两级逻辑优化

Two-level Logic Optimization

2025-4

主讲人：孙静翎

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

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）:
  - $f(x_1, x_2, \ldots, x_i, \ldots, x_n) = x_i fx_i + x_{i'} fx'_i$

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

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

- 判断一个函数是否输出恒为真（即恒为1）
- 递归范式:
  - 围绕某个变量展开
    - 若所有余因子都恒为真，则该函数是重言式
    - 若确定一个余因子不恒为真，则该函数不是重言式

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

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

•起始：

f = x′y+y′z+z'x

10 01 11

11 10 01

01 11 10

•对x取余因子：01 11 11

~~00 01 11~~ void

01 10 01

01 11 10

10 00 00

11 10 01

11 11 10

•对y取余因子：11 10 01

11 11 10

11 01 11

~~11 00 01~~ void

11 01 10

00 10 00

11 11 10

函数仅依赖一个变量，且在该变量域中有全为0的列，不为重言式。

=》确定一个余因子不恒为真，则该函数不是重言式

正式开始本周的内容

单调性

Unateness

- 函数 $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$
- 当以下式子成立时在变量x_i上具有正单调性:
  - $f_{xi} \geq f_{xi'}$
- 当以下式子成立时在变量x_i上具有负单调性:
  - $f_{xi} \leq f_{xi'}$

例子

函数: f = a + b + c

余子式:

$f_a = 1$

$f_{a'} = b + c$

b	c	f_a	$f_{a'}$
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

单调性

Unateness

- 函数 $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$
- 当以下式子成立时在变量x_i上具有正单调性:
  - $f_{xi} \geq f_{xi'}$
- 当以下式子成立时在变量x_i上具有负单调性:
  - $f_{xi} \leq f_{xi'}$
- 当一个函数在其所有变量上都是单调递增/递减时，它就是一个正/负单调函数

双相性

Binate

- 函数 f (x_1, x_2, …., x_i, …., x_n)
- 若在变量 x_i 上，f 既不是单调递增，也不是单调递减，则称函数在该变量上具有双相性（Binate）

例子

函数: $f = a' c + ab$

余子式:

$f_a = b$

$f_{a'} = c$

b	c	f_a	$f_{a'}$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	1

简单的单调性判断方法

- 如果某个变量 x 在所有蕴含项中：
  - 只以原变量（x）形式出现，说明是正单调变量；
  - 只以反变量（x'）形式出现，说明是负单调变量；
  - 如果同时出现 x 和 x’，说明是双相变量。

例子

函数: f = c+b+a’+ab’

启发式策略-重言式

Heuristics

- 如果函数 f 在变量 x 上是正单调的，则$f_{x'} \leq f_x$

=> 若 $f_{x'}$是重言式，则 f_x也一定是重言式

- 如果函数 f 在变量 x 上是负单调的，则$f_x \leq f_{x'}$

⇒若 f_x 是重言式，则 $f_{x'}$ 也一定是重言式

例子

函数: f = c+b+a′+ab′

b	c	$f_{c'}$	f_c
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	1

布尔运算操作

- 重言式判断
- 包含判断
- 取补

包含判断

Containment Judgment

- 定理：

  当且仅当 F_ϕ 是一个重言式时，函数F包含蕴含项ϕ。

- 推论：

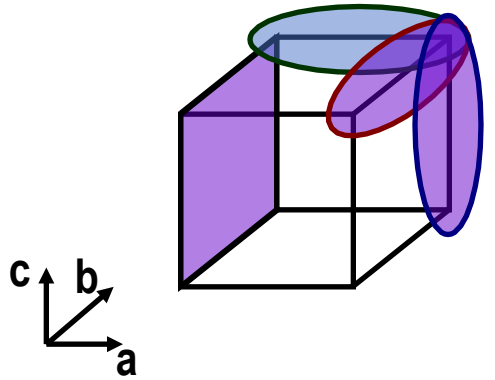
  F是否包含ϕ可以通过重言式判断法来验证。

例子

f = ab + ac + a′

- 检查ϕ= bc （11 01 01）是否被f包含
- f对ϕ取余因子:

01	11	11
01	11	11
10	11	11



- 是重言式 —— 因此ϕ 被 f 包含.

不是作业

请用矩阵表示法计算 xy′ 是否被以下布尔函数包含：

$$F = xz + x'y + y'z'$$

- 当且仅当 F_ϕ 是一个重言式时，函数F包含函数ϕ。

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

•起始：

F = xz + x'y + y'z'

•对xy’取余因子：

01 11 01

10 01 11

11 10 10

01 10 11

---

01 10 01

~~00 00 11~~ void

01 10 10

10 01 00

---

11 11 01

11 11 10

- 函数仅依赖一个变量，且在该变量域中无全为0的列，为重言式。
- 当且仅当 Fα 是一个重言式时， F 包含一个蕴含项 α。

=> F 包含xy’

布尔运算操作

- 重言式判断
- 包含判断
- 取补

余因子

Cofactor

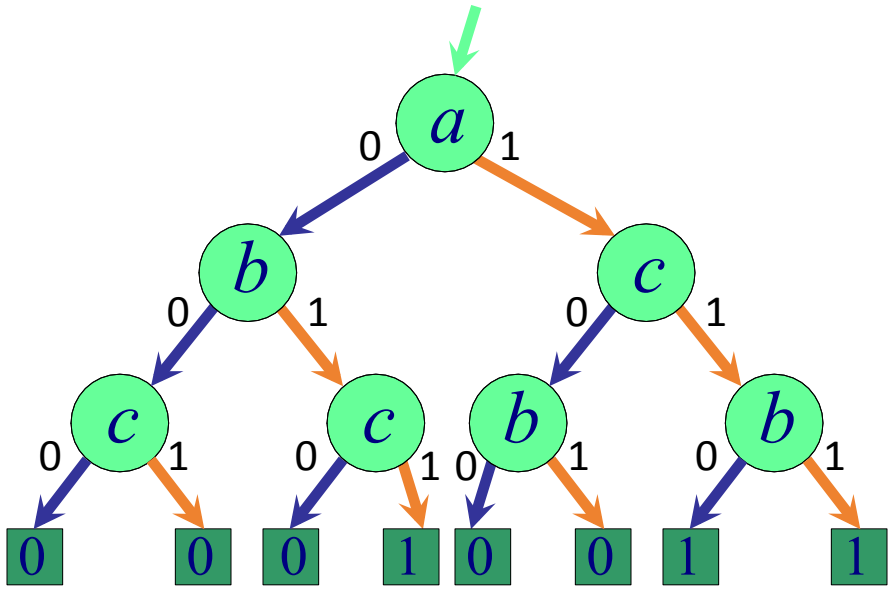
- 函数 $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$
- f 关于变量 x_i的余因子
  - $f_{xi} = f(x_1, x_2, \ldots, 1, \ldots, x_n)$
- f 关于变量 x_i'的余因子
  - $f_{xi'} = f(x_1, x_2, \ldots, 0, \ldots, x_n)$
- 布尔展开定理（Boole's expansion theorem）：
  - $f(x_1, x_2, \ldots, x_i, \ldots, x_n) = x_i fx_i + x_{i'} fx'_i$

取补

Complementation

递归范式：

$$f' = x\, f'_x + x'\, f'_{x'}$$

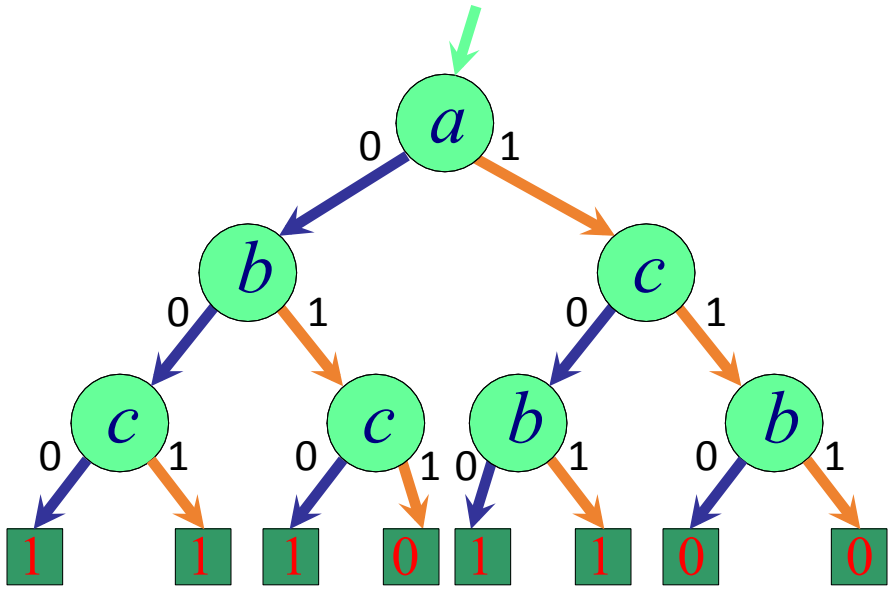


取补

Complementation

递归范式：

$$f' = x\, f'_x + x'\, f'_{x'}$$



取补

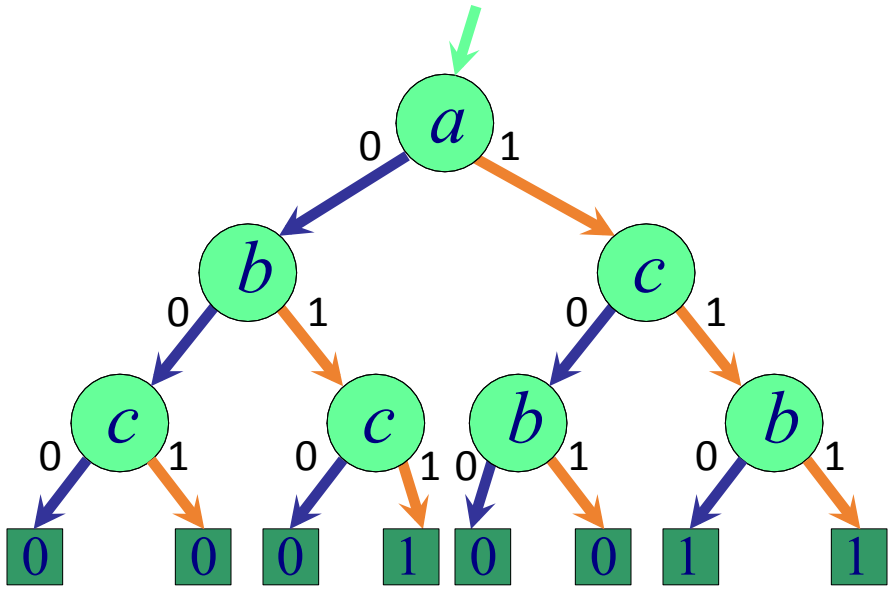
Complementation

函数: $f = ac + bc$

余因子:

$f_a = (1 + b)\, c = c$

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



取补

Complementation

函数: f = ac + bc

f’=(a’+c’)(b’+c’)

=a’b’+b’c’+a’c’+c’

=a’b’+c’

余因子:

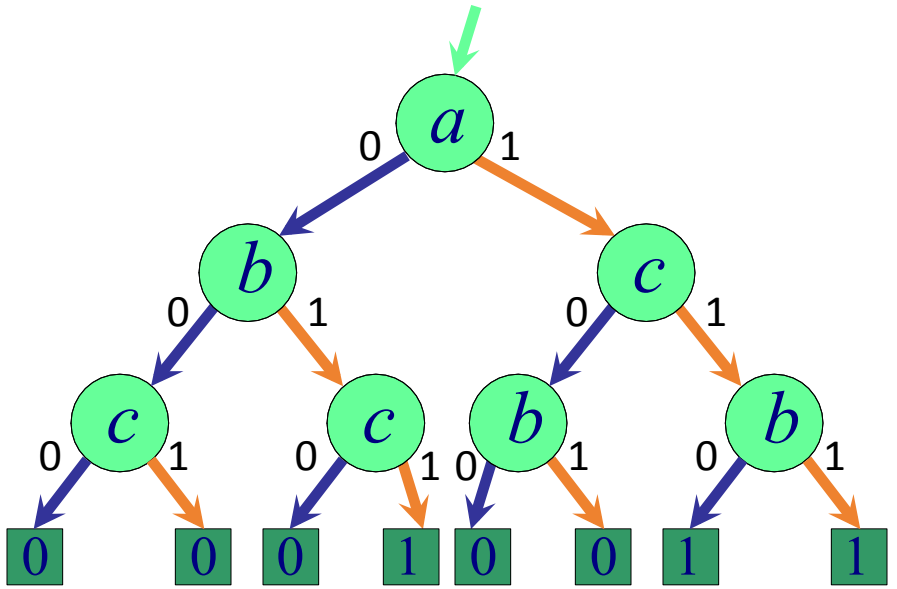
$f'_a = c'$

$f'_{a'} = (bc)' = b'+c'$

f’= x f’x + x’f’x’

= ac’+a’b’+a’c’

= a’b’+c’



取补

Complementation

递归范式：

$f' = x f'_x + x' f'_{x'}$

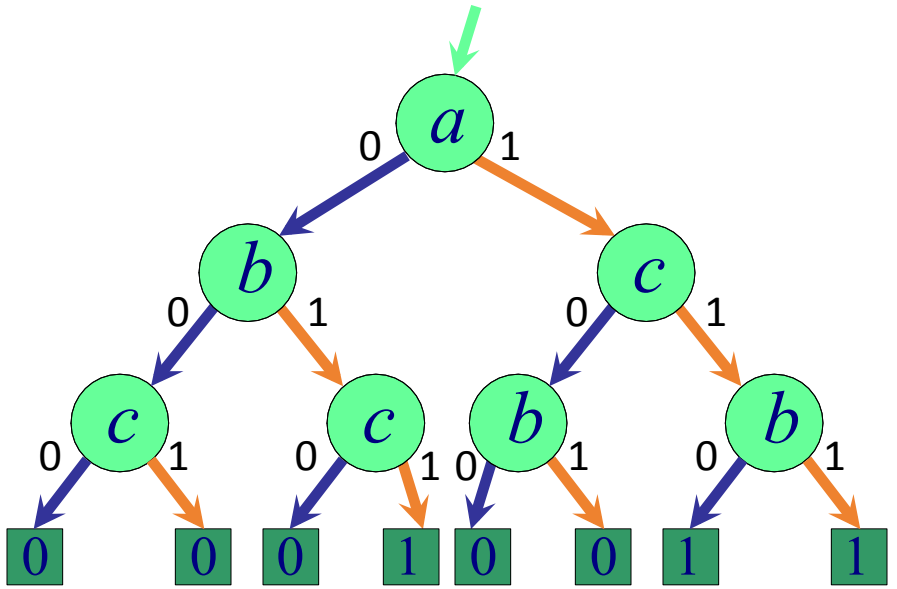
步骤：

1.选择变量

2.计算余因子

3.对余因子取补

4.递归，直到余因子可以直接确定其补值为止



取补-特殊情况

- F 为 void（即0）：
  - 补集为重言式（即1）
- F是重言式（即1）：
  - 补集为void（即0）
- F 仅包含一个蕴含项：
  - 补集可通过德摩根定律计算

$$\neg(P \wedge Q) \equiv (\neg P) \vee (\neg Q)$$

取补

Complementation

递归范式：

$f' = x f'_x + x' f'_{x'}$

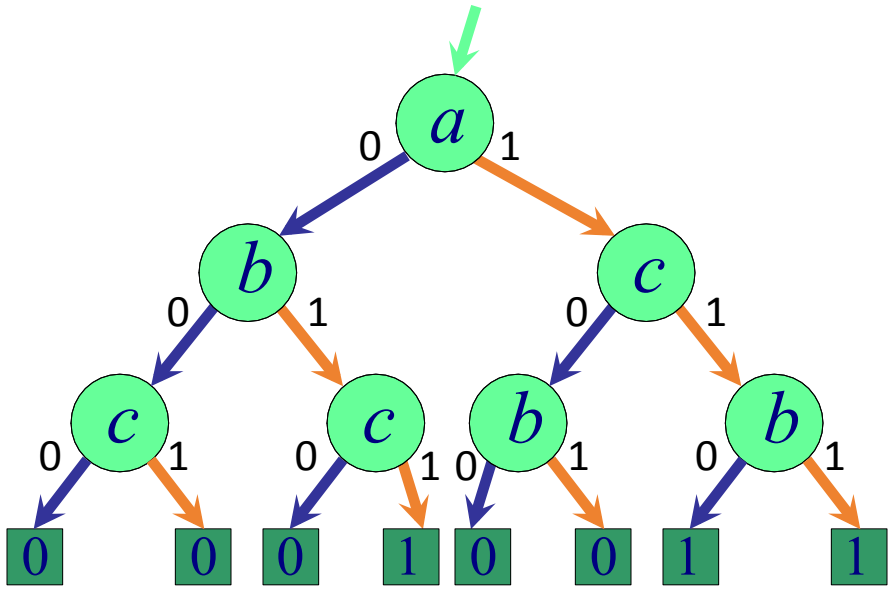
步骤：

1.选择变量

2.计算余因子

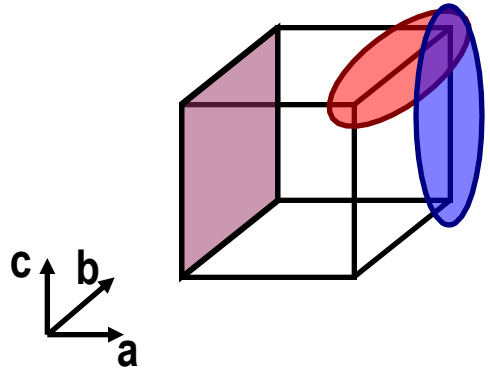
3.对余因子取补

4.递归，直到余因子可以直接确定其补值为止



例子

F = ab + ac + a′



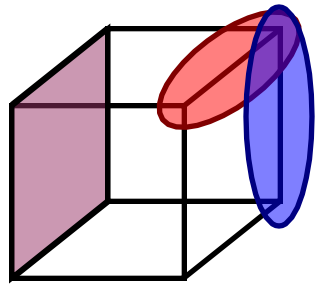
- 选择变量a
- f对a计算余因子:
  - $F_{a'}$ 是重言式，因此 $F'_{a'}$ 是 void.
  - F_a 为:

    11 01 11
    11 11 01

  - $F' = a F'_a + a' F'_{a'} = a F'_a + 0 = a F'_a$
  - 为方便，以下用Q代替F_a ,即F’=aQ’

例子

F = ab + ac + a′

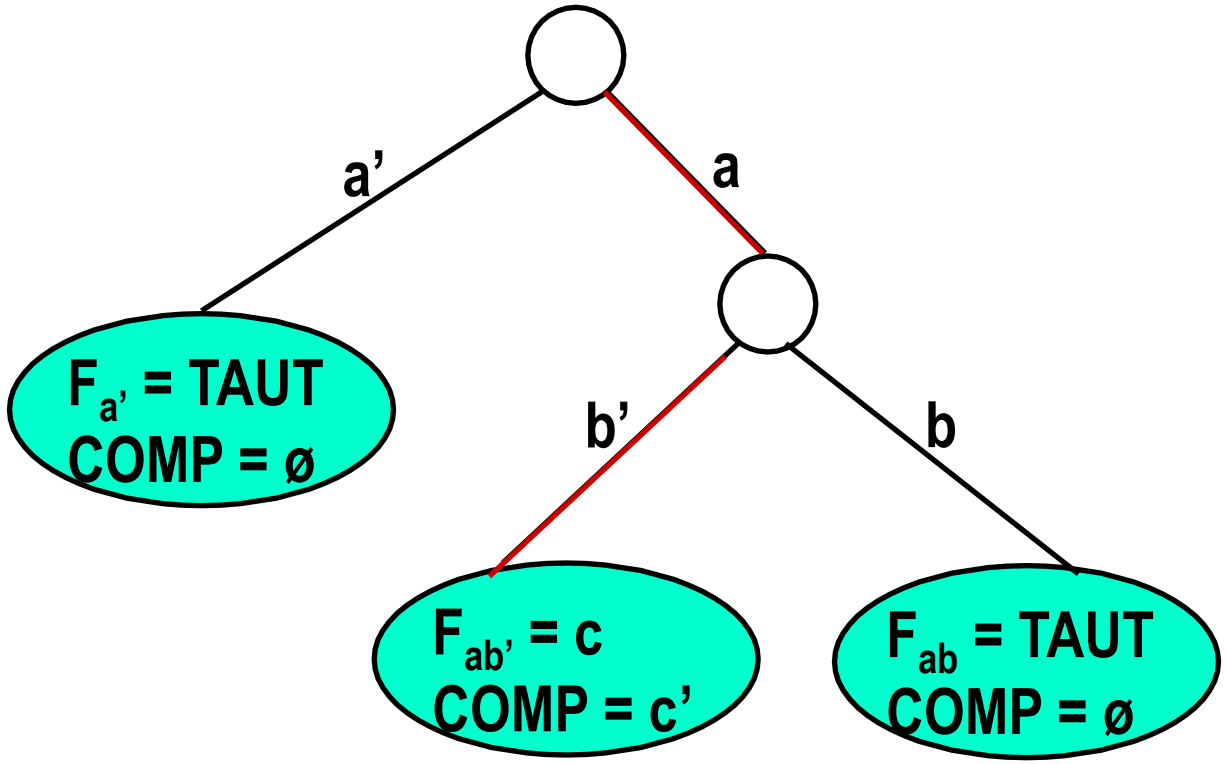


- 选择变量 b
- Q（即F_a）对b计算余因子:
  - Q_b 是重言式, 因此 Q'_b 是 void
  - $Q_{b'}$ = 11 11 01 , 因此 $Q_{b'}$ 是 11 11 10
- 合并结果:
  - $Q' = b\ Q'_b + b'\ Q'_{b'} = 0 + b'\ Q'_{b'} = 11\ 10\ 10$
  - F' = a Q' = 01 10 10
- 最终结果: F' = 01 10 10

例子

f = ab + ac + a′

- 递归搜索:



取补结果: a b’c’

不是作业

请用矩阵表示法计算以下布尔函数的补：

$$F = a + bc$$

- $f' = x\, f'_x + x'\, f'_{x'}$
- F 为 void ： 补集为重言式（即1）
- F是重言式：补集为void（即0）
- F 仅包含一个蕴含项 ：补集可通过德摩根定律计算

•起始：

$F = a + bc$

01 11 11

11 01 01

F_a ： 01 11 11

01 11 11

01 01 01

10 00 00

11 11 11

11 01 01

$F_{a'}$ ： 01 11 11

11 01 01

10 11 11

00 11 11 void

10 01 01

01 00 00

11 01 01

•F 仅包含一个蕴含项 ：补集可通过德摩根定律计算

-> $F'_{a'} = b'+c'$

11 10 11

11 11 10

覆盖矩阵中存在全为1的一行，为重言式

$F_a = 1$ -> $F'_a = 0$

•aF'_a

00 00 00

01 11 11

---

00 00 00

•$a'F'_{a'}$

11 10 11

11 11 10

10 11 11

---

10 10 11

10 11 10

•$F' = a\,F'_a + a'F'_{a'}$

10 10 11

10 11 10

启发式策略-求补

定理：

- 如果函数 f 在变量 x 上是正单调的（positive unate），则 $f' = f'_x + x' f'_{x'}$
- 如果函数 f 在变量 x 上是负单调的（negative unate），则 $f' = x f'_x + f'_{x'}$

推论：

- 选择有单调性的变量求补时，合并结果会更简单

启发式策略-重言式

Heuristics

- 如果函数 f 在变量 x 上是正单调的，则$f_{x'} \leq f_x$

启发式策略-重言式

Heuristics

- 如果函数 f 在变量 x 上是正单调的，则$f_{x'} \leq f_x$

$\Rightarrow f'_{x'} \geq f'_x$

例子

函数: f = a + b + c

余子式:

$f_a = 1$

$f_{a'} = b + c$

b	c	f_a	$f_{a'}$
0	0	1	0
0	1	1	1
1	0	1	1
1	1	1	1

例子

函数: f = a + b + c

余子式:

$f_a = 1$

$f_{a'} = b + c$

b	c	f'_a	$f'_{a'}$
0	0	0	1
0	1	0	0
1	0	0	0
1	1	0	0

启发式策略-重言式

Heuristics

- 如果函数 f 在变量 x 上是正单调的，则$f_{x'} \le f_x$

$\Rightarrow f'_x \le f'_{x'}$

$f' = xf'_x + x' f'_{x'}$

$= xf'_x + x'(f'_x + f'_{x'})$

$= xf'_x + x'f'_x + x' f'_{x'}$

$= (x+x')f'_x + x' f'_{x'}$

$= f'_x + x' f'_{x'}$

启发式策略-求补

定理：

- 如果函数 f 在变量 x 上是正单调的（positive unate），则
  $f' = f'_x + x' f'_{x'}$
- 如果函数 f 在变量 x 上是负单调的（negative unate），则
  $f' = x f'_x + f'_{x'}$

推论：

- 选择有单调性的变量求补时，合并结果会更简单

例子

请用矩阵表示法计算以下布尔函数的补：

$$F = a + bc$$

- aF'_a

00 00 00

01 11 11

---

00 00 00

- $a'F'_{a'}$

11 10 11

11 11 10

10 11 11

---

10 10 11

10 11 10

- $F' = a F'_a + a'F'_{a'}$

10 10 11

10 11 10

- F'_a

00 00 00

- $a'F'_{a'}$

11 10 11

11 11 10

10 11 11

---

10 10 11

10 11 10

- $F' = F'_a + a'F'_{a'}$

10 10 11

10 11 10

启发式两级逻辑最小化

定义

- 最小覆盖（Minimum cover）：
  - 具有最少蕴含项（implications）的函数覆盖
  - 全局最优
- 非冗余覆盖（Minimal cover or irredundant cover）：
  - 不完全是另一个函数覆盖的超集
  - 不能删除任何蕴含项
  - 局部最优
- 单个蕴含项下的非冗余覆盖（Minimal w.r.t. single-implicant containment）：
  - 没有蕴含项包含另一个蕴含项
  - 弱局部最优

启发式逻辑最小化

Heuristic logic minimization

- 获得“相对较小” 的非冗余覆盖
- 避免精确最小化中的瓶颈问题
  - 存储问题
  - 速度问题
- 常见的应用场景
  - 用作多级综合工具中的一部分

启发式最小化的基本原则

Heuristic minimization -- principles

- 从初始覆盖出发
  - 由设计者提供，或从硬件语言模型中提取
- 对当前覆盖进行修改
  - 使其成为质覆盖、无冗余覆盖
  - 重复迭代，直到获得较小的无冗余覆盖
- 通常覆盖的规模会减少
  - 对小规模覆盖的操作速度快


SALT
SOY SAUCE
VINEGAR

启发式最小化的操作

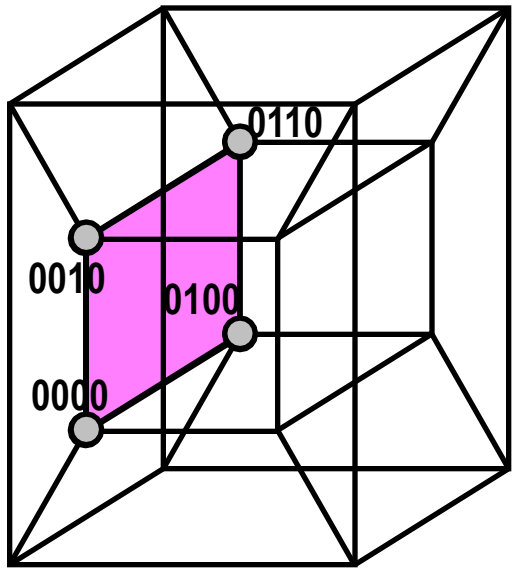
Heuristic minimization - operators

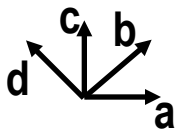
- 扩展（Expand）
  - 使蕴含项成为质蕴含项并移除被覆盖的蕴含项
- 缩减（Reduce）
  - 在保持覆盖范围不变的前提下，缩小每个蕴含项的大小
- 重构（Reshape）
  - 同时修改两个蕴含项：放大一个，缩小另一个
- 去冗余（Irredundant）
  - 移除覆盖中的冗余蕴含项

扩展的例子

F=a’b’c’d’+a’b’cd’+a’b’cd’+a’bcd’

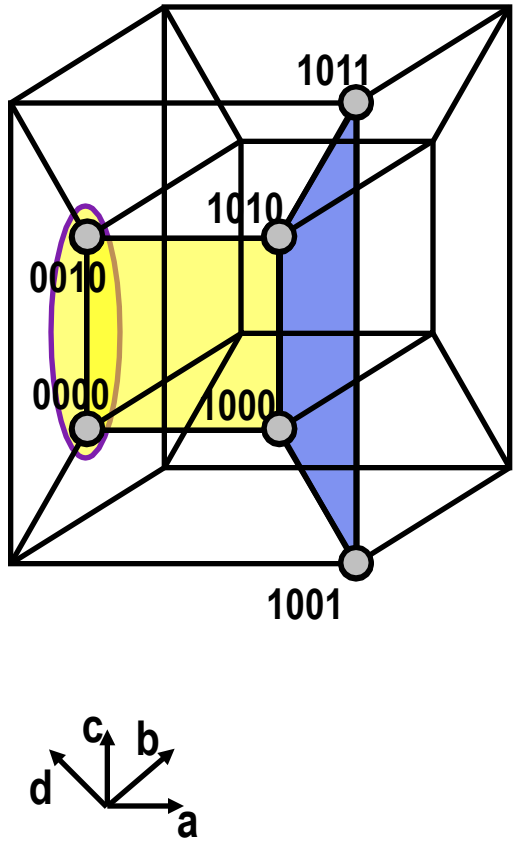
- Expand 0000 to α = 0**0.
  - Drop 0100, 0010, 0110 from the cover.





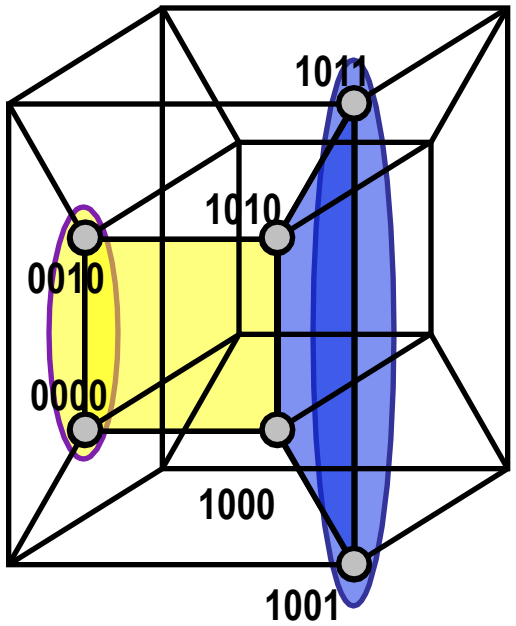
缩减的例子

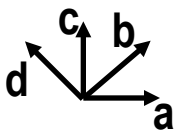
- {*0*0, 10**}
- Reduce β = *0*0 to β' = 00*0.



重构的例子

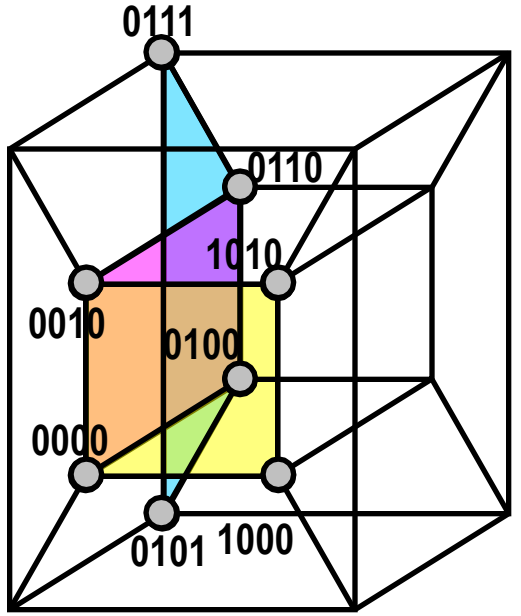
- {β'= 00*0, δ =10**}
- Reshape {β', δ} to: {β, δ'}.
  - Where δ' = 10*1, β = *0*0

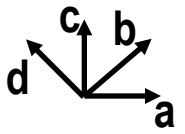




去冗余的例子

- 覆盖: {α=0**0, β=*0*0, γ=01**}.
- 其中α是冗余的
- 去除后得到覆盖：{β, γ}.





启发式最小化的操作

Heuristic minimization - operators

- 扩展（Expand）
  - 使蕴含项成为质蕴含项并移除被覆盖的蕴含项
- 缩减（Reduce）
  - 在保持覆盖范围不变的前提下，缩小每个蕴含项的大小
- 重构（Reshape）
  - 同时修改两个蕴含项：放大一个，缩小另一个
- 去冗余（Irredundant）
  - 移除覆盖中的冗余蕴含项

启发式最小化的操作

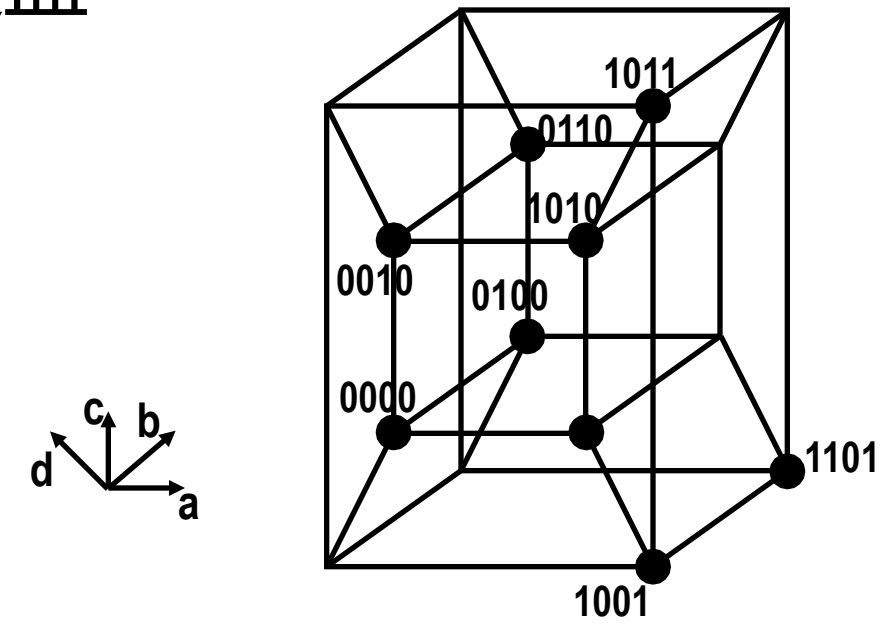
Heuristic minimization - operators

- MINI
  - 迭代执行 EXPAND、 REDUCE、 RESHAPE
  - MINI 只能保证最小的单个蕴含项上的非冗余覆盖
- Espresso
  - 迭代执行 EXPAND、 IRREDUNDANT、 REDUCE
  - Espresso 能保证无冗余覆盖，因为它包含了去冗余操作

例子

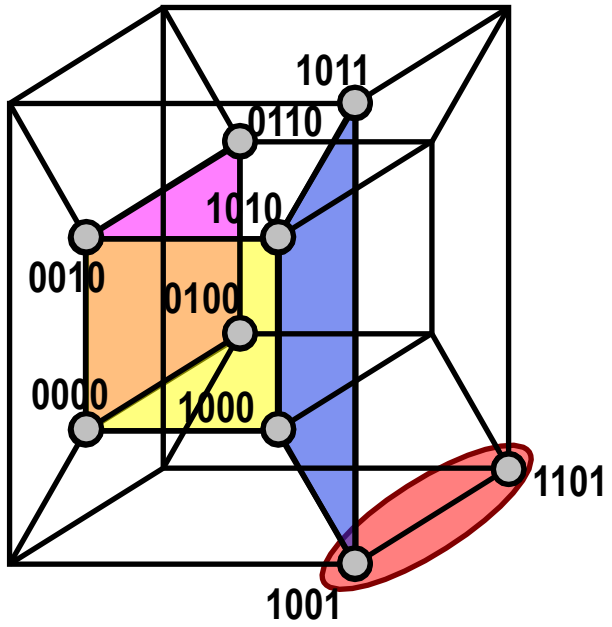
F=a’b’c’d’+a’b’cd’+a’bc’d’+a’bcd’+ab’c’d’+ab’cd’+ab’c’d+ab’cd+abc’d

- 初始覆盖

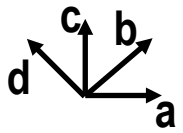


MINI的操作步骤-例子

- 1.扩展:

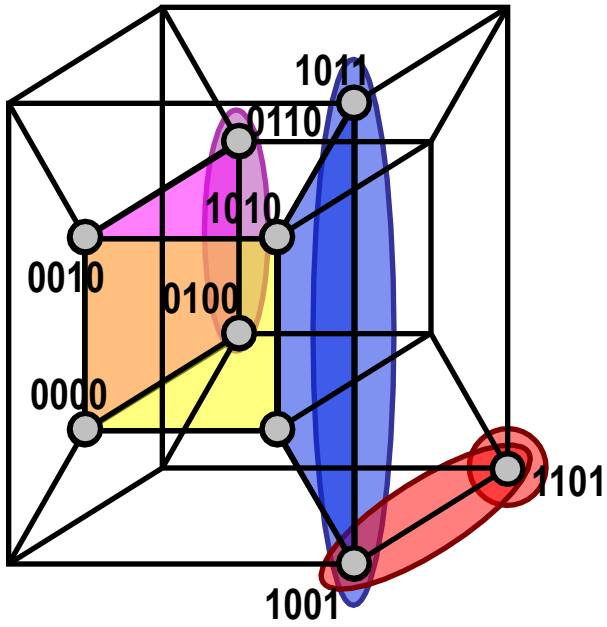


- 将 0000 扩展为 α = 0**0.
  - 删除被0**0覆盖的 0100, 0010, 0110.
- 将1000扩展为β = *0*0.
  - 删除被*0*0覆盖的1010.
- 将1001扩展为δ = 10**.
  - 删除被10**覆盖的1011.
- 将1101扩展为ε = 1*01.
- 获得的覆盖是: {α, β, δ, ε}.

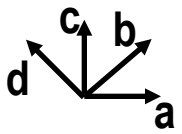


MINI的操作步骤-例子

- 1. 扩展:
  - 获得的覆盖是: {α,β,δ,ε}.
  - 质覆盖, 单个蕴含项下的非冗余覆盖.
- 2. 缩减:

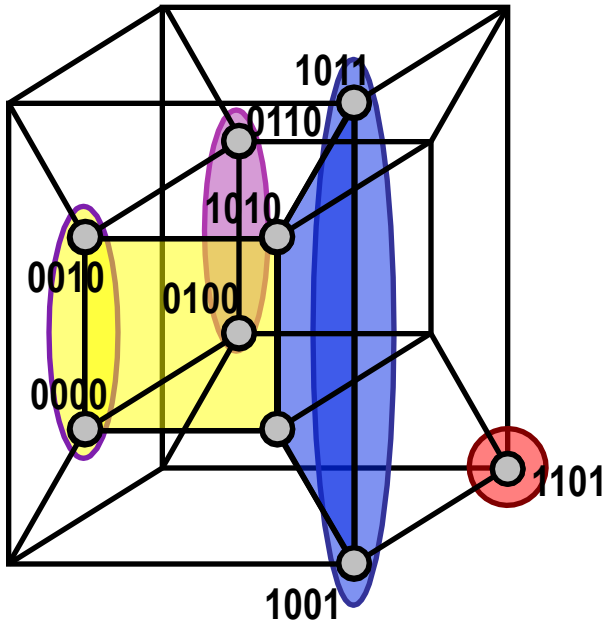


- 缩减α = 0**0为α’ = 01*0
- 缩减 δ = 10** 为 δ’ = 10*1.
- 缩减ε = 1*01 为 ε’ = 1101.
- 获得的覆盖是: {α’ β δ’ ε’}.

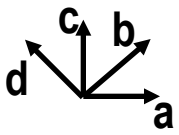


MINI的操作步骤-例子

- 1. 扩展:
  - 获得的覆盖是: {α,β,δ,ε}.
  - 质覆盖, 单个蕴含项下的非冗余覆盖.
- 2. 缩减:
  - 缩减α = 0**0为α’ = 01*0
  - 缩减 δ = 10** 为 δ’ = 10*1.
  - 缩减ε = 1*01 为 ε’ = 1101.
  - 获得的覆盖是: {α’ β,δ’,ε’}.
- 3. 重构:

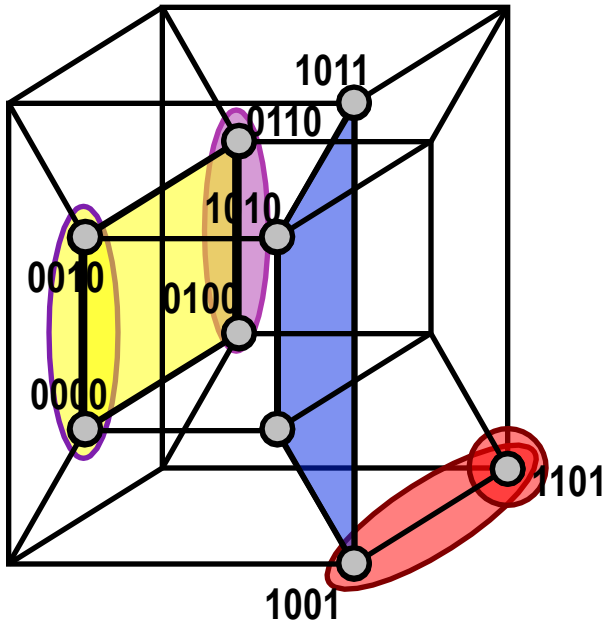


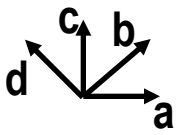
- 重构 {β, δ’} 为: {β’, δ}.
  - 其中 β’ = 00*0.
- 获得的覆盖是: {α’ β’ δ ε’’}.



MINI的操作步骤-例子

- 1. 扩展:
  - 获得的覆盖是: {α,β,δ,ε}.
  - 质覆盖, 单个蕴含项下的非冗余覆盖.
- 2. 缩减:
  - 缩减α = 0**0为α’ = 01*0
  - 缩减 δ = 10** 为 δ’ = 10*1.
  - 缩减ε = 1*01 为 ε’ = 1101.
  - 获得的覆盖是: {α’ β,δ’,ε’}.
- 3. 重构:
  - 重构 {β, δ’} 为: {β’, δ}，其中 β’ = 00*0.
  - 获得的覆盖是: {α’ β’,δ,ε’}.
- 4. 二次扩展:

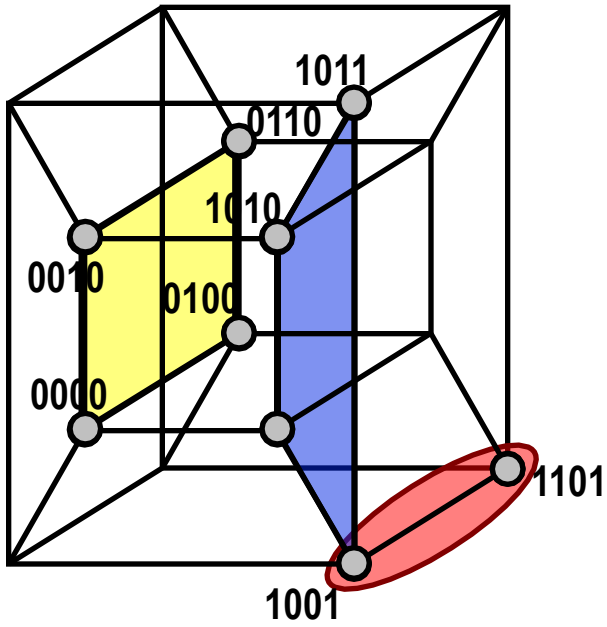




- 扩展 β = 00*0 为 β = 0**0，删除被0**0覆盖的01*0
- 扩展 ε’ = 1101 为 ε = 1*01.

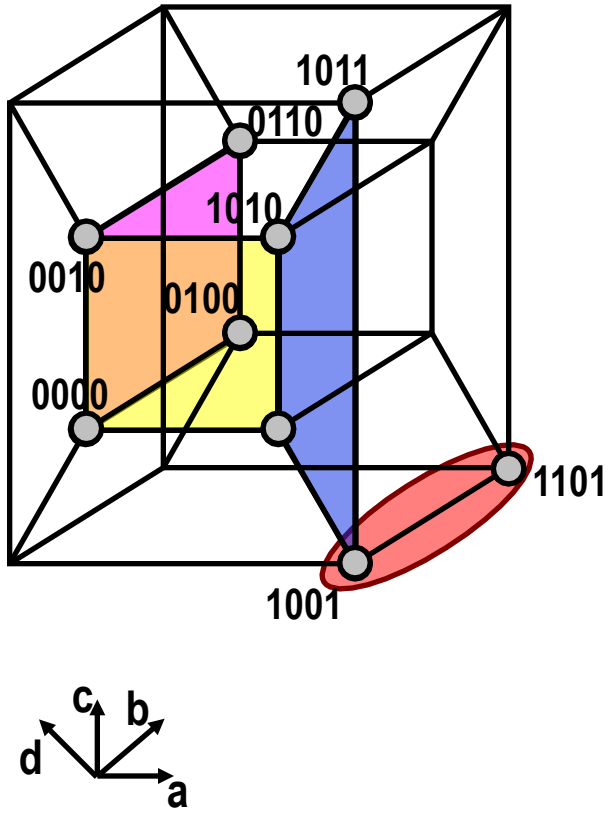
MINI的操作步骤-例子

- 1. 扩展:
  - 获得的覆盖是: {α,β,δ,ε}.
  - 质覆盖, 单个蕴含项下的非冗余覆盖.
- 2. 缩减:
  - 缩减α = 0**0为α’ = 01*0
  - 缩减 δ = 10** 为 δ’ = 10*1.
  - 缩减ε = 1*01 为 ε’ = 1101.
  - 获得的覆盖是: {α’ β,δ’,ε’}.
- 3. 重构:
  - 重构 {β, δ’} 为: {β’, δ}，其中 β’ = 00*0.
  - 获得的覆盖是: {α’ β’,δ,ε’}.
- 4. 二次扩展:
  - 扩展 β = 00*0 为 β = 0**0，删除被0**0覆盖的01*0
  - 扩展 ε’ = 1101 为 ε = 1*01.



ESPRESSO的操作步骤-例子

- 扩展:
  - 覆盖: {α, β, δ, ε}.
  - 质覆盖, 单个蕴含项下的非冗余覆盖.
- 去冗余:
  - 覆盖: {α, δ, ε}.
  - 质覆盖, 非冗余覆盖.



扩展 – 简单实现

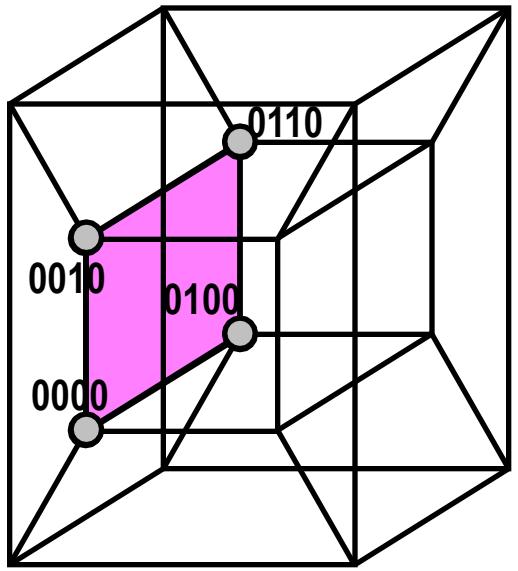
Expand - Naïve implementation

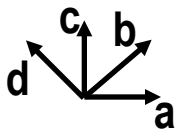
- 对于每个蕴含项
  - 对其中每个不为“无关项”（don’t care）的布尔文字
  - 如果可能，将其更改为“无关项”
  - 移除所有被扩展后的蕴含项覆盖的蕴含项

扩展的例子

F=a’b’c’d’+a’b’cd’+a’b’cd’+a’bcd’

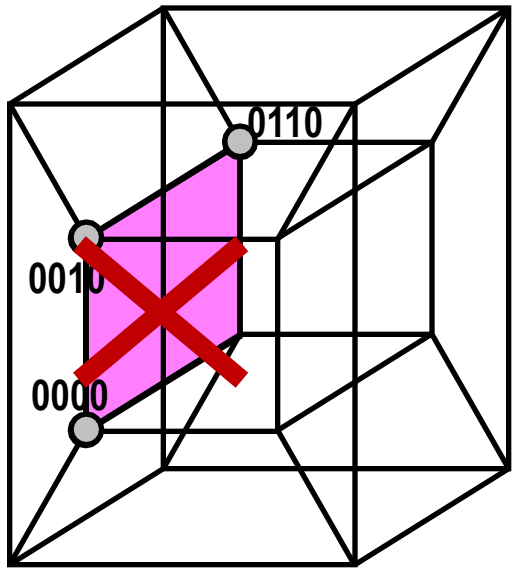
- Expand 0000 to α = 0**0.
  - Drop 0100, 0010, 0110 from the cover.

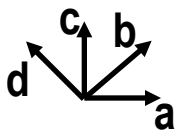




扩展的例子

F=a’b’c’d’+a’b’cd’+a’bcd’





扩展 – 简单实现

Expand - Naïve implementation

- 对于每个蕴含项
  - 对其中每个不为“无关项”（don’t care）的布尔文字
  - 如果可能，将其更改为“无关项”
  - 移除所有被扩展后的蕴含项覆盖的蕴含项
- 需要处理的问题：
  - 有效性检查
  - 决定扩展的顺序

有效性检查

Validity check

Espresso、 MINI

- 检查扩展后蕴含项与 OFF-SET的交集是否为空
- 该过程需要做求补操作

Presto

- 检查扩展后蕴含项是否包含于 ON-SET与 DC-SET的并集中
- 该问题可归约为递归重言式判定

例子

F = abc + abc’ + ab’c

检查abc(111)是否可以扩展为bc(*11)：

1. 求F’
2. 计算F’和bc的交是否为空

例子

$F = abc + abc' + ab'c$

检查abc(111)是否可以扩展为bc(*11)：

1. 求F'

对正单调变量a取余因子

$F' = F'_a + a'F'_{a'}$

$= F'_a + a'$

F_a:	$F_{a'}$:
01 01 01	01 01 01
01 01 10	01 01 10
01 10 01	01 10 01
01 11 11	10 11 11
01 01 01	00 01 01
01 01 10	00 01 10
01 10 01	00 10 01
10 00 00	
11 01 01	
11 01 10	
11 10 01	

$F_{a'}$=0 -> $F'_{a'}$=1

例子

$F = abc + abc' + ab'c$

检查abc(111)是否可以扩展为bc(*11)：

1. 求F'

$F' = F'_a + a'$

$= b\,F'_{ab} + b'\,F'_{ab'} + a'$

$= b'c' + a'$

F_{ab}:

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

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

11 11 01
11 11 10

F_{ab}=1 -> F'_{ab}=0

$F_{ab'}$:

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

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

$F_{ab'}$ 11 11 01

:

$F'_{ab'}$: 11 11 10

例子

F = abc + abc’ + ab’c

检查abc(111)是否可以扩展为bc(*11)：

1. F’ = b’c’+a’
2. 计算F’和bc的交是否为空

->交集不为空， abc不可以扩展为bc

F’ 和bc的交集：

11 10 10

10 11 11

11 01 01

---

11 00 00

10 01 01

例子

F = abc + abc’ + ab’c

检查abc(111)是否可以扩展为ac(1*1)：

1. F’ = b’c’+a’
2. 计算F’和ac的交是否为空

->交集为空， abc可以扩展为ac

F’ 和bc的交集：

11 10 10

10 11 11

01 11 01

---

01 10 00

00 11 01

有效性检查

Validity check

Espresso、 MINI

- 检查扩展后蕴含项与 OFF-SET的交集是否为空
- 该过程需要做求补操作

Presto

- 检查扩展后蕴含项是否包含于 ON-SET与 DC-SET的并集中
- 该问题可归约为递归重言式判定

例子 - Presto

F = abc + abc’ + ab’c

检查abc(111)是否可以扩展为bc(*11)：

1. 计算F是否包含bc

F_{bc}非重言式，F不包含bc
-> abc不可以扩展为bc

F_{bc}
:

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

---

01 01 01
01 01 00
01 00 01
00 10 10

---

01 11 11

非重言式

例子 - Presto

F = abc + abc’ + ab’c

检查abc(111)是否可以扩展为ac(1*1)：

1. 计算F是否包含ac

F_{ac}为重言式，F包含ac
-> abc可以扩展为ac

F_{ac}: 01 01 01
01 01 10
01 10 01
01 11 01

---

01 01 01
01 01 00
01 10 01
10 00 10

---

11 01 11
11 10 11

重言式

作业

请用矩阵表示法确认以下函数的a’bcd’能否扩展为a’bc：

$$f = a'bcd + a'bcd' + a'b'cd$$

Espresso、 MINI

- 检查扩展后蕴含项与 F’ 的交集是否为空

- Presto

- 检查扩展后蕴含项是否包含于 F