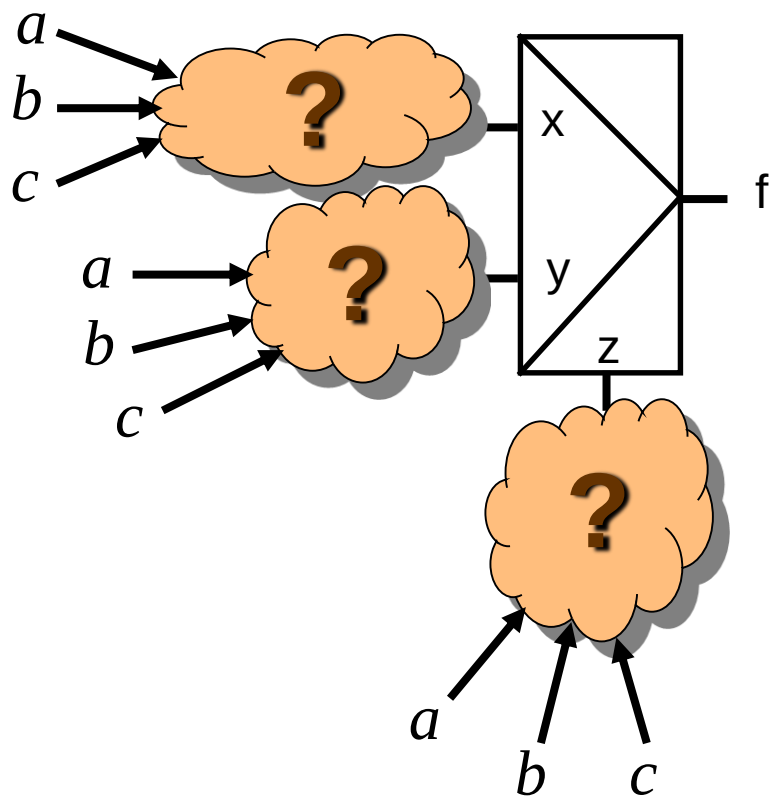
Extracting Functions from Boolean Relations

Jordi Cortadella
Universitat Politècnica de Catalunya

Boolean relation: example

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	*
1	1	0	1
1	1	1	0

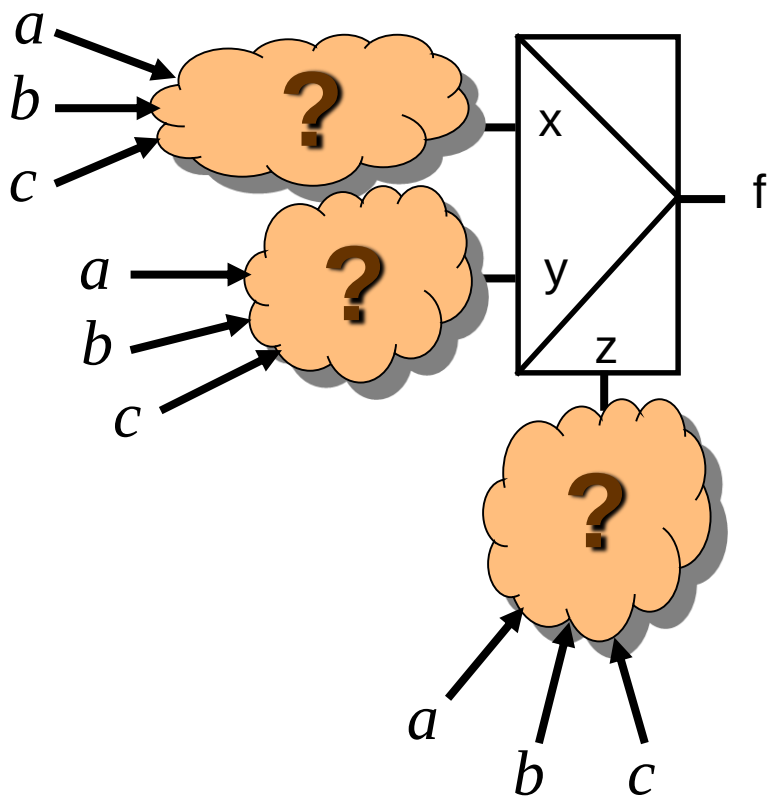
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	b	c	f	x y z
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	*	
1	1	0	1	
1	1	1	0	

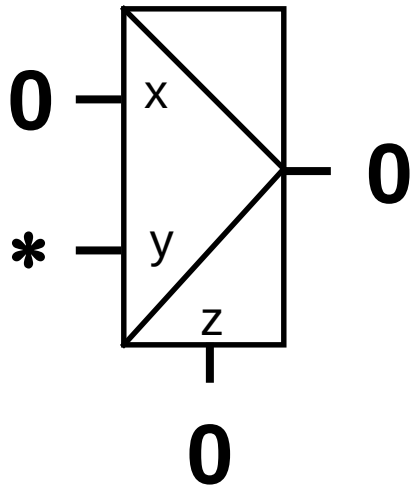
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	b	c	f	x y z
0	0	0	0	0∗0
0	0	1	1	
0	1	0	0	0∗0
0	1	1	1	
1	0	0	0	0∗0
1	0	1	∗	
1	1	0	1	
1	1	1	0	0∗0

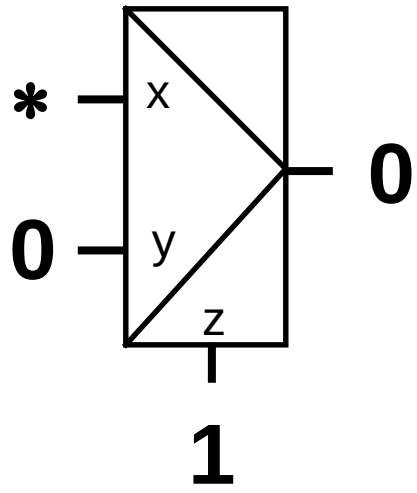
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	b	c	f	x y z
0	0	0	0	0∗0 | ∗01
0	0	1	1	
0	1	0	0	0∗0 | ∗01
0	1	1	1	
1	0	0	0	0∗0 | ∗01
1	0	1	∗	
1	1	0	1	
1	1	1	0	0∗0 | ∗01

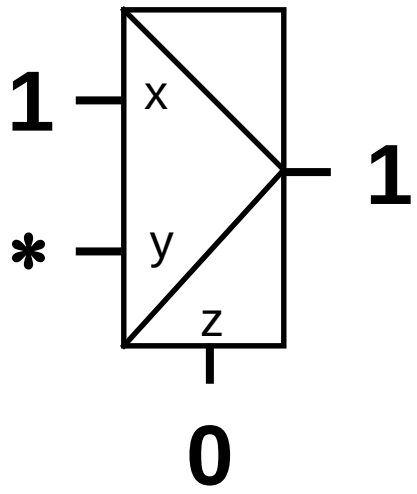
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	b	c	f	x y z
0	0	0	0	0∗0 | ∗01
0	0	1	1	1∗0
0	1	0	0	0∗0 | ∗01
0	1	1	1	1∗0
1	0	0	0	0∗0 | ∗01
1	0	1	∗	
1	1	0	1	1∗0
1	1	1	0	0∗0 | ∗01

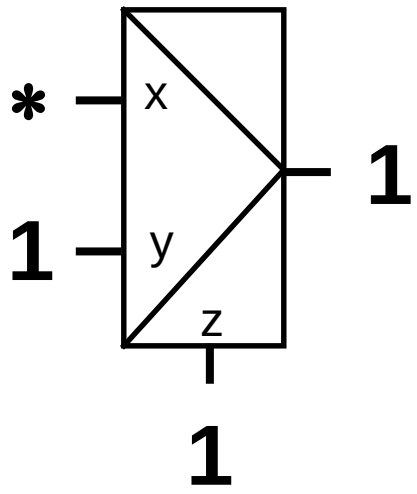
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	b	c	f	x y z
0	0	0	0	0∗0 | ∗01
0	0	1	1	1∗0 | ∗11
0	1	0	0	0∗0 | ∗01
0	1	1	1	1∗0 | ∗11
1	0	0	0	0∗0 | ∗01
1	0	1	∗	
1	1	0	1	1∗0 | ∗11
1	1	1	0	0∗0 | ∗01

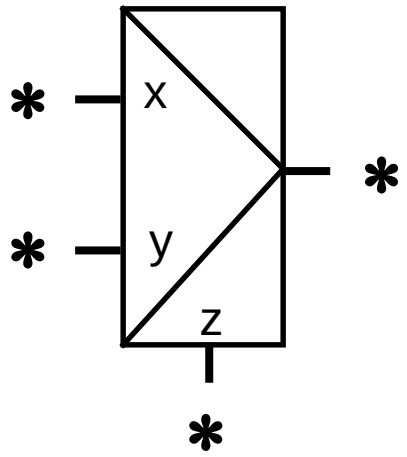
$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

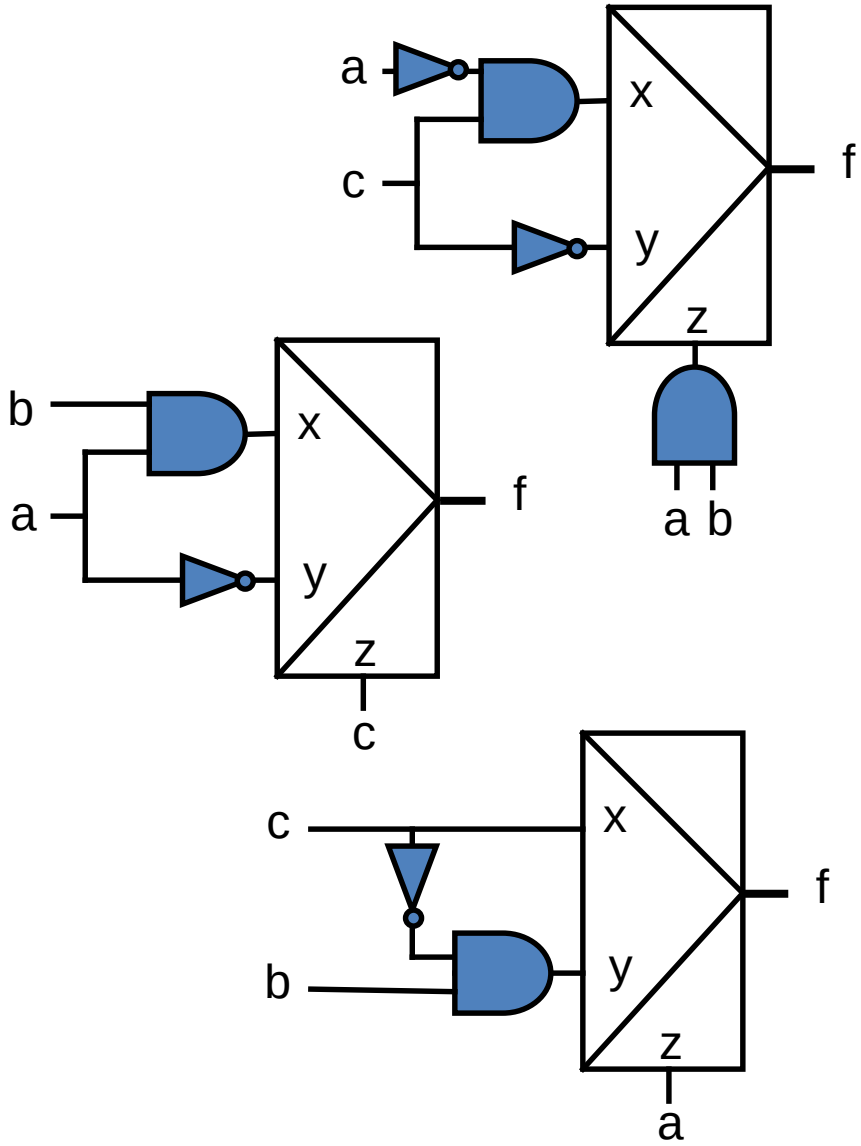
a	b	c	f	x y z
0	0	0	0	0∗0 | ∗01
0	0	1	1	1∗0 | ∗11
0	1	0	0	0∗0 | ∗01
0	1	1	1	1∗0 | ∗11
1	0	0	0	0∗0 | ∗01
1	0	1	*	∗ ∗ ∗
1	1	0	1	1∗0 | ∗11
1	1	1	0	0∗0 | ∗01

$$f = \bar{a}c + ab\bar{c}$$



Boolean relation: example

a	*b*	*c*	*x y z*
0	0	0	0∗0 | ∗01
0	0	1	1∗0 | ∗11
0	1	0	0∗0 | ∗01
0	1	1	1∗0 | ∗11
1	0	0	0∗0 | ∗01
1	0	1	∗ ∗ ∗
1	1	0	1∗0 | ∗11
1	1	1	0∗0 | ∗01



BR: characteristic function

$$R \subseteq B^n \times B^m$$

$$\chi_R : B^n \times B^m \rightarrow B$$

ab	fg
00	10
01	01 | 10
10	$*1$
11	$**$

$$\chi_R = \bar{a}\bar{b}f\bar{g} + \bar{a}b(f \oplus g) + a\bar{b}g + ab$$

Solving a Boolean Relation

Finding f and g such that χ_R becomes a tautology.

ab	fg
00	10
01	01 | 10
10	$*1$
11	$**$

Example: $f = 1 \quad g = a$

$$\chi_R = \bar{a}\bar{b}f\bar{g} + \bar{a}b(f \oplus g) + a\bar{b}g + ab$$

$$\chi_R = \bar{a}\bar{b}1\bar{a} + \bar{a}b(1 \oplus a) + a\bar{b}a + ab$$

$$\chi_R = 1$$

Solving a Boolean Relation

Finding f and g such that χ_R becomes a tautology.

ab	fg	
00	10	
01	01 | 10	**11**
10	$*1$	
11	$**$	

Example: $f = \bar{b} \quad g = a$

$$\chi_R = \bar{a}\bar{b}f\bar{g} + \bar{a}b(f \oplus g) + a\bar{b}g + ab$$

$$\chi_R = \bar{a}\bar{b}\bar{b}\bar{a} + \bar{a}b(\bar{b} \oplus a) + a\bar{b}a + ab$$

$$\chi_R = a + \bar{b} \quad \Rightarrow \quad \overline{\chi_R} = \bar{a}b$$

Exact 2-level solvers

The concept of prime must be redefined

- F. Somenzi and R.K. Brayton (1989)
  - Method similar to Quine-McCluskey
  - Extension of primes to c-primes
- S. Jeong and F. Somenzi (1992)
  - Formulate the problem as a Binate Covering Problem

Heuristic 2-level solvers

Based on ESPRESSO:

- A. Ghosh, S. Devadas and A. R. Newton (1992)
- Y. Watanabe and R. K. Brayton (1993)

GYOCRO

```
x:=Initial Solution;
do
    REDUCE(x);
    EXPAND(x);
    IRREDUNDANT(x);
while  x is improved;
```

Modern approaches

- Jie-Hong R. Jiang, Hsuan-Po Lin and Wei-Lun Hung, *Interpolating Functions from Large Boolean Relations*, ICCAD 2009.

- David Bañeres, Jordi Cortadella, and Mike Kishinevsky, *A Recursive Paradigm to Solve Boolean Relations*, DAC 2004.

Subclasses of Boolean Relations

a b	*x y*
00	10
01	11
10	01
11	10

Completely Specified Functions

a b	*x y*
00	10
01	11
10	*1
11	**

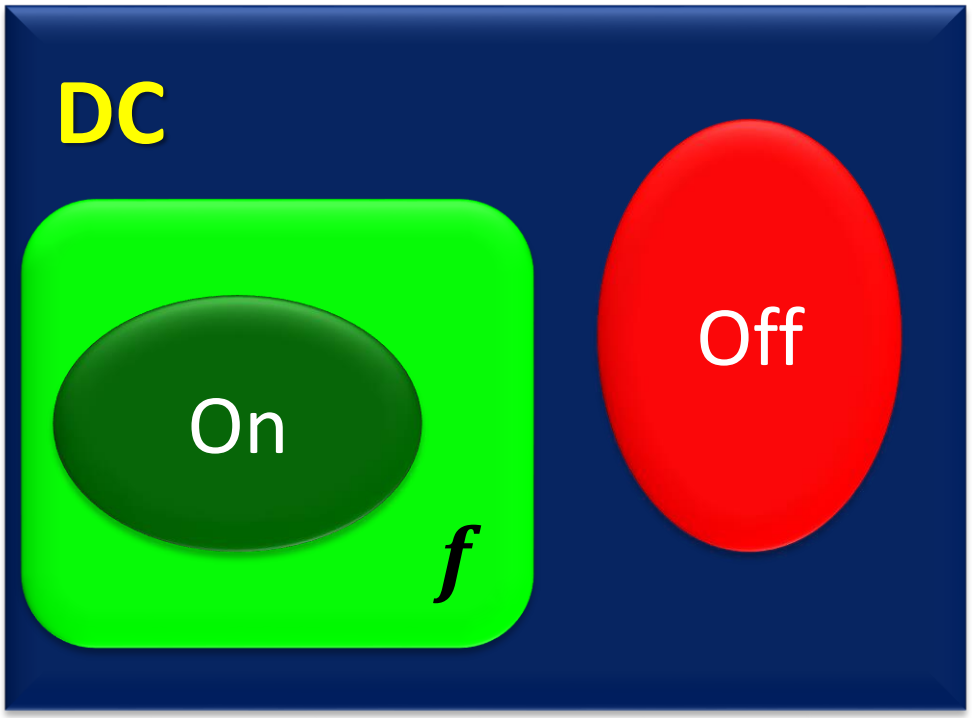
Incompletely Specified Functions

a b	*x y*
00	10
01	00 | 11
10	*1
11	*1 | 1*

General Boolean Relations

We have efficient ISF solvers

ISF solvers



exact, expensive

Quine-McCluskey (SOP, exact)

Scherzo (SOP, ZDDs)

Espresso (SOP, heuristic)

Minato-Morreale (BDDs)

Craig Interpolation (AIG, SAT)

inexpensive, inexact

Example

R

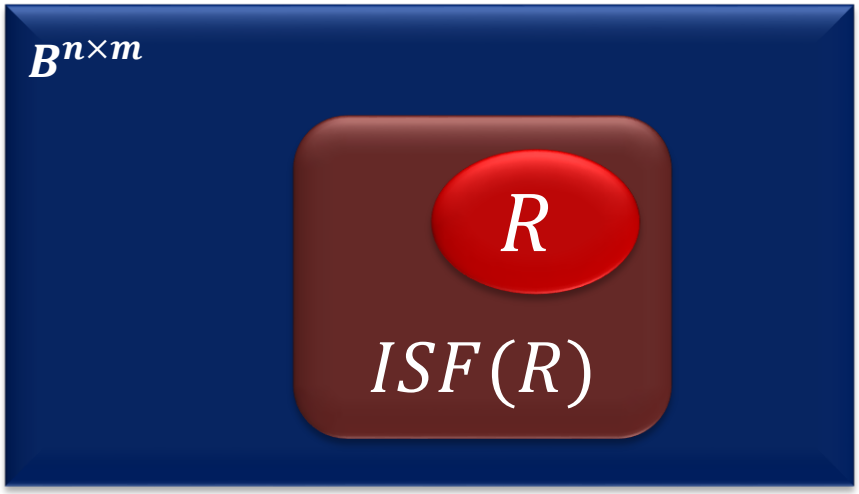
ab	xy
00	$\{10\}$
01	$\{00, 11\}$
10	$\{01, 10, 11\}$
11	$\{01\}$

ab	xy	xy	xy	xy
00	10	10	10	10
01	11	00	00	11
10	10	10	11	01
11	01	01	01	01
	$x = \bar{a} + \bar{b}$ $y = b$	$x = \bar{b}$ $y = ab$	$x = \bar{b}$ $y = a$	$x = \bar{a}$ $y = a + b$

Decoupling individual functions

$ISF(R)$

ab	xy
00	10
01	**
10	**
11	01

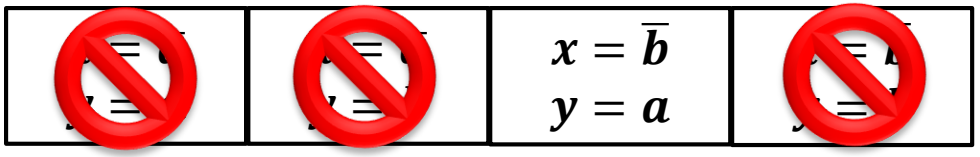


Decoupling the outputs

R

ab	xy
00	$\{10\}$
01	$\{00, 11\}$
10	$\{01, 10, 11\}$
11	$\{01\}$

ab	xy	xy	xy	xy
00	10	10	10	10
01	10	11	00	01
10	01	00	11	10
11	01	01	01	01



Decoupling individual functions

$ISF(R)$

ab	xy
00	10
01	**
10	**
11	01

$$ISF_x(a,b,x) = \exists y \ \chi_R(a,b,x,y)$$
$$= \chi_R(a,b,x,0) \vee \chi_R(a,b,x,1)$$

$$ISF_y(a,b,y) = \exists x \ \chi_R(a,b,x,y)$$
$$= \chi_R(a,b,0,y) \vee \chi_R(a,b,1,y)$$

Easy implementation with BDDs or AIGs

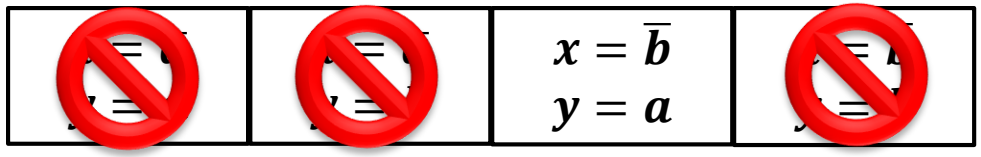
Decoupling the outputs

R

ab	xy
00	$\{10\}$
01	$\{00, 11\}$
10	$\{01, 10, 11\}$
11	$\{01\}$

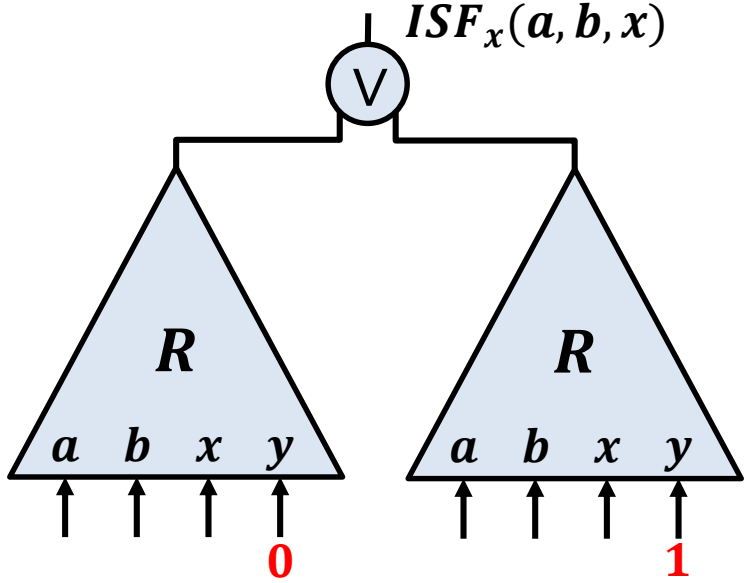
ab	xy	xy	xy	xy
00	10	10	10	10
01	10	11	00	01
10	01	00	11	10
11	01	01	01	01

Decoupling individual functions

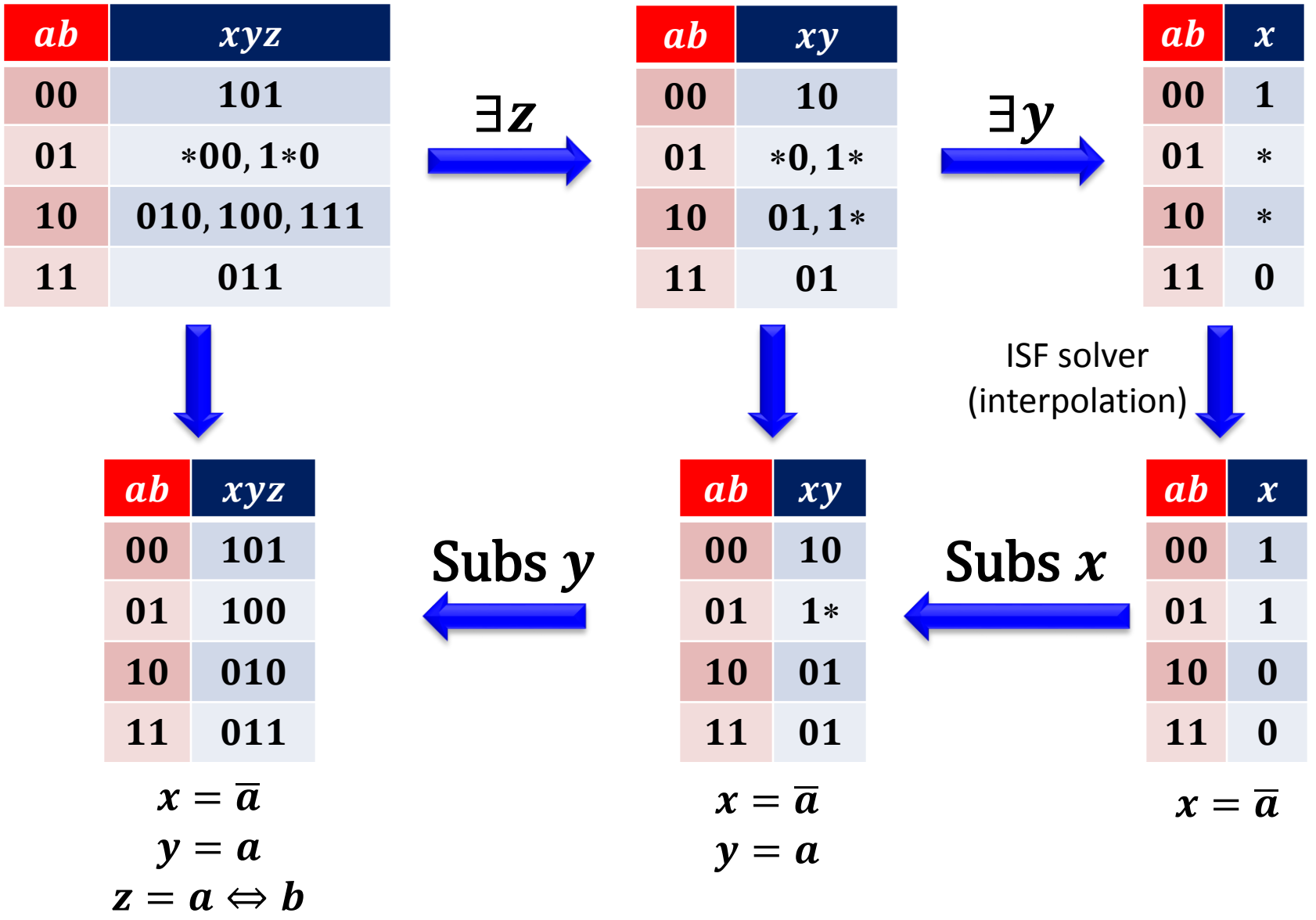


$ISF(R)$

ab	xy
00	10
01	**
10	**
11	01



ICCAD 2009: Reduce and substitute



ICCAD 2009: Reduce and substitute

ab	xyz
00	101
01	100
10	010
11	011

$x = \overline{a}$
$y = a$
$z = a \Leftrightarrow b$

ab	xyz
00	101
01	*00, 1*0
10	010, 100, 111
11	011

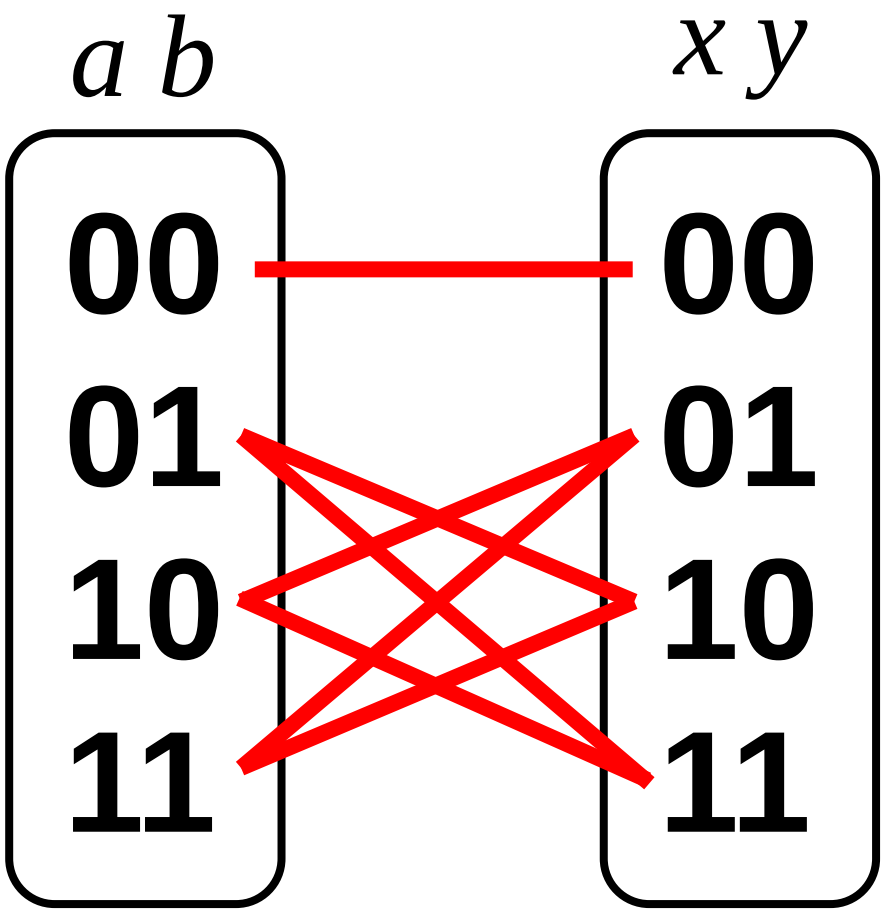
The final result depends on the order of the reductions

ab	xyz
00	101
01	000
10	111
11	011

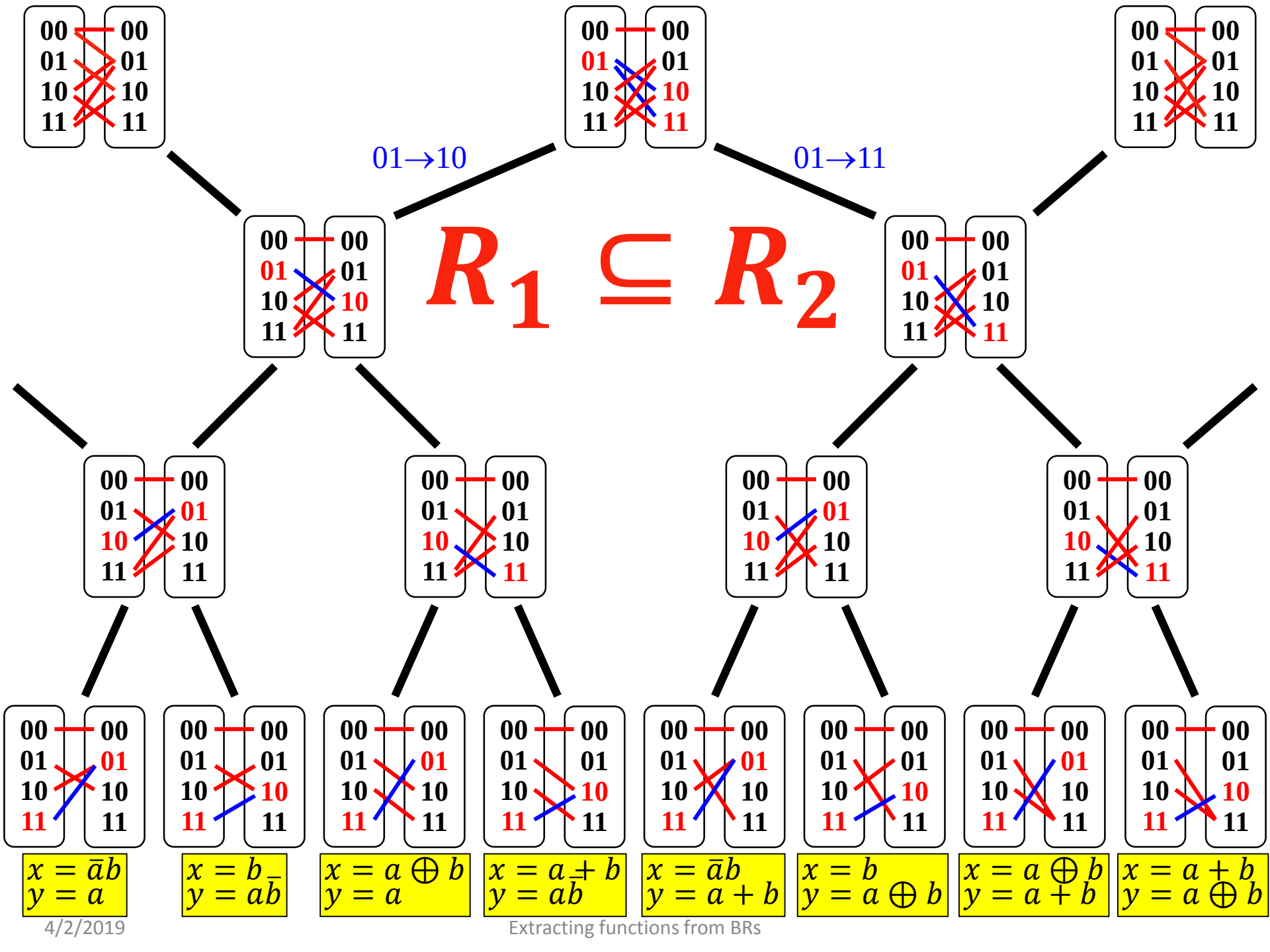
$z = a + \overline{b}$
$y = a$
$x = \overline{b}$

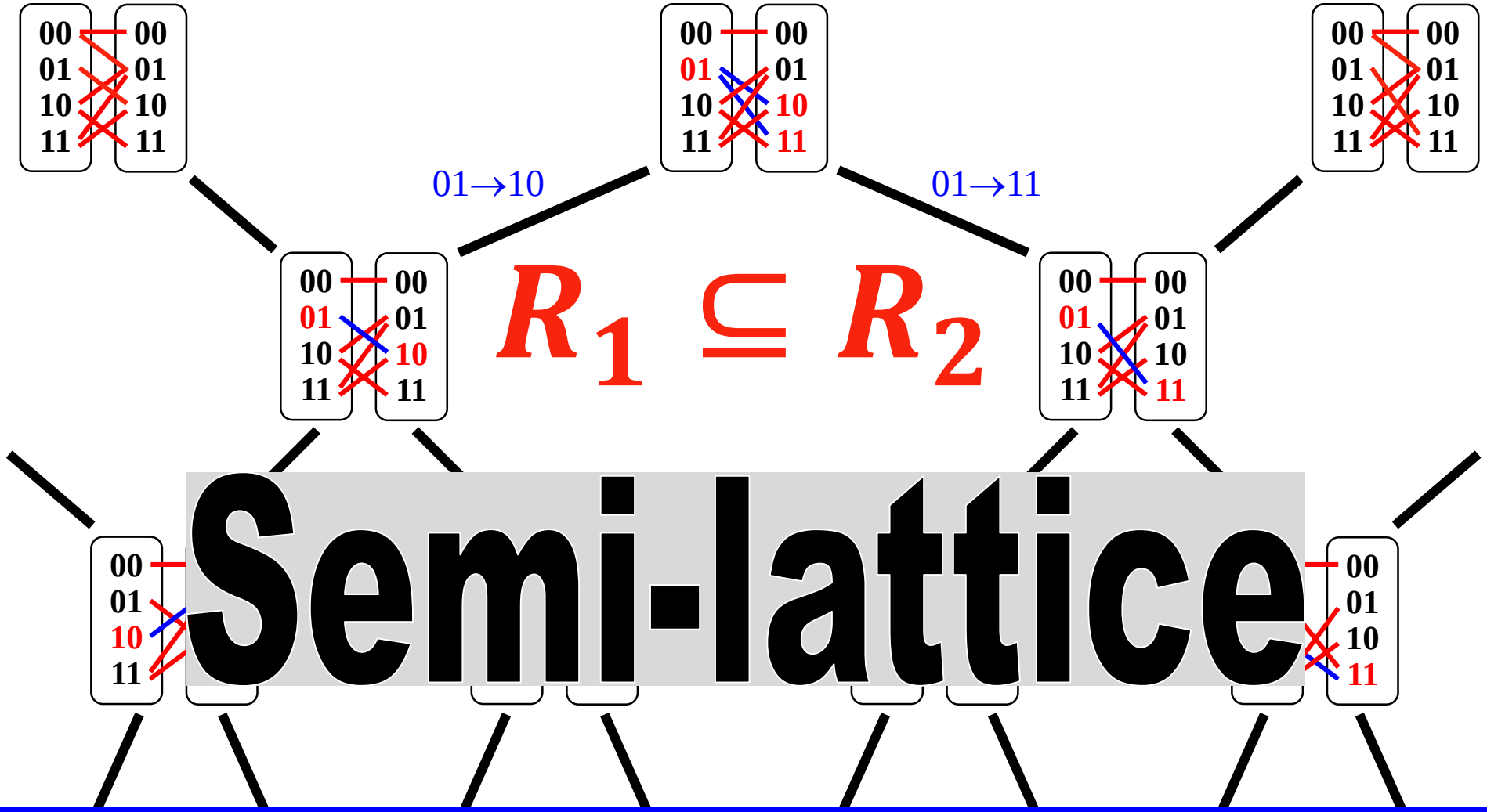
∃ abstractions: using AIGs

ISF minimization: using Craig interpolation



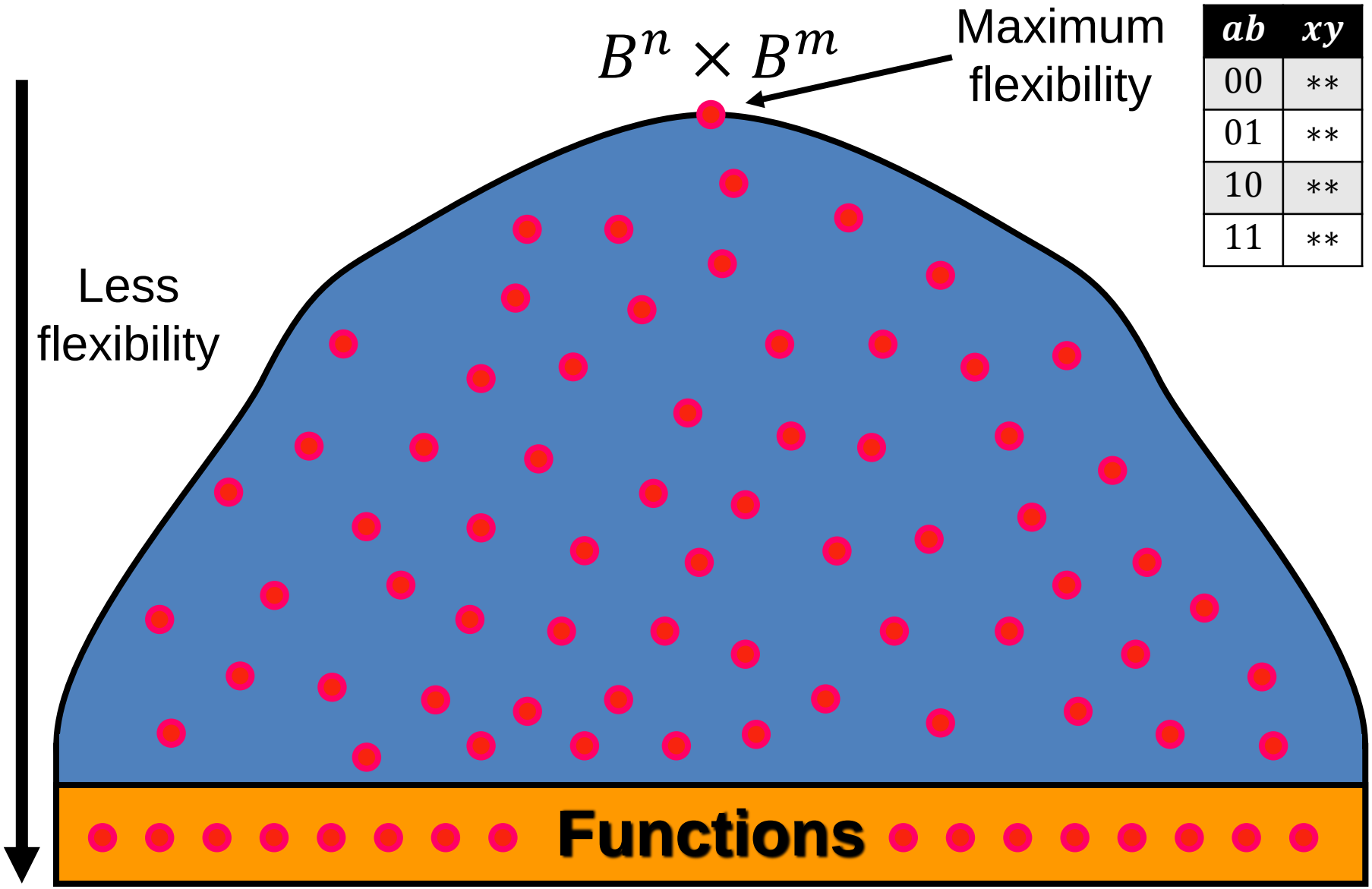
Boolean relation


01→10
01→11
$R_1 \subseteq R_2$
$x = \bar{a}b$ $y = a$
$x = b\bar{a}$ $y = a\bar{b}$
$x = a \oplus b$ $y = a$
$x = a + b$ $y = a\bar{b}$
$x = \bar{a}b$ $y = a + b$
$x = b$ $y = a \oplus b$
$x = a \oplus b$ $y = a + b$
$x = a + b$ $y = a \oplus b$

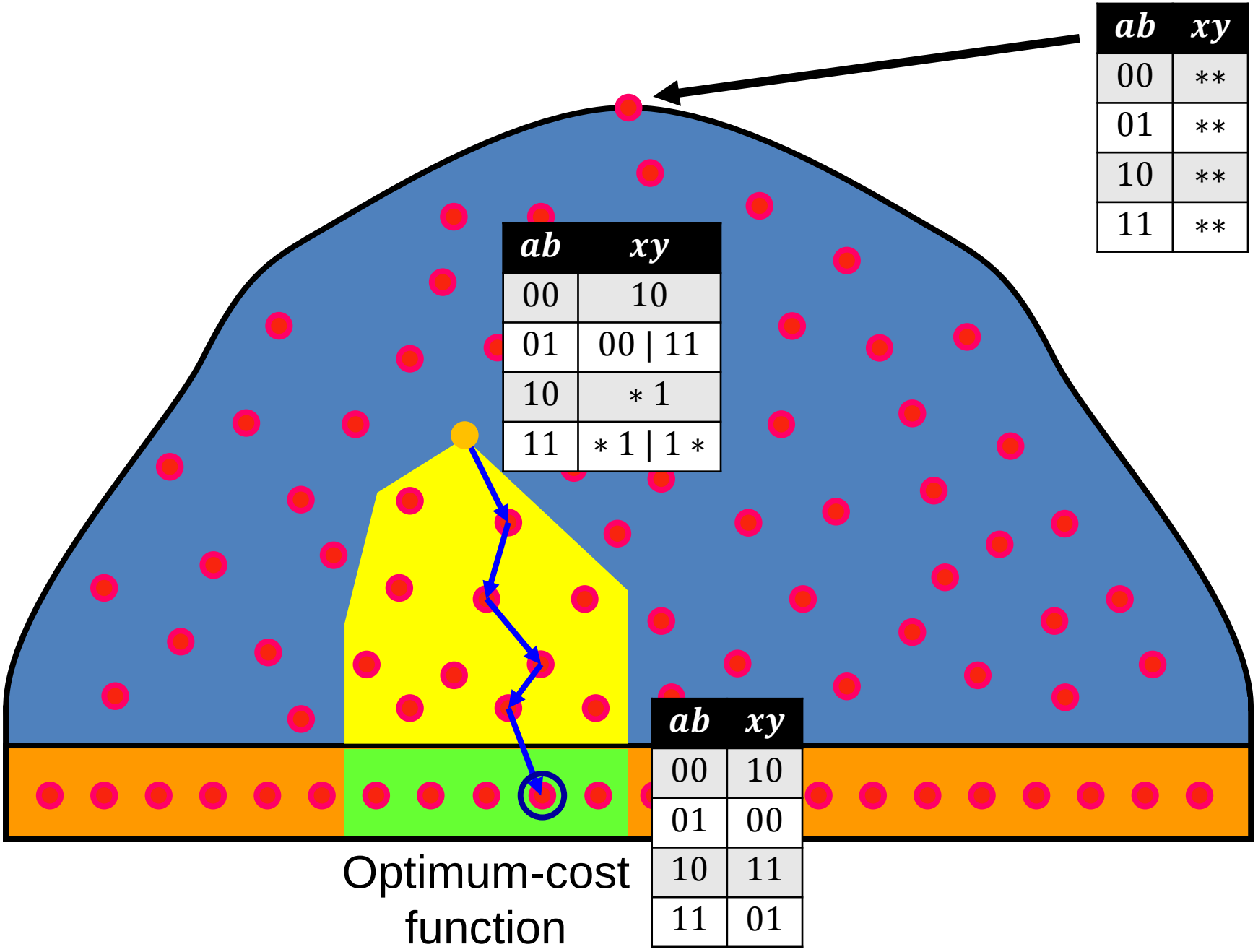


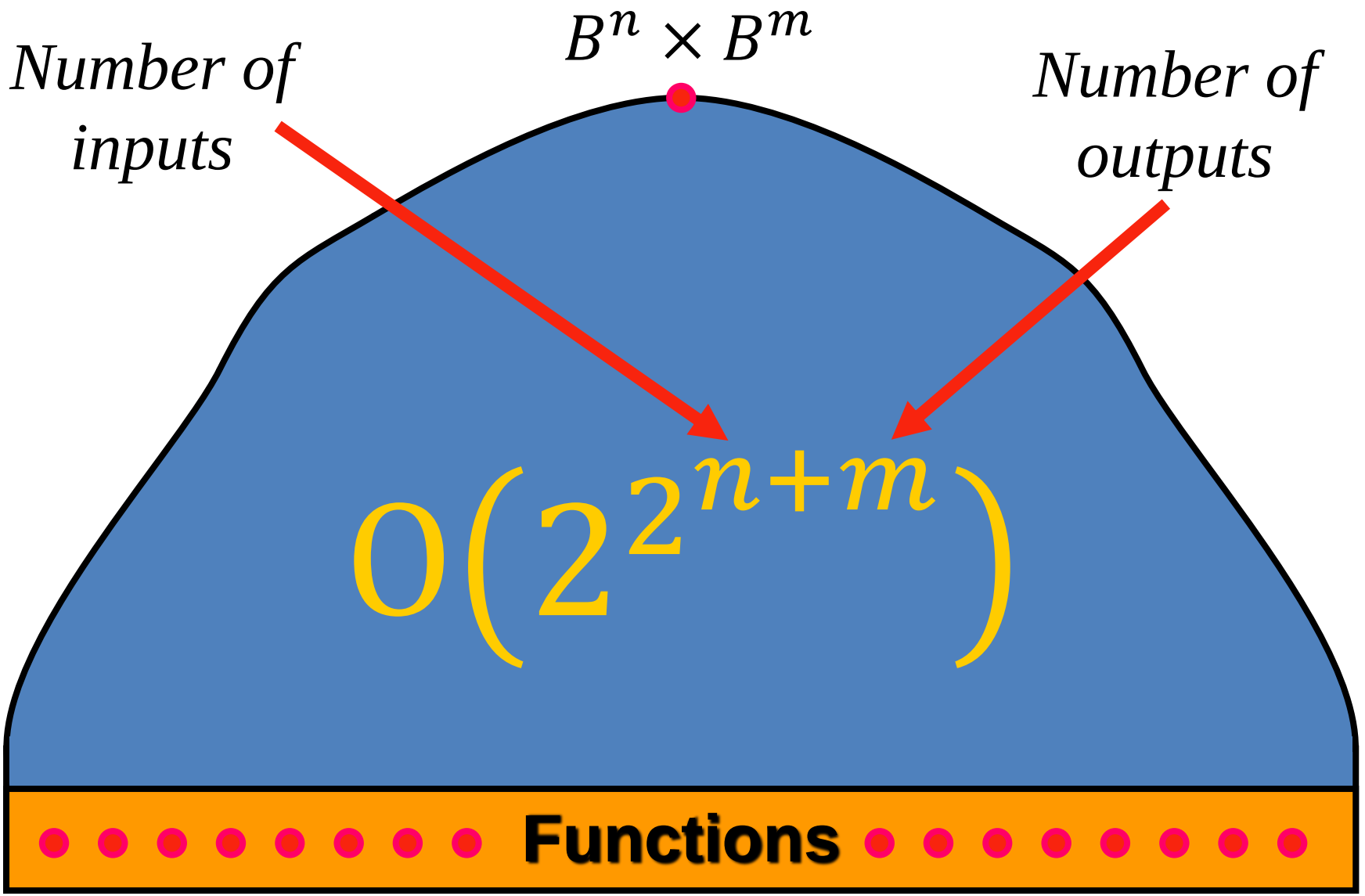
Functions

$x = \bar{a}b$ $y = a$	$x = b$ $y = a\bar{b}$	$x = a \oplus b$ $y = a$	$x = a + b$ $y = a\bar{b}$	$x = \bar{a}b$ $y = a + b$	$x = b$ $y = a \oplus b$	$x = a \oplus b$ $y = a + b$	$x = a + b$ $y = a \oplus b$

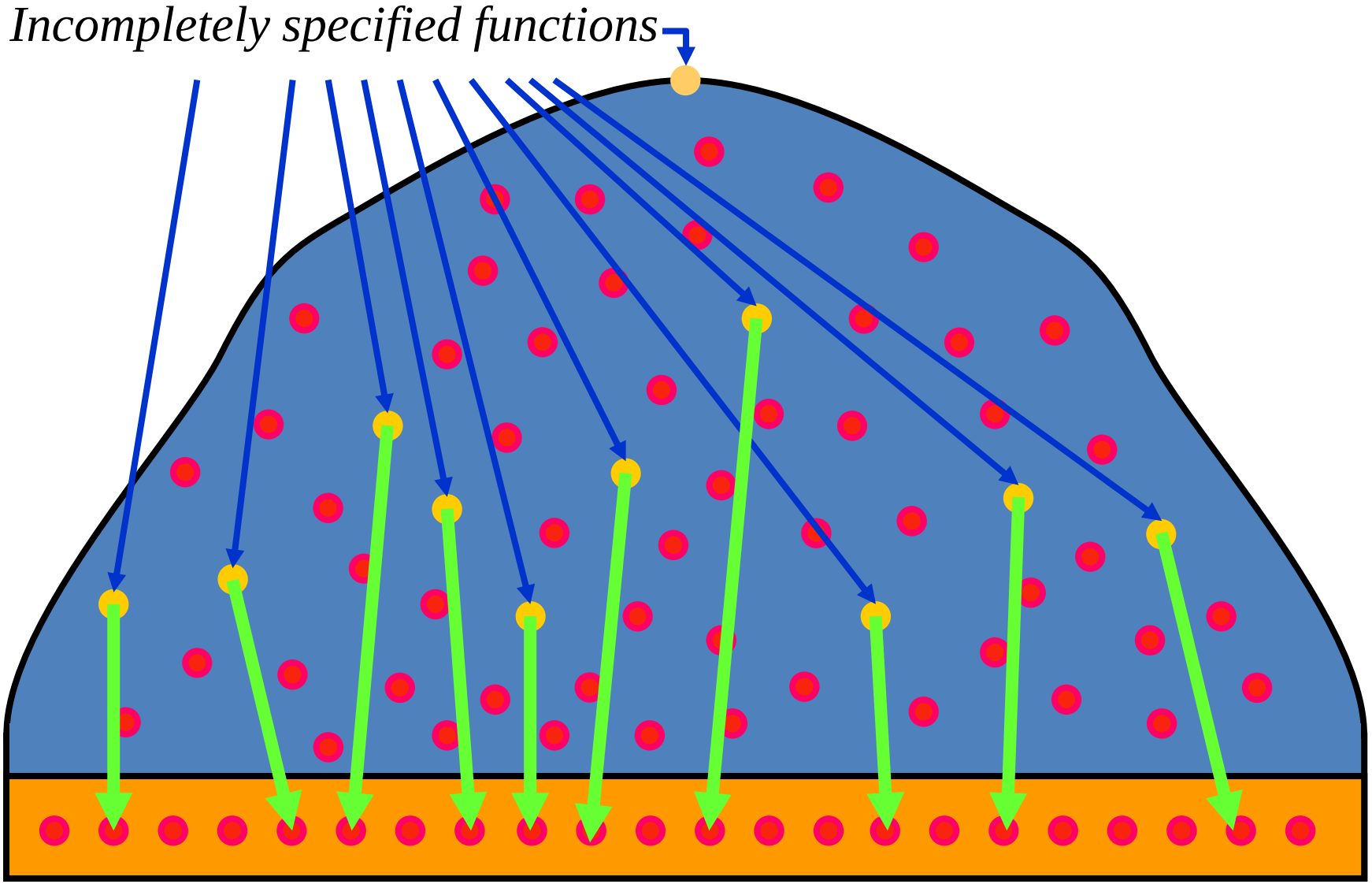


ab	xy
00	**
01	**
10	**
11	**

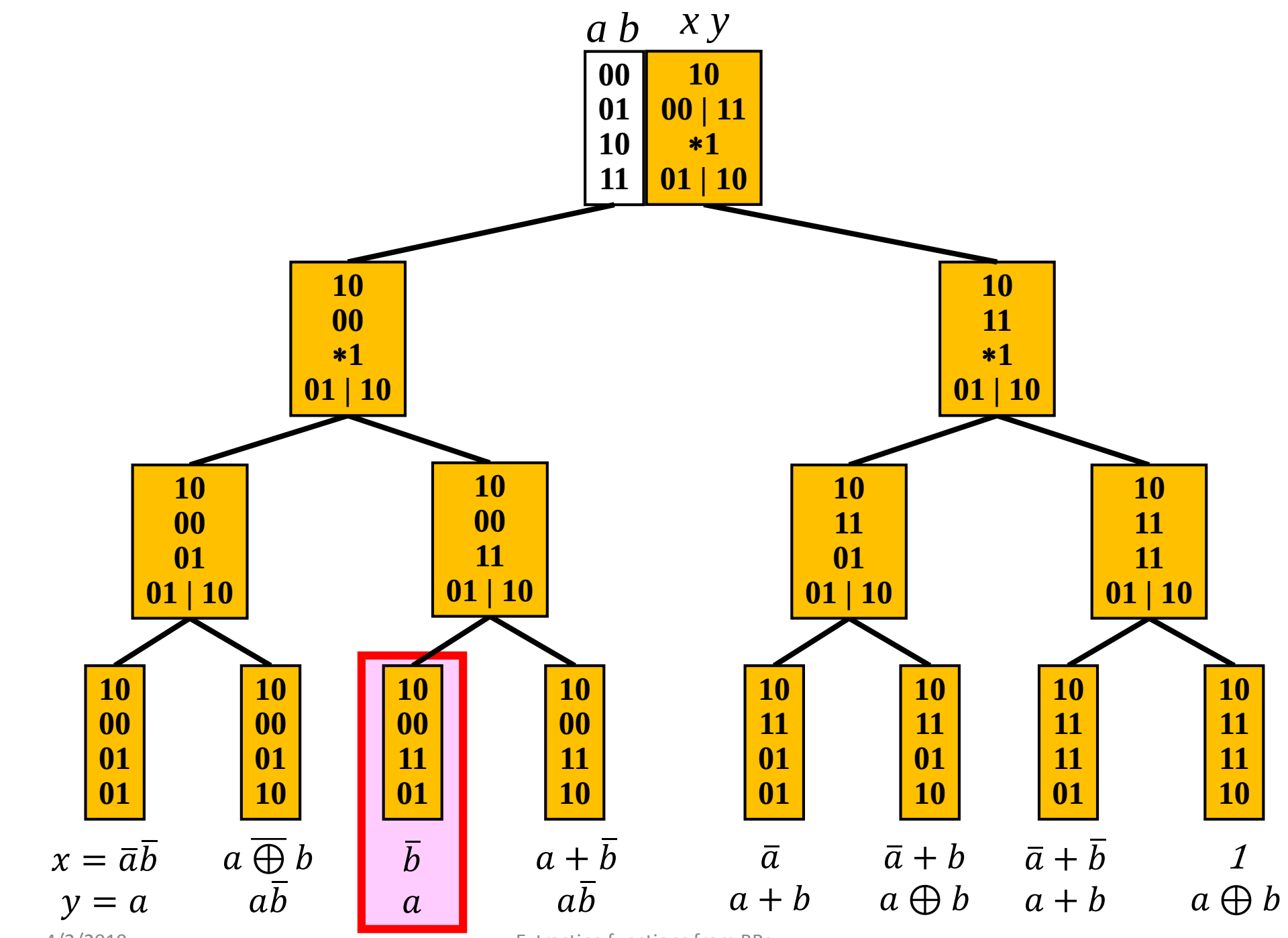

ab xy
00 **
01 **
10 **
11 **
ab xy
00 10
01 00 | 11
10 * 1
11 * 1 | 1 *
ab xy
00 10
01 00
10 11
11 01
Optimum-cost function



Size of the semi-lattice

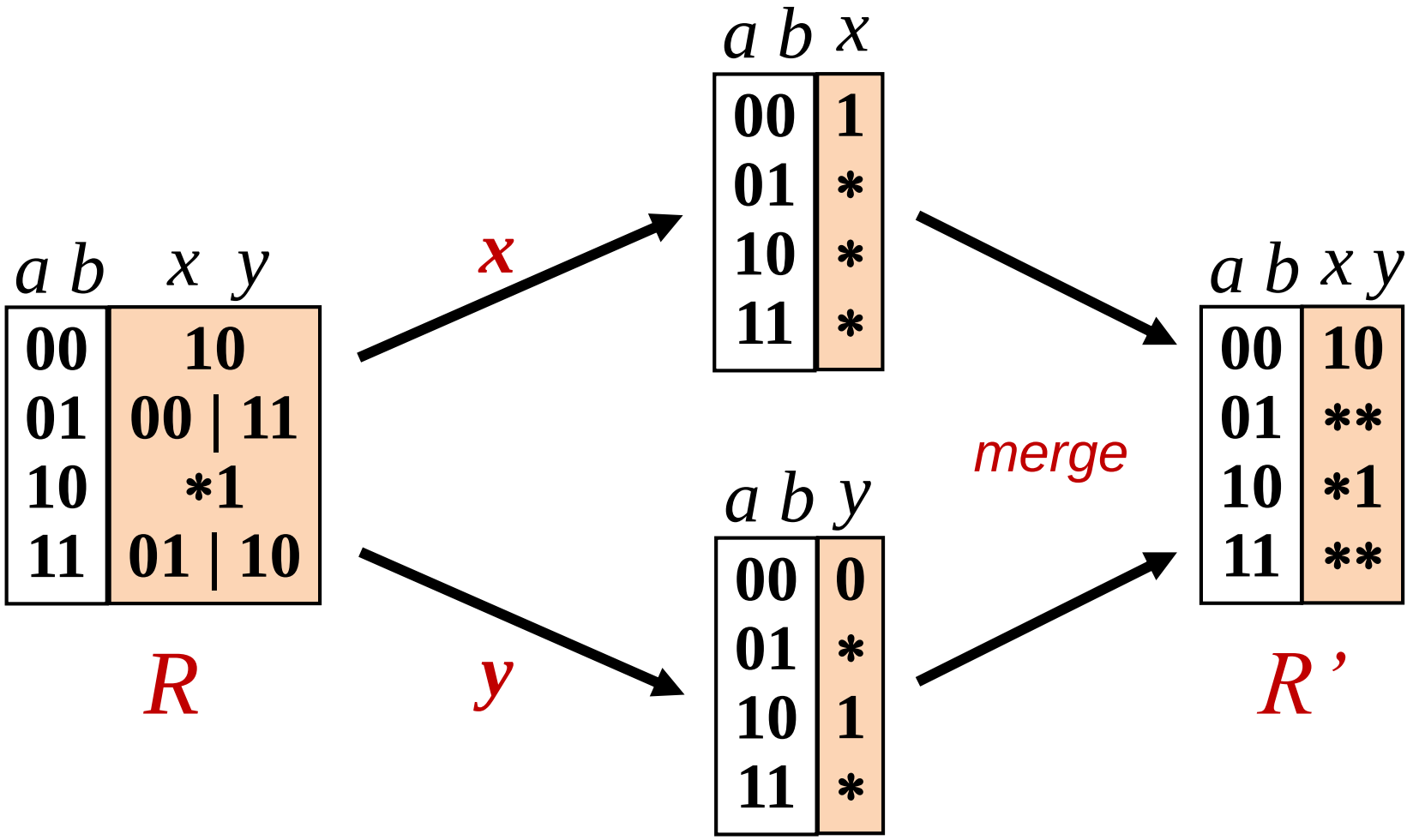


Shortcut: use ISF minimizers !
(ESPRESSO, Minato-Morreale, Scherzo)

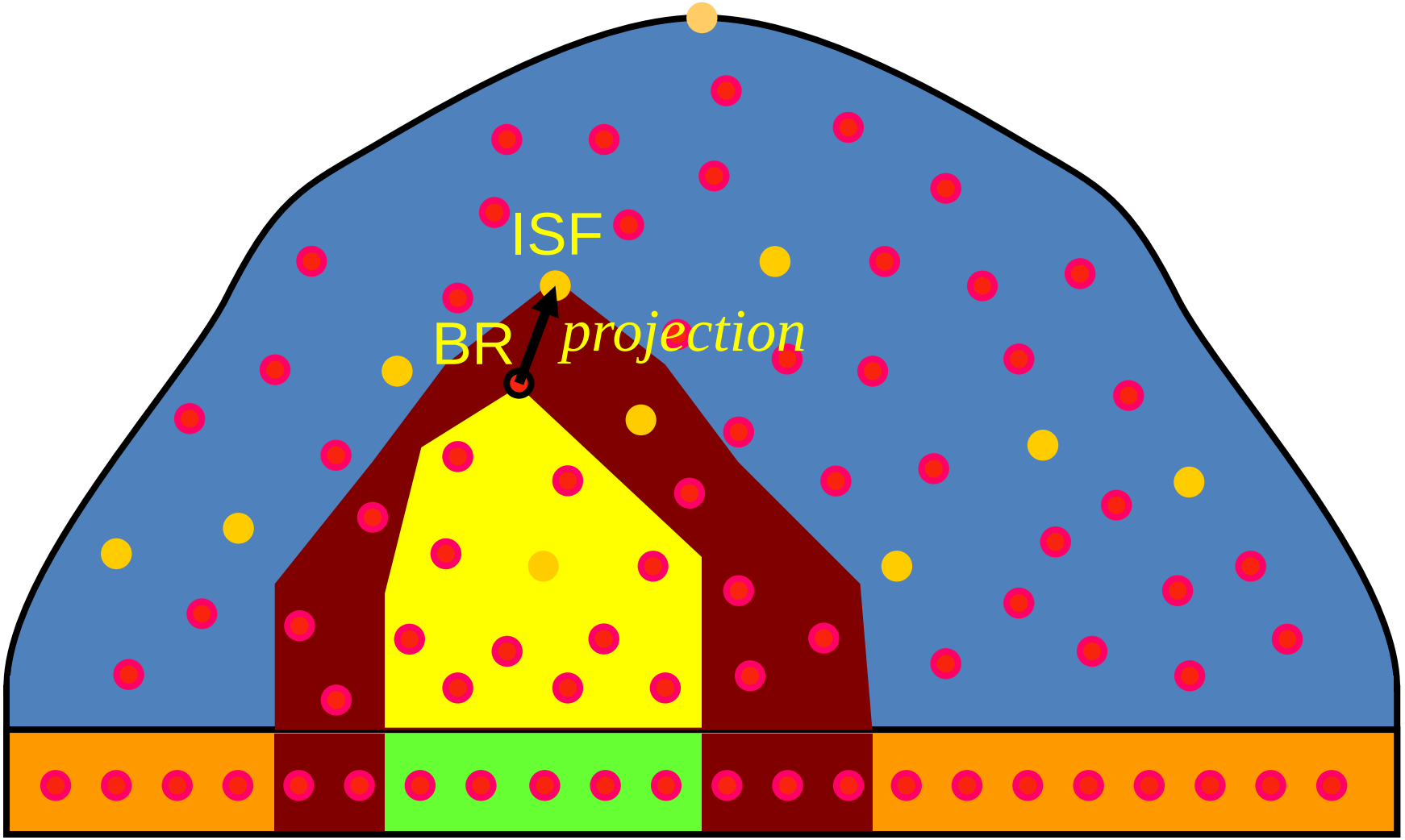

a b x y
00
01
10
11
10
00 | 11
*1
01 | 10
10
00
*1
01 | 10
10
11
*1
01 | 10
10
00
01
01 | 10
10
00
11
01 | 10
10
11
01
01 | 10
10
11
11
01 | 10
10
00
01
01
10
00
01
10
10
00
11
01
10
00
11
10
10
11
01
01
10
11
01
10
10
11
11
01
10
11
11
10
x = ā b̄
y = a
a ⊕ b
a b̄
b̄
a
a + b̄
a b̄
ā
a + b
ā + b
a ⊕ b
ā + b̄
a + b
1
a ⊕ b

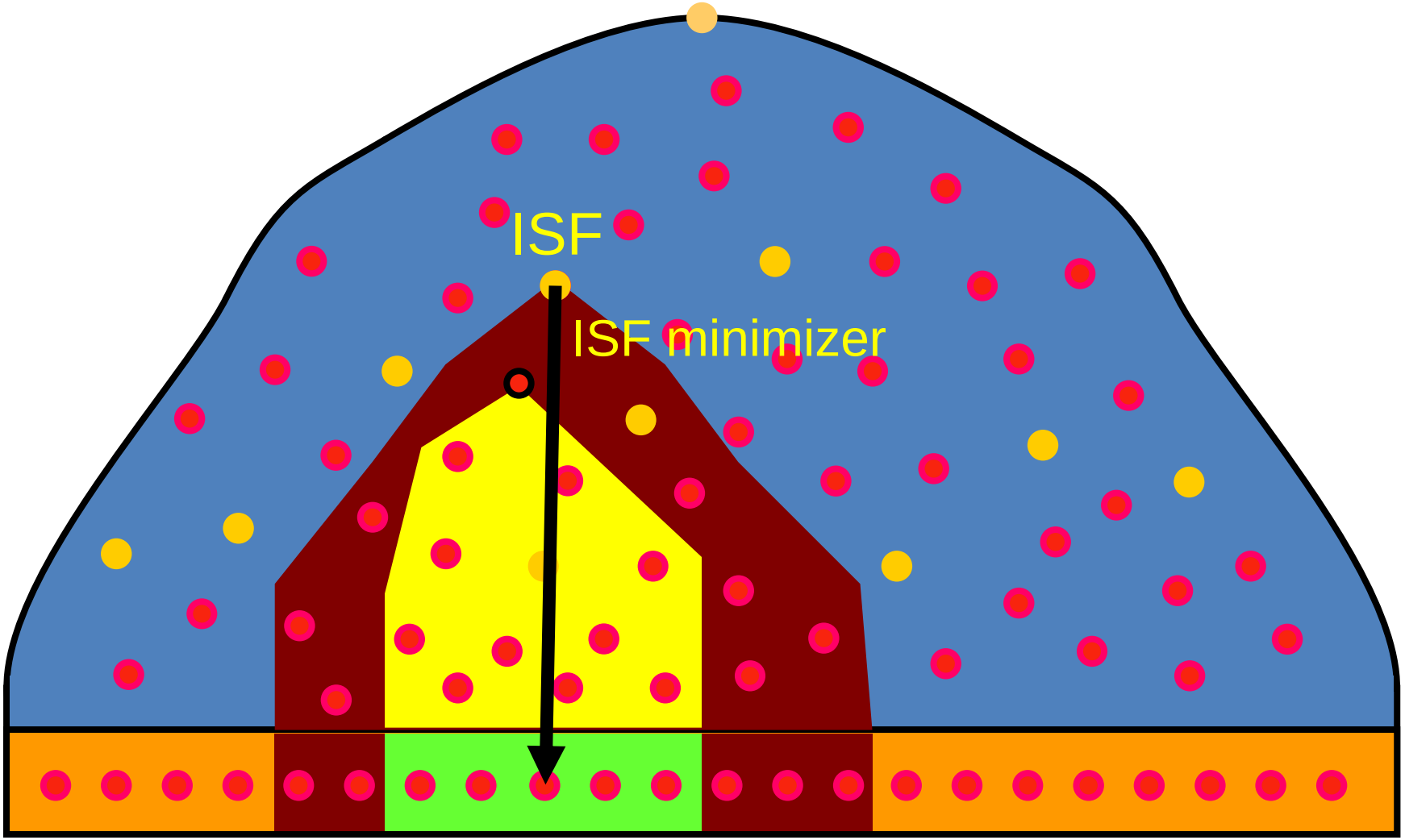
The recursive approach (DAC 2004, BREL)

Projections (∃ abs)

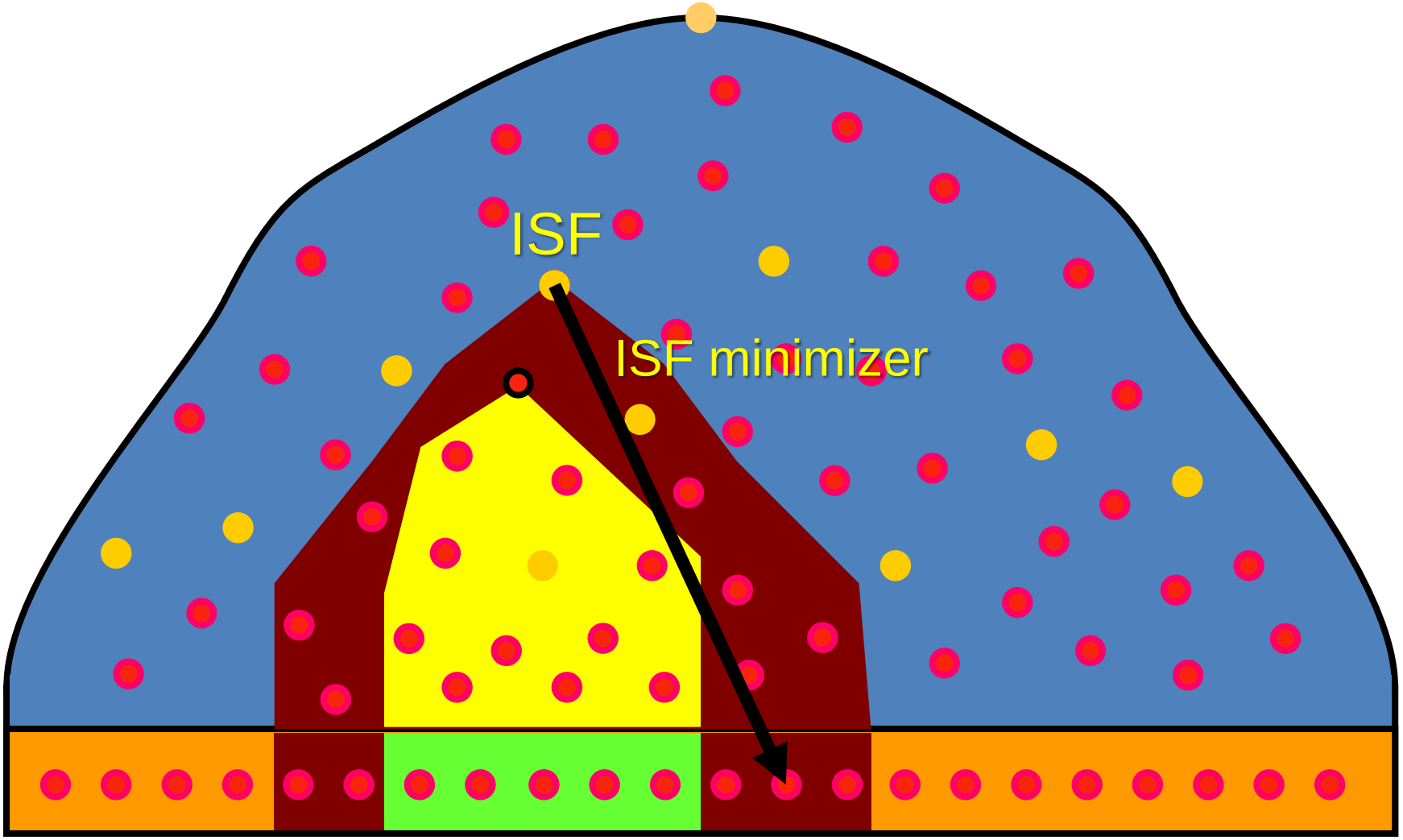


R' is an incompletely specified function, but …
has more flexibility than R $(R \subseteq R')$


ISF
BR
projection

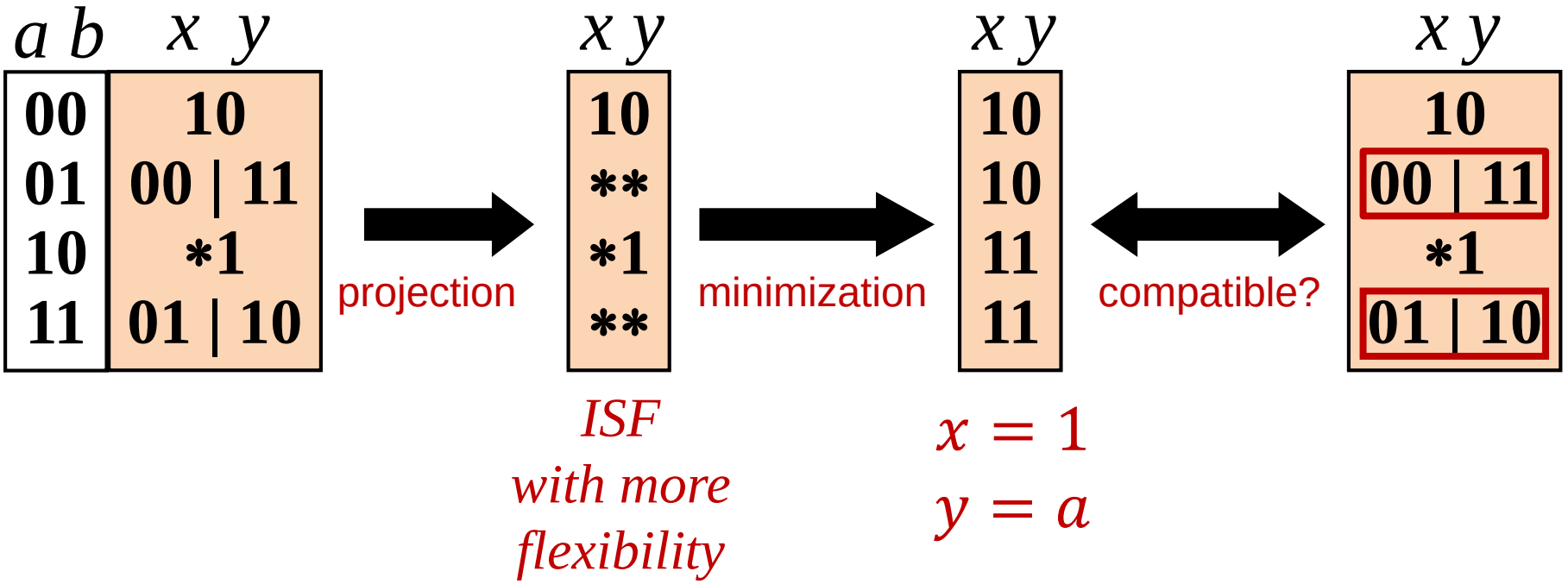


done !

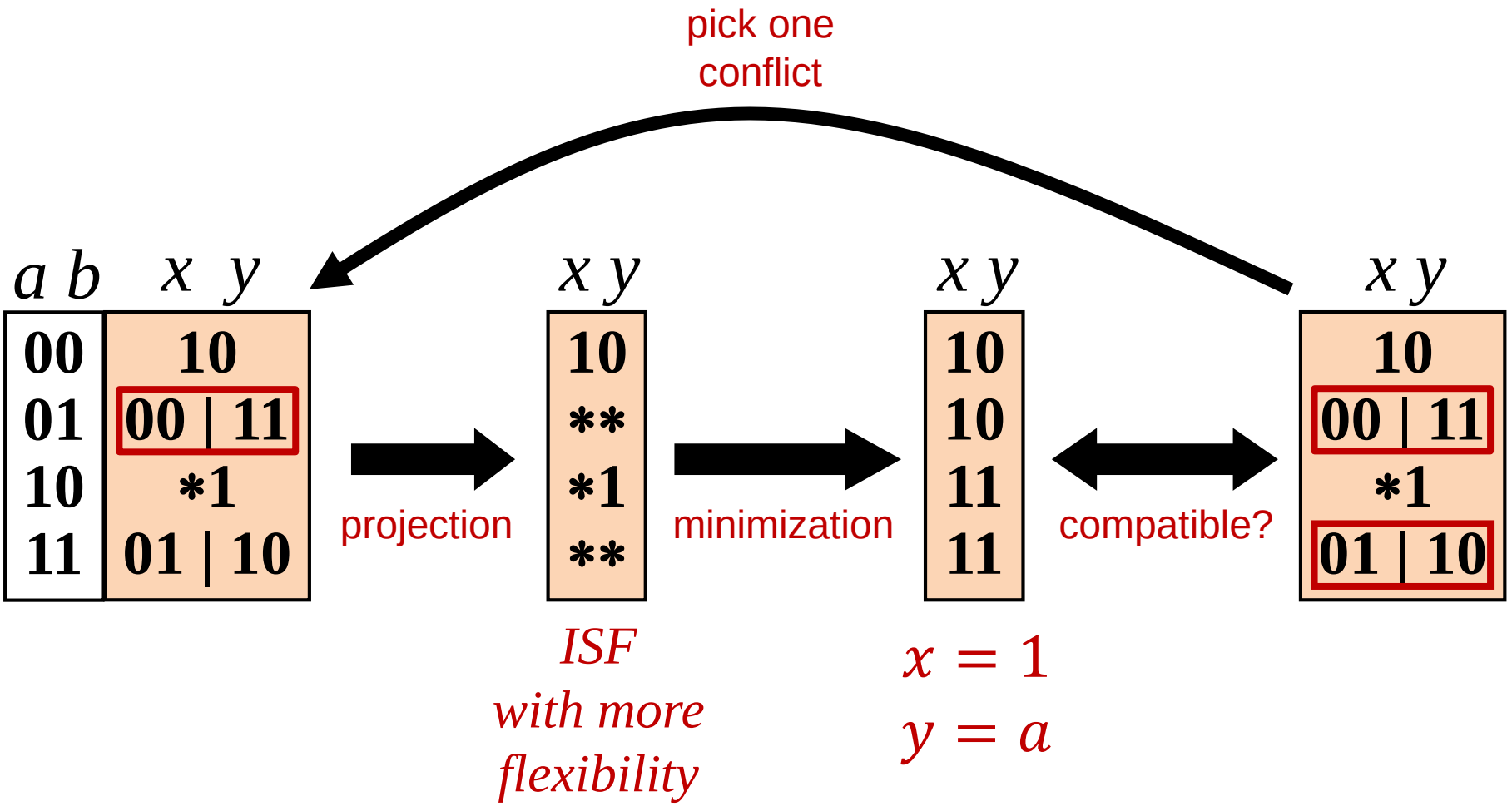


incompatible !

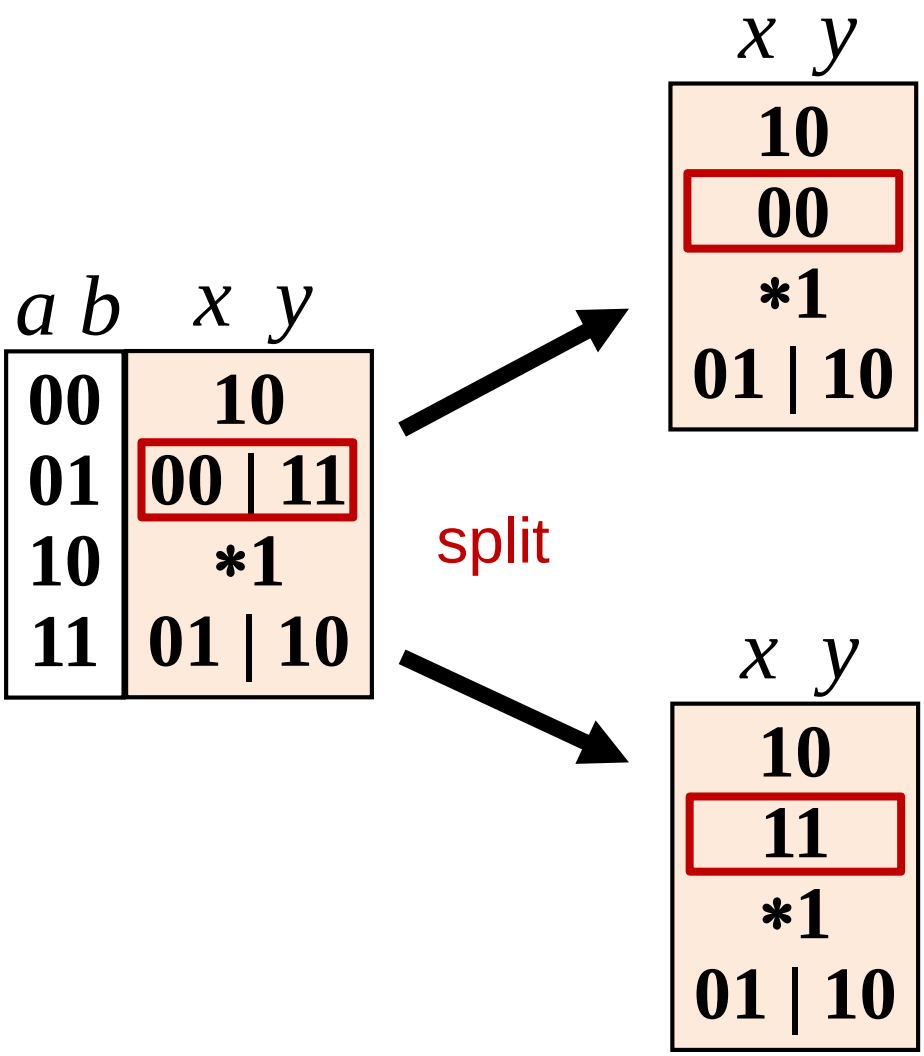
The recursive paradigm

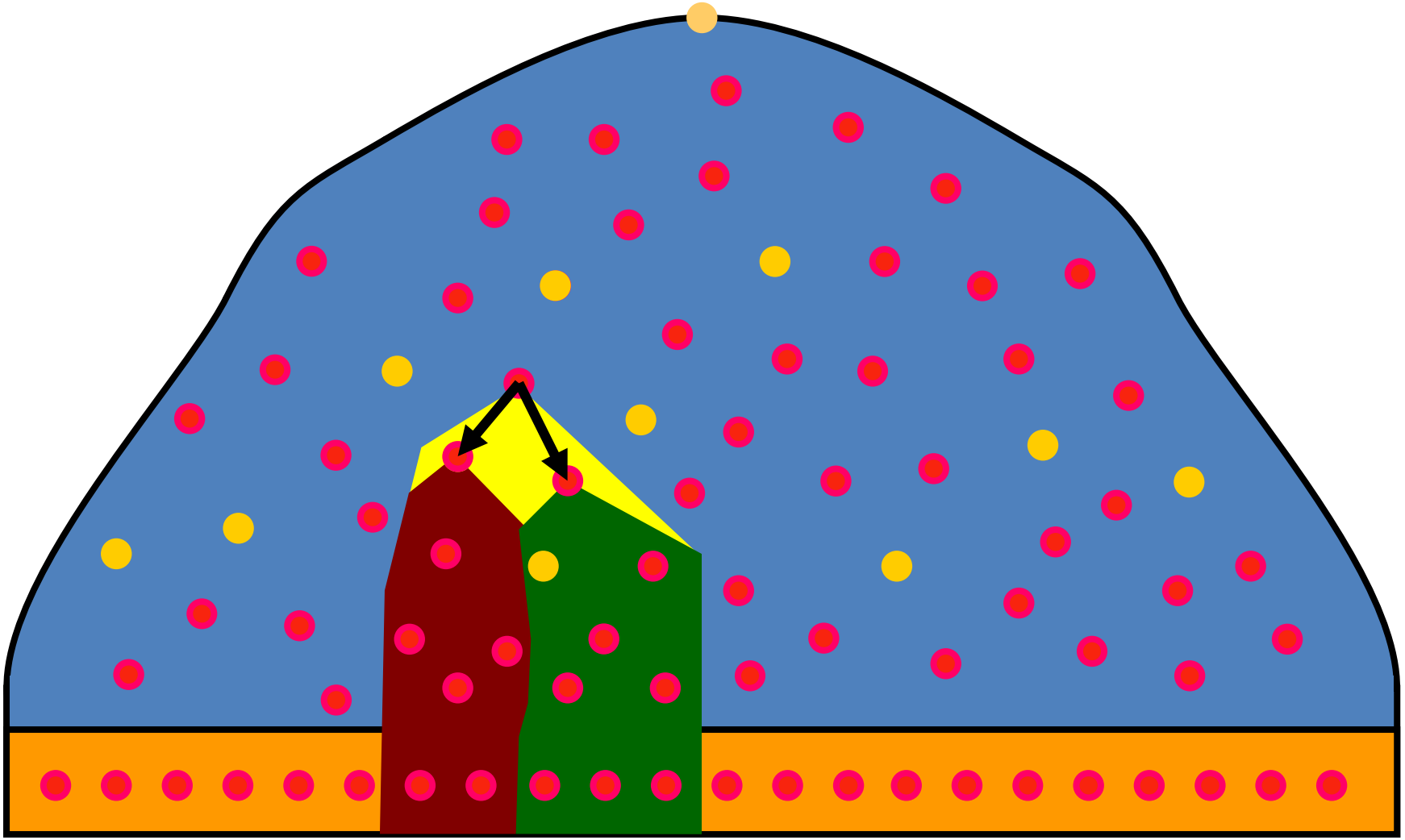


The recursive paradigm

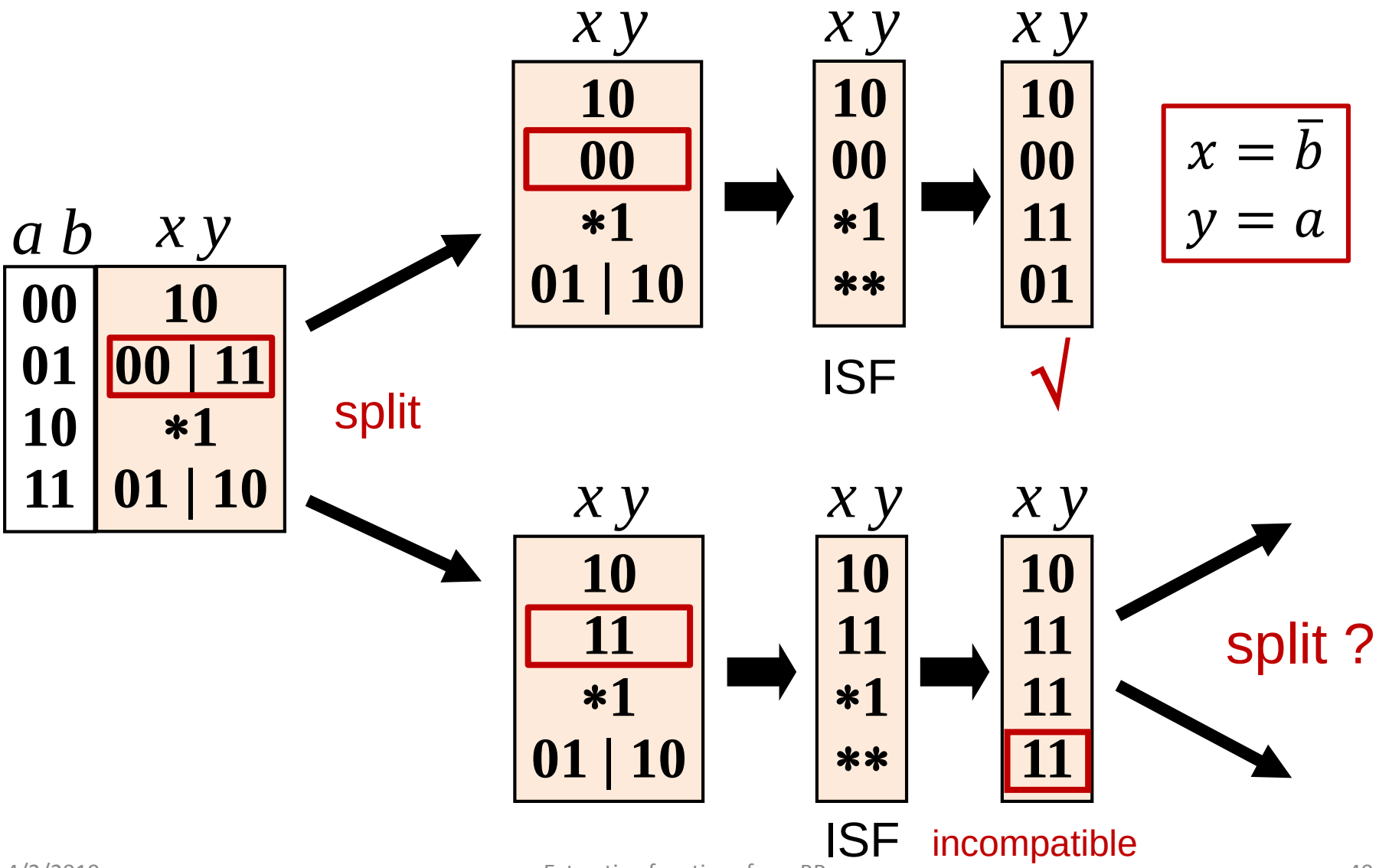


The recursive paradigm

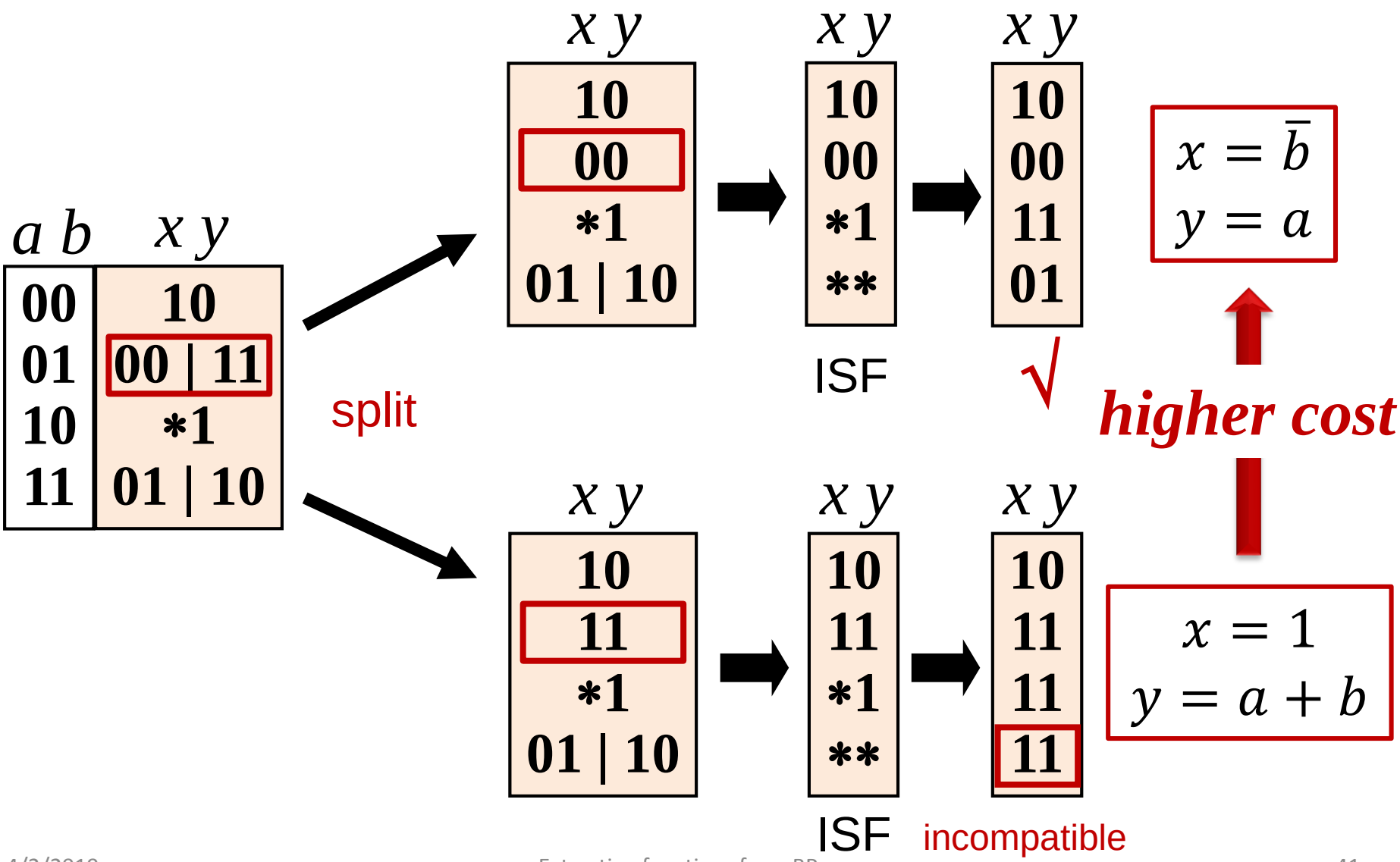



BR_1 BR_2

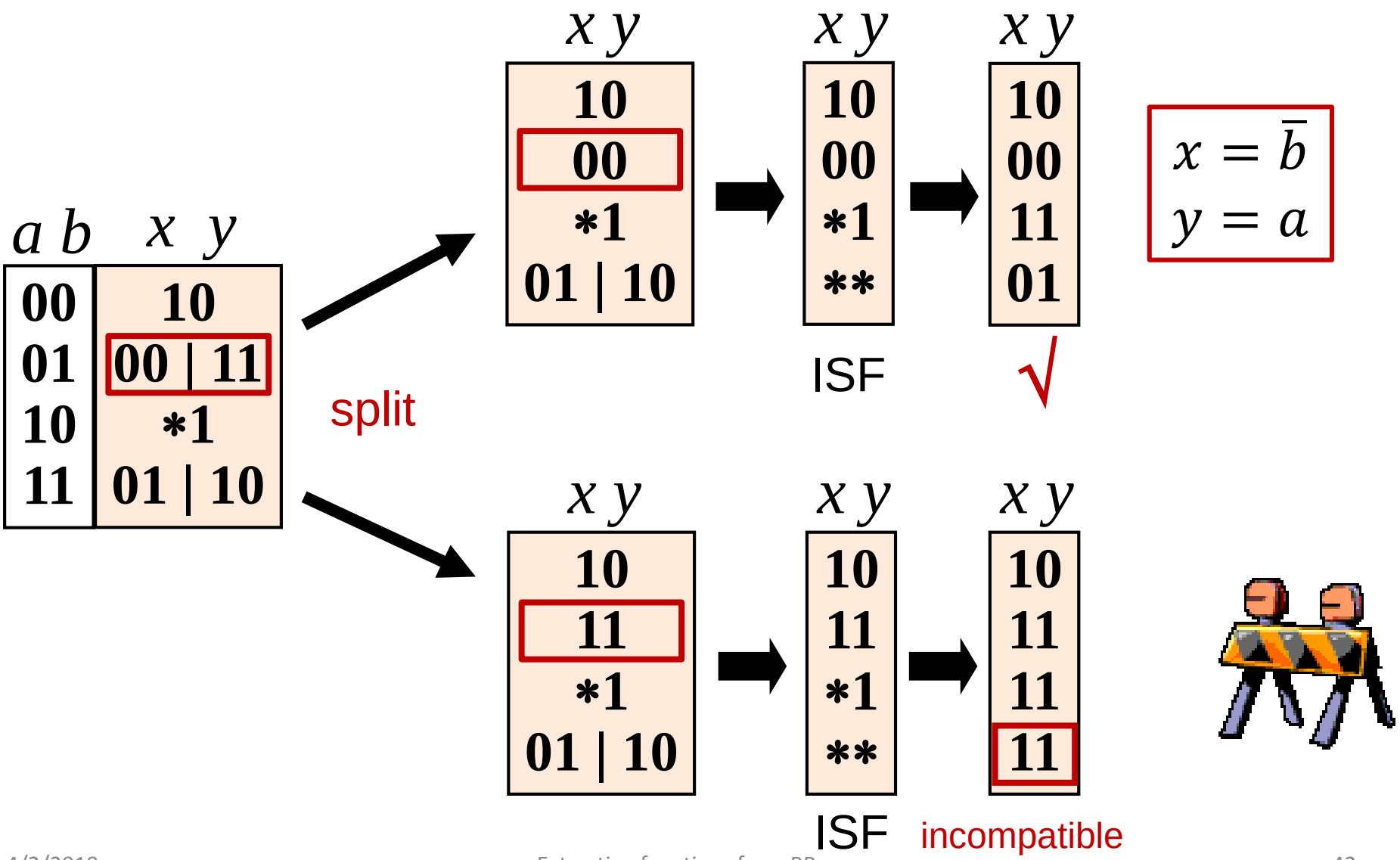
The recursive paradigm

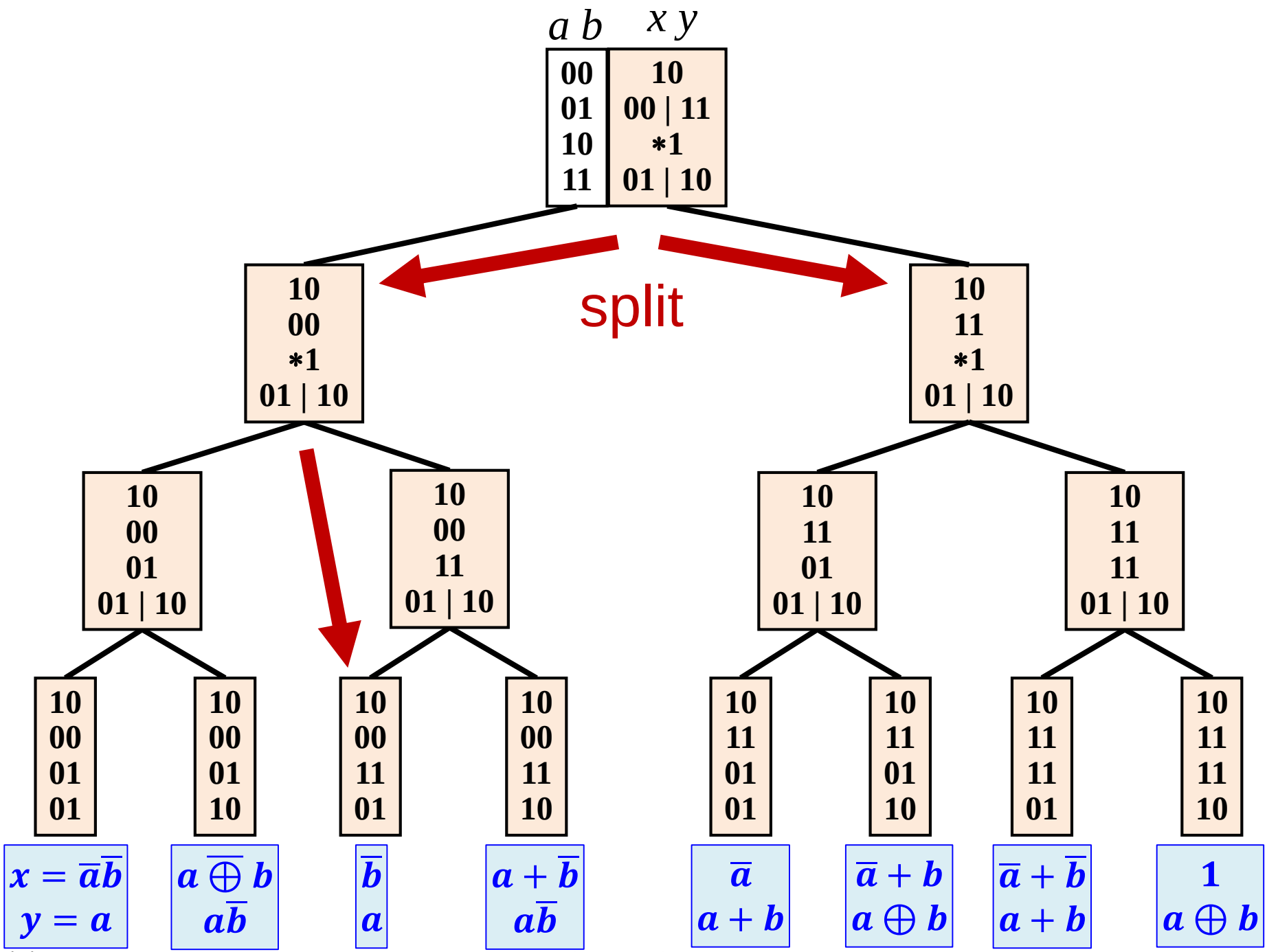


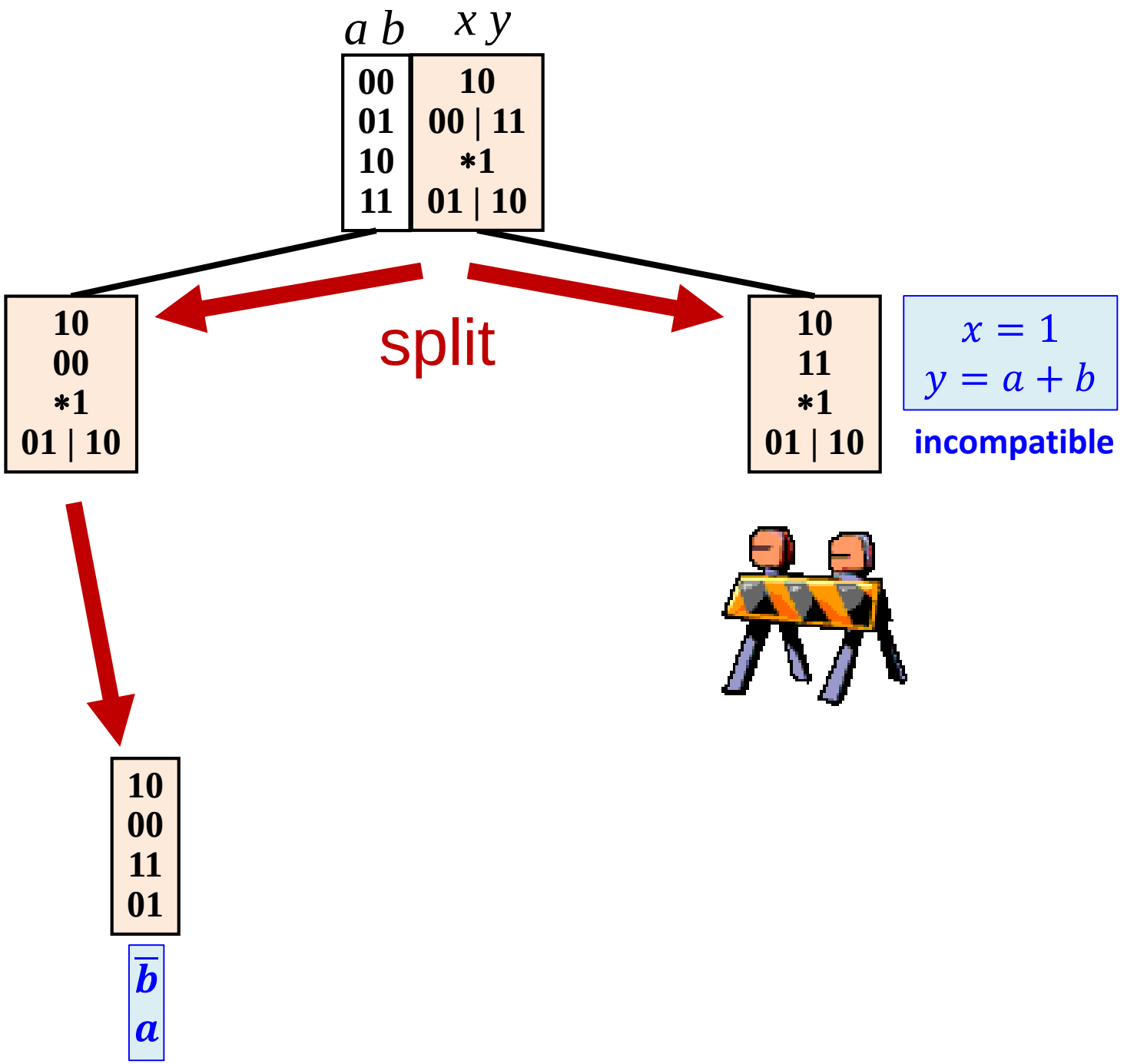
The recursive paradigm



The recursive paradigm



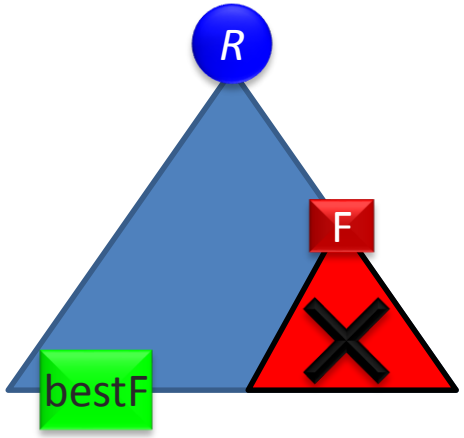

a b x y
00 01 10 11
10 00 | 11 *1 01 | 10
split
10 00 *1 01 | 10
10 11 *1 01 | 10
10 00 01 01 | 10
10 00 11 01 | 10
10 11 01 01 | 10
10 11 11 01 | 10
10 00 01 01
10 00 01 10
10 00 11 01
10 00 11 10
10 11 01 01
10 11 01 10
10 11 11 01
10 11 11 10
x = āb̄
y = a
a ⊕ b
ab̄
b̄
a
a + b̄
ab̄
ā
a + b
ā + b
a ⊕ b
ā + b̄
a + b
1
a ⊕ b

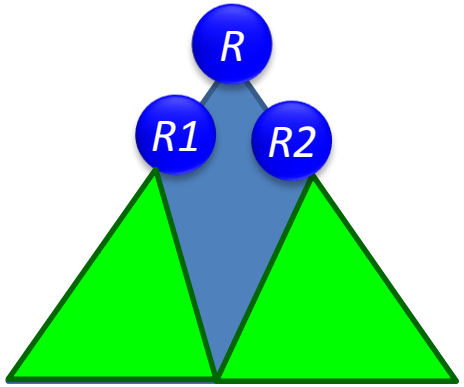

a b
x y
00
01
10
11
10
00 | 11
*1
01 | 10
split
10
00
*1
01 | 10
10
11
*1
01 | 10
x = 1
y = a + b
incompatible
10
00
11
01
b̄
a

BREL: algorithm

```
BREL(R, bestF)
  R'= ISF(R)  // obtains ISF: R' ⊇ R
  F = ISFsolver(R')    // obtains F
  // Prune search if larger cost
  if cost(F) ≥ cost(bestF): return;

  if F ⊆ R: bestF = F // no conflict
  else // split and recursive call
     conf = pickConflict(F, R)
     (R1, R2) = split(R, conf)
     BREL(R1, bestF)
     BREL(R2, bestF)
  endif
```





			Gyocro (ESPRESSO)			DAC 2004 (BREL)		
	PI	PO	LIT	AREA	CPU	LIT	AREA	CPU
int1	4	3	8	9280	0.03	9	8352	0.01
int10	6	4	32	44544	0.08	34	41296	0.02
c17i	5	3	34	34336	0.04	30	32016	0.01
she1	5	3	16	16240	0.04	15	17632	0.00
she2	5	5	31	30624	0.09	24	26488	0.01
she3	6	4	23	24592	0.08	21	21344	0.01
she4	5	6	56	62176	0.14	40	46864	0.03
gr	14	11	318	346608	3.43	313	322016	6.79
b9	16	5	321	382336	4.68	256	306240	0.19
int15	22	14	506	525248	21.94	459	472352	19.14
vtx	22	6	117	151728	30.10	101	94656	0.58
Normalized sum			**1.00**	**1.00**	**1.00**	**0.89**	**0.86**	**0.44**

ICCAD 2009: Craig interpolation

	Original				BDD				Xp			
Circuit	(#pi, #po)	#n	#l	#v	#n	#l	#v	time	#n	#l	#v	time
s5378	(214, 179)	624	12	1570	769	11	1561	200.2	1332	25	1561	49.1
s9234.1	(247, 211)	1337	25	3065	---	---	---	---	7696	55	2765	166.7
s13207	(700, 669)	1979	23	3836	---	---	---	---	5818	202	3554	897.9
s15850	(611, 597)	2648	36	15788	---	---	---	---	40078	136	13309	2596.9
s35932	(1763, 1728)	8820	12	7099	---	---	---	---	7360	25	6843	4811.1
s38584	(1464, 1452)	9664	26	19239	---	---	---	---	23726	331	17676	5476.7
b10	(28, 17)	167	11	159	199	9	152	0.1	193	8	152	1.0
b11	(38, 31)	482	21	416	1221	20	394	0.9	1562	52	394	5.5
b12	(126, 121)	953	16	1639	1619	15	1574	452.5	2261	23	1574	27.0
b13	(63, 53)	231	10	383	243	11	349	1.6	229	12	349	2.5
Ratio 1		1	1	1					3.35	4.53	0.91	
Ratio 2		1	1	1	1.65	0.94	0.97		2.27	1.71	0.97	
Ratio 3					1	1	1		1.38	1.82	1	

Conclusions

- Two options to solve Boolean Relations (applicable to other CS problems):
  - Exploit the knowledge we have about our problem and use smart algorithms.
  - "Admit" the *lack of knowledge about* of our problem and … (guess what) use Machine Learning.

- Our knowledge:
  - We know about SAT, AIGs, BDDs, ISF, Craig interpolants,…
  - The semi-lattice of Boolean Relations is *infested* by ISFs!

- Two ways:
  - Small and accurate (BDDs, recursive, Branch & Bound).
  - Large and less accurate (AIGs, SAT, Craig interpolants).

- Can both methods be combined?