两级逻辑优化

Two-level Logic Optimization

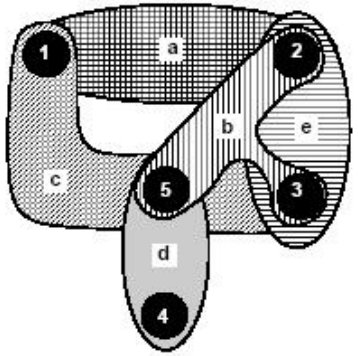
2025-4

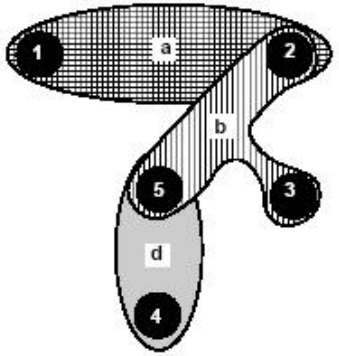
主讲人：孙静翎

覆盖问题

Covering problem

- 集合覆盖问题
  - 一个集合 S （最小项集）
  - 蕴含项集（implicant set）的集合 C
  - 在C 中选择最少的元素来覆盖 S
- 精确解法
  - Branch and bound algorithm 分枝定界算法
- 多种启发式近似方法

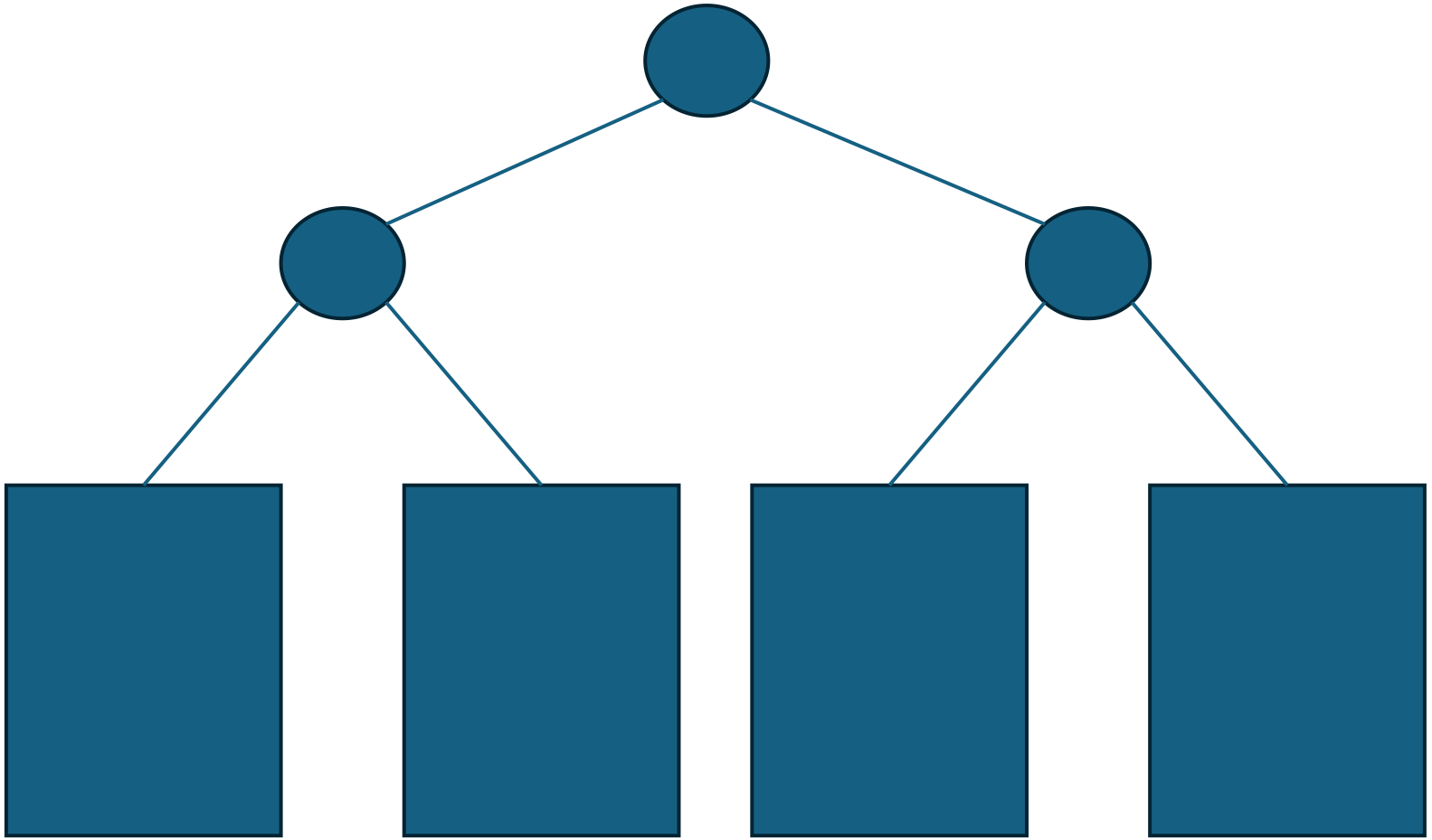




分枝定界算法

Branch and bound algorithm

- 在解决空间树中进行搜索
  - 潜在的指数增长
- 使用界定函数:
  - 如果从一组未来选择中得出的解决方案成本的下限超过了迄今为止看到的最佳解决方案的成本，那么停止该路径的搜索
  - 界定函数必须是可信的并可以快速被验证
- 良好的剪枝可以加快搜索过程



```
EXACT_COVER(A, x, b) {
    如果 current_estimate ⩾ |b|，则返回 b;
    对矩阵 A 进行化简并更新对应的 x;
    如果 A 没有任何行，则返回 x;
    选择一个用于分支的列 c;
    xc = x + 1;
    Ã = 删除列 c 以及与其相关的行后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    xc = x;
    Ã = 删除列 c 后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    返回 b;
}
```

	α(a'b')	β(b'c)	γ(ac)	δ(ab)
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

分枝定界算法

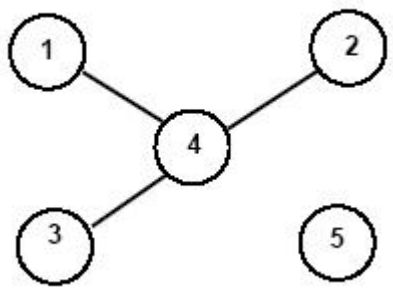
Branch and bound algorithm

- 行的独立关系
  - 行中有1的部分和另一行不重叠，则两行存在独立关系
  - 对每一行都需要独立的蕴含项
- 行的独立图
  - 把每一行看做一个结点
  - 两行如果有独立关系就在独立图中用线连接

例子

- 第四行和1，2，3是独立的

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$



分枝定界算法

Branch and bound algorithm

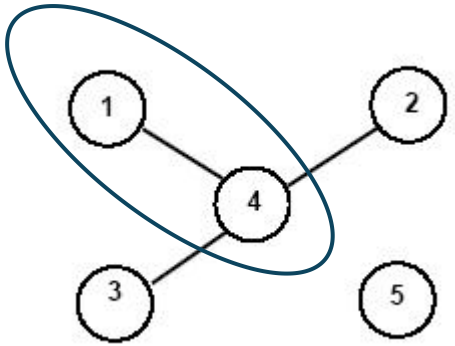
- 行的独立关系
  - 行中有1的部分和另一行不重叠，则两行存在独立关系
  - 对每一行都需要独立的蕴含项
- 行的独立图
  - 把每一行看做一个结点
  - 两行如果有独立关系就在独立图中用线连接
- 找到独立图中尽量大的团
- 团的结点数就是可以接受的近似（下界）

例子

- 第四行和1，2，3是独立的
- 最大的团的节点数是2
- 预测还最少需要的蕴含项是:

  最大团的结点数

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

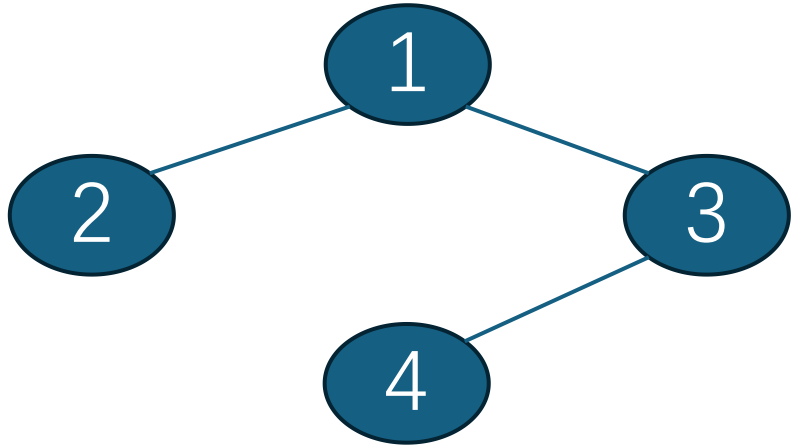


不是作业

	0*00	010*	01*1	0*11	001*
0100	1	1	0	0	0
0111	0	0	1	1	0
0011	0	0	0	1	1
0101	0	1	1	0	0

不是作业

	0*00	010*	01*1	0*11	001*
0100	1	1	0	0	0
0111	0	0	1	1	0
0011	0	0	0	1	1
0101	0	1	1	0	0



```
EXACT_COVER(A, x, b) {
    如果 current_estimate ⩾ |b|，则返回 b;
    对矩阵 A 进行化简并更新对应的 x;
    如果 A 没有任何行，则返回 x;
    选择一个用于分支的列 c;
    xc = x + 1;
    Ã = 删除列 c 以及与其相关的行后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    xc = x;
    Ã = 删除列 c 后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    返回 b;
}
```

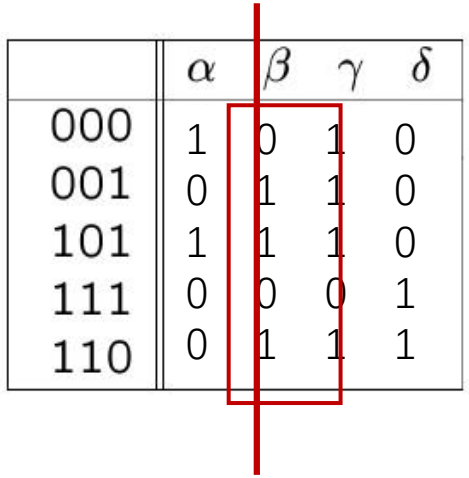
缩减矩阵的方法

- 找到基本要素:
  - 如果一行中只有一列为1
    - 选择该列
    - 从表中移除被覆盖的行

	α	β	γ	δ √
000	1	0	1	0
001	0	1	1	0
101	1	1	1	0
111	1	0	0	1
110	0	0	0	1

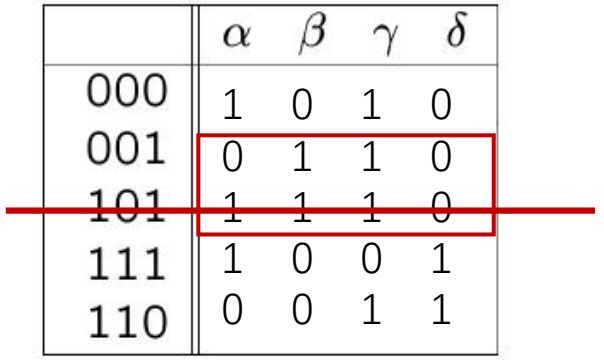
缩减矩阵的方法

- 列（蕴含项）支配：
- 如果对所有 k，都有 $a_{ki} \geqslant a_{kj}$
  - 移除列 j（被支配的）
- 被支配的蕴含项（j）的最小项已经被支配蕴含项（i）覆盖了



	α	β	γ	δ
000	1	0	1	0
001	0	1	1	0
101	1	1	1	0
111	0	0	0	1
110	0	1	1	1

- 行（最小项）支配：
- 如果对所有 k，都有 $a_{ik} \geqslant a_{jk}$
  - 移除行 i（支配的）
- 当一个蕴含项覆盖了被支配的最小项，它也覆盖了支配的最小项



	α	β	γ	δ
000	1	0	1	0
001	0	1	1	0
101	1	1	1	0
111	1	0	0	1
110	0	0	1	1

不是作业

	0*00	010*	01*1	0*11	001*
0000	1	0	0	0	0
0100	1	1	0	0	0
0111	0	0	1	1	0
0011	0	0	0	1	1
0010	0	0	0	0	1
0101	0	1	1	0	0

不是作业

√ √ √

	0*00	010*	01*1	0*11	001*
0000	1	0	0	0	0
0100	1	1	0	0	0
0111	0	0	1	1	0
0011	0	0	0	1	1
0010	0	0	0	0	1
0101	0	1	1	0	0

```
EXACT_COVER(A, x, b) {
    如果 current_estimate ⩾ |b|，则返回 b;
    对矩阵 A 进行化简并更新对应的 x;
    如果 A 没有任何行，则返回 x;
    选择一个用于分支的列 c;
    xc = x + 1;
    Ã = 删除列 c 以及与其相关的行后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    xc = x;
    Ã = 删除列 c 后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    返回 b;
}
```

```
EXACT_COVER(A, x, b) {
    如果 current_estimate ⩾ |b|，则返回 b;
    对矩阵 A 进行化简并更新对应的 x;
    如果 A 没有任何行，则返回 x;
    选择一个用于分支的列 c;
    xc = x + 1;
    Ã = 删除列 c 以及与其相关的行后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    xc = x;
    Ã = 删除列 c 后的 A;
    x~ = EXACT_COVER(Ã, xc, b);
    如果 |x~| < |b|，则
        b = x~;
    返回 b;
}
```

随堂作业

请求出以下布尔函数的最小覆盖：

F=A’B’C’D’+A’BC’D’+A’BC’D+A’BCD+A’B’CD+A’B’CD’

	0*00	010*	01*1	0*11	001*	00*0
0000	1	0	0	0	0	1
0100	1	1	0	0	0	0
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

质蕴含项表

	0*00	010*	01*1	0*11	001*	00*0
0000	1	0	0	0	0	1
0100	1	1	0	0	0	0
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

选中的质蕴含项：
0*00

留下第1列，x=x+1

x=0
b=
∞

→

x=1
b=
∞

√

	0*00	010*	01*1	0*11	001*	00*0
~~0000~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~	~~1~~
~~0100~~	~~1~~	~~1~~	~~0~~	~~0~~	~~0~~	~~0~~
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

选中的质蕴含项：
0*00

第2、6列被支配，删除

x=1
b=
∞

→

x=1
b=
∞

√

	0*00	010*	01*1	0*11	001*	00*0
0000	1	0	0	0	0	1
0100	1	1	0	0	0	0
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

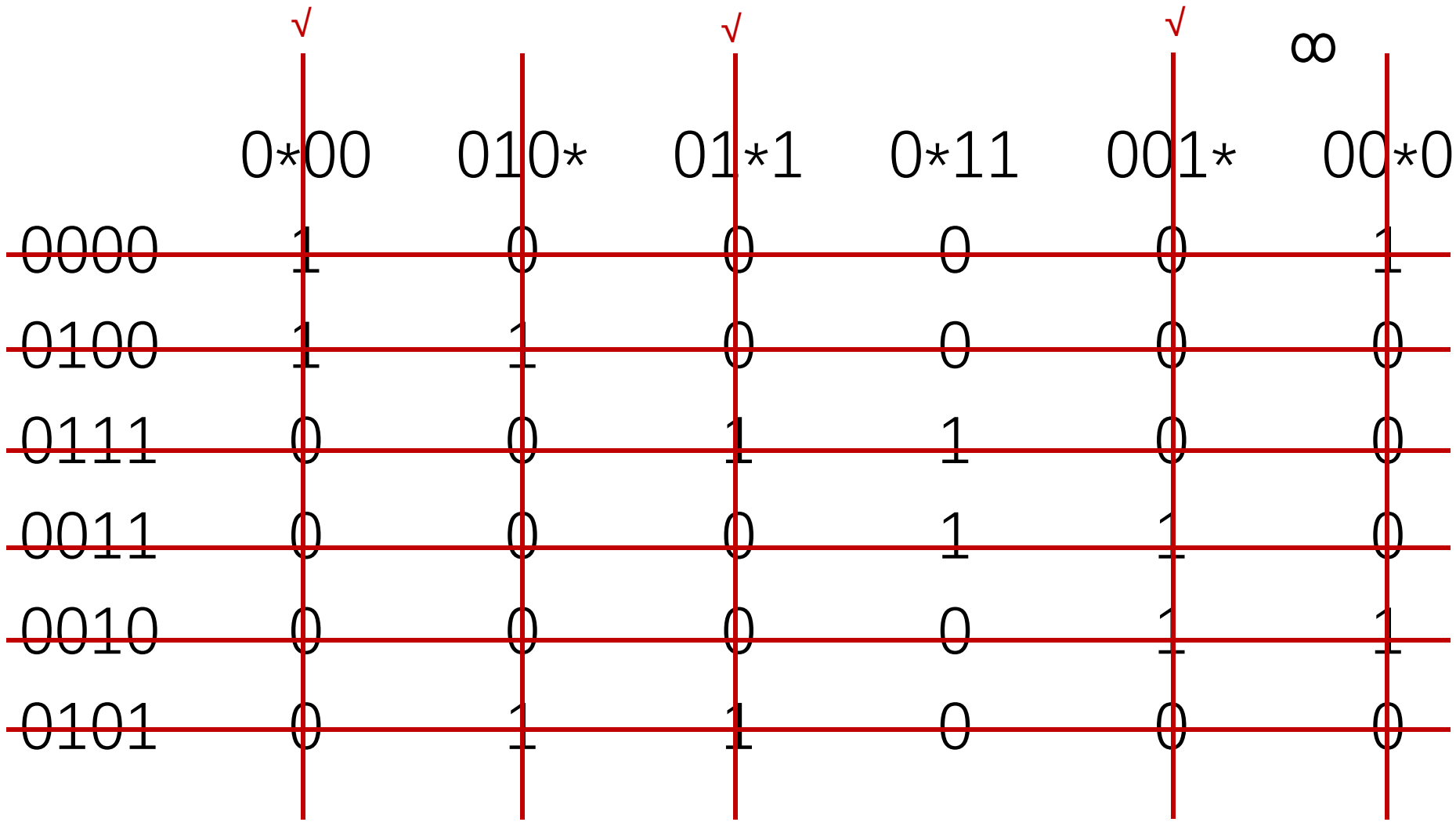
选中的质蕴含项：
0*00
01*1
001*

目前最优解：
0*00
01*1
001*

第3、5列是必要质蕴含项，x=x+2

x=1
b=∞

→ x=3
b=3



选中的质蕴含项：

目前最优解：
0*00
01*1
001*

丢掉第1列

x=0
b=3

→

x=0
b=3

	0*00	010*	01*1	0*11	001*	00*0
0000	1	0	0	0	0	1
0100	1	1	0	0	0	0
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

选中的质蕴含项：

目前最优解：

0*00

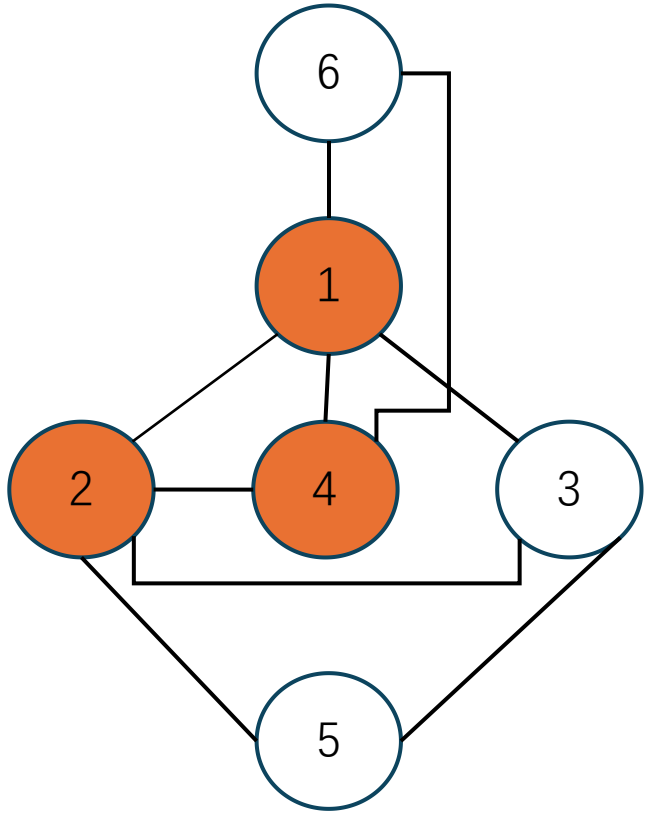
01*1

001*

找到一个团的结点数为3，x=x+3

x=0 b=3 → x=3 b=3

	010*	01*1	0*11	001*	00*0
0000	0	0	0	0	1
0100	1	0	0	0	0
0111	0	1	1	0	0
0011	0	0	1	1	0
0010	0	0	0	1	1
0101	1	1	0	0	0



选中的质蕴含项：

目前最优解：

0*00

01*1

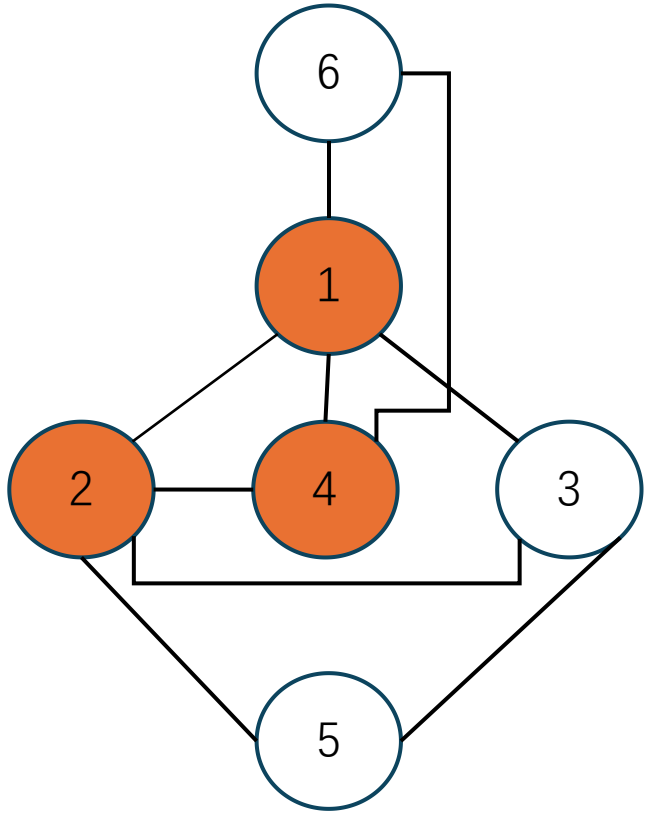
001*

3=目前求出的最优解3，直接结束

x=3

b=3

	010*	01*1	0*11	001*	00*0
0000	0	0	0	0	1
0100	1	0	0	0	0
0111	0	1	1	0	0
0011	0	0	1	1	0
0010	0	0	0	1	1
0101	1	1	0	0	0



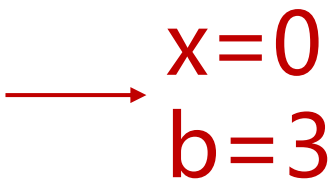
如果你不幸没有找到最大团

选中的质蕴含项：

目前最优解：
0*00
01*1
001*

丢掉第1列

x=0
b=3



	0*00	010*	01*1	0*11	001*	00*0
0000	1	0	0	0	0	1
0100	1	1	0	0	0	0
0111	0	0	1	1	0	0
0011	0	0	0	1	1	0
0010	0	0	0	0	1	1
0101	0	1	1	0	0	0

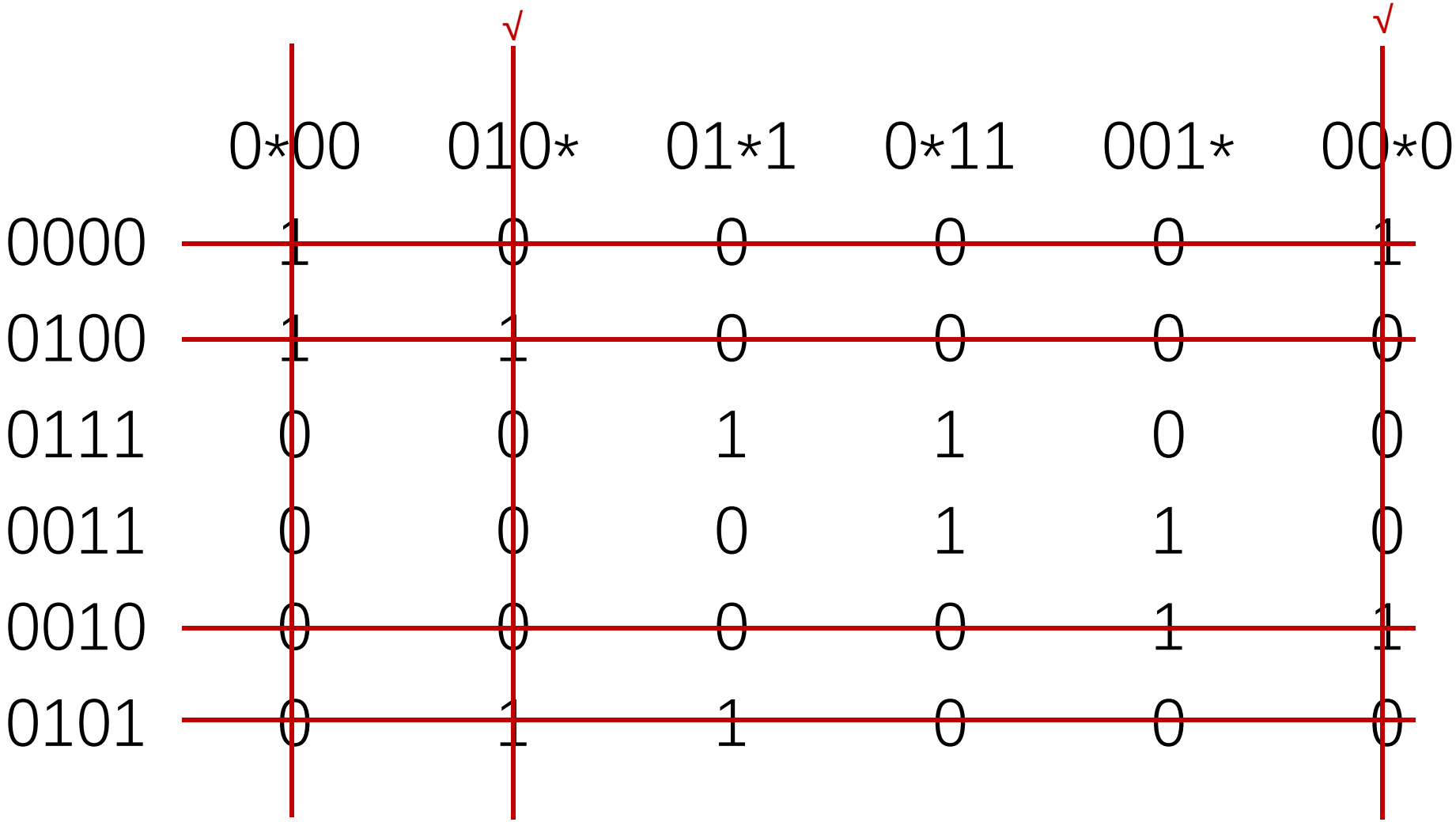
选中的质蕴含项：
010*
00*0
目前最优解：
0*00
01*1
001*

第2、6列是必要质蕴含项，x=x+2

x=0
b=3

→

x=2
b=3



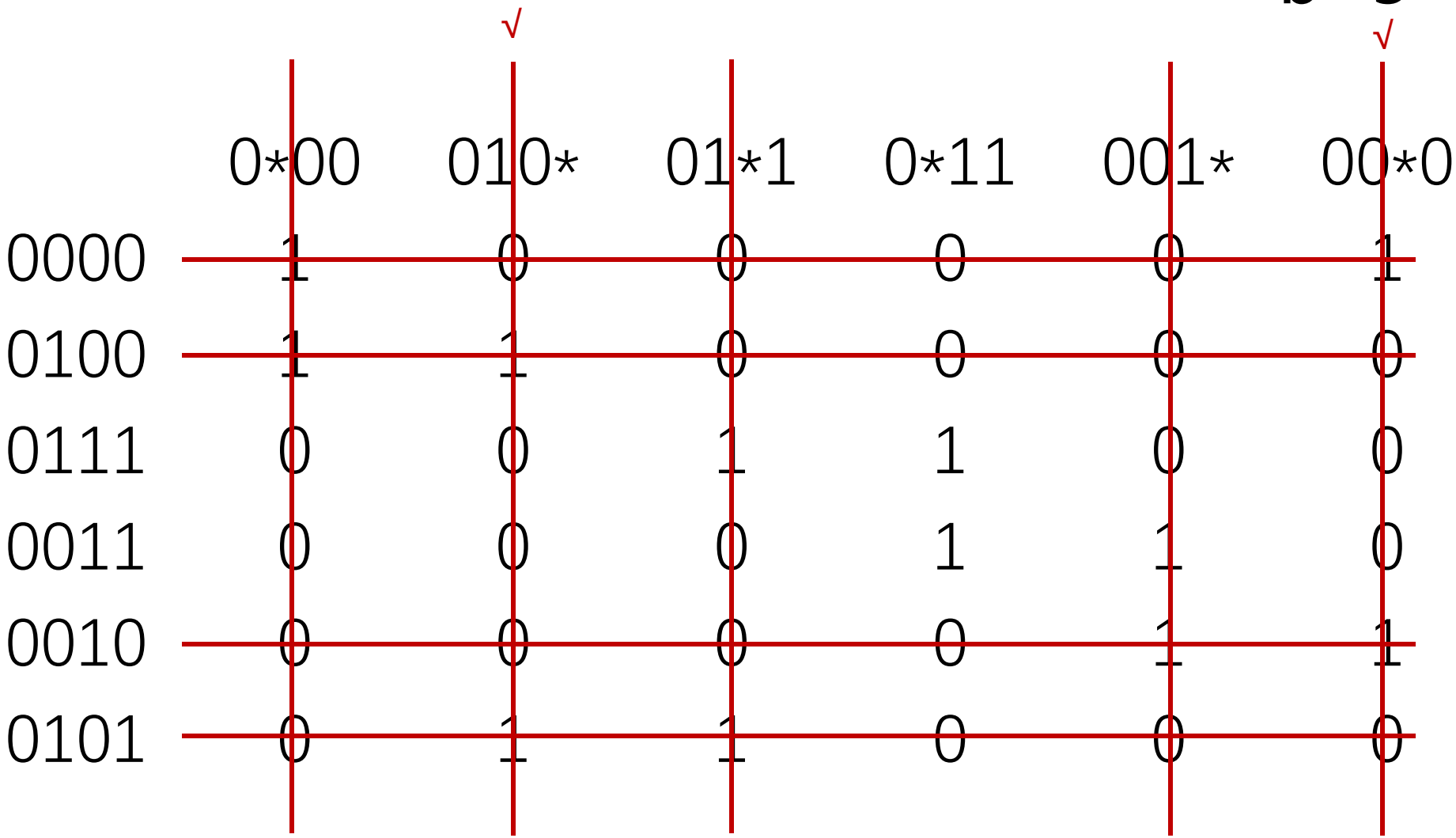
选中的质蕴含项：
010*
00*0
目前最优解：
0*00
01*1
001*

第3、5列被支配，删除

x=2
b=3

→

x=2
b=3



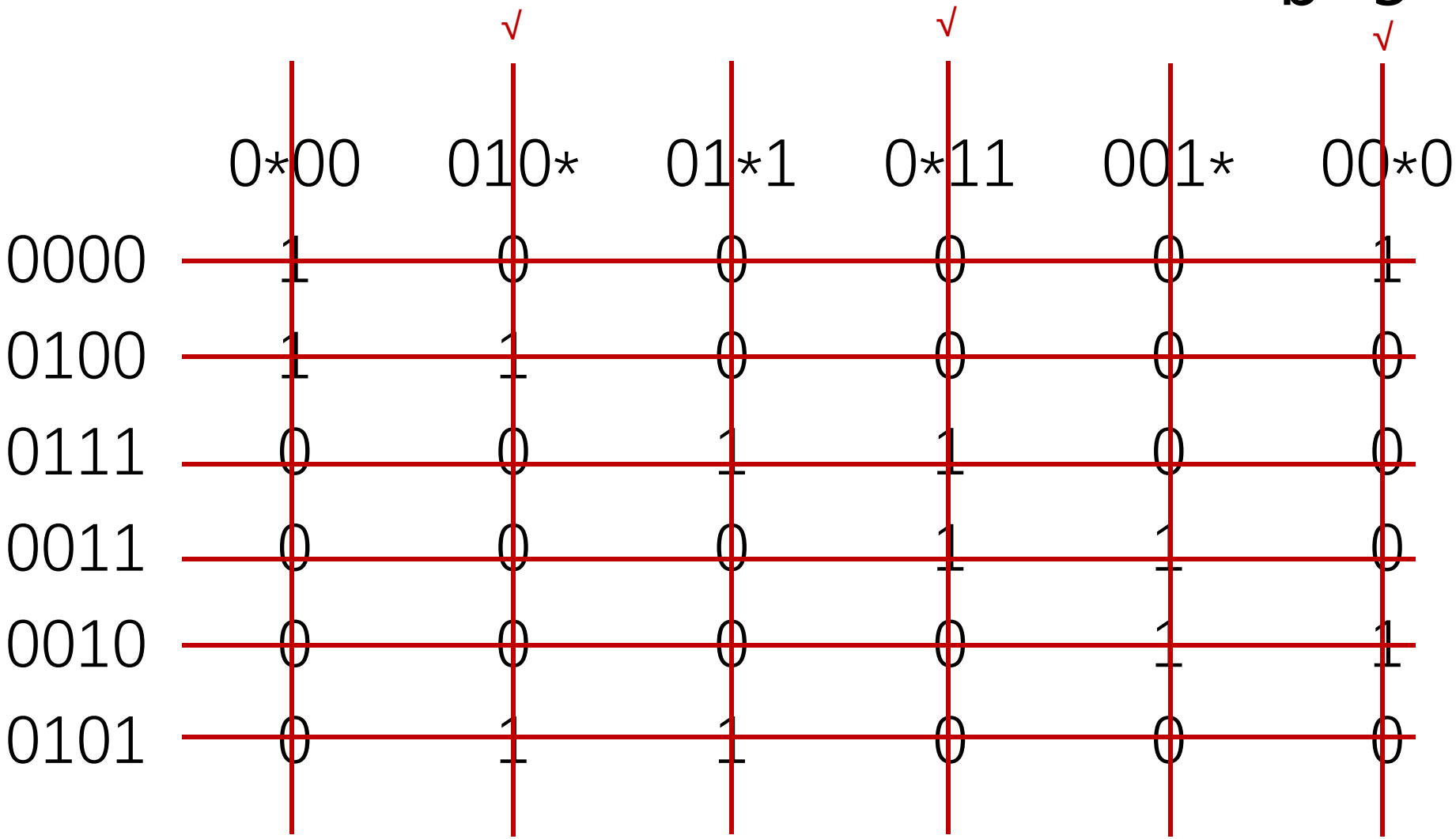
选中的质蕴含项：
010*
00*0
0*11

目前最优解：
0*00
01*1
001*

第4列是必要质蕴含项，x=x+1

x=2
b=3

x=3
b=3



正式开始本周的内容

Espresso-exact

Espresso-exact

- 精确的两级逻辑最小化器
- 利用分支定界算法
- 实现时用到了蕴含项表
- 对于大多数基准都能非常高效地得到最佳解

最低覆盖率的早期方法

Minimum cover early methods

- 简化表格
  - 迭代地识别必要元素，将它们保存在覆盖中。
  - 移除已覆盖的最小项。
- Petrick 方法
  - 以积之和（POS）形式写覆盖子句
  - 将积之和形式展开为和之积（SOP）的形式
  - 选择最小的立方项
  - 注意：展开子句的成本是指数级的

例子

Example

- f = a’b’c’ + a’b’c + ab’c +abc +abc’
- 蕴含项表:

	abc	f
α	00*	1
β	*01	1
γ	1*1	1
δ	11*	1

000: (α)

001: (α + β)

101: (β + γ)

111: (γ + δ)

110: (δ)

- pos形式:
  - (α)(α + β)(β + γ)(γ + δ)(δ) = 1
- 展开为sop形式:

$$(\alpha)(\alpha + \beta)(\beta + \gamma)(\gamma + \delta)(\delta)$$

$$=(\alpha + \alpha\ \beta)(\beta\ \gamma + \beta\ \delta + \gamma + \gamma\ \delta)\ \delta$$

$$=\alpha\ \beta\ \gamma\ \delta + \alpha\ \beta\ \delta + \alpha\ \gamma\ \delta + \alpha\ \gamma\ \delta + \alpha\ \beta\ \gamma\ \delta + \alpha$$

$$\beta\ \delta + \alpha\ \beta\ \gamma\ \delta + \alpha\ \beta\ \gamma\ \delta$$

$$=\alpha\ \beta\ \gamma\ \delta + \alpha\ \beta\ \delta + \alpha\ \gamma\ \delta$$

例子

Example

- f = a’b’c’ + a’b’c + ab’c +abc +abc’
- 蕴含项表:

	abc	f
α	00*	1
β	*01	1
γ	1*1	1
δ	11*	1

000: (α)

001: (α + β)

101: (β + γ)

111: (γ + δ)

110: (δ)

- pos形式:
  - (α)(α + β)(β + γ)(γ + δ)(δ) = 1
- 展开为sop形式:
  - αβγδ+αβδ+αγδ = 1
- 解:

{ α β δ } f = a’b’ + b’c + ab

或{ α γ δ } f = a’b’ + ac + ab

用ILP形式化

- 将表格视为布尔矩阵: A
- 选择一个布尔向量: x
- 确定 x 使得:
  - A * x ⩾ 1
  - 选择足够多的列来覆盖所有行
- 最小化 x 的范数（即最小化x中1的数量）

	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

$$\longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} >= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

矩阵表示

Matrix representation

- 布尔变量:

α、β、γ、δ

- 最小化:

α+β+γ+δ

- 约束:

α*1+β*0+γ*0+δ*0>=1

α*1+β*1+γ*0+δ*0>=1

α*0+β*1+γ*1+δ*0>=1

α*0+β*0+γ*1+δ*1>=1

α*0+β*0+γ*0+δ*1>=1

	α	β	γ	δ
000	1	0	0	0
001	1	1	0	0
101	0	1	1	0
111	0	0	1	1
110	0	0	0	1

$$\longrightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix} >= \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

例子

Example

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix} > = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

两级逻辑优化

Two-level logic minimization

- 精确逻辑最小化:
  - 目标是计算出一个最小覆盖。对于大型函数是困难的
  - Quine-McCluskey方法（奎因-麦克拉斯基算法）
- 启发式逻辑最小化:
  - 致力于在短时间内计算近似的最小覆盖，这通常足够快速但可能不是最优解。
  - MINI, PRESTO, ESPRESSO

余因子

回顾 - BDD

Binary Decision Diagram

一个以顶点v为根的BDD图对应于函数F_v

- 如果v是叶节点：
  - 如果值(v)为1，则$F_v = 1$
  - 如果值(v)为0，则$F_v = 0$
- 如果F是非叶节点（索引 index(v)=i）：

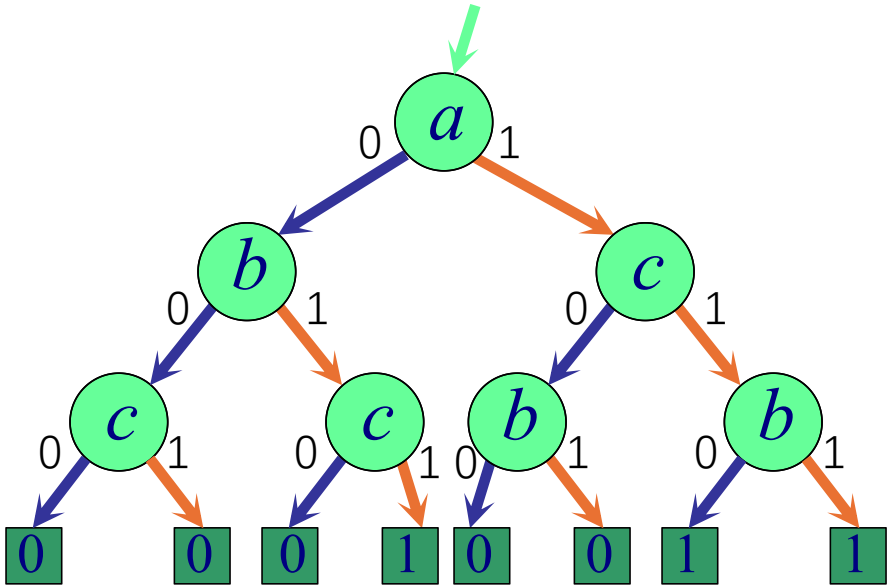
$$F_v(x_1,\ldots,x_n) = x_i' F_{low(v)}(x_{i+1},\ldots,x_n) + x_i' F_h$$

回顾 - BDD例子

Binary Decision Diagram

F= (a+b) c

a	b	c	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



- 每条边代表一个变量的决策
- 函数的值可以在叶子上找到
- 从根到叶的每条路径对应一个 真值表中的行

余因子

Cofactor

- 函数 $f (x_1, x_2, ..., x_i, ..., x_n)$
- f 关于变量 x_i的余因子
  - $f_{xi} = f (x_1, x_2, ..., 1, ..., x_n)$
- f 关于变量 x_i'的余因子
  - $f_{xi'} = f (x_1, x_2, ..., 0, ..., x_n)$

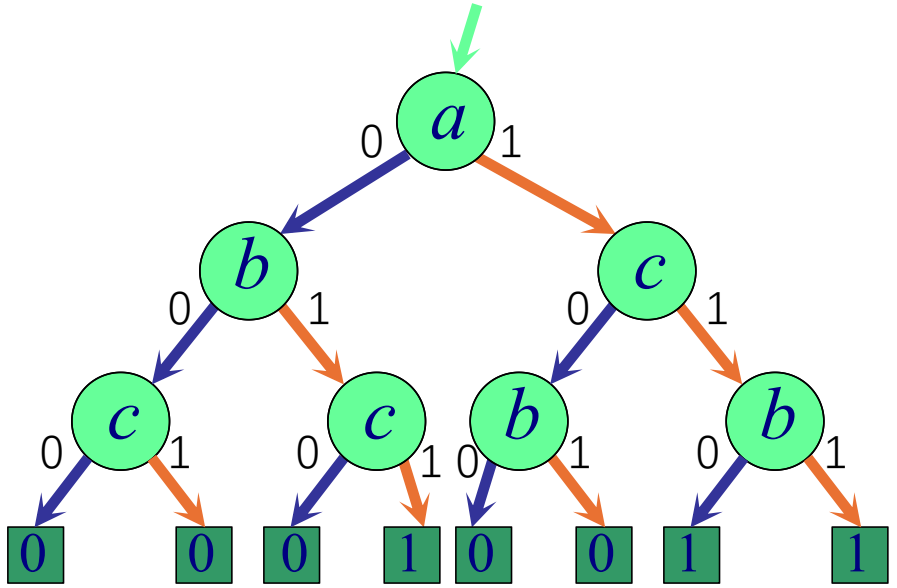
例子

函数: f = (a + b) c

余因子:

f_a = (1 + b) c = c+bc

$f_{a'}$ = (0 + b) c = bc



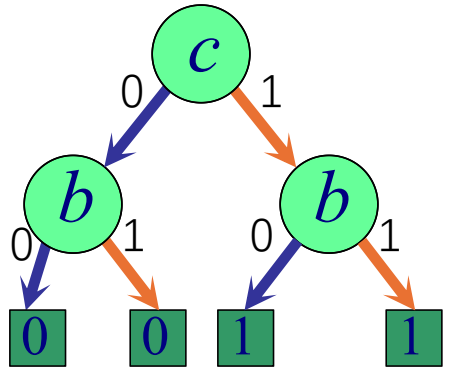
例子

函数: f = (a + b) c

余因子:

f_a = (1 + b) c = c+bc

$f_{a'}$ = (0 + b) c = bc



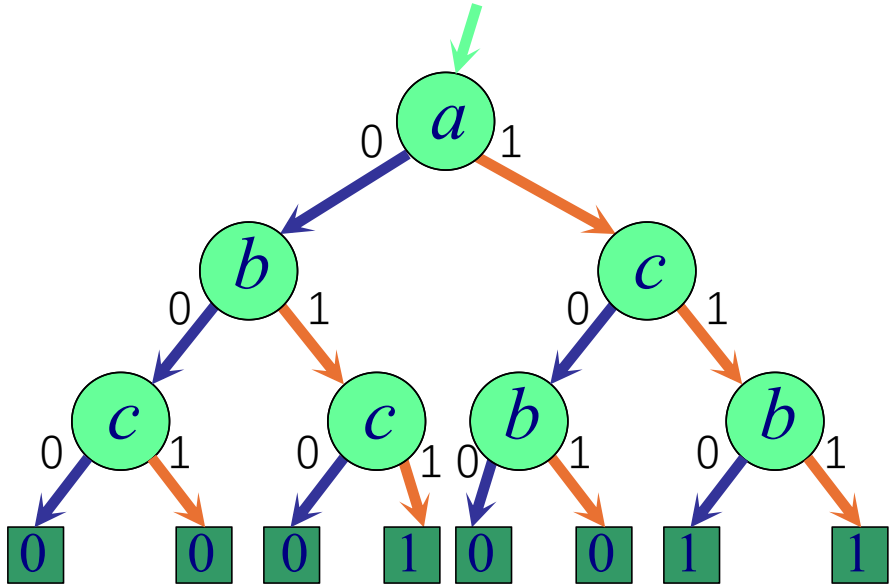
例子

函数: f = (a + b) c

余因子:

f_a = (1 + b) c = c+bc

$f_{a'}$ = (0 + b) c = bc



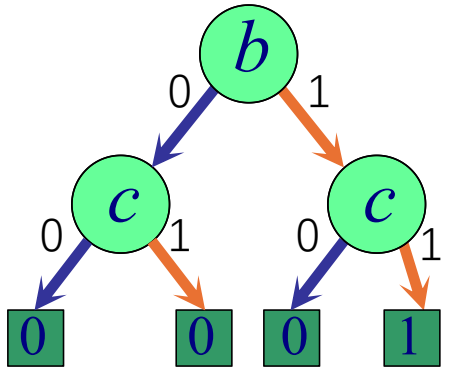
例子

函数: f = (a + b) c

余因子:

f_a = (1 + b) c = c+bc

$f_{a'}$ = (0 + b) c = bc



余因子

Cofactor

- 函数 $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$
- f 关于变量 x_i的余因子
  - $f_{xi} = f(x_1, x_2, \ldots, 1, \ldots, x_n)$
- f 关于变量 x_i'的余因子
  - $f_{xi'} = f(x_1, x_2, \ldots, 0, \ldots, x_n)$
- 布尔展开定理（Boole's expansion theorem）：
  - $f(x_1, x_2, \ldots, x_i, \ldots, x_n) = x_i\, fx_i + x_{i'}\, fx'_i$

例子

函数: f = (a + b) c

余因子:

f_a = (1 + b) c = c+bc

$f_{a'}$ = (0 + b) c = bc

布尔展开定理:

f = a f_a + a'$f_{a'}$

= a(c+bc) + a'(bc)

=ac+abc+a'bc

=ac+(a+a')bc

=ac+bc

=(a+b)c

矩阵表示与运算

逻辑覆盖的矩阵表示

Matrix representation of logic covers

逻辑最小化工具使用的表示方式

不同的格式:

- 通常每个蕴含项一行

符号:

- 0, 1, *, …
- 编码方式:

∅	00
0	10
1	01
*	11

例子

∅	00
0	10
1	01
*	11

- f = a’b + ab’ + ac’d

10 01 11 11
01 10 11 11
01 11 10 01

交集

Intersection

- Intersection（交集）：两个蕴含项的交集是它们共同覆盖的那部分输入组合。

- 例子：

ab’c与b’cd’的交集：ab’cd’

101*　*010　1010

例子 - Intersection：按位与

∅	00
0	10
1	01
*	11

101*

*010

01 10 01 11

11 10 01 10

---

01 10 01 10

1010

例子 - Intersection：按位与

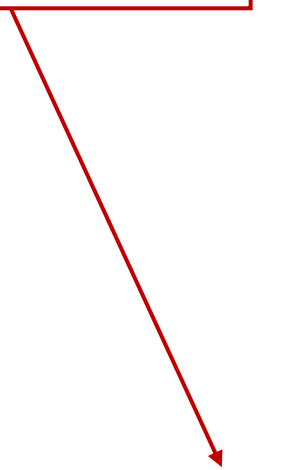
∅	00
0	10
1	01
*	11

a’c　　10 11 01

ab　　01 01 11

---

00 01 01



void

最小公共上界

Supercube

- Supercube（最小公共上界）：两个蕴含项的最小公共上界是能同时覆盖两个立方体所有输入组合的最小蕴含项。

- 例子：

ab’c与b’cd’的最小公共上界：b’c

101*　*010　*01*

例子 - Supercube：按位或

∅	00
0	10
1	01
*	11

101*

*010

01 10 01 11

11 10 01 10

11 10 01 11

01

例子 - Supercube：按位或

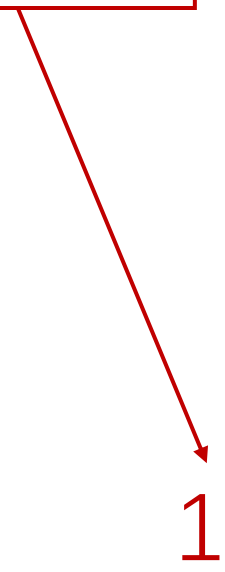
∅	00
0	10
1	01
*	11

a’ 10 11

ab 01 01

---

11 11



1

余因子

Cofactor

- 函数 f 关于 蕴含项φ 的余因子为：在满足 φ 条件的情况下，f 的行为。

- 例子：

  f = (a + b) c

  f_a = c+bc

  $f_{a'}$ = bc

例子 – 取余因子

- f = ac + bc
- ϕ = a
- f_a = c+bc

01 11 01
11 01 01
01 11 11

---

01 11 01
01 01 01
10 00 00

---

11 11 01
11 01 01

例子 – 取余因子

- f = ac + bc
- ϕ = a’
- $f_{a'}$ = bc

01 11 01
11 01 01
10 11 11

~~00 11 01~~ void
10 01 01
01 00 00
---
11 01 01

例子 – 取余因子

- f = ab + ac + ab'c' + a'
- ϕ = ab
- f_ϕ = 1

01 01 11
01 11 01
01 10 10
10 11 11
01 01 11

---

01 01 11
01 01 01
01 00 10 void
00 01 11 void
10 10 00

---

11 11 11
11 11 01

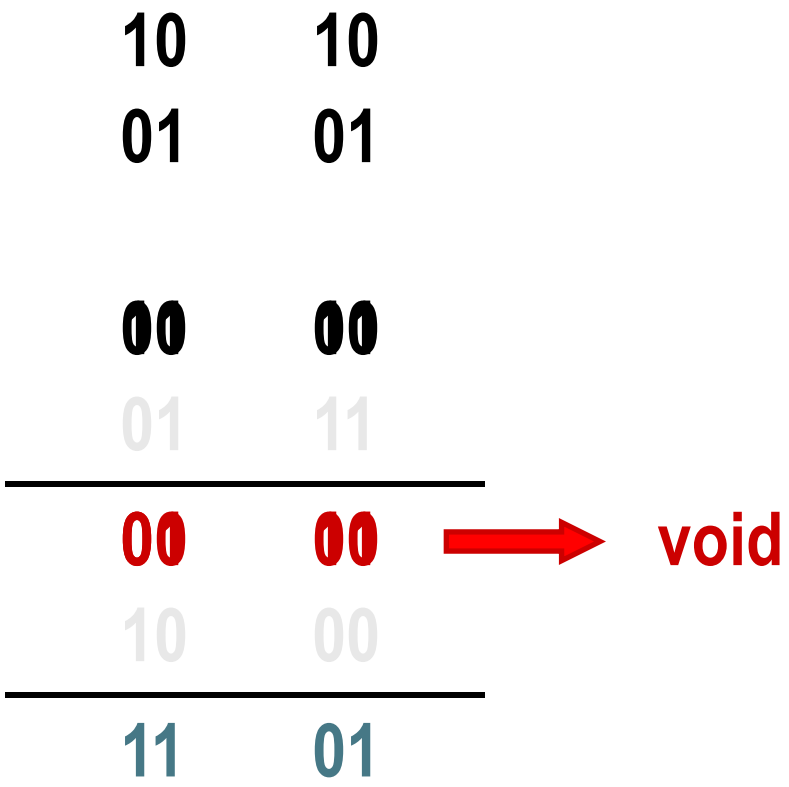
不是作业

∅	00
0	10
1	01
*	11

- f = a'b' + ab
- ϕ = a
- 请用矩阵表示法求f_ϕ

例子 – 取余因子

- f = a’b’ + ab
- f对a (01 11) 取余因子:
  - 第一行 – void
  - 第二行 – 11 01
- 余因子 f_ϕ = b



矩阵表示法的优势

Advantages of positional cube notation

使用二进制值:

- 每个符号使用两位
- 比一个字节（字符）更高效

可以应用二进制操作:

- Intersection：按位与（AND）
- Supercube：按位或（OR）
- 余因子: 先按位与（AND）再和全反按位或（OR）
- 二进制操作非常快速，并且可以并行化

布尔运算操作

- 重言式判断
- 包含判断
- 取补

重言式判断

Tautology Judgment

- 重言式：
  - 若覆盖矩阵中存在全为1的一行，则为重言式
  - 若函数仅依赖一个变量，且在该变量域中无全为0的列，则为重言式
- 非重言式：
  - 若函数仅依赖一个变量，且在该变量域中有全为0的列，则函数不是重言式

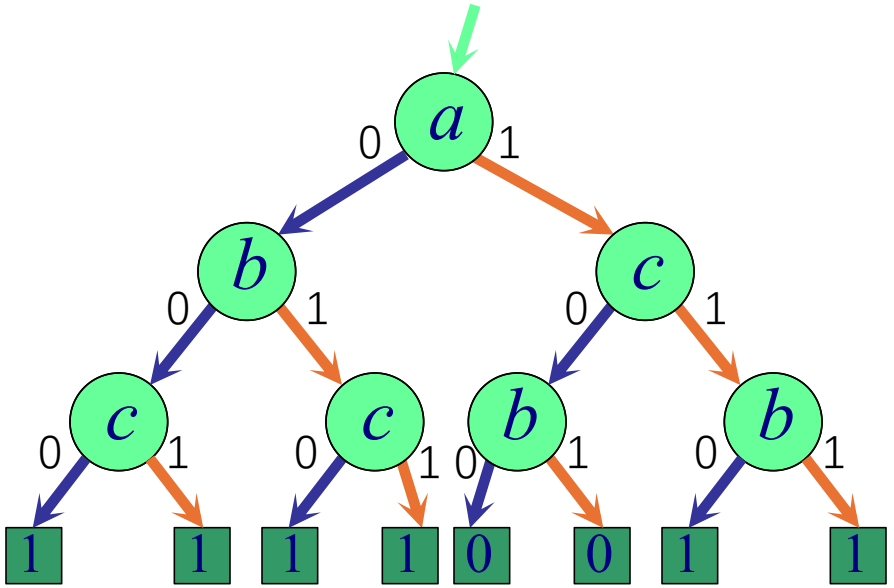
重言式判断

Tautology Judgment

- 判断一个函数是否输出恒为1
- 递归范式:
  - 围绕某个变量展开
  - 若所有余因子（cofactors）都恒为真，则该函数是重言式

重言式判断

Tautology Judgment



例子

$f = ab + ac + ab'c' + a'$

- 选择变量a
- f对a’取余因子:

  11 11 11 – 重言式.
- f对a取余因子:

  11 01 11

  11 11 01

  11 10 10

01 01 11
01 11 01
01 10 10
10 11 11
00 11 11

---

01 01 11
01 11 01
01 10 10
00 11 11

00 00 00

---

11 01 11
11 11 01
11 10 10

例子

f = ab + ac + ab′ c′ + a′

- 选择变量b
- f_a对b’取余因子:

11 11	01
11 11	10

- 没有一列全为0- Tautology
- f_a对b取余因子:

有一行全为1

- 函数是*重言式*

11	01	11
11	11	01
11	10	10
11	00	11
11	00	11
11	00	01
11	00	10
00	00	00
11	11	01
11	11	00

随堂作业

请用矩阵表示法计算以下布尔函数是否是重言式:

$$f = x'y+y'z+z'x$$

- 重言式:
  - 若覆盖矩阵中存在全为1的一行，则为重言式
  - 若函数仅依赖一个变量，且在该变量域中无全为0的列，则为重言式
- 非重言式:
  - 若函数仅依赖一个变量，且在该变量域中有全为0的列，则函数不是重言式