アルゴリズム論

平成22年11月11日

Satisfiability

充足可能性問題

- （**NP**完全）充足可能性問題　satisfiability or SAT

  ［入力］和積形式のブール式 $B(x_1, x_2, \cdots, x_n) = F_1 \bullet F_2 \bullet \cdots \bullet F_p$

  ［性質］$B = \mathrm{True}$ となるように各変数 x_k へ
  値（True or False）を割り当てできる

- （**NP**完全）3-充足可能性問題　3-satisfiability or 3SAT

  ［入力］和積形式のブール式 $B(x_1, x_2, \cdots, x_n) = F_1 \bullet F_2 \bullet \cdots \bullet F_p$,
  ただし，各 F_k は３個（以下）のリテラルの和

  ［性質］$B = \mathrm{True}$ となるように各変数 x_k へ
  値（True or False）を割り当てできる

ブール代数の公理

$a, b, c \in B$

交換律
$$\begin{cases} a \vee b = b \vee a \\ a \bullet b = b \bullet a \end{cases}$$

分配律
$$\begin{cases} a \bullet (b \vee c) = (a \bullet b) \vee (a \bullet c) \\ a \vee (b \bullet c) = (a \vee b) \bullet (a \vee c) \end{cases}$$

同一律
$$\begin{cases} a \vee 0 = a & [\text{零元の性質}] \\ a \bullet 1 = a & [\text{単位元の性質}] \end{cases}$$

補元律
$$\begin{cases} a \vee \overline{a} = 1 \\ a \bullet \overline{a} = 0 \end{cases}$$

[$\overline{a}$ を a の補元という]

乗法標準形

- ブール変数　値は 1(真) or 0(偽)
- リテラル　ブール変数 or その補元
- 基本和　一つのリテラル or
  同じ変数を含まないリテラルの和
- 乗法標準形　一つの基本和 or
  他を含まない基本和の積

$(x \vee y)(\overline{x} \vee \overline{y} \vee z)(y \vee z)$ ← 乗法標準形

$(x \vee \overline{y})(x \vee \overline{y} \vee z)(\overline{x} \vee y \vee z) = (x \vee \overline{y})(\overline{x} \vee y \vee z)$　吸収律

和積形式のブール式

$f(x,y,z)=(x\vee y)(\overline{x}\vee\overline{y}\vee z)(y\vee z)$　← 和の積

$f(x,y,z)=1$ となるように

各変数に値を割り当てることができるか？

$f(0,0,0)=(0\vee 0)(\overline{0}\vee\overline{0}\vee 0)(0\vee 0)=0\bullet 1\bullet 0=0$

$f(0,0,1)=(0\vee 0)(\overline{0}\vee\overline{0}\vee 1)(0\vee 1)=0\bullet 1\bullet 1=0$

$f(0,1,0)=(0\vee 1)(\overline{0}\vee\overline{1}\vee 0)(1\vee 0)=1\bullet 1\bullet 1=1$

変数の割り当て方は全部で 2^n（n は変数の個数）

和積形式：CNF conjunctive normal form　ともいう

積和形式：DNF disjunctive normal form　ともいう

積和形式 = 0 とできるか

例：$f(x,y,z) = x\,y\,\overline{z} \vee \overline{x}\,z \vee \overline{x}\,\overline{z} \vee x\,y\,z \vee x\,\overline{y} = 0$

$$f(0,0,0) = 0 \bullet 0 \bullet \overline{0} \vee \overline{0} \bullet 0 \vee \overline{0} \bullet \overline{0} \vee 0 \bullet 0 \bullet 0 \vee 0 \bullet \overline{0} = 1$$

$$f(0,0,1) = 0 \bullet 0 \bullet \overline{1} \vee \overline{0} \bullet 1 \vee \overline{0} \bullet \overline{1} \vee 0 \bullet 0 \bullet 1 \vee 0 \bullet \overline{0} = 1$$

$$f(0,1,1) = 0 \bullet 1 \bullet \overline{1} \vee \overline{0} \bullet 1 \vee \overline{0} \bullet \overline{1} \vee 0 \bullet 1 \bullet 1 \vee 0 \bullet \overline{1} = 1$$

$$f(1,1,1) = 1 \bullet 1 \bullet \overline{1} \vee \overline{1} \bullet 1 \vee \overline{1} \bullet \overline{1} \vee 1 \bullet 1 \bullet 1 \vee 1 \bullet \overline{1} = 1$$

実は，$f(x,y,z) = 0$ とすることはできない

$$\begin{aligned} f(x,y,z) &= x\,y\,\overline{z} \vee \overline{x}\,z \vee \overline{x}\,\overline{z} \vee x\,y\,z \vee x\,\overline{y} \\ &= (x\,y\,\overline{z} \vee x\,y\,z) \vee x\,\overline{y} \vee (\overline{x}\,z \vee \overline{x}\,\overline{z}) \\ &= x\,y(\overline{z} \vee z) \vee x\,\overline{y} \vee \overline{x}(z \vee \overline{z}) = x\,y \vee x\,\overline{y} \vee \overline{x} = x \vee \overline{x} = 1 \end{aligned}$$

充足可能性問題（SAT）

- 和積形式のブール式の値が 1 となる（充足する）ように，各変数に 1 または 0 を割り当てることができるか。

  SAT = satisfiability problem

- 問題 SAT は判定問題
- 問題 SAT は **NP** に属する

  証明書は，ブール式が 1 となるような各変数へ値の割り当て方

NP 完全（NP-complete）の定義

- クラス **NP** に属す判定問題 Q に対し，
  NP に属すすべての判定問題が Q に帰着できるとき，Q は **NP完全** であるという

  ① $Q \in \mathbf{NP}$

  ② $Q \triangleright Q'$ for any $Q' \in \mathbf{NP}$

- **NP完全** な問題は，クラス **NP** の中でも最も難しい問題ということになる

3SAT も NP-完全

- 3SAT :
  　個別問題である和積形式のブール式の各和項が，高々３個のリテラルの和からできているという条件がついた充足可能性問題

SAT ∈ NP-complete なので,

3SAT ▹ SAT であることを示せばよい

SAT は 3SAT に帰着できる

$f = F_1 \bullet F_2 \bullet \cdots \bullet F_i \bullet \cdots \bullet F_p$ (各 F_i は和項)

$g = G_1 \bullet G_2 \bullet \cdots \bullet G_j \bullet \cdots \bullet G_q$ (各 G_j は３個以下のリテラル)

f が充足可能 $\Leftrightarrow$ g が充足可能

例えば $F = x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6$ のとき

$G = (x_1 \vee x_2 \vee z_1)(x_3 \vee \overline{z}_1 \vee z_2)(x_4 \vee \overline{z}_2 \vee z_3)(x_5 \vee x_6 \vee \overline{z}_3)$

SAT のための f が与えられたら
このルールで g を作り
3SAT アルゴリズムで解けばよい

よって 3SAT ▹ SAT である

SAT は 3SAT に帰着できる

$F = x_1 \vee x_2 \vee \cdots \vee x_k \quad (k \geq 4)$

$G = (x_1 \vee x_2 \vee z_1)(x_3 \vee \overline{z}_1 \vee z_2)\cdots(x_r \vee \overline{z}_{r-2} \vee z_{r-1})\cdots(x_{k-1} \vee x_k \vee \overline{z}_{k-3})$

F が充足可能なとき

- $x_r = \mathrm{T}$ となる r がある
- $z_1 = \cdots = z_{r-2} = \mathrm{T}$, $z_{r-1} = \cdots = z_{k-2} = \mathrm{F}$ とすると $G = \mathrm{T}$

$$G = \cdots(x_{r-1} \vee \overline{z}_{r-3} \vee z_{r-2})(x_r \vee \overline{z}_{r-2} \vee z_{r-1})(x_{r+1} \vee \overline{z}_{r-1} \vee z_r)\cdots$$

G が充足可能なとき

- $x_1 = \cdots = x_k = \mathrm{F}$ と仮定する
- $G = z_1(\overline{z}_1 \vee z_2)(\overline{z}_2 \vee z_3)\cdots(\overline{z}_{k-4} \vee z_{k-3})\overline{z}_{k-3}$
  $= z_1 z_2(\overline{z}_2 \vee z_3)\cdots(\overline{z}_{k-4} \vee z_{k-3})\overline{z}_{k-3} = z_1 z_2 \cdots z_{k-3}\overline{z}_{k-3} = \mathrm{F}$
- G が充足可能であることに矛盾する

和積形式ブール式をリストで

- ブール式の変数は $x_1, x_2, \cdots, x_n$ とし，添字を変数名とする
- x_1 は“1”，$\overline{x}_2$ は“-2”と表す
- $(x_1 \vee \overline{x}_2 \vee x_4)(x_3 \vee \overline{x}_4 \vee x_5)(\overline{x}_2 \vee \overline{x}_4 \vee \overline{x}_5)(x_1 \vee \overline{x}_5)$ は
  [[1, -2, 4], [3, -4, 5], [-2, -4, -5], [1, -5]]
- $(x_{11} \vee \overline{x}_{21} \vee x_{41})(x_{13} \vee \overline{x}_{24} \vee x_{21})(\overline{x}_{23} \vee \overline{x}_{21} \vee \overline{x}_{31})(x_{22} \vee \overline{x}_{25})\cdots$

  変数リスト：[11, 13, 21, 22, 23, 24, 25, 31, 41]

  $\Rightarrow (x_1 \vee \overline{x}_3 \vee x_9)(x_2 \vee \overline{x}_6 \vee x_3)(\overline{x}_5 \vee \overline{x}_3 \vee \overline{x}_8)(x_4 \vee \overline{x}_7)\cdots$

countSats の高速化

- 各和項の変数名(1～n と -1～-n)に n を加えると
  0, 1, ..., n-1, [n], n+1, n+2, ..., 2n となる。
  和項の中に 0 はないので, n となることはない。
- これに対応して, 変数値のリスト values の長さも
  2n+1 にする必要がある。
- n = 4 とすると,
  values の初期値 = [1, 1, 1, 1, X, 0, 0, 0, 0]
  → [0, 1, 1, 1, X, 0, 0, 0, 1]
  → [1, 0, 1, 1, X, 0, 0, 1, 0]
  [0, 0, 0, 0, X, 1, 1, 1, 1] が最後

変数名を再構成

```
def restructure(A):   # fast version
    """ A で使われている変数名を再構成する """
    vrblList = []
    for term in A:    # term は和項：リテラルの和
        for vrbl in term:    # vrbl は和項のなかの変数
            if abs(vrbl) not in vrblList:
                vrblList.append(abs(vrbl))
    vrblList.sort()
    N = len(vrblList)    # 変数の個数
    for term in A:
        for k in range(len(term)):
            if term[k] > 0:
                term[k] = N + 1 + vrblList.index(term[k])
            else:
                term[k] = N - 1 - vrblList.index(-term[k])
    return vrblList
```

変数名再構成後

```
>>> A = [[11,-23],[12,32],[-12,-31],[-32,-11]]
>>> restructure(A)
[11, 12, 23, 31, 32]
>>> A
[[6, 2], [7, 10], [3, 1], [0, 4]]
```

再構成前	-32	-31	-23	-12	-11	×	11	12	23	31	32
再構成後	0	1	2	3	4	×	6	7	8	9	10
values	1	1	1	1	1	2	0	0	0	0	0
	0	1	1	1	1	2	0	0	0	0	1
last	0	0	0	0	0	2	1	1	1	1	1

satisfiability

```
def satisfiability(A):
    """ CNFリスト A の充足可能性を調べる """
    vrblList = restructure(A)
    if 0 in vrblList:
        print "variable 0 is not permitted"
        return []
    termNum, vrblNum = (len(A), len(vrblList))
    values = [1 for k in range(vrblNum)] + [2] + ¥
             [0 for k in range(vrblNum)]
    while countSats(A, values) != termNum:
        if nextVal(values) == False:
            original(A, vrblList)
            return []
    original(A, vrblList)
    return makeAnswer(values, vrblList)
```

countSats

```
def countSats(A, values):
    """ CNFリスト A のなかの和項が True になる個数 """
    count = 0
    for term in A:              # リスト A の各和項 term について
        for vrbl in term:       # 和項 term の各変数 vrbl について
            if values[vrbl] == 1:   # 一つでも 1 があれば
                count += 1          # 和項 term は True
                break
    return count
```

nextVal

```
def nextVal(values):
    """ 各変数の値を更新 """
    for k in range(len(values) / 2):
        if values[k] == 1:
            values[k] = 0
            values[-(k + 1)] = 1
            return True
        else:
            values[k] = 1
            values[-(k + 1)] = 0
    return False
```

original

```
def original(A, vrblList):
    """ CNFリスト A を元の状態に戻す """
    N = len(vrblList)
    for term in A:
        for k in range(len(term)):
            if term[k] < N:
                term[k] = -(vrblList[-(term[k] + 1)])
            else:
                term[k] = vrblList[term[k] - N - 1]
    return
```

makeAnswer

```
def makeAnswer(values, vrblList):
    """ 充足する変数の割り当て値を返す """
    answer = []
    N = len(vrblList)
    for k in range(N):
        if values[k + 1 + N] == 1:
            answer.append(vrblList[k])
        else:
            answer.append(-vrblList[k])
    return answer
```

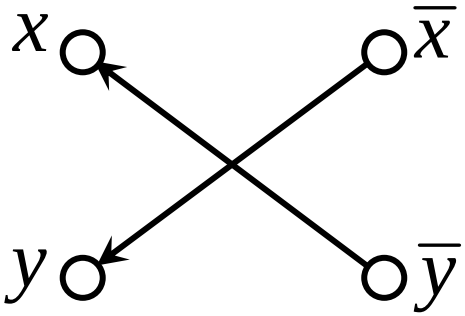
課題 sat1：充足可能性問題

- 関数 countSats と nextVal を作成して，充足可能性問題を解くプログラム satisfiability を完成せよ

課題 sat2

- ［課題 sat2］２充足可能性問題 2SAT はクラス **P** に属すことを示せ［難しい］

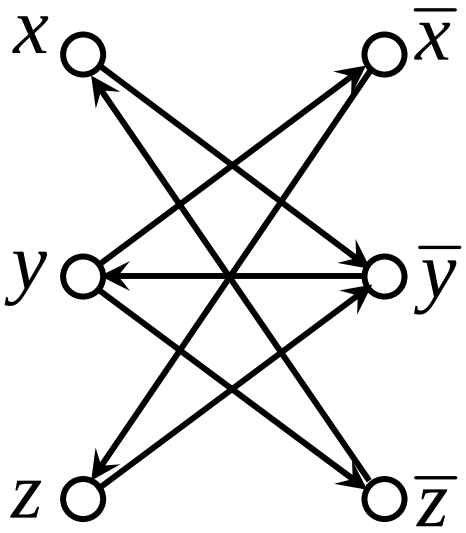
$$[x \vee y = 1] \equiv [x = 0 \Rightarrow y = 1] \wedge [y = 0 \Rightarrow x = 1]$$
$$\equiv [\overline{x} = 1 \Rightarrow y = 1] \wedge [\overline{y} = 1 \Rightarrow x = 1]$$



$$x = x \vee x$$
$$[x \vee x = 1] \equiv [\overline{x} = 1 \Rightarrow x = 1] \wedge [\overline{x} = 1 \Rightarrow x = 1]$$
$$\equiv [\overline{x} = 0] \equiv [x = 1]$$

課題 sat2

$$y(\overline{x} \vee \overline{y})(\overline{y} \vee \overline{z})(z \vee x)$$



x から $\overline{x}$ への道があり
かつ, $\overline{x}$ から x への道がある

$\Downarrow$

充足不可能

$d(x, y)$: x から y への距離

$\exists x,\ d(x, \overline{x}) < \infty,\ d(\overline{x}, x) < \infty \Rightarrow$ 充足不可能

$\forall x,\ d(x, \overline{x}) = \infty$ or $d(\overline{x}, x) = \infty \Rightarrow$ 充足可能

課題 sat3

- ［課題 sat3］和積形式のブール式が充足可能であるかどうかを答える多項式時間アルゴリムがある。これを利用して，変数の個数 n の多項式時間で変数の充足割り当て(の一つ)を見つけるにはどうしたらよいか。