して実際に計算してみれば，等価でない場合はいつか異なる結果が出るため，判定が出来る．しかし，結果が等しい場合は
  - まだ結果が異なるような入力を試していないだけのか
  - 等価なので，そもそもすべての入力で結果が等しくなるのか

  を区別するのが難しい…気がする…

こんなプログラムができたらいいな.

- プログラムの停止性を判定するプログラム

```
int Halt(string P, string D) {
   if (P(D)を実行すると有限時間内に停止する) {
      return 1;
   } else {
      return 0;
   }
}
```

これさえあれば，無限ループにハマるか
どうかを事前に確かめられるのに.....

こんなプログラムができたらいいな.

- プログラムの等価性を判定するプログラム

```
int Equiv(string P1, string P2) {
  if (P1とP2は等価である) {
    return 1;
  } else {
    return 0;
  }
}
```

あらゆる入力に対して
同じ動作をすること

これさえあれば，プログラミング演習の
レポートの採点がすごく楽になるのに
…..

計算不能性

- 残念なことに，
  プログラム `Halt(P, D)`, `Equiv(P1, P2)` は，
  いずれも**存在しないこと**が証明できる．

- つまり，関数 `Halt`, `Equiv` は**計算不能**である．

証明（背理法）
プログラム Halt が存在したとすると…

- 次のようなプログラム Q が作れる.
- Q は，プログラム P にデータ P を入力したときに
  - 停止するならば無限ループへ（停止しない），
  - 停止しないならば停止する

```
int Q(string P) {
  int i = 0;
  if (Halt(P, P) == 1) {
    while (i < 100) i = i * 2;
  }
  return 0;
}
```

対角線論法を用いた停止性の証明のツボ

- 少年Aいわく
  「俺は，自分自身を愛するような奴は嫌いだ！」

好き？	太郎	健一	哲也	真一	秀夫	伸吾	英樹		少年A
太郎	Y	N	Y	Y	Y	N	N		N
健一	N	N	Y	Y	Y	N	Y		Y
哲也	N	Y	Y	N	N	N	N		N
真一	Y	N	Y	N	N	N	N		Y
秀夫	Y	N	Y	N	N	N	Y		Y
伸吾	N	Y	Y	Y	Y	N	N		Y
英樹	Y	N	Y	N	N	N	N		Y

少年Aは，少年A自身を好きなのか???

プログラムの停止性

- `Halt` があったとすると、
  下の表を埋めることができる

プログラム

データ	停止?	P1	P2	P3	P4	P5	P6	P7	…	
	D1	Y	N	Y	Y	Y	N	N		
	D2	N	N	Y	Y	Y	N	Y		
	D3	N	Y	Y	N	N	N	N		
	D4	Y	N	Y	N	N	N	N		
	D5	Y	N	Y	N	N	N	Y		
	D6	N	Y	Y	Y	Y	N	N		
	D7	Y	N	Y	N	N	N	N		
	⋮									

プログラムの停止性

- プログラムも文字列だから，データとみなせる．

プログラム

データ	停止?	P1	P2	P3	P4	P5	P6	P7	…	
	P1	Y	N	Y	Y	Y	N	N		
	P2	N	N	Y	Y	Y	N	Y		
	P3	N	Y	Y	N	N	N	N		
	P4	Y	N	Y	N	N	N	N		
	P5	Y	N	Y	N	N	N	Y		
	P6	N	Y	Y	Y	Y	N	N		
	P7	Y	N	Y	N	N	N	N		
	⋮									

プログラムの停止性

- **プログラム Q は，**
  プログラム P にデータ P を入力したとき
  - P(P) が停止するならば無限ループへ（停止しない），
  - P(P) が停止しないならば停止する．

プログラム

データ: 停止?	P1	P2	P3	P4	P5	P6	P7	P*
P1	Y	N	Y	Y	Y	N	N	N
P2	N	N	Y	Y	Y	N	Y	Y
P3	N	Y	Y	N	N	N	N	N
P4	Y	N	Y	N	N	N	N	Y
P5	Y	N	Y	N	N	N	Y	Y
P6	N	Y	Y	Y	Y	N	N	Y
P7	Y	N	Y	N	N	N	N	Y

プログラムの停止性

…?

プログラム

データ: 停止?	P1	P2	P3	P4	P5	P6	P7	P*
P1	Y	N	Y	Y	Y	N	N	N
P2	N	N	Y	Y	Y	N	Y	Y
P3	N	Y	Y	N	N	N	N	N
P4	Y	N	Y	N	N	N	N	Y
P5	Y	N	Y	N	N	N	Y	Y
P6	N	Y	Y	Y	Y	N	N	Y
P7	Y	N	Y	N	N	N	N	Y

Q(Q)は，停止するか???

Equivが存在しないことの証明

```
int R<P,D>(string x) {
  P(D);
  return 0;
}
```

```
int CONST(string x) {
  return 0;
}
```

```
int Halt(string P, string D) {
  if (Equiv(R<P,D>, CONST) == 1) {
    return 1;
  } else {
    return 0;
  }
}
```

Equiv の存在を仮定すると
Halt を構成できてしまう
これは矛盾