Logic Synthesis

Rectangles and Rectangle Covering

Võ Thị Khánh Vân 09070478
Phan Văn Long Điền 10070474

Content

- Rectangle
- Rectangle – Kernel
- Kernel Intersection
- PING-PONG Algorithm
- Rectangle Algorithm

Content

- Rectangle
- Rectangle – Kernel
- Kernel Intersection
- PING-PONG Algorithm
- Rectangle Algorithm

Definitions (1/6)

- ***Rectangle:***
  - Rectangle (R, C) của ma trận B, $B_{ij} \in \{0,1,*\}$, là một tập con các hàng R và tập con các cột C sao cho $B_{ij} \in \{1,*\}$ với $i \in R$ và $j \in C$.

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
5	1	0	1	1	1

Definitions (2/6)

- ***Contain:***
  - Rectangle (R1, C1) gọi là chứa rectangle (R2, C2) nếu (R2 $\subseteq$ R1 và C2 $\subset$ C1) hoặc (R2 $\subset$ R1 và C2 $\subseteq$ C1).

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
5	1	0	1	1	1

Definitions (3/6)

- ***Prime rectangle:***
  - Rectangle (R,C) của ma trận B được gọi là prime rectangle nếu nó không bị chứa bởi bất kỳ rectangle nào của B

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
5	1	0	1	1	1

Definitions (4/6)

- ***Corectangle:***
  - Corectangle của 1 rectangle (R, C) là cặp (R, C') trong đó C' là tập các cột không thuộc C

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
5	1	0	1	1	1

Definitions (5/6)

- ***Rectangle cover:***
  - Tập các rectangle ($\{R^K, C^K\}$) tạo thành rectangle cover của ma trận B nếu với $i \in R^K$ và $j \in C^K \Rightarrow B_{ij} = 1$.
  - Mỗi giá trị 1 của B phải bị phủ bởi ít nhất 1 rectangle của cover.

	1	2	3	4	5
1	1	1	1	0	0
2	1	*	1	0	*
3	0	1	1	0	1
5	1	0	1	1	1

Definitions (6/6)

- ***Minimum weighted rectangle cover:***
  - Mỗi rectangle (R_k, C_k) được gán cho trọng số hoặc chi phí định nghĩa bởi hàm w(R_k, C_k). Trọng số của một rectangle cover được định nghĩa:

$$\sum_k w\left(R^k, C^k\right)$$

mà $$w(R,C) = \begin{cases} |C| & \text{nếu } |R| = 1 \\ |C| + |R| & \text{nếu } |R| > 1 \end{cases}$$

  - Minimum weighted rectangle cover của một ma trận là rectangle cover có tổng trọng số các rectangle nhỏ nhất.

Content

- Rectangle
- **Rectangle – Kernel**
- Kernel Intersection
- PING-PONG Algorithm
- Rectangle Algorithm

Cube-Literal Matrix (1/2)

- Biểu diễn biểu thức luận lý bằng ma trận cube-literal:
  - Mỗi hàng tương ứng với cube
  - Mỗi cột tương ứng với literal
  - B_{ij} = 1: biến thứ j xuất hiện trong cube thứ i của biểu thức

$$g = a.b.e + a.c.d + b.c.d$$

	1	**2**	**3**	**4**	**5**
a.b.e	1	1	0	0	1
a.c.d	1	0	1	1	0
b.c.d	0	1	1	1	0

Cube-Literal Matrix (1/2)

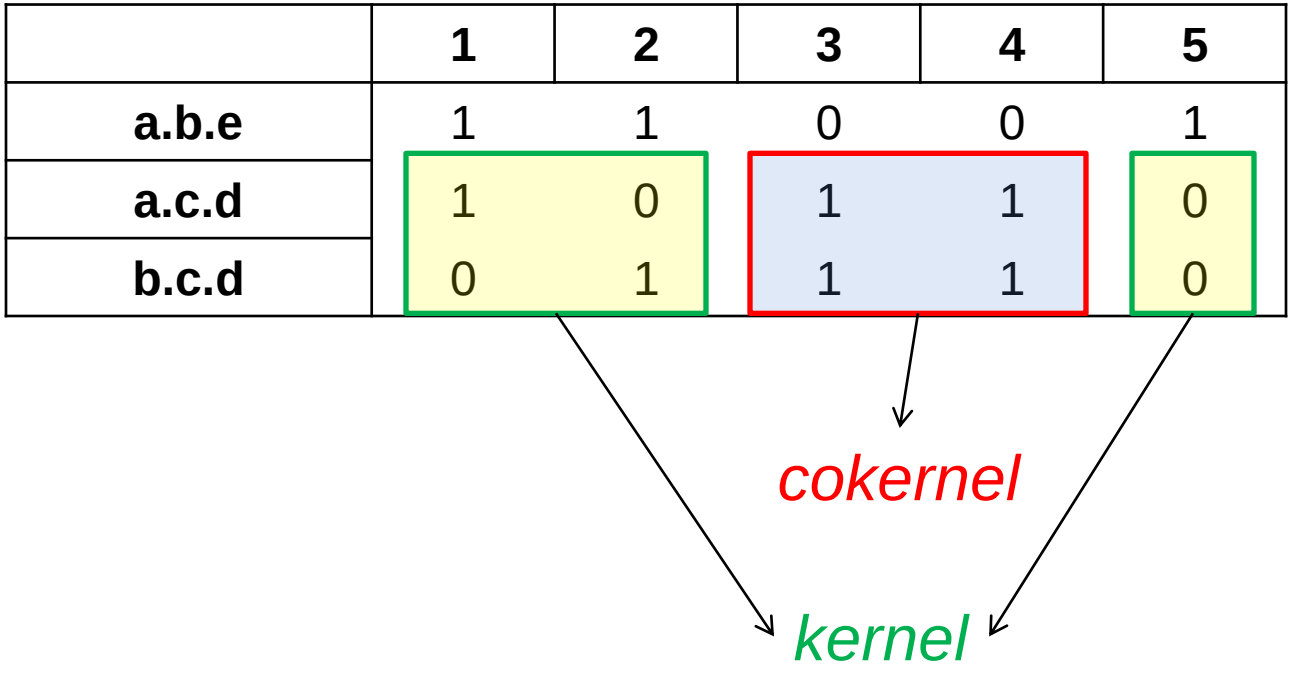
$$g = a.b.e + a.c.d + b.c.d$$

	1	2	3	4	5
a.b.e	1	1	0	0	1
a.c.d	1	0	1	1	0
b.c.d	0	1	1	1	0

- Rectangle (R,C) = ({2,3},{3,4}) = cube *c.d*
- Corectangle (R,C') = ({2,3},{1,2,5}) = *(a + b) = g/(c.d)*
- Chọn prime rectangle biểu diễn cube chung lớn nhất giữa các toán hạng của biểu thức luận lý.
- Kết quả của phép chia biểu thức cho cube trên là một cube-free → kernel của biểu thức.

Rectangle – Kernel

- Định nghĩa:
  - Mỗi *prime rectangle* là một *cokernel*
  - Mỗi *corectangle* của một *prime rectangle* là một *kernel*



Common-Cube Extraction

- F = a.b.c + a.b.d + e.g
- G = a.b.f.g
- H = b.d + e.f

	a	b	c	d	e	f	g
F_1 a.b.c	1	1	1	0	0	0	0
F_2 a.b.d	1	1	0	1	0	0	0
F_3 e.g	0	0	0	0	1	0	1
G a.b.f.g	1	1	0	0	0	1	1
H_1 b.d	0	1	0	1	0	0	0
H_2 e.f	0	0	0	0	1	1	0

- X = a.b $\Rightarrow$

F = X.c + X.d + e.g

G = X.f.g

H = b.d + e.f

Common-Cube Extraction

- Giá trị v(R, C) của một rectangle:
  - số biến được rút gọn sau khi áp dụng extraction cho cube biểu diễn bởi rectangle

$$v(R,C) = \left|\{(i,j) \mid B_{ij} = 1, i \in R, j \in C\}\right| - w(R,C)$$

		a	b	c	d	e	f	g
F_1	a.b.c	1	1	1	0	0	0	0
F_2	a.b.d	1	1	0	1	0	0	0
F_3	e.g	0	0	0	0	1	0	1
G	a.b.f.g	1	1	0	0	0	1	1
H_1	b.d	0	1	0	1	0	0	0
H_2	e.f	0	0	0	0	1	1	0

Số biến: 6

w(R,C) = 5

$\Rightarrow$ Số biến rút gọn: 1

Content

- Rectangle
- Rectangle – Kernel
- **Kernel Intersection**
- PING-PONG Algorithm
- Rectangle Algorithm

Kernel Intersection (1/7)

- ***Ma trận cokernel-cube:***
  - Mỗi hàng tương ứng với một cokernel
  - Mỗi cột tương ứng với một cube trong một kernel

- Ví dụ:

  F = a.f + b.f + a.g + c.g + a.d.e + b.d.e + c.d.e

  G = a.f + b.f + a.c.e + b.c.e

  H = a.d.e + c.d.e

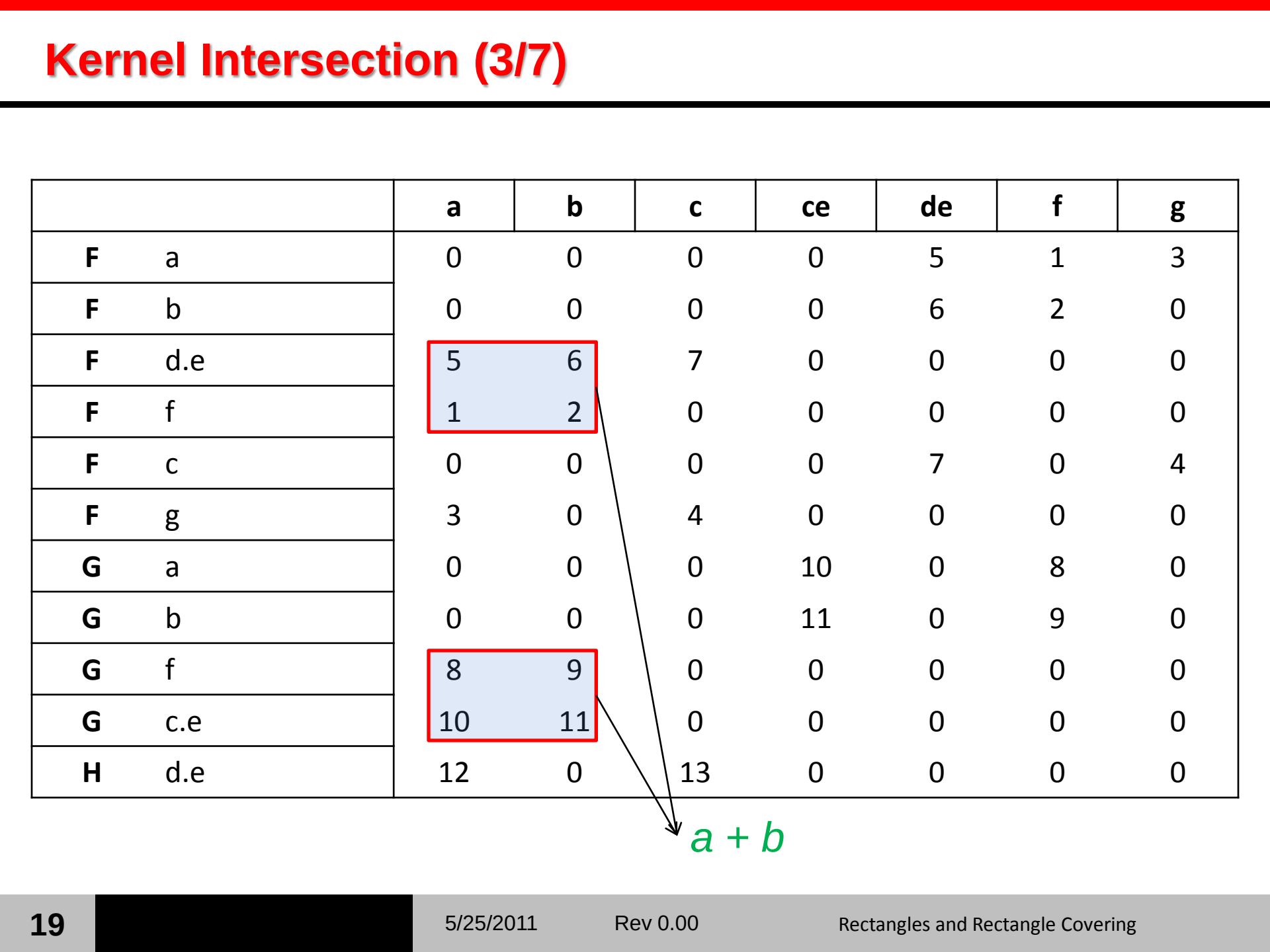
Kernel Intersection (2/7)

Cube	Index
a.f	1
b.f	2
a.g	3
c.g	4
a.d.e	5
b.d.e	6
c.d.e	7
a.f	8
b.f	9
a.c.e	10
b.c.e	11
a.d.e	12
c.d.e	13

Function	Cokernel	Kernel
F	a	d.e + f+ g
F	b	d.e + f
F	d.e	a + b + c
F	f	a + b
F	c	d.e + g
F	g	a + c
G	a	c.e + f
G	b	c.e + f
G	f	a + b
G	c.e	a + b
H	d.e	a + c

Kernel Intersection (3/7)

		a	b	c	ce	de	f	g
F	a	0	0	0	0	5	1	3
F	b	0	0	0	0	6	2	0
F	d.e	5	6	7	0	0	0	0
F	f	1	2	0	0	0	0	0
F	c	0	0	0	0	7	0	4
F	g	3	0	4	0	0	0	0
G	a	0	0	0	10	0	8	0
G	b	0	0	0	11	0	9	0
G	f	8	9	0	0	0	0	0
G	c.e	10	11	0	0	0	0	0
H	d.e	12	0	13	0	0	0	0



Kernel Intersection (4/7)

$$F = a.f + b.f + a.g + c.g + a.d.e + b.d.e + c.d.e$$
$$G = a.f + b.f + a.c.e + b.c.e$$
$$H = a.d.e + c.d.e$$

$\Rightarrow$

$$F = d.e.X + f.X + a.g + c.g + c.d.e$$
$$G = c.e.X + f.X$$
$$H = a.d.e + c.d.e$$
$$X = a + b$$

Kernel Intersection (5/7)

- Trọng số của rectangle trong ma trận cokernel-cube: tổng số literal ứng với rectangle sau khi áp dụng extraction trên rectangle tương ứng.

$$w(R,C) = \sum_{i \in R} w_i^r + \sum_{j \in C} w_j^c$$

w_i^r : 1 + số literal của cokernel hàng thứ i

w_j^c : số literal của kernel-cube cột thứ j

Kernel Intersection (6/7)

- Giá trị của một rectangle (R, C) trong ma trận cokernel-cube: số literal được rút gọn sau khi áp dụng extraction trên rectangle tương ứng.

$$v(R,C) = \sum_{i \in R,\, j \in C} V_{ij} - w(R,C)$$

V_{ij} : số literal của cube được tạo bởi cokernel hàng i và kernel-cube cột thứ j của ma trận cokernel-cube

Kernel Intersection (7/7)

		a	b	c	ce	de	f	g
F	a	0	0	0	0	5	1	3
F	b	0	0	0	0	6	2	0
F	d.e	5	6	7	0	0	0	0
F	f	1	2	0	0	0	0	0
F	c	0	0	0	0	7	0	4
F	g	3	0	4	0	0	0	0
G	a	0	0	0	10	0	8	0
G	b	0	0	0	11	0	9	0
G	f	8	9	0	0	0	0	0
G	c.e	10	11	0	0	0	0	0
H	d.e	12	0	13	0	0	0	0

F = a.f + b.f + a.g + c.g + a.d.e + b.d.e + c.d.e
G = a.f + b.f + a.c.e + b.c.e
H = a.d.e + c.d.e

$\Rightarrow$F = d.e.X + f.X + a.g + c.g + c.d.e
G = c.e.X + f.X
H = a.d.e + c.d.e
X = a + b

- w(R,C) = (3+2+2+3)+(1+1) = 12
- v(R,C) = 3+3+2+2+2+2+3+3-12 = 8

Content

- Rectangle
- Rectangle – Kernel
- Kernel Intersection
- **PING-PONG Algorithm**
- Rectangle Algorithm

PING-PONG Algorithm (1/13)

- ***Mục tiêu***: tìm một Rectangle có giá trị lớn nhất mà không cần phải liệt kê ra tất cả các Prime Rectangle
- ***Ví dụ***: Tìm Rectangle có v lớn nhất trên ma trận cokernel-cube của hàm F

$$F = ac + ad + ae + ag + bc + bd + be + bf + ce + cf + df + dg$$

PING-PONG Algorithm (2/13)

- ***Bước 1***: Vẽ ma trận cokernel-cube tương ứng cho biểu thức

$$F = ac + ad + ae + ag + bc + bd + be + bf + ce + cf + df + dg$$

	a	b	c	d	e	f	g
a	0	0	2	2	2	0	2
b	0	0	2	2	2	2	0
c	2	2	0	0	2	2	0
d	2	2	0	0	0	2	2
e	2	2	2	0	0	0	0
f	0	2	2	2	0	0	0
g	2	0	0	2	0	0	0

Kernel-cube

cokernel

Vij : số literal của cube tạo bởi cokernel hàng i và kernel-cube cột j

PING-PONG Algorithm (3/13)

- ***Bước 2***: Chọn hàng tốt nhất làm hạt giống

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Chọn bất kì hàng nào trong 4 hàng 1, 2, 3, 4 vì có cùng value:

$$V(R,C) = \sum_{i \in R, j \in C} V_{ij} - \sum_{i \in R} w_i^r - \sum_{j \in C} w_j^c = (2x4) - (1+1) - (1x4) = 8 - 2 - 4 = 2$$

Chọn hàng 1: Rectangle: ({1},{3,4,5,7}) V(R,C) = **2**

PING-PONG Algorithm (4/13)

- ***Bước 3***: Mở rộng rectangle hạt giống theo hướng tăng số lượng các hàng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm hàng cho đến khi chỉ còn 1 cột trong tập cột C

Hạt giống : Rectangle: ({1},{3,4,5,7}) V(R,C) = **2**

Thêm hàng 2: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5**

PING-PONG Algorithm (5/13)

- ***Bước 3***: Mở rộng rectangle hạt giống theo hướng tăng số lượng các hàng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm hàng cho đến khi chỉ còn 1 cột trong tập cột C

Hạt giống : Rectangle: ({1},{3,4,5,7}) V(R,C) = **2**

Thêm hàng 2: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5**

Thêm hàng 6: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

PING-PONG Algorithm (6/13)

- ***Bước 3***: Mở rộng rectangle hạt giống theo hướng tăng số lượng các hàng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm hàng cho đến khi chỉ còn 1 cột trong tập cột C

Hạt giống : Rectangle: ({1},{3,4,5,7}) V(R,C) = **2**

Thêm hàng 2: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5**

Thêm hàng 6: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

Thêm hàng 5: Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

PING-PONG Algorithm (7/13)

- ***Bước 3***: Mở rộng rectangle hạt giống theo hướng tăng số lượng các hàng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Hạt giống : Rectangle: ({1},{3,4,5,7}) V(R,C) = **2**

Thêm hàng 2: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5** ⇒ chọn

Thêm hàng 6: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

Thêm hàng 5: Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

PING-PONG Algorithm (8/13)

- ***Bước 4***: Chọn hạt giống mới là cột có giá trị lớn nhất trong rectangle được chọn ở bước 3

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Hạt giống : Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

PING-PONG Algorithm (9/13)

- ***Bước 5***: Lặp lại bước 3 nhưng hoán vị giữa cột và dòng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm cột cho đến khi chỉ còn 1 hàng trong tập hàng R

Hạt giống : Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

Thêm cột 4: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

PING-PONG Algorithm (10/13)

- ***Bước 5***: Lặp lại bước 3 nhưng hoán vị giữa cột và dòng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm cột cho đến khi chỉ còn 1 hàng trong tập hàng R

Hạt giống : Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

Thêm cột 4: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

Thêm cột 5: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5**

PING-PONG Algorithm (11/13)

- ***Bước 5***: Lặp lại bước 3 nhưng hoán vị giữa cột và dòng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Thêm cột cho đến khi chỉ còn 1 hàng trong tập hàng R

Hạt giống : Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

Thêm cột 4: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

Thêm cột 5: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5**

Thêm cột 7: Rectangle: ({1},{3,4,5,7}) V(R,C) = 8 – 2 – 4 = **2**

PING-PONG Algorithm (12/13)

- ***Bước 5***: Lặp lại bước 3 nhưng hoán vị giữa cột và dòng ⇒ chọn rectangle tốt nhất

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Hạt giống : Rectangle: ({1,2,5,6},{3}) V(R,C) = 8 – 8 – 1 = **-1**

Thêm cột 4: Rectangle: ({1,2,6},{3,4}) V(R,C) = 12 – 6 – 2 = **4**

Thêm cột 5: Rectangle: ({1,2},{3,4,5}) V(R,C) = 12 – 4 – 3 = **5** ⇒ chọn

Thêm cột 7: Rectangle: ({1},{3,4,5,7}) V(R,C) = 8 – 2 – 4 = **2**

PING-PONG Algorithm (13/13)

- ***Bước 6***: Nếu rectangle được chọn ở bước 3 khác với ở bước 5 thì lặp lại quá trình. Ngược lại đó là rectangle cần tìm

		1	2	3	4	5	6	7
		a	b	c	d	e	f	g
1	a	0	0	2	2	2	0	2
2	b	0	0	2	2	2	2	0
3	c	2	2	0	0	2	2	0
4	d	2	2	0	0	0	2	2
5	e	2	2	2	0	0	0	0
6	f	0	2	2	2	0	0	0
7	g	2	0	0	2	0	0	0

Note:

Nếu lặp lại, chọn seed row là 1 hàng trong rectangle tốt nhất hiện tại

⇒ **Kết thúc! Rectangle cần tìm**: ({1,2},{3,4,5})

Content

- Rectangle
- Rectangle – Kernel
- Kernel Intersection
- PING-PONG Algorithm
- **Rectangle Algorithm**

Generating Prime Rectangle (1/2)

GENERATE_RECTANGLES(M):

{

Tìm tất cả các trivial rectangle và lưu trữ lại;

rect = new rectangle;

RECTANGLE1(*M, 0, rect*);

}

- Mỗi prime rectangle tương ứng với một cặp kernel – cokernel của hàm ban đầu

Generating Prime Rectangle (2/2)

RECTANGLE1 (*M, index, rect*):

```
{
    for ( mỗi cột c của M ) {
      if ( c có ≥ 2 phần tử và c.index ≥ index ) {
            M1 = new matrix;
            for( mỗi phần tử p trong cột c )
            copy hàng tương ứng với p trong M vào M1;

            rect1 = new rectangle;
            Cột của rect1 ← cột của rect;
            Hàng của rect1 tương ứng với các phần tử của c trong M;

            if ( các index của tất cả các cột toàn 1 trong M1 ≥ c.index ) {
                  Xoá các cột toàn 1 trong M1 và thêm vào rect1;
                  Lưu rect1 trong vào tập các prime rectangle;
                  RECTANGLE1(M1, c.index, rect1);
            }
      }
    }
}
```

Rectangle Algorithm (1/4)

$F = a.b.c.d.g + a.b.c.d.h + a.b.c.e + a.b.c.f + a.b.i$

Ma trận cube-literal:

	1	2	3	4	5	6	7	8	9
	a	b	c	d	e	f	g	h	i
1	1	1	1	1	0	0	1	0	0
2	1	1	1	1	0	0	0	1	0
3	1	1	1	0	1	0	0	0	0
4	1	1	1	0	0	1	0	0	0
5	1	1	0	0	0	0	0	0	1

Gọi **RECTANGLE1 (***M, 0, rect*)

– *Xét cột 1*: với 5 hàng bằng 1 (≥ 2) và *c.index* ≥ 0 ⇒ tạo ma trận mới *M1* được gán bằng *M*. Tạo rectangle mới *rect1* với giá trị khởi tạo ({1,2,3,4,5}, {1})

– *Cột 2* của *M1* có tất cả các hàng bằng 1 ⇒ thêm vào *rect1*

⇒ prime rectangle ({1,2,3,4,5}, {1,2})

Kernel (cokernel) tương ứng: c.d.g + c.d.h + c.e + c.f + i (a.b)

Rectangle Algorithm (2/4)

$F = a.b.c.d.g + a.b.c.d.h + a.b.c.e + a.b.c.f + a.b.i$

Gọi đệ quy **RECTANGLE1** (*M, c.index, rect*) với M như sau:

	3	4	5	6	7	8	9
	c	**d**	**e**	**f**	**g**	**h**	**i**
1	1	1	0	0	1	0	0
2	1	1	0	0	0	1	0
3	1	0	1	0	0	0	0
4	1	0	0	1	0	0	0
5	0	0	0	0	0	0	1

c.index = 1, *rect* = ({1,2,3,4,5},{1,2})

– *Xét cột 3*: với 4 hàng bằng 1 (≥ 2) và *c.index* ≥ 1 ⇒ tạo ma trận mới *M1* được gán bằng *M* bỏ đi hàng 5. Tạo *rect1* được gán giá trị ({1,2,3,4}, {1,2})

– *Cột 3* của *M1* có tất cả các hàng bằng 1 ⇒ thêm vào *rect1*

⇒ prime rectangle ({1,2,3,4}, {1,2,3})

Kernel (cokernel) tương ứng: d.g + d.h + e + f (a.b.c)

Rectangle Algorithm (3/4)

F = a.b.c.d.g + a.b.c.d.h + a.b.c.e + a.b.c.f + a.b.i

Gọi đệ quy **RECTANGLE1** (*M, c.index, rect*) với M như sau:

	4	5	6	7	8	9
	d	**e**	**f**	**g**	**h**	**i**
1	1	0	0	1	0	0
2	1	0	0	0	1	0
3	0	1	0	0	0	0
4	0	0	1	0	0	0

c.index = 3, *rect* = ({1,2,3,4},{1,2,3})

– *Xét cột 4*: với 2 hàng bằng 1 (≥ 2) và *c.index* ≥ 3 ⇒ tạo ma trận mới *M1* được gán bằng *M* bỏ đi hàng 3 và 4. Tạo *rect1* được gán giá trị ({1,2}, {1,2,3})

– *Cột 4* của *M1* có tất cả các hàng bằng 1 ⇒ thêm vào *rect1*

⇒ prime rectangle ({1,2}, {1,2,3,4})

Kernel (cokernel) tương ứng: g + h (a.b.c.d)

Rectangle Algorithm (4/4)

Quay lại gọi đệ quy **RECTANGLE1** cho cột thứ 2, chỉ số cột bắt đầu bằng 0, M là ma trận ban đầu

	1	2	3	4	5	6	7	8	9
	a	b	c	d	e	f	g	h	i
1	1	1	1	1	0	0	1	0	0
2	1	1	1	1	0	0	0	1	0
3	1	1	1	0	1	0	0	0	0
4	1	1	1	0	0	1	0	0	0
5	1	1	0	0	0	0	0	0	1

Gọi **RECTANGLE1 (***M, 0, rect*)

– *Xét cột 2*: tất cả các hàng bằng 1 (≥ 2) và *c.index* ≥ 0 ⇒ tạo ma trận mới *M1* được gán bằng *M*. Tạo *rect1* được gán giá trị ({1,2,3,4,5}, {2})

– *Cột 1* của *M1* có tất cả các hàng bằng 1,

tuy nhiên *index < c.index* (điều kiện dừng đệ quy) ⇒ dừng giải thuật

Thank you!

References

- Chapter 7, Logic Synthesis –Srinivas Devadas, Abhijit Ghosh, Kurt Keutzer