アルゴリズム特論（第1回）

北海道大学　大学院　情報科学研究科
アルゴリズム研究室　　湊　真一

アルゴリズム特論 2006年度講義予定

- 前半（担当：湊助教授）
  - 大規模データ処理アルゴリズムの技法
  - 論理関数と組合せ最適化
- 後半（担当：トーマス教授）
  - 計算論的学習アルゴリズムの基礎
  - 言語学習とそのアルゴリズム

前半の講義内容

- 第1回　論理関数の処理とデータ表現
- 第2回　積和形論理式
- 第3回　積和形論理式の簡単化
- 第4回　二分決定グラフ(BDD)
- 第5回　BDD処理系の実装技術
- 第6回　BDDの変数順序づけ
- 第7回　BDDの応用
- 第8回　様々なBDD

講義ノートURL:
http://www-alg.ist.hokudai.ac.jp/~minato/class-j.html

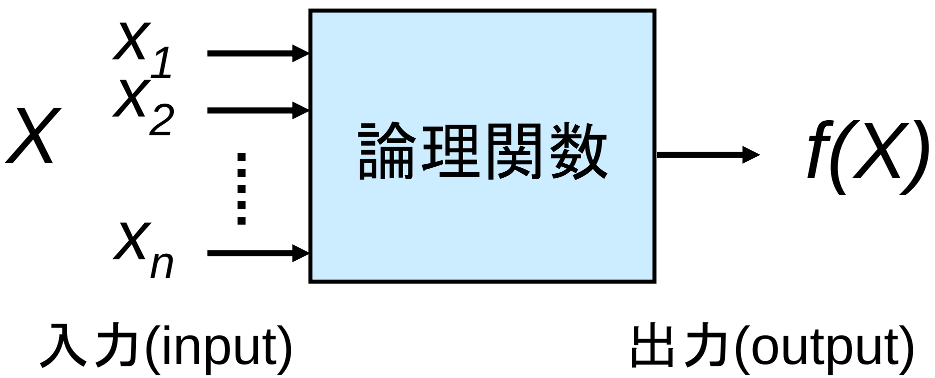
本日（第1回）の内容

論理関数の処理とデータ表現

- 論理関数とは
  - ２値と多値、論理式と論理関数、命題論理と述語論理
- 論理関数の処理とは
  - 論理式/回路からの論理関数データ生成、各種論理演算
  - 恒真/恒偽判定、等価性判定、包含性判定
  - 解探索、最小化
- 論理関数の表現法
  - 求められる性質
  - 関数の総数・情報理論的ビット量・完全ランダム関数
  - カルノー図、真理値表、積和形、BDD

論理関数とは

- 関数(Function)：
  集合A（定義域）から集合B（値域）への写像、かつ
  Aの要素に対してBの要素がただ1つに定まる関係
- 論理関数（Logic Function/Boolean Function）：
  B={0,1}のとき、f: B^n→Bという関数fを論理関数
  （正確には二値論理関数。Switching Functionとも呼ぶ）
- 一般には多値論理関数(Multi-valued Boolean Function)もある。



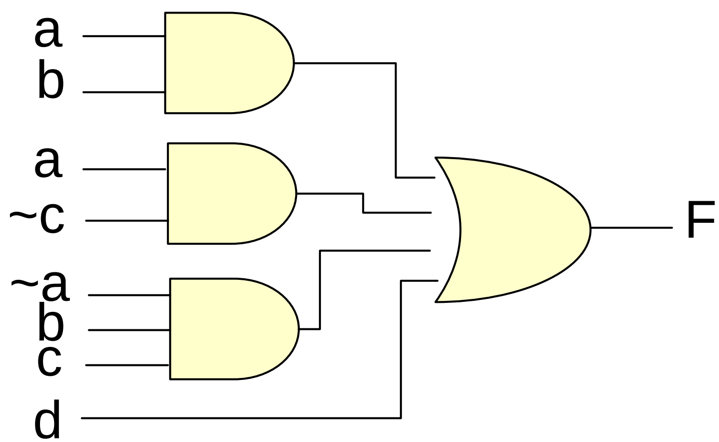
論理式(Boolean Expression)と論理関数

- 論理式（Boolean Expression）: 論理変数と論理演算（AND,OR,NOT等）の組合せで書かれた数式

1つの論理式は1つの論理関数を表現している

(例) F = a b + a~c + ~a b c + d

a	b	c	d	F
1	1	−	−	1
1	−	0	−	1
0	1	1	−	1
−	−	−	1	1



複数の異なる論理式が同じ論理関数を表す場合がある

(例) F1 = x~y + x z + ~x y + ~x~z
F2 = x~y + ~x y + y z + ~y ~z
F3 = x~y + ~x~z + y z

論理関数と命題論理・述語論理

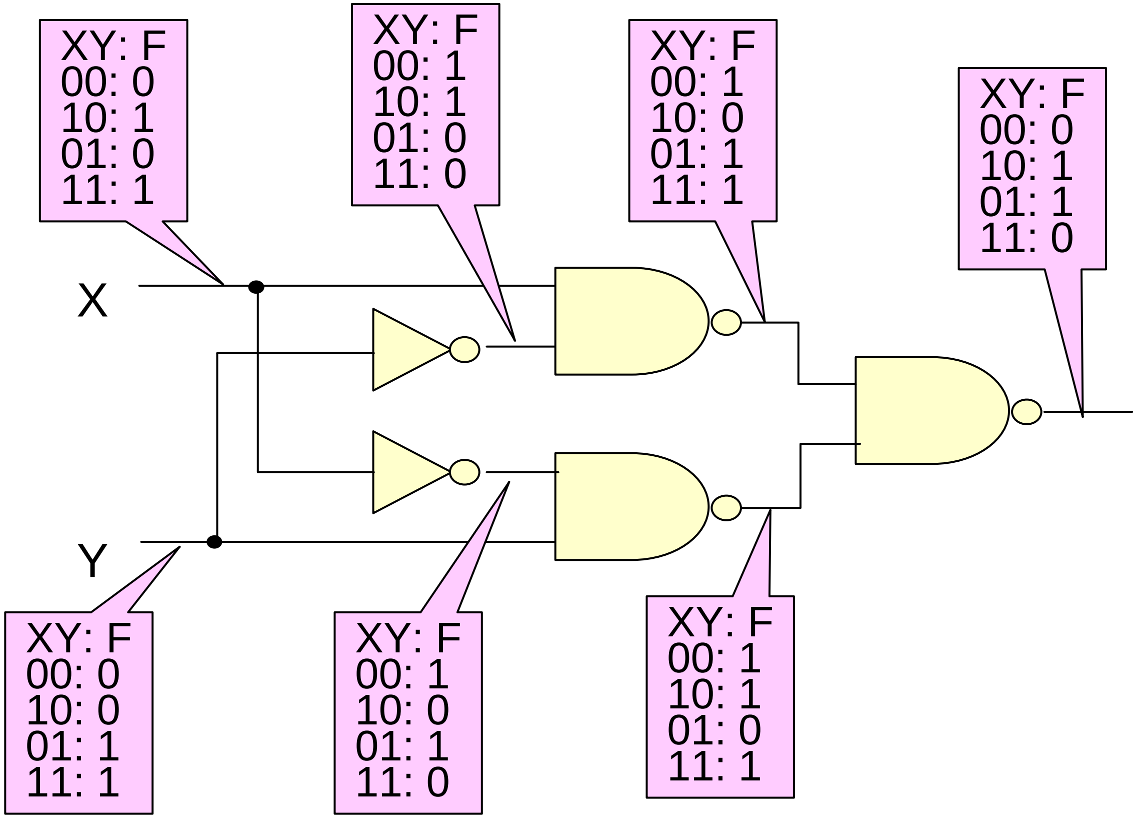
- 命題論理(Propositional Logic):
  真/偽を表す命題(proposition)変数とそれらの論理演算からなる式。
  二値論理関数と等価な概念
  （例） 大学生ならば人間である
  （Student⇒Human) ≡ （~Student + Human）

- 述語論理(Predicate Logic):
  述語(predicate)とそれらの論理演算からなる式。述語は主語に当たる変数（二値とは限らない）を持つ。二値論理関数よりも高度な概念
  （例） Xが大学生ならばXは人間である:
  (Student(X)⇒Human（X）) ≡ (~Student(X) + Human(X))

論理関数の処理とは(１)

現実の様々な情報処理の中で必要となる論理関数の操作

- 論理式/論理回路から論理関数データを作る操作（２つの論理関数データ同士の演算）



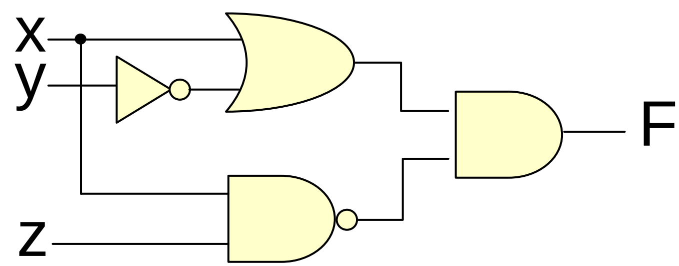
論理関数の処理とは（2）

- 恒真/恒偽判定 （→　co-NP完全問題）
  （例） $x\,y + {\sim}x + {\sim}y\,{\sim}z + z$
- 等価性判定　$(F \equiv G) \Leftrightarrow (F\,G + {\sim}F{\sim}G \equiv 1)$
  $x{\sim}y + x\,z + {\sim}x\,y + {\sim}x{\sim}z$
  $x{\sim}y + {\sim}x\,y + y\,z + {\sim}y\,{\sim}z$
  $x{\sim}y + {\sim}x{\sim}z + y\,z$
- 包含性判定　$(F \Rightarrow G) \Leftrightarrow ({\sim}F + G \equiv 1)$
- 充足解探索:
  – 与えられた論理関数が1になるような変数値の組合せをどれか1つ求める問題 （→　NP完全問題）
  – 最適化問題:充足解の中で最もコストが小さい（または大きい）解を求める問題 （→　NP完全問題）

小規模な論理関数を表現する方法

zw \ xy	00	01	11	10
00	1	1	0	0
01	0	1	1	1
11	1	1	1	1
10	1	0	0	0

カルノー図表現



論理回路図表現

- 人間の目で見通しが良い図的表現が多く使われていた。

計算機上での論理関数表現の要求条件

- 表現がコンパクトであること
  - n入力の論理関数は2^{2^n}通り存在。
    → 固定長データで識別するには
    少なくとも2^nビットは必要。（例：真理値表）
  - 現実には全ての論理関数が均等に現れる訳ではない。
    → よく現れる関数がコンパクトに表現できる
    可変長データが使われる。
- 論理演算処理が高速に行えること
  - 2つの論理関数データの等価性判定が高速に行えるか
  - AND, OR, NOT等の論理演算が効率よく実行できるか
- 人間が見やすいことはあまり重要でない。

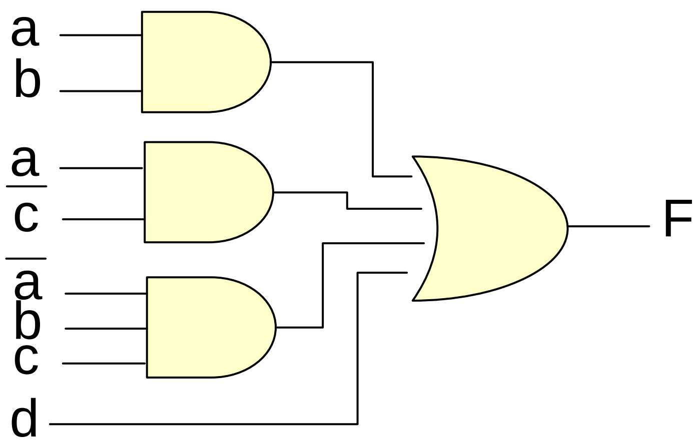
真理値表表現

$x_1x_2x_3 \dots x_n$	F_1	F_2	F_3
000 … 0	1	0	1
100 … 0	0	1	0
010 … 0	1	1	0
110 … 0	0	0	0
001 … 0	1	1	0
101 … 0	0	1	0
011 … 0	1	1	0
⋮	⋮	⋮	⋮
111 … 1	1	0	0

- カルノー図と本質的には同じだが、見やすく並べる必要はない （単純1次元配列）
- ベクトル計算・並列計算に向く
- どんな簡単な論理関数にも、入力数に対して指数の記憶量と時間を要する（30変数程度が限界）
- どんな複雑な（ランダムな）論理関数でも同じ時間で処理できる

積和形論理式

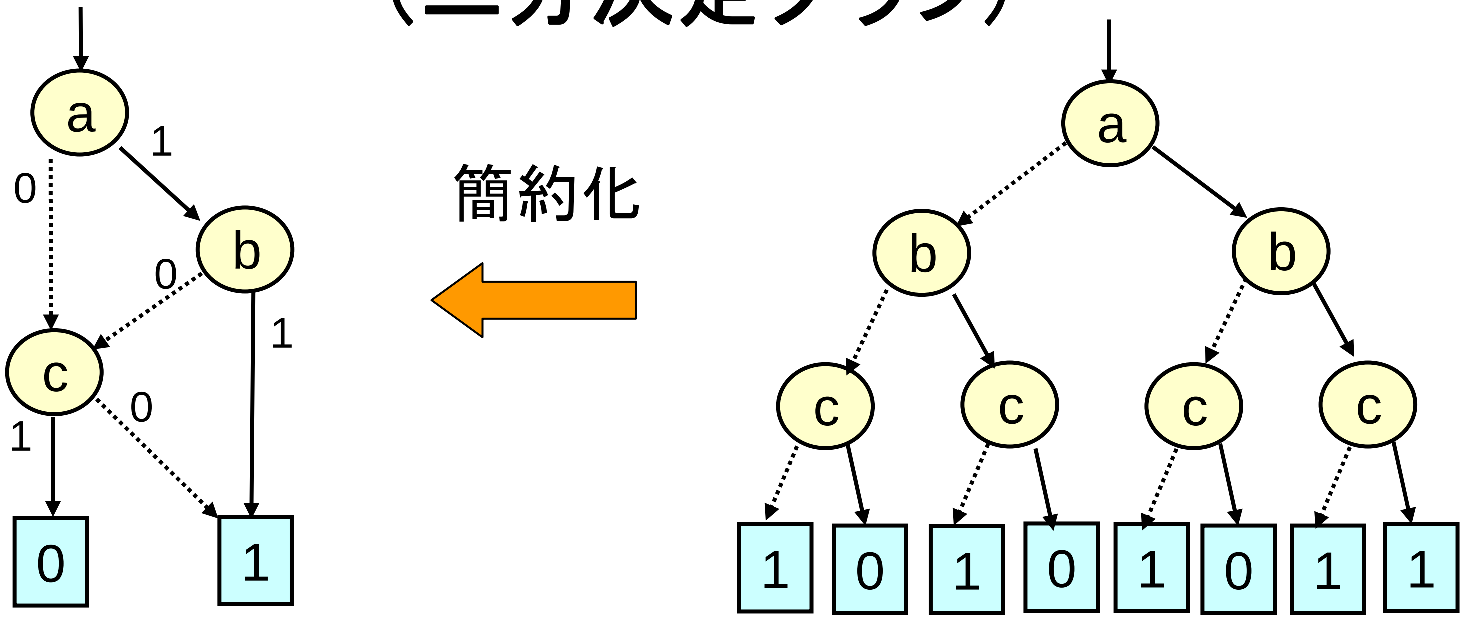
$$ab + a\bar{c} + \bar{a}bc + d$$



a	b	c	d	F
1	1	–	–	1
1	–	0	–	1
0	1	1	–	1
–	–	–	1	1

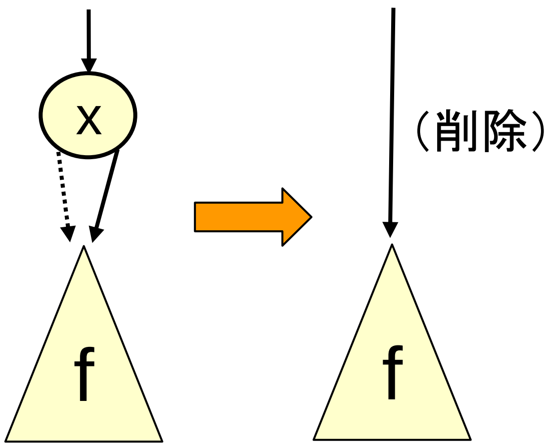
- 肯定･否定の2種類の入力変数を用いた積和2段の論理式で表す方法。
- 記憶量は論理式に現れる文字数にほぼ比例。
- 簡単な式でかける論理関数は効率よく処理できる。
- 表現が一意でなく等価性判定に時間がかかる。
- 否定演算やXOR演算が苦手
- BDDが出現するまでは最も多く使われた（今も使われている）

BDD（Binary Decision Diagram)
（二分決定グラフ）

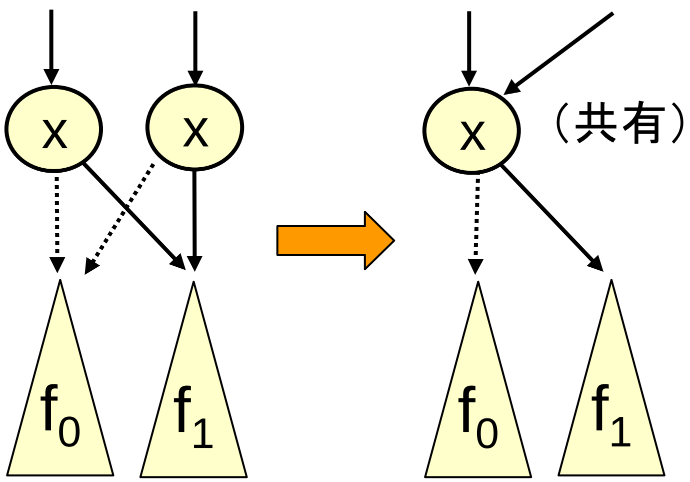


- 二分木状の「場合分けグラフ」によるデータ構造（計算機上ではポインタの配列で表現）
- 1980年代後半から急速に研究が進み実用化

BDDの簡約化規則



- 展開する変数順序を固定
- 冗長な節点を全て削除
- 等価な節点を全て共有



「既約」なBDDが得られる
(Reduced Ordered BDD)

BDDの特徴

- 論理関数に対してグラフの形が一意に定まる。
  - 等価性判定が非常に容易
- 多くの実用的な論理関数がコンパクトに表現できる。
  - パリティ関数や加減算回路も効率よく表現
  - 性質の良い関数では数百入力まで扱える
- 論理関数同士の演算が、グラフのサイズにほぼ比例する計算時間で実行できる。
  - 否定演算も容易
- グラフのサイズが小さくならない場合もある。
  - 乗算回路のBDDは指数サイズ
- 変数の順序づけが悪いとグラフが大きくなる。
  - 比較的良い順序づけを得る方法がいくつか実用化（厳密最小化はNP完全問題）

本日のまとめ

論理関数の処理とデータ表現

- 論理関数とは
  - 2値と多値、論理式と論理関数、命題論理と述語論理
- 論理関数の処理とは
  - 論理式/回路からの論理関数データ生成、各種論理演算
  - 恒真/恒偽判定、等価性判定、包含性判定
  - 解探索、最小化
- 論理関数の表現法
  - 求められる性質
  - 関数の総数・情報理論的ビット量・完全ランダム関数
  - カルノー図、真理値表、積和形、BDD